Package ‘stcos’

May 27, 2020

Type Package

Title Space-Time Change of Support

Version 0.3.0

Author Andrew M. Raim [aut, cre],
Scott H. Holan [aut, res],
Jonathan R. Bradley [aut, res],
Christopher K. Wikle [aut, res]

Maintainer Andrew M. Raim <andrew.raim@gmail.com>

URL https://github.com/holans/ST-COS

Description
Spatio-temporal change of support (STCOS) methods are designed for statistical inference on geographic and time domains which differ from those on which the data were observed. In particular, a parsimonious class of STCOS models supporting Gaussian outcomes was introduced by Bradley, Wikle, and Holan <doi:10.1002/sta4.94>. The 'stcos' package contains tools which facilitate use of STCOS models.

License GPL (>= 2)

Depends R (>= 3.3), Rcpp, Matrix, sf, dplyr

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.0

Encoding UTF-8

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-05-27 12:20:03 UTC

R topics documented:

acs_sf *Shapes and ACS estimates for Boone County, MO.*

Description

An sf object with ACS estimates for:

- Boone County, Missouri
- Table B19013
- Block group level geography
- Years 2013 - 2017

Usage

```
acs5_2013

acs5_2014

acs5_2015

acs5_2016

acs5_2017
```

Format

`sf` objects.

An object of class `sf` (inherits from `data.frame`) with 87 rows and 9 columns.

An object of class `sf` (inherits from `data.frame`) with 87 rows and 9 columns.

An object of class `sf` (inherits from `data.frame`) with 85 rows and 9 columns.

An object of class `sf` (inherits from `data.frame`) with 87 rows and 9 columns.

An object of class `sf` (inherits from `data.frame`) with 87 rows and 9 columns.

Details

Shapefiles were gathered via the `tigris` package, and ACS estimates were downloaded from the American FactFinder `http://factfinder.census.gov`. Data were assembled on 2/28/2019. See `data-prep-aff.R` in the Columbia example code for details.

`adjacency_matrix` *Sparse adjacency matrix between two sets of areas.*

Description

A convenience function to convert output from `sf::st_touches` to a sparse matrix as defined in the `Matrix` package.

Usage

```
adjacency_matrix(dom)
```

Arguments

`dom`	An `sf` object representing a domain of areal units.

Details

Returns a matrix `A` whose (i,j)th entry contains a 1 if areal units `dom[i,]` and `dom[j,]` are adjacent; 0 otherwise.

Value

An adjacency matrix

Examples

```
data("acs_sf")
dom = acs5_2013[1:4,]
A = adjacency_matrix(dom)
```

`areal_spacetime_bisquare` *Areal Space-Time Bisquare Basis*

Description

Space-Time bisquare basis on areal data.

Usage

```
areal_spacetime_bisquare(dom, period, knots, w_s, w_t, control = NULL)
```

Arguments

`dom`	An `sf` or `sfc` object with areas $A_1, \ldots, A_n$ to evaluate.
`period`	A numeric vector of time periods $v_1, \ldots, v_m$ to evaluate for each area.
`knots`	Spatio-temporal knots $(\boldsymbol{c}_1, g_1), \ldots, (\boldsymbol{c}_r, g_r)$ for the basis. See "Details".
`w_s`	Spatial radius for the basis.
`w_t`	Temporal radius for the basis.
`control`	A `list` of control arguments. See "Details".

Details

Notes about arguments:

- `knots` may be provided as either an `sf` or `sfc` object, or as a matrix of points.
- If an `sf` or `sfc` object is provided for `knots`, r three-dimensional `POINT` entries are expected in `st_geometry(knots)`. Otherwise, `knots` will be interpreted as an $r \times 3$ numeric matrix.
- If `knots` is an `sf` or `sfc` object, it is checked to ensure the coordinate system matches `dom`.

For each area A in the given domain, and time period $\boldsymbol{v} = (v_1, \ldots, v_m)$ compute the basis functions

$$\psi_j^{(m)}(A, \boldsymbol{v}) = \frac{1}{m} \sum_{k=1}^{m} \frac{1}{|A|} \int_A \psi_j(\boldsymbol{u}, v_k) d\boldsymbol{u},$$

for $j = 1, \ldots, r$. Here, $\varphi_j(\boldsymbol{u}, v)$ represent spacetime_bisquare basis functions defined at the point level using $\boldsymbol{c}_j$, g_j, w_s, and w_t.

The basis requires an integration which may be computed using one of two methods. The `mc` method uses a Monte Carlo approximation

$$\psi_j^{(m)}(A, \boldsymbol{v}) \approx \frac{1}{m} \sum_{k=1}^{m} \frac{1}{Q} \sum_{q=1}^{Q} \psi_j(\boldsymbol{u}_q, v_k),$$

based on a random sample of locations $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_Q$ from a uniform distribution on area A. The `rect` method uses a simple quadrature approximation

$$\psi_j^{(m)}(A, \boldsymbol{v}) \approx \frac{1}{m} \sum_{k=1}^{m} \frac{1}{|A|} \sum_{a=1}^{n_x} \sum_{b=1}^{n_y} \psi_j(\boldsymbol{u}_{ab}, v_k) I(\boldsymbol{u}_{ab} \in A) \Delta_x \Delta_y.$$

Here, the bounding box `st_bbox(A)` is divided evenly into a grid of $n_x \times n_y$ rectangles, each of size $\Delta_x \times \Delta_y$. Each $\boldsymbol{u}_{ab} = (u_a, u_b)$ is a point from the (a, b)th rectangle, for $a = 1, \ldots, n_x$ and $b = 1, \ldots, n_y$.

Due to the treatment of A_i and $\boldsymbol{c}_j$ as objects in a Euclidean space, this basis is more suitable for coordinates from a map projection than coordinates based on a globe representation.

The `control` argument is a list which may provide any of the following:

- `method` specifies computation method: `mc` or `rect`. Default is `mc`.
- `mc_reps` is number of repetitions to use for `mc`. Default is 1000.
- `nx` is number of x-axis points to use for `rect` method. Default is 50.
- `ny` is number of y-axis points to use for `rect` method. Default is 50.
- `report_period` is an integer; print a message with progress each time this many areas are processed. Default is `Inf` so that message is suppressed.
- `verbose` is a logical; if `TRUE` print descriptive messages about the computation. Default is `FALSE`.
- `mc_sampling_factor` is a positive number; an oversampling factor used to compute `blocksize` in the rdomain function. I.e., `blocksize = ceiling(mc_sampling_factor * mc_reps)`. Default is 1.2.

Value

A sparse $n \times r$ matrix whose ith row is $\boldsymbol{s}_i^\top = \Big(\psi_1^{(m)}(A_i), \ldots, \psi_r^{(m)}(A_i)\Big)$.

See Also

Other bisquare: `areal_spatial_bisquare()`, `spacetime_bisquare()`, `spatial_bisquare()`

Examples

```
set.seed(1234)

# Create knot points
seq_x = seq(0, 1, length.out = 3)
seq_y = seq(0, 1, length.out = 3)
seq_t = seq(0, 1, length.out = 3)
knots = expand.grid(x = seq_x, y = seq_y, t = seq_t)
knots_sf = st_as_sf(knots, coords = c("x","y","t"), crs = NA, dim = "XYM", agr = "constant")

# Create a simple domain (of rectangles) to evaluate
shape1 = matrix(c(0.0,0.0, 0.5,0.0, 0.5,0.5, 0.0,0.5, 0.0,0.0), ncol=2, byrow=TRUE)
shape2 = shape1 + cbind(rep(0.5,5), rep(0.0,5))
shape3 = shape1 + cbind(rep(0.0,5), rep(0.5,5))
```

```
shape4 = shape1 + cbind(rep(0.5,5), rep(0.5,5))
sfc = st_sfc(
   st_polygon(list(shape1)),
   st_polygon(list(shape2)),
   st_polygon(list(shape3)),
   st_polygon(list(shape4))
)
dom = st_sf(data.frame(geoid = 1:length(sfc), geom = sfc))

rad = 0.5
period = c(0.4, 0.7)
areal_spacetime_bisquare(dom, period, knots, w = rad, w_t = 1)
areal_spacetime_bisquare(dom, period, knots_sf, w_s = rad, w_t = 1)

# Plot the (spatial) knots and the (spatial) domain at which we evaluated
# the basis.
plot(knots[,1], knots[,2], pch = 4, cex = 1.5, col = "red")
plot(dom[,1], col = NA, add = TRUE)

# Draw a circle representing the basis' radius around one of the knot points
tseq = seq(0, 2*pi, length=100)
coords = cbind(rad * cos(tseq) + seq_x[2], rad * sin(tseq) + seq_y[2])
lines(coords, col = "red")
```

areal_spatial_bisquare

Areal Spatial Bisquare Basis

Description

Spatial bisquare basis on areal data.

Usage

```
areal_spatial_bisquare(dom, knots, w, control = NULL)
```

Arguments

dom	An sf or sfc object with areas $A_1, \ldots, A_n$ to evaluate.
knots	Knots $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_r$ for the basis. See "Details".
w	Radius for the basis.
control	A list of control arguments. See "Details".

Details

Notes about arguments:

- `knots` may be provided as either an `sf` or `sfc` object, or as a matrix of points.
- If an `sf` or `sfc` object is provided for `knots`, r two-dimensional `POINT` entries are expected in `st_geometry(knots)`. Otherwise, `knots` will be interpreted as an $r \times 2$ numeric matrix.
- If `knots` is an `sf` or `sfc` object, it is checked to ensure the coordinate system matches `dom`.

For each area A in the given domain, compute an the basis functions

$$\bar{\varphi}_j(A) = \frac{1}{|A|} \int_A \varphi_j(\boldsymbol{u}) d\boldsymbol{u}$$

for $j = 1, \ldots, r$. Here, $\varphi_j(\boldsymbol{u})$ represent spatial_bisquare basis functions defined at the point level using $\boldsymbol{c}_j$ and w.

The basis requires an integration which may be computed using one of two methods. The `mc` method uses

$$\bar{\varphi}_j(A) \approx \frac{1}{Q} \sum_{q=1}^{Q} \varphi_j(\boldsymbol{u}_q),$$

based on a random sample of locations $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_Q$ from a uniform distribution on area A. The `rect` method uses a simple quadrature approximation

$$\bar{\varphi}_j(A) \approx \frac{1}{|A|} \sum_{a=1}^{n_x} \sum_{b=1}^{n_y} \varphi_j(\boldsymbol{u}_{ab}) I(\boldsymbol{u}_{ab} \in A) \Delta_x \Delta_y.$$

Here, the bounding box `st_bbox(A)` is divided evenly into a grid of $n_x \times n_y$ rectangles, each of size $\Delta_x \times \Delta_y$. Each $\boldsymbol{u}_{ab} = (u_a, u_b)$ is a point from the (a, b)th rectangle, for $a = 1, \ldots, n_x$ and $b = 1, \ldots, n_y$.

Due to the treatment of A_i and $\boldsymbol{c}_j$ as objects in a Euclidean space, this basis is more suitable for coordinates from a map projection than coordinates based on a globe representation.

The `control` argument is a list which may provide any of the following:

- `method` specifies computation method: `mc` or `rect`. Default is `mc`.
- `mc_reps` is number of repetitions to use for `mc`. Default is 1000.
- `nx` is number of x-axis points to use for `rect` method. Default is 50.
- `ny` is number of y-axis oints to use for `rect` method. Default is 50.
- `report_period` is an integer; print a message with progress each time this many areas are processed. Default is `Inf` so that message is suppressed.
- `verbose` is a logical; if `TRUE` print descriptive messages about the computation. Default is `FALSE`.
- `mc_sampling_factor` is a positive number; an oversampling factor used to compute `blocksize` in the rdomain function. I.e., `blocksize = ceiling(mc_sampling_factor * mc_reps)`. Default is 1.2.

Value

A sparse $n \times r$ matrix whose ith row is $\boldsymbol{s}_i^\top = \Big(\bar{\varphi}_1(A_i), \ldots, \bar{\varphi}_r(A_i)\Big)$.

See Also

Other bisquare: areal_spacetime_bisquare(), spacetime_bisquare(), spatial_bisquare()

Examples

```
set.seed(1234)

# Create knot points
seq_x = seq(0, 1, length.out = 3)
seq_y = seq(0, 1, length.out = 3)
knots = expand.grid(x = seq_x, y = seq_y)
knots_sf = st_as_sf(knots, coords = c("x","y"), crs = NA, agr = "constant")

# Create a simple domain (of rectangles) to evaluate
shape1 = matrix(c(0.0,0.0, 0.5,0.0, 0.5,0.5, 0.0,0.5, 0.0,0.0), ncol=2, byrow=TRUE)
shape2 = shape1 + cbind(rep(0.5,5), rep(0.0,5))
shape3 = shape1 + cbind(rep(0.0,5), rep(0.5,5))
shape4 = shape1 + cbind(rep(0.5,5), rep(0.5,5))
sfc = st_sfc(
   st_polygon(list(shape1)),
   st_polygon(list(shape2)),
   st_polygon(list(shape3)),
   st_polygon(list(shape4))
)
dom = st_sf(data.frame(geoid = 1:length(sfc), geom = sfc))

rad = 0.5
areal_spatial_bisquare(dom, knots, rad)
areal_spatial_bisquare(dom, knots_sf, rad)

# Plot the knots and the points at which we evaluated the basis
plot(knots[,1], knots[,2], pch = 4, cex = 1.5, col = "red")
plot(dom[,1], col = NA, add = TRUE)

# Draw a circle representing the basis' radius around one of the knot points
tseq = seq(0, 2*pi, length=100)
coords = cbind(rad * cos(tseq) + seq_x[2], rad * sin(tseq) + seq_y[2])
lines(coords, col = "red")
```

---

autocov_VAR1 *Compute the autocovariance matrix for a VAR(1) process.*

Description

Compute the autocovariance matrix for a VAR(1) process.

Usage

```
autocov_VAR1(A, Sigma, lag_max)
```

Arguments

`A`	Coefficient matrix A of the autoregression term.
`Sigma`	Covariance matrix $\boldsymbol{\Sigma}$ of the errors.
`lag_max`	maximum number of lags to compute.

Details

Computes the autocovariance matrix $\boldsymbol{\Gamma}(h)$ of the m-dimensional VAR(1) process

$$\boldsymbol{Y}_t = \boldsymbol{A}\boldsymbol{Y}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$$

For the required computation of $\boldsymbol{\Gamma}(0)$, this function solves the $m^2 \times m^2$ system

$$\mathrm{vec}(\boldsymbol{\Gamma}(0)) = [\boldsymbol{I} - \boldsymbol{A} \otimes \boldsymbol{A}]^{-1}\mathrm{vec}(\boldsymbol{\Sigma}).$$

without directly computing $m^2 \times m^2$ matrices.

Value

An array `Gamma` of dimension `c(m,m,lag_max + 1)`, where the slice `Gamma[,,h]` represents the autocovariance at lag `h = 0,1,...,lag_max`.

Examples

```
U = matrix(NA, 3, 3)
U[,1] = c(1, 1, 1) / sqrt(3)
U[,2] = c(1, 0, -1) / sqrt(2)
U[,3] = c(0, 1, -1) / sqrt(2)
B = U %*% diag(c(0.5, 0.2, 0.1)) %*% t(U)
A = (B + t(B)) / 2
Sigma = diag(x = 2, nrow = 3)
autocov_VAR1(A, Sigma, lag_max = 5)
```

`car_precision` *CAR Precision Matrix*

Description

A convenience function to compute the CAR precision matrix based on a given adjacency matrix.

Usage

```
car_precision(A, tau = 1, scale = FALSE)
```

Arguments

`A`	An adjacency matrix.
`tau`	The CAR dependency parameter $\tau \in [0, 1]$. See "Value". Default: `1`.
`scale`	Whether to scale matrix entries. See "Value". Default: `FALSE`.

Details

Suppose $\boldsymbol{A}$ is an $n \times n$ adjacency matrix and $\boldsymbol{D} = \text{Diag}(\boldsymbol{A}\boldsymbol{1}) = \text{Diag}(a_{1+}, \ldots, a_{n+})$. If `scale` is `FALSE`, return the CAR precision matrix

$$\boldsymbol{Q} = \boldsymbol{D} - \tau\boldsymbol{A}.$$

If `scale` is `TRUE`, return a scaled version

$$\tilde{\boldsymbol{Q}} = \boldsymbol{D}^{-1}\boldsymbol{Q}.$$

An error is thrown if `scale = TRUE` and any of $\{a_{1+}, \ldots, a_{n+}\}$ are equal to 0. Taking $\tau = 1$ corresponds to the Intrinsic CAR precision matrix.

Typically in a modeling context, the precision matrix will be multiplied by a scaling parameter; e.g., a CAR model for random effects $\boldsymbol{\phi}$ could be

$$f(\boldsymbol{\phi} \mid \alpha) \propto \alpha^{-q} \exp\left\{-\frac{1}{2\alpha^2}\boldsymbol{\phi}^\top \boldsymbol{Q}\boldsymbol{\phi}\right\}.$$

where $q = \text{rank}(Q)$.

Value

CAR precision matrix.

Examples

```
data("acs_sf")
dom = acs5_2013[1:4,]
A = adjacency_matrix(dom)
Q = car_precision(A)
```

`columbia_neighbs` *City of Columbia neighborhoods.*

Description

An `sf` object containing the geometry of four neighborhoods in the City of Columbia, Boone County, Missouri. Based on shapefiles provided by the Office of Information Technology / GIS, City of Columbia, Missouri.

Usage

```
columbia_neighbs
```

Format

An `sf` object with 4 features (neighborhoods).

`Covariance Approximation` *Best Approximation to Covariance Structure*

Description

Compute the best positive approximant for use in the STCOS model, under several prespecified covariance structures.

Usage

```
cov_approx_randwalk(Delta, S)

cov_approx_blockdiag(Delta, S)
```

Arguments

`Delta`	Covariance ($n \times n$) for observations within a time point for the process whose variance we wish to approximate.
`S`	Design matrix ($N \times r$) of basis functions evaluated on the fine-level process over $T = N/n$ time points.

Details

Let $\boldsymbol{\Sigma}$ be an $N \times N$ symmetric and positive-definite covariance matrix and $\boldsymbol{S}$ be an $N \times r$ matrix with rank r. The objective is to compute a matrix $\boldsymbol{K}$ which minimizes the Frobenius norm

$$\|\boldsymbol{\Sigma} - \boldsymbol{S}\boldsymbol{C}\boldsymbol{S}^\top\|_{\mathrm{F}},$$

over symmetric positive-definite matrices $\boldsymbol{C}$. The solution is given by

$$\boldsymbol{K} = (\boldsymbol{S}^\top \boldsymbol{S})^{-1} \boldsymbol{S}^\top \boldsymbol{\Sigma} \boldsymbol{S} (\boldsymbol{S}^\top \boldsymbol{S})^{-1}.$$

In the STCOS model, $\boldsymbol{S}$ represents the design matrix from a basis function computed from a fine-level support having n areas, using T time steps. Therefore $N = nT$ represents the dimension of covariance for the fine-level support.

We provide functions to handle some possible structures for target covariance matrices of the form

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Gamma}(1,1) & \cdots & \boldsymbol{\Gamma}(1,T) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Gamma}(T,1) & \cdots & \boldsymbol{\Gamma}(T,T) \end{pmatrix},$$

where each $\boldsymbol{\Gamma}(s,t)$ is an $n \times n$ matrix.

- `cov_approx_randwalk` assumes $\boldsymbol{\Sigma}$ is based on the autocovariance function of a random walk

  $$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\Delta}).$$

  so that

  $$\boldsymbol{\Gamma}(s,t) = \min(s,t)\boldsymbol{\Delta}.$$

- `cov_approx_blockdiag` assumes $\boldsymbol{\Sigma}$ is based on

  $$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\Delta}).$$

  which are independent across t, so that

  $$\boldsymbol{\Gamma}(s,t) = I(s=t)\boldsymbol{\Delta},$$

The block structure is used to reduce the computational burden, as N may be large.

`DIC` *Deviance Information Criterion*

Description

Generic function to calculate Deviance Information Criterion (DIC) for a given model object.

Usage

```
DIC(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Additional arguments.

Value

A numeric value of the DIC.

gibbs_stcos *Gibbs Sampler for STCOS Model*

Description

Gibbs Sampler for STCOS Model

Usage

```
gibbs_stcos(
  z,
  v,
  H,
  S,
  Kinv,
  R,
  report_period = R + 1,
  burn = 0,
  thin = 1,
  init = NULL,
  fixed = NULL,
  hyper = NULL
)

## S3 method for class 'stcos_gibbs'
logLik(object, ...)

## S3 method for class 'stcos_gibbs'
DIC(object, ...)

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

## S3 method for class 'stcos_gibbs'
fitted(object, H, S, ...)

## S3 method for class 'stcos_gibbs'
predict(object, H, S, ...)
```

Arguments

`z`	Vector which represents the outcome; assumed to be direct estimates from the survey.
`v`	Vector which represents direct variance estimates from the survey.
`H`	Matrix of overlaps between source and fine-level supports.
`S`	Design matrix for basis decomposition.

`Kinv`	The precision matrix $\boldsymbol{K}^{-1}$ of the random coefficient $\boldsymbol{\eta}$
`R`	Number of MCMC reps.
`report_period`	Gibbs sampler will report progress each time this many iterations are completed.
`burn`	Number of the `R` draws to discard at the beginning of the chain.
`thin`	After burn-in period, save one out of every `thin` draws.
`init`	A list containing the following initial values for the MCMC: `sig2mu`, `sig2xi`, `sig2K`, `muB`, `eta`, `xi`. Any values which are not specified are set to arbitrary choices.
`fixed`	A list specifying which parameters to keep fixed in the MCMC. This can normally be left blank. If elements `sig2mu`, `sig2xi`, or `sig2K` are specified they should be boolean, where TRUE means fixed (i.e. not drawn). If elements `muB`, `eta`, or `xi` are specified, they should each be a vector of indicies; the specified indices are to be treated as fixed (i.e. not drawn).
`hyper`	A list containing the following hyperparameter values: `a_sig2mu`, `a_sig2K`, `a_sig2xi`, `b_sig2mu`, `b_sig2K`, `b_sig2xi`. Any hyperparameters which are not specified are set to a default value of 2.
`object`	A result from `gibbs_stcos`.
`...`	Additional arguments.
`x`	A result from `gibbs_stcos`.

Details

Fits the model

$$\boldsymbol{Z} = \boldsymbol{H}\boldsymbol{\mu}_B + \boldsymbol{S}\boldsymbol{\eta} + \boldsymbol{\xi} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathrm{N}(0, \boldsymbol{V}),$$

$$\boldsymbol{\eta} \sim \mathrm{N}(\boldsymbol{0}, \sigma_K^2 \boldsymbol{K}), \quad \boldsymbol{\xi} \sim \mathrm{N}(0, \sigma_\xi^2 \boldsymbol{I}),$$

$$\boldsymbol{\mu}_B \sim \mathrm{N}(\boldsymbol{0}, \sigma_\mu^2 \boldsymbol{I}), \quad \sigma_\mu^2 \sim \mathrm{IG}(a_\mu, b_\mu),$$

$$\sigma_K^2 \sim \mathrm{IG}(a_K, b_K), \quad \sigma_\xi^2 \sim \mathrm{IG}(a_\xi, b_\xi),$$

using a Gibbs sampler with closed-form draws.

Helper functions produce the following outputs:

- `logLik` computes the log-likelihood for each saved draw.
- `DIC` computes the Deviance information criterion for each saved draw.
- `print` displays a summary of the draws.
- `fitted` computes the mean $E(Y_i)$ for each observation $i = 1, \ldots, n$, for each saved draw.
- `predict` draws Y_i for each observation $i = 1, \ldots, n$, using the parameter values for each saved Gibbs sampler draw.

Value

`gibbs_stcos` returns an `stcos` object which contains draws from the sampler. Helper functions take this object as an input and produce various outputs (see details).

Examples

```
## Not run:
demo = prepare_stcos_demo()
out = gibbs_stcos(demo$z, demo$v, demo$H, demo$S, solve(demo$K),
    R = 100, burn = 0, thin = 1)
print(out)
logLik(out)
DIC(out)
fitted(out, demo$H, demo$S)
predict(out, demo$H, demo$S)

## End(Not run)
```

`licols` *licols*

Description

Extract a linearly independent set of columns of a matrix.

Usage

```
licols(X, tol = 1e-10, quiet = FALSE)
```

Arguments

`X`	A matrix.
`tol`	A tolerance for rank estimation. Default is 1e-10.
`quiet`	logical; if FALSE, print a warning about computation time if `X` is large.

Details

An R version of a Matlab `licols` function given in this MathWorks forum post.

Value

`Xsub` contains the extracted columns of `X` and `idx` contains the indices (into X) of those columns. The elapsed time is stored in `elapsed.sec`.

Examples

```
x = 0:19 %% 3 + 1
Z = model.matrix(~ as.factor(x) - 1)
X = cbind(1, Z)
licols(X)
```

mle_stcos *MLE for STCOS Model*

Description

MLE for STCOS Model

Usage

```
mle_stcos(
  z,
  v,
  H,
  S,
  K,
  init = NULL,
  optim_control = list(),
  optim_method = "L-BFGS-B"
)
```

Arguments

z	Vector which represents the outcome; assumed to be direct estimates from the survey.
v	Vector which represents direct variance estimates from the survey. The diagonal of the matrix $\boldsymbol{V}$ described in the details.
H	Matrix of overlaps between source and fine-level supports.
S	Design matrix for basis decomposition.
K	Variance of the random coefficient $\boldsymbol{\eta}$
init	A list containing the initial values in the MCMC for `sig2xi` and `sig2K`. If not specified, we select an arbitrary initial value.
optim_control	This is passed as the `control` argument to `optim`. Note that the value `fnscale` is ignored if specified.
optim_method	Method to be used for likelihood maximization by `optim`. Default is `L-BFGS-B`.

Details

Maximize the likelihood of the STCOS model

$$f(\boldsymbol{z} \mid \boldsymbol{\mu}_B, \sigma_K^2, \sigma_\xi^2) = \mathrm{N}(\boldsymbol{z} \mid \boldsymbol{H}\boldsymbol{\mu}_B, \boldsymbol{\Delta}), \quad \boldsymbol{\Delta} = \sigma_\xi^2 \boldsymbol{I} + \boldsymbol{V} + \sigma_K^2 \boldsymbol{S}\boldsymbol{K}\boldsymbol{S}^\top,$$

by numerical maximization of the profile likelihood

$$\ell(\sigma_K^2, \sigma_\xi^2) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Delta}| - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{\mu}}_B)^\top \boldsymbol{\Delta}^{-1}(\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{\mu}}_B)$$

using $\hat{\boldsymbol{\mu}}_B = (\boldsymbol{H}^\top \boldsymbol{\Delta}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^\top \boldsymbol{\Delta}^{-1}\boldsymbol{z}$.

Value

A list containing maximum likelihood estimates.

Examples

```
## Not run:
demo = prepare_stcos_demo()
mle_out = mle_stcos(demo$z, demo$v, demo$S, demo$H, demo$K)
sig2K_hat = mle_out$sig2K_hat
sig2xi_hat = mle_out$sig2xi_hat
mu_hat = mle_out$mu_hat

## End(Not run)
```

`overlap_matrix` *Matrix of overlaps between two sets of areas.*

Description

A convenience function to convert output from `sf::st_intersection` to a sparse matrix as defined in the `Matrix` package.

Usage

```
overlap_matrix(dom1, dom2, proportion = TRUE)
```

Arguments

`dom1`	An `sf` object representing a domain of areal units.
`dom2`	An `sf` object representing a domain of areal units.
`proportion`	Logical; if `TRUE`, normalize so that rows sum to 1. Otherwise areas are returned.

Details

Returns a matrix `H` whose (i,j)th entry represent the area of the overlap between areal units `dom1[i,]` and `dom2[j,]`.

Value

An matrix of overlaps.

Examples

```
data("acs_sf")
dom1 = acs5_2013[1:10,]
dom2 = acs5_2016[1:10,]
H1 = overlap_matrix(dom1, dom2)
H2 = overlap_matrix(dom1, dom2, proportion = FALSE)
```

`prepare_stcos_demo` *Prepare Demo Data for STCOS Model*

Description

Create demo data based on ACS example, making a few simple model choices. The purpose of this function is to facilitate examples in other functions. Uses functions in the package to create model terms from shapefiles.

Usage

```
prepare_stcos_demo(num_knots_sp = 200, basis_mc_reps = 200, eigval_prop = 0.65)
```

Arguments

`num_knots_sp`	Number of spatial knots to use in areal space-time basis.
`basis_mc_reps`	Number of monte carlo reps to use in areal space-time basis.
`eigval_prop`	Proportion of variability to keep in dimension reduction of basis expansions.

Value

A list containing the following:

- `z` direct estimates.
- `v` direct variance estimates.
- `H` overlap matrix.
- `S` design matrix of basis expansion.
- `K` covariance matrix of the random effect.

Examples

```
## Not run:
out = prepare_stcos_demo()

## End(Not run)
```

`rdomain` *Draw uniformly distributed points from a set of areas*

Description

An alternative to `sf::st_sample` which draws uniformly distributed points using a simple accept-reject method.

Usage

```
rdomain(n, dom, blocksize = n, itmax = Inf)
```

Arguments

`n`	Number of points desired in the final sample.
`dom`	An `sf` object representing a domain of areal units.
`blocksize`	Number of candidate points to draw on each pass of accept-reject sampling (see details). Defaults to `n`.
`itmax`	Maximum number of accept-reject samples to attempt. Defaults to `Inf`.

Details

Draws a sample of `blocksize` points uniformly from a bounding box on `dom`, and accepts only the points which belong to `dom`. This yields a uniform sample on `dom`. The process is repeated until `n` accepted draws are obtained, or until it has been attempted `itmax` times. If `itmax` iterations are reached without accepting `n` draws, an error is thrown.

This seems to be an order of magnitude faster than the current implementation of `st_sample`, although the latter can accomplish the same objective and is more general. The improved performance is worthwhile when used in the areal basis functions, which sample repeatedly from the domain.

Performance will degrade when areal units have small area relative to their bounding box, as many candidate points may need to be discarded. For example, this will occur if `dom` contains a set of small scattered islands in an ocean. In this case, it would be more efficient to sample from each island at a time.

Value

An `sf` object with 2-dimensional points.

Examples

```
dom = acs5_2013[c(1,5,8,12),]
pts = rdomain(10000, dom)
```

`spacetime_bisquare` *Space-Time Bisquare Basis*

Description

Space-time bisquare basis on point data.

Usage

```
spacetime_bisquare(dom, knots, w_s, w_t)
```

Arguments

`dom`	Space-time points $(\boldsymbol{u}_1, v_1), \ldots, (\boldsymbol{u}_n, v_n)$ to evaluate. See "Details".
`knots`	Spatio-temporal knots $(\boldsymbol{c}_1, g_1), \ldots, (\boldsymbol{c}_r, g_r)$ for the basis. See "Details".
`w_s`	Spatial radius for the basis.
`w_t`	Temporal radius for the basis.

Details

Notes about arguments:

- Both `dom` and `knots` may be provided as either `sf` or `sfc` objects, or as matrices of points.
- If an `sf` or `sfc` object is provided for `dom`, n three-dimensional `POINT` entries are expected in `st_geometry(dom)`. Otherwise, `dom` will be interpreted as an $n \times 3$ numeric matrix.
- If an `sf` or `sfc` object is provided for `knots`, r three-dimensional `POINT` entries are expected in `st_geometry(knots)`. Otherwise, `knots` will be interpreted as an $r \times 3$ numeric matrix.
- If both `dom` and `knots_s` are given as `sf` or `sfc` objects, they will be checked to ensure a common coordinate system.

For each $(\boldsymbol{u}_i, v_i)$, compute the basis functions

$$\psi_j(\boldsymbol{u}, v) = \left[2 - \frac{\|\boldsymbol{u} - \boldsymbol{c}_j\|^2}{w_s^2} - \frac{|v - g_j|^2}{w_t^2}\right]^2 \cdot I(\|\boldsymbol{u} - \boldsymbol{c}_j\| \leq w_s) \cdot I(|v - g_j| \leq w_t)$$

for $j = 1, \ldots, r$.

Due to the treatment of $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$ as points in a Euclidean space, this basis is more suitable for coordinates from a map projection than coordinates based on a globe representation.

Value

A sparse $n \times r$ matrix whose ith row is

$$\boldsymbol{s}_i^\top = \Big(\psi_1(\boldsymbol{u}_i, v_i), \ldots, \psi_r(\boldsymbol{u}_i, v_i)\Big).$$

See Also

Other bisquare: areal_spacetime_bisquare(), areal_spatial_bisquare(), spatial_bisquare()

Examples

```
set.seed(1234)

# Create knot points
seq_x = seq(0, 1, length.out = 3)
seq_y = seq(0, 1, length.out = 3)
seq_t = seq(0, 1, length.out = 3)
knots = expand.grid(x = seq_x, y = seq_y, t = seq_t)
knots_sf = st_as_sf(knots, coords = c("x","y","t"), crs = NA, dim = "XYM", agr = "constant")

# Points to evaluate
x = runif(50)
y = runif(50)
t = sample(1:3, size = 50, replace = TRUE)
pts = data.frame(x = x, y = y, t = t)
dom = st_as_sf(pts, coords = c("x","y","t"), crs = NA, dim = "XYM", agr = "constant")

rad = 0.5
spacetime_bisquare(cbind(x,y,t), knots, w_s = rad, w_t = 1)
spacetime_bisquare(dom, knots_sf, w_s = rad, w_t = 1)

# Plot the (spatial) knots and the points at which we evaluated the basis
plot(knots[,1], knots[,2], pch = 4, cex = 1.5, col = "red")
text(x, y, labels = t, cex = 0.75)

# Draw a circle representing the basis' radius around one of the knot points
tseq = seq(0, 2*pi, length=100)
coords = cbind(rad * cos(tseq) + seq_x[2], rad * sin(tseq) + seq_y[2])
lines(coords, col = "red")
```

---

spatial_bisquare *Spatial Bisquare Basis*

Description

Spatial bisquare basis on point data.

Usage

```
spatial_bisquare(dom, knots, w)
```

Arguments

`dom`	Points $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ to evaluate. See "Details".
`knots`	Knots $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_r$ for the basis. See "Details".
`w`	Radius for the basis.

Details

Notes about arguments:

- Both `dom` and `knots` may be provided as either `sf` or `sfc` objects, or as matrices of points.
- If an `sf` or `sfc` object is provided for `dom`, n two-dimensional `POINT` entries are expected in `st_geometry(dom)`. Otherwise, `dom` will be interpreted as an $n \times 2$ numeric matrix.
- If an `sf` or `sfc` object is provided for `knots`, r two-dimensional `POINT` entries are expected in `st_geometry(knots)`. Otherwise, `knots` will be interpreted as an $r \times 2$ numeric matrix.
- If both `dom` and `knots` are given as `sf` or `sfc` objects, they will be checked to ensure a common coordinate system.

For each $\boldsymbol{u}_i$, compute the basis functions

$$\varphi_j(\boldsymbol{u}) = \left[1 - \frac{\|\boldsymbol{u} - \boldsymbol{c}_j\|^2}{w^2}\right]^2 \cdot I(\|\boldsymbol{u} - \boldsymbol{c}_j\| \leq w)$$

for $j = 1, \ldots, r$.

Due to the treatment of $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$ as points in a Euclidean space, this basis is more suitable for coordinates from a map projection than coordinates based on a globe representation.

Value

A sparse $n \times r$ matrix whose ith row is $\boldsymbol{s}_i^\top = \Big(\varphi_1(\boldsymbol{u}_i), \ldots, \varphi_r(\boldsymbol{u}_i)\Big)$.

See Also

Other bisquare: `areal_spacetime_bisquare()`, `areal_spatial_bisquare()`, `spacetime_bisquare()`

Examples

```
set.seed(1234)

# Create knot points
seq_x = seq(0, 1, length.out = 3)
seq_y = seq(0, 1, length.out = 3)
knots = expand.grid(x = seq_x, y = seq_y)
knots_sf = st_as_sf(knots, coords = c("x","y"), crs = NA, agr = "constant")

# Points to evaluate
x = runif(50)
y = runif(50)
pts = data.frame(x = x, y = y)
dom = st_as_sf(pts, coords = c("x","y"), crs = NA, agr = "constant")
```

```
rad = 0.5
spatial_bisquare(cbind(x,y), knots, rad)
spatial_bisquare(dom, knots, rad)

# Plot the knots and the points at which we evaluated the basis
plot(knots[,1], knots[,2], pch = 4, cex = 1.5, col = "red")
points(x, y, cex = 0.5)

# Draw a circle representing the basis' radius around one of the knot points
tseq = seq(0, 2*pi, length=100)
coords = cbind(rad * cos(tseq) + seq_x[2], rad * sin(tseq) + seq_y[2])
lines(coords, col = "red")
```

---

stcos — *stcos: Space-Time Change of Support*

Description

An R Package for Space-Time Change of Support (STCOS) modeling.

Details

Supports building and running STCOS and related models. A guide on package use is given in <arXiv:1904.12092>.

References

Jonathan R. Bradley, Christopher K. Wikle, and Scott H. Holan (2015). Spatio-temporal change of support with application to American Community Survey multi-year period estimates. STAT 4 pp.255-270. https://doi.org/10.1002/sta4.94.

Andrew M. Raim, Scott H. Holan, Jonathan R. Bradley, and Christopher K. Wikle (2017). A model selection study for spatio-temporal change of support. In JSM Proceedings, Government Statistics Section. Alexandria, VA: American Statistical Association, pp.1524-1540.

Andrew M. Raim, Scott H. Holan, Jonathan R. Bradley, and Christopher K. Wikle (2020+). Spatio-Temporal Change of Support Modeling with R. https://arxiv.org/abs/1904.12092.

Index