Package ‘stcos’

December 6, 2019

**Type** Package

**Title** Space-Time Change of Support

**Version** 0.2.1

**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)

**Imports** R6

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

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-12-06 06:10:02 UTC

**R topics documented:**

acs_sf .......................................................... 2
adjacency_matrix ........................................... 3
ArealSpaceTimeBisquareBasis .................................................. 4
ArealSpatialBisquareBasis ...................................................... 5
autocov_VAR1 ................................................................. 7
columbia_neighbs ............................................................. 8
Covariance Approximation ...................................................... 8
DIC ................................................................................. 9
gibbs_stcos ................................................................. 10
licols ................................................................. 12
mle_stcos ................................................................. 13
overlap_matrix ............................................................. 14
prepare_stcos_demo ......................................................... 15
rdomain ................................................................. 16
SpaceTimeBisquareBasis ....................................................... 17
SpatialBisquareBasis .......................................................... 18
stcos ................................................................. 19

Index 21

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.
adjacency_matrix

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::stTouches 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 $dom[i,j]$ and $dom[j,i]$ are adjacent; 0 otherwise.

Value

An adjacency matrix

Examples

data("acs_sf")

dom = acs5_2013[1:4,]

A = adjacency_matrix(dom)
ArealSpaceTimeBisquareBasis

Areal SpaceTime Bisquare Basis

Description

An R6Class representing the space-time bisquare basis.

Usage

bs = ArealSpaceTimeBisquareBasis$new(knots_x, knots_y, knots_t, w_s, w_t, mc_reps)
bs$compute(dom, period)
bs$get_dim()
bs$get_cutpoints()
bs$get_ws()
bs$get_wt()

Arguments

- knots_x numeric vector; x-coordinates of knot points.
- knots_y numeric vector; y-coordinates of knot points.
- knots_t numeric vector; time coordinate of knot points.
- w_s numeric; spatial radius for the basis.
- w_t numeric; temporal radius for the basis.
- dom an sf object; areal units to evaluate.
- period numeric vector; time coordinates for points to evaluate.

Methods

- new Create a new SpaceTimeBisquareBasis object.
- get_dim Get the number of cutpoints used to construct this basis.
- get_cutpoints Get the cutpoints used to construct this basis.
- get_ws Get the spatial radius used to construct this basis.
- get_wt Get the temporal radius used to construct this basis.
- compute Evaluate this basis on specific areal units and periods.

Examples

set.seed(1234)
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(seq_x, seq_y, seq_t)
# Create a simple domain from rectangles
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))
period = c(0.4, 0.7)

bs = ArealSpaceTimeBisquareBasis$new(knots[,1], knots[,2], knots[,3],
    w_s = 0.5, w_t = 1, mc_reps = 200)
bs$compute(dom, period)
bs$dim()
bs$cutpoints()
bs$ws()
bs$wt()

# 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)
rad = bs$ws()
coords = cbind(rad * cos(tseq) + seq_x[2], rad * sin(tseq) + seq_y[2])
lines(coords, col = "red")

---

**ArealSpatialBisquareBasis**

**Areal Spatial Bisquare Basis**

**Description**

An R6Class representing the spatial bisquare basis.

**Usage**

bs = ArealSpatialBisquareBasis$new(knots_x, knots_y, w, mc_reps)
bs$compute(dom)
bs$dim()
bs$cutpoints()
bs$ws()
Arguments

- knots_x numeric vector; x-coordinates of knot points.
- knots_y numeric vector; y-coordinates of knot points.
- w numeric; radius for the basis.
- dom an sf object; areal units to evaluate.

Methods

- new Create a new ArealSpatialBisquareBasis object.
- get_dim Get the number of cutpoints used to construct this basis.
- get_cutpoints Get the cutpoints used to construct this basis.
- get_w Get the radius used to construct this basis.
- compute Evaluate this basis on specific areal units.

Examples

```r
set.seed(1234)
seq_x = seq(0, 1, length.out = 3)
seq_y = seq(0, 1, length.out = 3)
knots = merge(seq_x, seq_y)

# Create a simple domain from rectangles
shape1 = matrix(c(0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0, 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))

bs = ArealSpatialBisquareBasis$new(knots[,1], knots[,2], w = 0.5,
  mc_reps = 200)
bs$compute(dom)
bs$get_dim()
bs$get_cutpoints()
bs$get_w()

# 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)
rad = bs$get_w()
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 $\Sigma$ of the errors.
- **lag_max**: maximum number of lags to compute.

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

$$Y_t = AY_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma)$$

For the required computation of $\Gamma(0)$, this function avoids directly solving the $m^2 \times m^2$ system

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

Value
An array $\Gamma$ of dimension $c(m, m, \text{lag\_max} + 1)$, where the slice $\Gamma[,,h]$ represents the autocovariance at lag $h = 0, 1, \ldots, \text{lag\_max}$.

Examples

```r
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)
```
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 $\Sigma$ be an $N \times N$ symmetric and positive-definite covariance matrix, which we would like to approximate in the STCOS model. Let $S$ be the $N \times r$ design matrix of the basis function computed on $T$ time steps of the fine-level support. The number of observations at each time point is assumed to be $n$, so that the number of total observations is $N = nT$.

The objective is to compute a symmetric positive-definite matrix $K$ which minimizes

$$||\Sigma - SCS^\top||_F,$$

where $|| \cdot ||_F$ represents the Frobenius norm. The solution is given by

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

We provide functions to handle some possible structures for the target covariance, which are all in the form

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

- `cov_approx_randwalk` assumes $\Sigma$ is based on the autocovariance function of a random walk
  $$Y_{t+1} = Y_t + \epsilon_t, \quad \epsilon_t \sim N(0, \Delta),$$
  so that
  $$\Gamma(s,t) = \min(s,t)\Delta.$$

- `cov_approx_blockdiag` assumes $\Sigma$ is based on
  $$Y_{t+1} = Y_t + \epsilon_t, \quad \epsilon_t \sim N(0, \Delta),$$
  which are independent across $t$, so that
  $$\Gamma(s,t) = I(s = t)\Delta.$$

In any case $\Sigma$ may be large and we avoid computing it in its entirety.

---

**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 $K^{-1}$ of the random coefficient $\eta$

R Number of MCMC reps.

report_period Gibbs sampler will report progress each time this many iterations are completed.
**gibbs_stcos**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>burn</td>
<td>Number of the ( R ) draws to discard at the beginning of the chain.</td>
</tr>
<tr>
<td>thin</td>
<td>After burn-in period, save one out of every ( thin ) draws.</td>
</tr>
<tr>
<td>init</td>
<td>A list containing the following initial values for the MCMC: ( \text{sig2mu}, \text{sig2xi}, \text{sig2K}, \mu_B, \eta, \xi ). Any values which are not specified are set to arbitrary choices.</td>
</tr>
<tr>
<td>fixed</td>
<td>A list specifying which parameters to keep fixed in the MCMC. This can normally be left blank. If elements ( \text{sig2mu}, \text{sig2xi}, \text{sig2K} ) are specified they should be boolean, where TRUE means fixed (i.e. not drawn). If elements ( \mu_B, \eta, \xi ) are specified, they should each be a vector of indices; the specified indices are to be treated as fixed (i.e. not drawn).</td>
</tr>
<tr>
<td>hyper</td>
<td>A list containing the following hyperparameter values: ( a_{\text{sig2mu}}, a_{\text{sig2K}}, a_{\text{sig2xi}}, b_{\text{sig2mu}}, b_{\text{sig2K}}, b_{\text{sig2xi}} ). Any hyperparameters which are not specified are set to a default value of 2.</td>
</tr>
<tr>
<td>object</td>
<td>A result from ( \text{gibbs_stcos} ).</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments.</td>
</tr>
<tr>
<td>x</td>
<td>A result from ( \text{gibbs_stcos} ).</td>
</tr>
</tbody>
</table>

**Details**

Fits the model

\[
Z = H\mu_B + S\eta + \xi + \varepsilon, \quad \varepsilon \sim N(0, V),
\]

\[
\eta \sim N(0, \sigma_K^2 K), \quad \xi \sim N(0, \sigma_\xi^2 I),
\]

\[
\mu_B \sim N(0, \sigma_\mu^2 I), \quad \sigma_\mu^2 \sim \text{IG}(a_\mu, b_\mu),
\]

\[
\sigma_K^2 \sim \text{IG}(a_K, b_K), \quad \sigma_\xi^2 \sim \text{IG}(a_\xi, b_\xi),
\]

using a Gibbs sampler with closed-form draws.

Helper functions produce the following outputs:

- \( \logLik \) computes the log-likelihood for each saved draw.
- \( \text{DIC} \) computes the Deviance information criterion for each saved draw.
- \( \text{print} \) displays a summary of the draws.
- \( \text{fitted} \) computes the mean \( E(Y_i) \) for each observation \( i = 1, \ldots, n \), for each saved draw.
- \( \text{predict} \) 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 \text{stcos} object which contains draws from the sampler. Helper functions take this object as an input and produce various outputs (see details).
Examples

```r
## 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

Description

Extract a linearly independent set of columns of a matrix.

Usage

```r
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

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

**Description**

MLE for STCOS Model

**Usage**

```r
mle_stcos(z, v, H, S, K, init = NULL, optim_control = list(), optim_method = "L-BFGS-B")
mle_stcos_v2(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 $V$ 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 $\eta$
- `init`: A list containing the initial values in the MCMC for $\text{sig2xi}$ and $\text{sig2K}$. 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

$$f(z \mid \mu_B, \sigma^2_K, \sigma^2_\xi) = N(z \mid H\mu_B, \Delta), \quad \Delta = \sigma^2_\xi I + V + \sigma^2_K SKS^T,$$

by numerical maximization of the profile likelihood

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

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

**Value**

A list containing maximum likelihood estimates.
## 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**

```r
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 \( \text{dom1}[i,] \) and \( \text{dom2}[j,] \).

**Value**

An matrix of overlaps.

**Examples**

```r
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. Uses functions in the package to create model terms from shapefiles.

Usage

```r
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

```r
## 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 such as `ArealSpatialBisquareBasis` and `ArealSpaceTimeBisquareBasis`, which sample repeatedly from the domain.

Another departure from `st_sample` is that `rdomain` returns an `n` by 2 matrix of coordinates rather than an `sf` object.

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 `n` by 2 matrix of coordinates of the sampled points.
Example

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

# Convert the points to an sf object if desired
dat = data.frame(x = pts[,1], y = pts[,2])
pts_sf = st_as_sf(dat, coords = c("x", "y"), crs = st_crs(dom))
```

Description

An R6Class representing the space-time bisquare basis.

Usage

```
bs = SpaceTimeBisquareBasis$new(knots_x, knots_y, knots_t,
    w_s, w_t)
bs$compute(x, y, time)
bs$get_dim()
bs$get_cutpoints()
bs$get_ws()
bs$get_wt()
```

Arguments

- `knots_x` numeric vector; x-coordinates of knot points.
- `knots_y` numeric vector; y-coordinates of knot points.
- `knots_t` numeric vector; time coordinate of knot points.
- `w_s` numeric; spatial radius for the basis.
- `w_t` numeric; temporal radius for the basis.
- `x` numeric vector; x-coordinates for points to evaluate.
- `y` numeric vector; y-coordinates for points to evaluate.
- `time` numeric vector; time coordinates for points to evaluate.

Methods

- `new` Create a new SpaceTimeBisquareBasis object.
- `get_dim` Get the number of cutpoints used to construct this basis.
- `get_cutpoints` Get the cutpoints used to construct this basis.
- `get_ws` Get the spatial radius used to construct this basis.
- `get_wt` Get the temporal radius used to construct this basis.
- `compute` Evaluate this basis on specific points.
Examples

```r
set.seed(1234)
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(seq_x, seq_y, seq_t)
x = runif(50)
y = runif(50)
t = sample(1:3, size = 50, replace = TRUE)

bs = SpatialBisquareBasis$new(knots[,1], knots[,2], knots[,3], w_s = 0.5, w_t = 1)
bs$compute(x, y, t)
bs$dim()
bs$cutpoints()
bs$w()
bs$wt()

# 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)
rad = bs$w()
coords = cbind(rad * cos(tseq) + seq_x[2], rad * sin(tseq) + seq_y[2])
lines(coords, col = "red")
```

SpatialBisquareBasis  Spatial Bisquare Basis

Description

An R6Class representing the spatial bisquare basis.

Usage

```r
bs = SpatialBisquareBasis$new(knots_x, knots_y, w)
bs$compute(x, y, time)
bs$dim()
bs$cutpoints()
bs$w()
```

Arguments

- `knots_x` numeric vector; x-coordinates of knot points.
- `knots_y` numeric vector; y-coordinates of knot points.
- `w` numeric; radius for the basis.
- `x` numeric vector; x-coordinates for points to evaluate.
- `y` numeric vector; y-coordinates for points to evaluate.
Methods

- **new** Create a new SpatialBisquareBasis object.
- **get_dim** Get the number of cutpoints used to construct this basis.
- **get_cutpoints** Get the cutpoints used to construct this basis.
- **get_w** Get the radius used to construct this basis.
- **compute** Evaluate this basis on specific points.

Examples

```r
set.seed(1234)
seq_x = seq(0, 1, length.out = 3)
seq_y = seq(0, 1, length.out = 3)
knots = merge(seq_x, seq_y)
x = runif(50)
y = runif(50)

bs = SpatialBisquareBasis$new(knots[,1], knots[,2], w = 0.5)
bs$compute(x, y)
bs$get_dim()
bs$get_cutpoints()
bs$get_w()

# 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)
rad = bs$get_w()
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


---
References


Index

*Topic datasets
- acs_sf, 2
  - columbia_neighbs, 8
- acs5_2013(acs_sf), 2
- acs5_2014(acs_sf), 2
- acs5_2015(acs_sf), 2
- acs5_2016(acs_sf), 2
- acs5_2017(acs_sf), 2
- acs_sf, 2
- adjacency_matrix, 3
- ArealSpaceTimeBisquareBasis, 4, 16
- ArealSpatialBisquareBasis, 5, 16
- autocov_VAR1, 7
- columbia_neighbs, 8
- cov_approx_blockdiag (Covariance Approximation), 8
- cov_approx_randwalk (Covariance Approximation), 8
- Covariance Approximation, 8
- DIC, 9
- DIC.stcos_gibbs (gibbs_stcos), 10
- fitted.stcos_gibbs (gibbs_stcos), 10
- gibbs_stcos, 10
- licols, 12
- logLik.stcos_gibbs (gibbs_stcos), 10
- mle_stcos, 13
- mle_stcos_v2 (mle_stcos), 13
- overlap_matrix, 14
- predict.stcos_gibbs (gibbs_stcos), 10
- prepare_stcos Demo, 15
- print.stcos_gibbs (gibbs_stcos), 10
- rdomain, 16

SpaceTimeBisquareBasis, 17
SpatialBisquareBasis, 18
stcos, 19
stcos-package (stcos), 19