Package ‘spconf’

November 3, 2024

Title Computing Scales of Spatial Smoothing for Confounding Adjustment

Version 1.0.1

Author Kayleigh Keller [aut, cre],
Maddie Rainey [aut]

Maintainer Kayleigh Keller <kayleigh.keller@colostate.edu>

Description Computes the effective range of a smoothing matrix,
which is a measure of the distance to which smoothing occurs.
This is motivated by the application of spatial splines for adjusting
for unmeasured spatial confounding in regression models, but the
calculation of effective range can be applied to smoothing
matrices in other contexts. For algorithmic details, see Rainey and Keller
(2024) ``spconfShiny: an R Shiny application...'' <doi:10.1371/journal.pone.0311440>
and Keller and Szpiro (2020)
``Selecting a Scale for Spatial Confounding Adjustment'' <doi:10.1111/rssa.12556>.

Depends R (>= 3.5)

Imports flexclust, mgcv

Suggests splines, testthat (>= 2.1.0)

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

Repository CRAN

Date/Publication 2024-11-03 12:20:02 UTC

Contents

computeS *Compute Smoothing Matrix*

Description

Calculates the smoothing (or "hat") matrix from a design matrix.

Usage

```
computeS(x, inds = 1:nrow(x))
```

Arguments

x	Matrix of spline values, assumed to have full rank. A data frame is coerced into a matrix.
inds	Column indices of smoothing matrix to return (corresponding to rows in x).

Details

Given a matrix X of spline values, this computes S=X(X'X)^(-1)X'. When x has many rows, this can be quite large. The inds argument can be used to return a subset of columns from S.

Value

An N-by-n matrix, where n is the length of inds and N is the number of rows in x.

See Also

compute_effective_range

Examples

```
# Simple design matrix case
X <- cbind(1, rep(c(0, 1), each=4))
S <- computeS(X)
# More complex example
xloc <- runif(n=100, min=0, max=10)
X <- splines::ns(x=xloc, df=4, intercept=TRUE)
S <- computeS(X)
S2 <- computeS(X, inds=1:4)
```

computeTPRS *Create TPRS basis*

Description

Compute TPRS basis for given spatial coordinates

Usage

```
computeTPRS(coords, maxdf, rearrange = TRUE, intercept = FALSE)

arrangeTPRS(tprs, intercept = FALSE)
```

Arguments

`coords`	Data frame containing the coordinates.
`maxdf`	Largest number of splines to include in TPRS basis
`rearrange`	Logical indicator of whether to rearrange the columns of TPRS basis.
`intercept`	Logical indicator of whether or not to remove the intercept column from the basis when `rearrange` is `TRUE`.
`tprs`	Matrix of TPRS basis values (from `computeTPRS`).

Details

`computeTPRS` creates a thin-plate regression spline (TPRS) basis from a two-dimensional set of coordinate locations using the `mgcv` package.

The output from `mgcv` is structured to have the linear terms as the last columns of the matrix. Use `arrangeTPRS()` to arrange the matrix columns to be in order of increasing resolution. Specifically, it moves the last two columns to the left of the matrix and the third-from last column, which corresponds to the intercept, is optionally removed.

Value

An n-by-k matrix of spline basis functions where n is the number of rows in `coords` and k is equal to `maxdf`

Examples

```
x <- runif(100)
y <- runif(100)
mat <- computeTPRS(data.frame(x, y), maxdf=4)
```

compute_effective_range *Compute effective range*

Description

Calculates the effective range for a spline basis matrix.

Usage

```
compute_effective_range(
  X,
  coords = X[, c("x", "y")],
  df = 3,
  nsamp = min(1000, nrow(X)),
  smoothedCurve = FALSE,
  newd = seq(0, 1, 100),
  scale_factor = 1,
  returnFull = FALSE,
  cl = NULL,
  namestem = "tprs",
  inds = NULL,
  verbose = FALSE,
  span = 0.1
)

compute_effective_range_nochecks(
  X,
  inds,
  newd,
  D,
  smoothedCurve = FALSE,
  scale_factor = 1,
  returnFull = FALSE,
  cl = NULL,
  span = 0.1
)
```

Arguments

X	Matrix of spline values. See `namestem` for expected column names.
coords	Matrix of point coordinates. Defaults to the `x` and `y` columns of `X`, but can have a different number of columns for settings with different dimensions.
df	Degrees of freedom for which effective range should be computed.
nsamp	Number of observations from `X` from which to sample. Defaults to minimum of 1,000 and `nrow(X)`.

`smoothedCurve`	Should the effective range be computed using the procedure introduced by Keller and Szpiro, 2020, (`TRUE`) or the procedure introduced by Rainey and Keller, 2024, (`FALSE`). See Details.
`newd`	Distance values at which to make loess predictions. Should correspond to distances in the same units as `coords`. Only needed when `smoothedCurve` is `TRUE`.
`scale_factor`	Factor by which range should be scaled. Often physical distance corresponding to resolution of grid. Defaults to 1, so that range is reported on the same scale as distance in `coords`. Only needed when `smoothedCurve` is `TRUE`.
`returnFull`	Should the mean and median curves be returned (`TRUE`), or just the range value of where they first cross zero (`FALSE`).
`cl`	Cluster object, or number of cluster instances to create. Defaults to no parallelization.
`namestem`	Stem of names of columns of X corresponding to evaluated splines. Defaults to `""`, meaning names of the form `1`, `2`, ...
`inds`	Indices of observations to use for computation. Passed to `computeS`.
`verbose`	Control printing of a df counter to console.
`span`	Passed to `fitLoess`. If too small, then can lead to unstable loess estimates. Only needed when `smoothedCurve` is `TRUE`.
`D`	Distance matrix for coordinates.

Details

Using the given spline basis and the inputted coordinates, the effective bandwidth is computed for the given degrees of freedom. This is accomplished by computing a distance matrix from the coordinates and a smoothing matrix from the basis. Setting `smoothedCurve = TRUE` (see Keller and Szpiro, 2020, for details), for each column of smoothing weights, a LOESS curve is fit to the smoothing weights as a function of the distances, and the distance where the curve first crosses zero is obtained. Setting `smoothedCurve = FALSE` (see Rainey and Keller, 2024, for details), for each column of smoothing weights, the smallest distance that corresponds with the first negative smoothing weight is obtained. Then, for both procedures, the median of the obtained distances is reported as the effective bandwidth.

The columns of `X` are selected by name, and so are assumed to have a numeric value in the column name that indicates the spline number. For example, the columns containing the first three splines should be "1", "2", and "3". IF there is a fixed character prefix, that can be supplied via `namestem`. For example, if the columns are "s1", "s2", "s3", then set `namestem="s"`.

Value

The effective bandwidth for each value of `df`. If `returnFull = FALSE`, then this is a vector of the same length as `df`. If `returnFull = TRUE` and `smoothedCurve = TRUE`, this is a list that additionally contains values of the pointwise median and mean of the smoothed curves.

References

Keller and Szpiro (2020). Selecting a scale for spatial confounding adjustment. Journal of the Royal Statistical Society, Series A https://doi.org/10.1111/rssa.12556.

Rainey and Keller (2024). spconfShiny: An R Shiny application for calculating the spatial scale of smoothing splines for point data. PLOS ONE https://doi.org/10.1371/journal.pone.0311440

See Also

compute_lowCurve

Examples

```
M <- 16
tprs_df <- 10
si <- seq(0, 1, length=M+1)[-(M+1)]
gridcoords <- expand.grid(x=si, y=si)
tprsX <- computeTPRS(coords = gridcoords, maxdf = tprs_df+1)
compute_effective_range(X=tprsX, coords=gridcoords, df=3:10, smoothedCurve=FALSE)


xloc <- runif(n=100, min=0, max=10)
X <- splines::ns(x=xloc, df=4, intercept=TRUE)
colnames(X) <- paste0("s", 1:ncol(X))
xplot <- 0:10
compute_effective_range(X=X, coords=as.matrix(xloc), df=2:4, newd=xplot,
                        namestem="s", smoothedCurve = TRUE)
```

---

compute_lowCurve *Compute loess curves for smoothing matrix*

Description

Calculates a loess curve for the smoothing matrix entries, as a function of distance between points.

Usage

```
compute_lowCurve(S, D, newd, cl = NULL, span = 0.1)
```

Arguments

S	Smoothing matrix, or a subset of columns from a smoothing matrix.
D	Distance matrix, or a subset of columns from a distance matrix.
newd	Distances to use for loess prediction.
cl	Cluster object, or number of cluster instances to create. Defaults to no parallelization.
span	Passed to fitLoess

Details

For each column in S, a loess curve is fit to the values as a function of the distances between points, which are taken from the columns of D. Thus, the order of rows and columns in S should match the order of rows and columns in D. For a large number of locations, this procedure may be somewhat slow. The cl argument can be used to parallelize the operation using clusterMap.

Value

List with three elements: n-by-N matrix, where n is the length of newd and N is the number of columns in S; a vector of length n giving the median curve value; a vector of length n giving the mean curve value.

See Also

computeS fitLoess

Examples

```
xloc <- runif(n=100, min=0, max=10)
X <- splines::ns(x=xloc, df=4, intercept=TRUE)
S <- computeS(X)
d <- as.matrix(dist(xloc))
xplot <- 0:10
lC <- compute_lowCurve(S, D=d, newd=xplot)
matplot(xplot, lC$SCurve, type="l", col="black")
points(xplot, lC$SCurveMedian, type="l", col="red")
```

---

find_first_zero_cross *Find zero*

Description

Calculates the zero of a function by linear interpolation between the first two points either side of zero.

Usage

```
find_first_zero_cross(x)
```

Arguments

x	Function values, assumed to be ordered

Value

Index of first value of x that lies below 0. Decimal values will be returned using a simple interpolation of the two values straddling 0.

See Also

find_zeros_cross, compute_effective_range

---

find_zeros_cross *Find distance to first zero*

Description

For a set of distance and smoothing matrix values, determines the smallest distance that corresponds with negative value for each column of the smoothing matrix.

Usage

```
find_zeros_cross(D, S)
```

Arguments

D	Distance matrix, or a subset of columns from a distance matrix.
S	Smoothing matrix, or a subset of columns from a smoothing matrix.

Value

Vector of length equal to the number of columns in `D` and `S`. Each value is the smallest observed distance (from a column of `D`) that has a negative value in the corresponding column of `S`.

---

fitLoess *Fit a loess curve*

Description

Wrapper function for fitting and predicting from `loess()`.

Usage

```
fitLoess(y, x, newx = x, span = 0.5, ...)
```

Arguments

y	Dependent variable values
x	Independent variable values
newx	Values of x to use for prediction.
span	Controls the amount of smoothing. Passed to loess; see that function for details.
...	Additional arguments passed to loess

Value

A vector of the same length of newx providing the predictions from a loess smooth.

Examples

```
x <- seq(0, 5, length=50)
y <- cos(4*x) + rnorm(50, sd=0.5)
xplot <- seq(0, 5, length=200)
lfit <- fitLoess(y=y, x=x, newx=xplot, span=0.2)
plot(x, y)
points(xplot, lfit, type="l")
```

Index