# Package: spEDM (via r-universe)

March 10, 2025

**Title** Spatial Empirical Dynamic Modeling

**Version** 1.5

**Description** Inferring causal associations in cross-sectional earth system data through empirical dynamic modeling (EDM), with extensions to convergent cross mapping from Sugihara et al. (2012) <doi:10.1126/science.1227079>, partial cross mapping as outlined in Leng et al. (2020) <doi:10.1038/s41467-020-16238-0>, and cross mapping cardinality as described in Tao et al. (2023)<doi:10.1016/j.fmre.2023.01.007>.

**License** GPL-3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**URL** `https://stscl.github.io/spEDM/`, `https://github.com/stscl/spEDM`

**BugReports** `https://github.com/stscl/spEDM/issues`

**Depends** R (>= 4.1.0)

**LinkingTo** Rcpp, RcppThread, RcppArmadillo

**Imports** dplyr, ggplot2, methods, sdsfun (>= 0.7.0), sf, terra

**Suggests** knitr, Rcpp, RcppThread, RcppArmadillo, rmarkdown, spData

**VignetteBuilder** knitr

**Config/pak/sysreqs** libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

**Repository** https://stscl.r-universe.dev

**RemoteUrl** https://github.com/stscl/spEDM

**RemoteRef** HEAD

**RemoteSha** fce3b8c5c0eb680453cea4d86d469e45bec6dc3d

# Contents

## `detectThreads` *detect the number of available threads*

### Description

detect the number of available threads

### Usage

```
detectThreads()
```

### Value

An integer

### Examples

```
detectThreads()
```

## `embedded` *embedding spatial cross sectional data*

### Description

embedding spatial cross sectional data

### Usage

```
## S4 method for signature 'sf'
embedded(data, target, E = 3, tau = 1, nb = NULL, trend.rm = FALSE)

## S4 method for signature 'SpatRaster'
embedded(data, target, E = 3, tau = 1, trend.rm = FALSE)
```

### Arguments

`data` The observation data.

`target` Name of target variable.

`E` (optional) Dimensions of the embedding.

`tau` (optional) Step of spatial lags.

`nb` (optional) The neighbours list.

`trend.rm` (optional) Whether to remove the linear trend.

### Value

A matrix

### Examples

```
columbus = sf::read_sf(system.file("shapes/columbus.gpkg", package="spData"))
embedded(columbus,target = "CRIME", E = 3)
```

## gccm *geographical convergent cross mapping*

### Description

geographical convergent cross mapping

### Usage

```
## S4 method for signature 'sf'
gccm(
  data,
  cause,
  effect,
  libsizes,
  E = 3,
  tau = 1,
  k = E + 2,
  theta = 1,
  algorithm = "simplex",
  lib = NULL,
  pred = NULL,
  nb = NULL,
  threads = detectThreads(),
  parallel.level = "low",
  bidirectional = TRUE,
  trend.rm = TRUE,
  progressbar = TRUE
)

## S4 method for signature 'SpatRaster'
gccm(
  data,
  cause,
  effect,
  libsizes,
  E = 3,
  tau = 1,
  k = E + 2,
  theta = 1,
  algorithm = "simplex",
  lib = NULL,
  pred = NULL,
  threads = detectThreads(),
  parallel.level = "low",
  bidirectional = TRUE,
  trend.rm = TRUE,
  progressbar = TRUE
)
```

## Arguments

`data` The observation data.

`cause` Name of causal variable.

`effect` Name of effect variable.

`libsizes` A vector of library sizes to use.

`E` (optional) Dimensions of the embedding.

`tau` (optional) Step of spatial lags.

`k` (optional) Number of nearest neighbors to use for prediction.

`theta` (optional) Weighting parameter for distances, useful when `algorithm` is `smap`.

`algorithm` (optional) Algorithm used for prediction.

`lib` (optional) Libraries indices.

`pred` (optional) Predictions indices.

`nb` (optional) The neighbours list.

`threads` (optional) Number of threads.

`parallel.level` (optional) Level of parallelism, `low` or `high`.

`bidirectional` (optional) whether to identify bidirectional causal associations.

`trend.rm` (optional) Whether to remove the linear trend.

`progressbar` (optional) whether to print the progress bar.

## Value

A list

`xmap` cross mapping prediction results

`varname` names of causal and effect variable

`bidirectional` whether to identify bidirectional causal associations

## Examples

```
columbus = sf::read_sf(system.file("shapes/columbus.gpkg", package="spData"))

g = gccm(columbus,"HOVAL","CRIME",libsizes = seq(5,45,5),E = 6)
g
plot(g, ylimits = c(0,0.85))
```

---

# `multiview` *multiview embedding forecast*

---

## Description

multiview embedding forecast

## Usage

```
## S4 method for signature 'sf'
multiview(
  data,
  columns,
  target,
  nvar,
  lib = NULL,
  pred = NULL,
  E = 3,
  tau = 1,
  k = E + 2,
  nb = NULL,
  top = NULL,
  threads = detectThreads(),
  trend.rm = TRUE
)

## S4 method for signature 'SpatRaster'
multiview(
  data,
  columns,
  target,
  nvar,
  lib = NULL,
  pred = NULL,
  E = 3,
  tau = 1,
  k = E + 2,
  top = NULL,
  threads = detectThreads(),
  trend.rm = TRUE
)
```

### Arguments

| | |
|---|---|
| `data` | The observation data. |
| `columns` | Names of individual variables. |
| `target` | Name of target variable. |
| `nvar` | Number of variable combinations. |
| `lib` | (optional) Libraries indices. |
| `pred` | (optional) Predictions indices. |
| `E` | (optional) Dimensions of the embedding. |
| `tau` | (optional) Step of spatial lags. |
| `k` | (optional) Number of nearest neighbors used for prediction. |
| `nb` | (optional) The neighbours list. |
| `top` | (optional) Number of reconstructions used for MVE forecast. |
| `threads` | (optional) Number of threads. |
| `trend.rm` | (optional) Whether to remove the linear trend. |

### Value

A vector (when input is sf object) or matrix

### Examples

```
columbus = sf::read_sf(system.file("shapes/columbus.gpkg", package="spData"))

multiview(columbus,
          columns = c("INC","CRIME","OPEN","PLUMB","DISCBD"),
          target = "HOVAL", nvar = 3)
```

## `simplex` *simplex forecast*

### Description

simplex forecast

### Usage

```
## S4 method for signature 'sf'
simplex(
  data,
  target,
  lib = NULL,
  pred = NULL,
  E = 1:10,
  tau = 1,
  k = E + 2,
  nb = NULL,
  threads = detectThreads(),
  trend.rm = TRUE
)

## S4 method for signature 'SpatRaster'
simplex(
  data,
  target,
  lib = NULL,
  pred = NULL,
  E = 1:10,
  tau = 1,
  k = E + 2,
  threads = detectThreads(),
  trend.rm = TRUE
)
```

### Arguments

| | |
|---|---|
| `data` | The observation data. |
| `target` | Name of target variable. |
| `lib` | (optional) Libraries indices. |
| `pred` | (optional) Predictions indices. |
| `E` | (optional) Dimensions of the embedding. |
| `tau` | (optional) Step of spatial lags. |
| `k` | (optional) Number of nearest neighbors used for prediction. |

`nb` (optional) The neighbours list.

`threads` (optional) Number of threads.

`trend.rm` (optional) Whether to remove the linear trend.

### Value

A list

`xmap` self mapping prediction results

`varname` name of target variable

### Examples

```
columbus = sf::read_sf(system.file("shapes/columbus.gpkg", package="spData"))

simplex(columbus,target = "CRIME")
```

## smap *smap forecast*

### Description

smap forecast

### Usage

```
## S4 method for signature 'sf'
smap(
  data,
  target,
  lib = NULL,
  pred = NULL,
  E = 3,
  tau = 1,
  k = E + 2,
  theta = c(0, 1e-04, 3e-04, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 0.5, 0.75, 1, 1.5, 2, 3,
    4, 6, 8),
  nb = NULL,
  threads = detectThreads(),
  trend.rm = TRUE
)

## S4 method for signature 'SpatRaster'
smap(
  data,
  target,
  lib = NULL,
  pred = NULL,
  E = 3,
  tau = 1,
  k = E + 2,
 theta = c(0, 1e-04, 3e-04, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 0.5, 0.75, 1, 1.5, 2, 3,
    4, 6, 8),
  threads = detectThreads(),
  trend.rm = TRUE
)
```

## Arguments

| | |
|---|---|
| `data` | The observation data. |
| `target` | Name of target variable. |
| `lib` | (optional) Libraries indices. |
| `pred` | (optional) Predictions indices. |
| `E` | (optional) Dimensions of the embedding. |
| `tau` | (optional) Step of spatial lags. |
| `k` | (optional) Number of nearest neighbors used for prediction. |
| `theta` | (optional) Weighting parameter for distances. |
| `nb` | (optional) The neighbours list. |
| `threads` | (optional) Number of threads. |
| `trend.rm` | (optional) Whether to remove the linear trend. |

## Value

A list

`xmap` self mapping prediction results

`varname` name of target variable

## Examples

```
columbus = sf::read_sf(system.file("shapes/columbus.gpkg", package="spData"))

smap(columbus,target = "INC")
```

# Index