Package 'ICvectorfields'

May 7, 2026

Title Vector Fields from Spatial Time Series of Population Abundance

Version 0.1.2

Description Functions for converting time series of spatial abundance or density data in raster format to vector fields of population movement using the digital image correlation technique. More specifically, the functions in the package compute cross-covariance using discrete fast Fourier transforms for computational efficiency. Vectors in vector fields point in the direction of highest two dimensional cross-covariance. The package has a novel implementation of the digital image correlation algorithm that is designed to detect persistent directional movement when image time series extend beyond a sequence of two raster images.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Suggests ggnewscale, ggplot2, knitr, metR, ncf, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports fftwtools, Rcpp, terra (>= 1.5-21)

Depends R (>= 2.10)

VignetteBuilder knitr

BugReports https://github.com/goodsman/ICvectorfields/issues

LinkingTo Rcpp

NeedsCompilation yes

Author Devin Goodsman [aut, cre] (ORCID: <https://orcid.org/0000-0003-1935-5779>)

Maintainer Devin Goodsman <goodsman@ualberta.ca>

Repository CRAN

Date/Publication 2022-02-26 22:30:02 UTC

Contents

`DispField` *Displacement fields based on 2D cross-covariance*

Description

Calculates a displacement field based on the cross-covariance of two input rasters presumably representing spatial population abundance or density at two different instances of time.

Usage

```
DispField(inputrast1, inputrast2, factv1, facth1, restricted = FALSE)
```

Arguments

`inputrast1`	a raster as produced by terra::rast
`inputrast2`	a raster of equivalent dimension to inputrast1 as produced by terra::rast
`factv1`	an odd integer for the vertical dimension of sub-grids
`facth1`	an odd integer for the horizontal dimension of sub-grids
`restricted`	logical (TRUE or FALSE)

Details

The input rasters are first converted to equivalent matrices. The function then divides the domain up into sub-grids of size factv1 X facth1, which are vertical and horizontal sub-grid dimensions.

If restricted is set to FALSE (the default), the function computes cross-covariance between each sub-grid of the first input raster and the entirety of the second input raster and then uses the location of maximum cross-covariance to estimate displacement in the vertical and horizontal directions from the centre of each sub-grid.

If restricted is set to TRUE, the function uses cross-covariance between each sub-grid in the first input raster and the equivalent sub-grid in the second input raster to estimate vertical and horizontal displacement.

Reference coordinates and cell size are extracted from the first input raster such that the locations from whence displacement is estimated as well as displacement estimates can be expressed in the units of the projected coordinates.

The coordinates are assumed to increase vertically and horizontally from the lower left corner of the two-dimensional domain.

Caution is warranted when defining the sub-grid dimensions because the function can produce erroneous results when sub-grids are too small.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, centx, centy, dispx, and dispy. The rowcent and colcent column names are the row and column indices for the center of each sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; centx and centy are the projected coordinates of the centre of the subgrid derived from the raster input files; dispx and dispy are the displacement in the horizontal and vertical directions in the same units as the projected coordinates of the raster input files.

See Also

`DispFieldbb` for a similar function using a bounding box to define a focal region, `DispFieldST` for a version designed to quantify persistent directional movement when the time series features more than two time instances, `DispFieldSTall` for a version designed to quantify persistent directional movement when velocity is variable in space, and `Xcov2D` for demonstration of how two-dimensional cross-covariance is used to determine displacement (see examples of Xcov2D function documentation).

Examples

```
(Mat1 <- matrix(rep(c(1:5, 0, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat2 <- matrix(rep(c(0, 1:5, 0, 0, 0), 9), nrow = 9, byrow = TRUE))

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)
```

```
(VFdf1 <- DispField(rast1, rast2, factv1 = 9, facth1 = 9))
# The second raster is shifted right by 1 unit relative to the first raster
# dispx = 1
```

DispFieldbb *Displacement fields based on 2D cross-covariance using bounding box*

Description

Calculates a displacement field based on the cross-covariance of two input rasters presumably representing spatial population abundance or density at two different instances of time. This version differs from DispField in that the user defines a bounding box that determines a single sub-grid. The center of the bounding box is the location from whence displacement is estimated.

Usage

```
DispFieldbb(
  inputrast1,
  inputrast2,
  rowmn,
  rowmx,
  colmn,
  colmx,
  restricted = FALSE
)
```

Arguments

inputrast1	a raster as produced by terra::rast
inputrast2	a raster of equivalent dimension to inputrast1 as produced by terra::rast
rowmn	an integer denoting the minimum row index of the sub-grid
rowmx	an integer denoting the maximum row index of the sub-grid
colmn	an integer denoting the minimum column index of the sub-grid
colmx	an integer denoting the maximum column index of the sub-grid
restricted	logical (TRUE or FALSE)

Details

The input rasters are first converted to equivalent matrices. If restricted is set to FALSE (the default), the function computes cross-covariance between the sub-grid of the first input raster and the entirety of the second input raster and then uses the location of maximum cross-covariance to estimate displacement in the vertical and horizontal directions from the centre of the sub-grid.

If restricted is set to TRUE, the function uses cross-covariance between the sub-grid of the first input raster and the equivalent sub-grid of the second input raster to estimate vertical and horizontal displacement.

Reference coordinates and cell size are extracted from the first input raster such that the locations from whence displacement is estimated as well as displacement estimates can be expressed in the units of the projected coordinates.

The coordinates are assumed to increase vertically and horizontally from the lower left corner of the two-dimensional domain.

Caution is warranted when defining the bounding box because the function can produce erroneous results when the bounding box is too small.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, centx, centy, dispx, and dispy. The rowcent and colcent column names are the row and column indices for the center of the sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; centx and centy are the projected coordinates of the centre of the subgrid derived from the raster input files; dispx and dispy are the displacement in the horizontal and vertical directions in the same units as the projected coordinates of the raster input files.

See Also

DispField for a similar function with a grid of focal regions, DispFieldSTbb for a version designed to quantify persistent directional movement when the time series features more than two time instances, DispFieldSTbball for a version designed to quantify persistent directional movement when velocity is variable in space, and Xcov2D for demonstration of how two-dimensional cross-covariance is used to determine displacement (see examples of Xcov2D function documentation).

Examples

```
rseq <- stats::runif(72)
Mat1 <- matrix(rep(0, 81), nrow = 9)
Mat2 <- Mat1
Mat1[1:9, 1:8] <- rseq
Mat1
Mat2[1:9, 2:9] <- rseq
Mat2

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

(VFdf1 <- DispFieldbb(rast1, rast2, 2, 8, 2, 8))
# The second raster is shifted right by 1 unit relative to the first raster
# dispx = 1
```

DispFieldST *Displacement fields for spatiotemporal data when velocity is spatially constant*

Description

This is an implementation of a novel algorithm that differs from more traditional digital image correlation implementations that are applied in the DispField and DispFieldbb functions. The function calculates a displacement field representing persistent movement based on the cross-covariance in a raster stack (in this case a sequential series of rasters) presumably representing spatial population abundance or density at more than two different instances of time. If analysis is restricted to only two time instances, DispField is more appropriate.

Usage

```
DispFieldST(inputstack1, lag1, factv1, facth1, restricted = FALSE)
```

Arguments

inputstack1	a raster stack with each raster layer representing an instance of time. The raster stack should be organized such that the first raster in the stack is the first observed spatial dataset and time progresses forward with the third dimension index of the raster stack. The raster stack should contain only numeric values. Any NA value will be converted to a zero
lag1	an integer time lag
factv1	an odd integer for the vertical dimension of subgrids
facth1	an odd integer for the horizontal dimension of subgrids
restricted	logical (TRUE or FALSE)

Details

The input rasters in the raster stack are first converted to equivalent matrices, which together represent a three-dimensional array with two spatial dimensions and one time dimension. The prescribed lag is applied to the three dimensional array derived from the raster stack by first producing two equivalent arrays and then removing appropriate numbers of layers from the top of one and the bottom of the other. These are referred to as unlagged and lagged spatiotemporal arrays in the description that follows.

Prior to computing displacement based on direction of maximum cross-covariance, the function divides the spatial domain up into sub-grids of size factv1 X facth1, which are vertical and horizontal sub-grid spatial dimensions.

The function converts three dimensional lagged and unlagged spatiotemporal arrays to two-dimensional lagged and unlagged spatiotemporal matrices by averaging along one of the spatial dimensions (either rows or columns) to obtain two pairs of two-dimensional matrices in which one dimension is spatial (either rows or columns) and one dimension is temporal. One of each pair corresponds to the unlagged spatiotemporal array and the other corresponds to the lagged spatiotemporal array.

Displacement in the vertical direction is computed using unlagged and lagged matrices that have been averaged along rows and displacement in the horizontal direction is computed using unlagged and lagged matrices that have been averaged along columns.

If restricted is set to FALSE (the default), the function computes cross-covariance between each sub-grid of the unlagged row-averaged spatiotemporal matrix and the whole row-averaged lagged spatiotemporal matrix and between each sub-grid of the unlagged column-averaged spatiotemporal matrix and the entirety corresponding lagged matrix.

If restricted is set to TRUE, the function uses cross-covariance between lagged and unlagged version of row-averaged and column averaged spatiotemporal matrices that have all been either row or column-averaged within sub-grids to estimate vertical and horizontal displacement.

Regardless of whether restricted is set to TRUE or FALSE, for each sub-grid, displacement in the x and y direction is divided by the shift in the time dimension to produce orthogonal velocity vetors. Note that for this reason, the lag1 argument of the function does not necessarily determine the time lag that is used to produce each orthoganal velocity vector.

Reference coordinates and cell size are extracted from the first raster stack such that the locations from whence displacement is estimated as well as displacement (or velocity) estimates can be expressed in the units of the projected coordinates.

The coordinates are assumed to increase vertically and horizontally from the lower left corner of the two-dimensional domain.

Caution is warranted when defining the sub-grid dimensions because the function can produce erroneous results when sub-grids are too small.

In addition, results can be quite sensitive to specification of the time lag. If velocities are highly variable in space or over time, avoid specifying a single time lag by calling the related `DispFieldSTall` function.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, centx, centy, dispx, and dispy. The rowcent and colcent column names are the row and column indices for the center of each sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; centx and centy are the projected coordinates of the centre of the subgrid derived from the raster input files; dispx and dispy are the orthoganal velocity vectors in units of space per timestep in the horizontal and vertical directions in the same spatial units as the projected coordinates of the raster input files.

See Also

`DispField` for a similar function with a grid of focal regions for only two time instances, `DispFieldSTbb` for a version designed to quantify persistent directional movement when the time series features more than two time instances but using a bounding pox to define a focal region, see `DispFieldSTall` for a version designed to quantify persistent directional movement when velocity is variable in space, and `Xcov2D` for demonstration of how two-dimensional cross-covariance is used to determine displacement (see examples of Xcov2D function documentation).

Examples

```
(Mat1 <- matrix(rep(c(1:5, 0, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat2 <- matrix(rep(c(0, 1:5, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat3 <- matrix(rep(c(0, 0, 1:5, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat4 <- matrix(rep(c(0, 0, 0, 1:5, 0), 9), nrow = 9, byrow = TRUE))

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)
rast3 <- terra::rast(Mat3)
terra::plot(rast3)
rast4 <- terra::rast(Mat4)
terra::plot(rast4)

teststack1 <- c(rast1, rast2, rast3, rast4)
(VFdf2 <- DispFieldST(teststack1, lag1 = 1, factv1 = 9, facth1 = 9))
# block is moving rightward at a speed of 1 unit of space per unit of time
# dispx = 1
```

`DispFieldSTall` *Diplacement fields for spatiotemporal data when velocity varies spatially*

Description

This is an implementation of a novel algorithm that differs from more traditional digital image correlation implementations that are applied in the `DispField` and `DispFieldbb` functions. This version is similar to the `DispFieldST` function except that it does not require a specific time lag. Instead the user specifies a maximum time lag and the function computes displacement vectors using the time lag that produces the maximum speed (magnitude of displacement divided by time lag). The function calculates a displacement field representing persistent movement based on the cross-covariance in a raster stack (in this case a sequential series of rasters) presumably representing spatial population abundance or density at more than two different instances of time. If analysis is restricted to only two time instances, `DispField` is more appropriate.

Usage

```
DispFieldSTall(inputstack1, lagmax, factv1, facth1, restricted = FALSE)
```

Arguments

`inputstack1`	a raster stack with each raster layer representing an instance of time. The raster stack should be organized such that the first raster in the stack is the first observed spatial dataset and time progresses forward with the third dimension index of the raster stack. The raster stack should contain only numeric values. Any NA value will be converted to a zero

lagmax	an integer representing the maximum time lag
factv1	an odd integer for the vertical dimension of subgrids
facth1	an odd integer for the horizontal dimension of subgrids
restricted	logical (TRUE or FALSE)

Details

The DispFieldSTall function has the same inner workings as the DispFieldST function except that instead of specifying a specific time lag, the user specifies a maximum time lag. The function then cycles through all lags up to the maximum time lag and chooses the for each location the maximum speed. The DispFieldSTall function is more appropriate than DispFieldST when velocity is variable in space.

Caution is warranted when defining the sub-grid dimensions because the function can produce erroneous results when sub-grids are too small.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, centx, centy, dispx, and dispy. The rowcent and colcent column names are the row and column indices for the center of each sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; centx and centy are the projected coordinates of the centre of the subgrid derived from the raster input files; dispx and dispy are the orthoganal velocity vectors in units of space per timestep in the horizontal and vertical directions in the same spatial units as the projected coordinates of the raster input files.

See Also

DispField for a similar function with a grid of focal regions for only two time instances, DispFieldST for a version designed to quantify persistent directional movement when the time series features more than two time instances and the velocity is constant in space, DispFieldSTbball for a version designed to quantify persistent directional movement when velocity is variable in space and the focal region is defined using a bounding box, and Xcov2D for demonstration of how two-dimensional cross-covariance is used to determine displacement (see examples of Xcov2D function documentation).

Examples

```
(Mat1 <- matrix(rep(c(1:5, 0, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat2 <- matrix(rep(c(0, 1:5, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat3 <- matrix(rep(c(0, 0, 1:5, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat4 <- matrix(rep(c(0, 0, 0, 1:5, 0), 9), nrow = 9, byrow = TRUE))

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)
rast3 <- terra::rast(Mat3)
```

```
terra::plot(rast3)
rast4 <- terra::rast(Mat4)
terra::plot(rast4)

teststack1 <- c(rast1, rast2, rast3, rast4)
(VFdf4 <- DispFieldSTall(teststack1, lagmax = 2, factv1 = 9, facth1 = 9))
# block is moving rightward at a speed of 1 unit of space per unit of time
# dispx = 1
```

`DispFieldSTbb` *Displacement fields for spatiotemporal data using a bounding box*

Description

This is an implementation of a novel algorithm that differs from more traditional digital image correlation implementations that are applied in the `DispField` and `DispFieldbb` functions. The function calculates a displacement field representing persistent movement based on the cross-covariance in a raster stack (in this case a sequential series of rasters) presumably representing spatial population abundance or density at more than two different instances of time. If analysis is restricted to only two time instances, `DispFieldbb` is more appropriate.

Usage

```
DispFieldSTbb(
  inputstack1,
  lag1,
  rowmn,
  rowmx,
  colmn,
  colmx,
  restricted = FALSE
)
```

Arguments

`inputstack1`	a raster stack with each raster layer representing an instance of time. The raster stack should be organized such that the first raster in the stack is the first observed spatial dataset and time progresses forward with the third dimension index of the raster stack. The raster stack should contain only numeric values. Any NA value will be converted to a zero
`lag1`	an integer time lag
`rowmn`	an integer for the minimum row index of the bounding box
`rowmx`	an integer for the maximum row index of the bounding box
`colmn`	an integer for the minimum column index of the bounding box
`colmx`	an integer for the maximum column index of the bounding box
`restricted`	logical (TRUE or FALSE)

Details

The input rasters in the raster stack are first converted to equivalent matrices, which together represent a three-dimensional array with two spatial dimensions and one time dimension. The prescribed lag is applied to the three dimensional array derived from the raster stack by first producing two equivalent arrays and then removing appropriate numbers of layers from the top of one and the bottom of the other. These are referred to as unlagged and lagged spatiotemporal arrays in the description that follows.

The function converts three dimensional lagged and unlagged spatiotemporal arrays to two-dimensional lagged and unlagged spatiotemporal matrices by averaging along one of the spatial dimensions (either rows or columns) to obtain two pairs of two-dimensional matrices in which one dimension is spatial (either rows or columns) and one dimension is temporal. One of each pair corresponds to the unlagged spatiotemporal array and the other corresponds to the lagged spatiotemporal array. Displacement in the vertical direction is computed using unlagged and lagged matrices that have been averaged along rows and displacement in the horizontal direction is computed using unlagged and lagged matrices that have been averaged along columns.

If restricted is set to FALSE (the default), the function computes cross-covariance between the values within the bounding box of the unlagged row-averaged spatiotemporal matrix and the whole row-averaged lagged spatiotemporal matrix and between the values within the bounding box of the unlagged column-averaged spatiotemporal matrix and the entirety corresponding lagged matrix.

If restricted is set to TRUE, the function uses cross-covariance between lagged and unlagged version of row-averaged and column averaged spatiotemporal matrices that have all been either row or column-averaged within the bounding box to estimate vertical and horizontal displacement.

Regardless of whether restricted is set to TRUE or FALSE, for each sub-grid, displacement in the x and y direction is divided by the shift in the time dimension to produce orthogonal velocity vetors. Note that for this reason, the lag1 argument of the function does not necessarily determine the time lag that is used to produce each orthoganal velocity vector.

Reference coordinates and cell size are extracted from the first raster stack such that the locations from whence displacement is estimated as well as displacement (or velocity) estimates can be expressed in the units of the projected coordinates.

The coordinates are assumed to increase vertically and horizontally from the lower left corner of the two-dimensional domain.

Caution is warranted when defining the sub-grid dimensions because the function can produce erroneous results when sub-grids are too small.

#' In addition, results can be quite sensitive to specification of the time lag. If velocities are highly variable in space or over time, avoid specifying a single time lag by calling the related `DispFieldSTbball` function.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, centx, centy, dispx, and dispy. The rowcent and colcent column names are the row and column indices for the center of each sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; centx and centy are the projected coordinates of the centre of the subgrid derived from the raster input files; dispx and dispy are the orthoganal velocity vectors in units of space per timestep

in the horizontal and vertical directions in the same spatial units as the projected coordinates of the raster input files.

See Also

DispField for a similar function with a grid of focal regions for only two time instances, DispFieldST for a version designed to quantify persistent directional movement when the time series features more than two time instances but using a grid to define focal regions, see DispFieldSTbball for a version designed to quantify persistent directional movement when velocity is variable in space, and Xcov2D for demonstration of how two-dimensional cross-covariance is used to determine displacement (see examples of Xcov2D function documentation).

Examples

```
rseq <- stats::runif(54)
Mat1 <- matrix(rep(0, 9*9), nrow = 9)
Mat2 <- Mat1; Mat3 <- Mat1; Mat4 <- Mat1
Mat1[1:9, 1:6] <- rseq
Mat1
Mat2[1:9, 2:7] <- rseq
Mat2
Mat3[1:9, 3:8] <- rseq
Mat3
Mat4[1:9, 4:9] <- rseq
Mat4

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)
rast3 <- terra::rast(Mat3)
terra::plot(rast3)
rast4 <- terra::rast(Mat4)
terra::plot(rast4)

teststack1 <- c(rast1, rast2, rast3, rast4)
(VFdf3 <- DispFieldSTbb(teststack1, lag1 = 1, 2, 8, 2, 8))
# block is moving rightward at a speed of 1 unit of space per unit of time
# dispx = 1
```

DispFieldSTbball *Diplacement fields using bounding box when velocity varies spatially*

Description

This is an implementation of a novel algorithm that differs from more traditional digital image correlation implementations that are applied in the DispField and Dispfieldbb functions. This version is similar to the DispFieldSTbb function except that it does not require a specific time

lag. Instead the user specifies a maximum time lag and the function computes displacement vectors using the time lag that produces the maximum speed (magnitude of displacement divided by time lag). The function calculates a displacement field representing persistent movement based on the cross-covariance in a raster stack (in this case a sequential series of rasters) presumably representing spatial population abundance or density at more than two different instances of time. If analysis is restricted to only two time instances, DispFieldbb is more appropriate.

Usage

```
DispFieldSTbball(
  inputstack1,
  lagmax,
  rowmn,
  rowmx,
  colmn,
  colmx,
  restricted = FALSE
)
```

Arguments

inputstack1	a raster stack with each raster layer representing an instance of time. The raster stack should be organized such that the first raster in the stack is the first observed spatial dataset and time progresses forward with the third dimension index of the raster stack. The raster stack should contain only numeric values. Any NA value will be converted to a zero
lagmax	an integer representing the maximum time lag
rowmn	an integer denoting the minimum row index of the sub-grid
rowmx	an integer denoting the maximum row index of the sub-grid
colmn	an integer denoting the minimum column index of the sub-grid
colmx	an integer denoting the maximum column index of the sub-grid
restricted	logical (TRUE or FALSE)

Details

The DispFieldSTbball function has the same inner workings as the DispFieldSTbb function except that instead of specifying a specific time lag, the user specifies a maximum time lag. The function then cycles through all lags up to the maximum time lag and chooses the for each location the maximum speed. The DispFieldSTbball function is more appropriate than DispFieldSTbb when velocity is variable in space.

Caution is warranted when defining the bounding box dimensions because the function can produce erroneous results when the bounding box is too small.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, centx, centy, dispx, and dispy. The rowcent and colcent column names are the row

and column indices for the center of each sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; centx and centy are the projected coordinates of the centre of the subgrid derived from the raster input files; dispx and dispy are the orthoganal velocity vectors in units of space per timestep in the horizontal and vertical directions in the same spatial units as the projected coordinates of the raster input files.

See Also

DispFieldbb for a similar function with focal region defined using a bounding box for only two time instances, DispFieldSTbb for a version designed to quantify persistent directional movement when velocity is constant in space and the focal region is defined using a bounding box, see DispFieldSTall for a version designed to quantify persistent directional movement when velocity is variable in space and focal regions are defined based on a grid, and Xcov2D for demonstration of how two-dimensional cross-covariance is used to determine displacement (see examples of Xcov2D function documentation).

Examples

```
(Mat1 <- matrix(rep(c(1:5, 0, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat2 <- matrix(rep(c(0, 1:5, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat3 <- matrix(rep(c(0, 0, 1:5, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat4 <- matrix(rep(c(0, 0, 0, 1:5, 0), 9), nrow = 9, byrow = TRUE))

# Rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)
rast3 <- terra::rast(Mat3)
terra::plot(rast3)
rast4 <- terra::rast(Mat4)
terra::plot(rast4)

teststack1 <- c(rast1, rast2, rast3, rast4)
(VFdf5 <- DispFieldSTbball(teststack1, lagmax = 2, 1, 9, 1, 9))
# block is moving rightward at a speed of 1 unit of space per unit of time
# dispx = 1
```

DispMoransI *Calculate statistics in source or sink regions*

Description

Functions for computing the statistics which may be driving variables of movement that has been quantified using the DispField or DispFieldbb functions. The same raster data as were supplied to the aforementioned functions must be supplied to these in addition to a raster layer for which statistics are sought. Then for each region of interest defined when DispField or DispFieldbb

were called, these functions compute statistics for presumed source (sourceloc = TRUE) locations or presumed sink locations (sourceloc = FALSE). Note that in the DispMornasI function, defining radius using distance means that a radius of one corresponds to the rook's neighbourhood.

Usage

```
DispMoransI(inputrast1, inputrast2, statrast, vfdf, sourceloc = TRUE, rad1)

DispStats(
  inputrast1,
  inputrast2,
  statrast,
  vfdf,
  sourceloc = TRUE,
  statistic = "var"
)
```

Arguments

inputrast1	a raster as produced by terra::rast
inputrast2	a raster of equivalent dimension to inputrast1 as produced by terra::rast
statrast	a raster of equivalent dimension to inputrast1 as produced by terra::rast which contains the variable that will be used to compute statistics
vfdf	a data frame returned by the DispField or DispFieldbb functions, which contains all of the information necessary for defining regions of interest as well as the displacement estimates
sourceloc	logical (TRUE or FALSE) indicating whether statistics are to be returned at source or sink locations
rad1	an ingeger indicating the neighbourhood radius for Moran's I statistic calculations in rows/columns. Any cell within a distance of rad1 cells of the focal cell is considered to be in its neighbourhood.
statistic	desired output statistic: It should be one of "mean", "var", or "sum". Default setting is var.

Value

A data frame is returned with all of the same columns as the vfdf input data frame plus an additional column containing the computed statistic in each region of interest defined in vfdf.

Examples

```
# Illustrating use of DispMoransI:

(Mat1 <- matrix(c(0.1,1,0.1,0,0,0,0,0,0,
                  1,0.1,1,0,0,0,0,0,0,
                  0.1,1,0.1,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
```

```
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0),
                 nrow = 9))
(Mat2 <- matrix(c(0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0.1,1,0.1,0,0,0,0,0,0,
                  1,0.1,1,0,0,0,0,0,0,
                  0.1,1,0.1,0,0,0,0,0,0),
                 nrow = 9))

# Note that rasterizing a matrix causes it to be rotated 90 degrees.
# Therefore, any shift in the x direction is in fact now a shift in the y direction
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

(VFdf1 <- DispField(rast1, rast2, factv1 = 3, facth1 = 3))
# The second raster is shifted down by -0.6666667 units relative to the first raster
# dispy = -0.6666667 (the width of each box is 0.1111111).

# Now to compute the statistics at the source: the Moran's I of the original values
# in each region of interest (should be minus one in first row)
(VFdf2 <- DispMoransI(rast1, rast2, rast1, VFdf1, sourceloc = TRUE, rad1 = 1))

# Illustrating use of DispStats:

(Mat1 <- matrix(c(1,1,1,0,0,0,0,0,0,
                  1,1,1,0,0,0,0,0,0,
                  1,1,1,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0),
                 nrow = 9))
(Mat2 <- matrix(c(0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,
                  1,1,1,0,0,0,0,0,0,
                  1,1,1,0,0,0,0,0,0,
                  1,1,1,0,0,0,0,0,0),
                 nrow = 9))
```

```
# Note that rasterizing a matrix causes it to be rotated 90 degrees.
# Therefore, any shift in the x direction is in fact now a shift in the y direction
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

(VFdf1 <- DispField(rast1, rast2, factv1 = 3, facth1 = 3))
# The second raster is shifted down by -0.6666667 units relative to the first raster
# dispy = -0.6666667 (the width of each box is 0.1111111).

# Now to compute the statistics at the source: the mean of the original values
# in each region of interest (should be one in first row)
(VFdf2 <- DispStats(rast1, rast2, rast1, VFdf1, sourceloc = TRUE, statistic = "mean"))
# sum in each region of interest (should be nine in first row)
(VFdf3 <- DispStats(rast1, rast2, rast1, VFdf1, sourceloc = TRUE, statistic = "sum"))
# variance in each region of interest (should be zero in all rows)
(VFdf4 <- DispStats(rast1, rast2, rast1, VFdf1, sourceloc = TRUE, statistic = "var"))
```

`GetRowCol` *Retrieve matrix row and column indices*

Description

Here is a function that will find the row and column indices of a matrix that are associated with a vector index.

Usage

```
GetRowCol(Index, dim1, dim2)
```

Arguments

`Index`	an integer vector index
`dim1`	integer row dimension of the matrix from which the row and column indices are to be extracted
`dim2`	integer column dimension of the matrix from which the row and column indices are to be extracted

Details

Often when applying functions like the R function which.max(matrix) to a matrix, a vector index is returned when the coder would prefer to have a row and column indices. This function converts the vector index to row and column indices.

The function assumes that the elements of the matrix are filled by column (byrow = FALSE), which is the default R matrix behaviour.

Value

a numeric vector of length two with two integers indicating row and column respectively

Examples

```
GetRowCol(6, dim1 = 3, dim2 = 3) # should return c(3, 2)
```

`MoransI` *Efficiently compute Moran's I statistic*

Description

Compute Moran's I for a matrix. A fast implementation of Moran's I for gridded data, with neighbours defined based on a radial distance. Note that when using radius to define the neighbourhood, a radius of one corresponds to the rook's neibhourhood. There is currently no equivalent to queen's neighbourhood.

Usage

```
MoransI(mat1, r1)
```

Arguments

`mat1`	a matrix of values; NA/Inf values must be coded as NA and are ignored
`r1`	an integer representing the distance (radius), within which nearby cells are considered neighbours in units of rows/columns

Value

a single numeric value for Moran's I

Examples

```
(TestMat <- matrix(c(1, 0, 1, 0, 1,
                     0, 1, 0, 1, 0,
                     1, 0, 1, 0, 1,
                     0, 1, 0, 1, 0,
                     1, 0, 1, 0, 1),
                   nrow =5))
# the code below should return -1
MoransI(TestMat, r1 = 1)
```

PatternDetect *Detect Patterns in Vector Fields*

Description

Detect patterns in vector fields represented on a grid by looking in the rook's neighbourhood of each grid cell. Four patterns are detected: convergences occur when the vectors in the four adjacent cells in the rook's neighbourhood point towards the focal cell; divergences occur when the vectors in the four adjacent cells point away from the focal cell; Partial convergences occur when three of the four vectors point towards the focal cell and the final vector points neither towards nor away from the focal cell; Partial divergences occur when three of the four vectors point away the focal grid cell and the final vector points neither towards nor away from the focal grid. For all of the patterns above a sub-pattern is specified if all arrows point clockwise or counter-clockwise.

Usage

```
PatternDetect(vfdf)
```

Arguments

vfdf	A data frame as returned by DispField, DispFieldST, or DispFieldSTall with at least five rows (more is better)

Value

A data frame as returned by DispField, DispFieldST, or DispFieldSTall, with three additional columns. The first additional column is called Pattern in which the patterns around each focal cell are categorized as convergence, divergence, partial convergence, partial divergence, or NA. The second additional column, called SubPattern, indicates whether all arrows point clockwise or counter-clockwise. The third additional column is called PatternCt, which contains a one if all four neighbourhood grid cells contain displacement estimates, and a NA otherwise.

Examples

```
# creating convergence/divergence patterns
Mat1 <- matrix(rep(0,9*9), nrow = 9)
Mat1[3, c(4, 6)] <- 1
Mat1[7, c(4, 6)] <- 1
Mat1[c(4, 6), 3] <- 1
Mat1[c(4, 6), 7] <- 1
Mat1

Mat2 <- matrix(rep(0,9*9), nrow = 9)
Mat2[2, c(4, 6)] <- 1
Mat2[8, c(4, 6)] <- 1
Mat2[c(4, 6), 2] <- 1
Mat2[c(4, 6), 8] <- 1
Mat2
```

```
# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

# Detecting a divergence
(VFdf1 <- DispField(rast1, rast2, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf1 <- PatternDetect(VFdf1))

# Detecting a convergence
(VFdf2 <- DispField(rast2, rast1, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf2 <- PatternDetect(VFdf2))
```

PixelCt *Count populated pixels in a raster stack*

Description

In order to produce reliable results, the cross-covariance approach implemented in DispField. DispFieldST, and DispFieldSTall needs a certain minimum number of non-zero or non-NA valued pixels in pairs of images or pairs of arrays derived from a raster stack that it uses to compute cross-covariance. The user may define a threshold such as ten percent of the pixels within each sub-grid. This function allows the user to assess whether the minimum threshold number of non-zero pixels per sub-grid are surpassed by returning the number of non-zero pixels within each sub-grid over all of the time instances in the user-supplied raster stack. The user may choose to disregard or mistrust displacement or velocity estimates derived from sub-grids with insufficient numbers of non-zero pixels. This function is designed to be called before or after one of the functions referenced above in order to enable the user to quantify their confidence in displacement or velocity estimates.

Usage

```
PixelCt(inputstack1, factv1, facth1)
```

Arguments

inputstack1	a raster stack with each raster layer representing an instance of time. The raster stack should be organized such that the first raster in the stack is the first observed spatial dataset and time progresses forward with the third dimension index of the raster stack. The raster stack should contain only numeric values. Any NA value will be converted to a zero
factv1	an odd integer for the vertical dimension of subgrids
facth1	an odd integer for the horizontal dimension of subgrids

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, and PixelCt. The rowcent and colcent column names are the row and column indices for the center of each sub-grid; frowmin and frowmax are the sub-grid minimum and maximum row indices; fcolmin and fcolmax are the sub-grid minimum and maximum column indices; pixelct is the count of non-zero pixels in the sub-grid over the entire time period covered by the input raster stack.

Examples

```
# below the example in the DispField function documentation is reproduced
(Mat1 <- matrix(rep(c(1:5, 0, 0, 0, 0), 9), nrow = 9, byrow = TRUE))
(Mat2 <- matrix(rep(c(0, 1:5, 0, 0, 0), 9), nrow = 9, byrow = TRUE))

# Note that rasterizing a matrix causes it to be rotated 90 degrees.
# Therefore, any shift in the x direction is in fact now a shift in the y direction
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

(Confdf1 <- PixelCt(c(rast1, rast2), factv1 = 9, facth1 = 9))
# This should return a pixel count of 54: This is the number
# of pixels that were occupied in either the first or second
# time instance.
```

RastStackData *Creating a raster stack from formatted datasets*

Description

This function converts the data that accompany the ICvectorfields R package to a raster stack. The raster stack is the only accepted data input format for the following ICvectorfields functions: DispFieldST, DispFieldSTbb, DispFieldSTall, DispFieldSTbball.

Usage

```
RastStackData(inputdf)
```

Arguments

inputdf	a data frame object in which the first column is longitude (or x coordinate), the second column is latitude (or y coordinate), and all of the subsequent columns represent a measure of population abundance or density at a unique instance of time. Each row of the input data frame, therefore, represents a unique spatial location, which should be on an evenly spaced grid. Note, however, that not all grid locations need to have observations; some grid locations can have values of NA or can be missing entirely.

Details

Once a raster stack has been created, individual layers can be subsetted using rasterstack[[index]], where index is an integer index for the third dimension of the raster stack.

Value

The function returns a raster stack constructed using inputdf. Each layer in the stack corresponds to a column of the input dataset (after the first two columns, which are longitude and latitude). The extent of all of the rasters in the stack is constructed using the minimum and maximum longitudes and latitudes.

Examples

```
# creating random data in the correct data format
xyzdf <- expand.grid(x = c(1:3), y = c(1:3))
xyzdf$z1 <- runif(9)
xyzdf$z2 <- runif(9)
xyzdf$z3 <- runif(9)

zstack <- RastStackData(xyzdf)

dim(zstack)
terra::plot(zstack[[1]])
terra::plot(zstack[[2]])
terra::plot(zstack[[3]])
```

RooksGradient *Calculate Gradient Statistics in the Rook's Neighbourhood*

Description

The movement of populations into adjacent cells may sometimes be influenced by the gradient of some predictive variable. This function enables the calculation of a simple gradient statistic in the rook's neighbourhood of each cell in a dataset. The statistic must first be computed for each grid cell using SubgridStats. Then for each grid, the RooksGradient function computes the arithmetic average of the difference between the statistic at the focal grid cell and the statistic in the four (or fewer) adjacent neighbours in its Rook's neibourhood. This arithmetically averaged difference is then returned under the column header of 'Gradient'. A negative gradient estimate indicates that the statistic in the central cell is higher than that in neighbouring cells whereas a positive gradient estimate indicates the opposite.

Usage

```
RooksGradient(vfdf, statistic = "mean")
```

Arguments

vfdf	A data frame as returned by SubgridStats
statistic	desired output statistic: It should be one of "mean", "var", or "sum". Default setting is mean.

Value

A data frame similar to vfdf except that it includes an additional column called Gradient as described above.

Examples

```
# creating pattern patterns
Mat1 <- matrix(rep(0,9*9), nrow = 9)
Mat1[c(4:6), c(4:6)] <- 2
Mat1[c(4:6), c(1:3)] <- 1
Mat1[c(1:3), c(4:6)] <- 1
Mat1[c(7:9), c(4:6)] <- 1
Mat1[c(4:6), c(7:9)] <- 1
Mat1

Rast1 <- terra::rast(Mat1)
terra::plot(Rast1)

# calculating the mean in 9 subgrids
(statsdf1 <- SubgridStats(Rast1, factv1 = 3, facth1 = 3, statistic = "mean"))

# computing the gradient statistic on the mean
(graddf1 <- RooksGradient(statsdf1, statistic = "mean"))
# the Gradient statistic in the central grid in row 5 should
# be equal to negative one
```

RooksNeighCt *Define a subset of grid locations with non-overlapping rook neighborhoods*

Description

This function prunes the data frame returned by the PatternDetect function such that it includes only rook's neighborhoods that do not overlap. Pruning is done by sequential removal of observations that are too near one another and as a result are overlapping. Locations that are the most highly connected are removed first.

Usage

```
RooksNeighCt(vfdf)
```

Arguments

vfdf	A data frame as returned by PatternDetect

Details

The reason for this function's existence is to facilitate probabilistic calculations regarding whether certain patterns are occurring more or less often that would be expected by chance. If rook's neighborhoods in which patterns are observed overlap, then the assumption of probabilistic independence is necessarily incorrect. Thus, overlap invalidates any calculation of the probability of occurrence of a particular pattern if that calculation assumes independence. The pruning actions of this function enable the user to more safely assume probabilistic independence.

Value

A data frame similar to vfdf except that it includes only grid locations with speed estimates in all four adjacent grid locations in their rook's neighborhood. An additional column called IndPatternCt is appended which contains NA values for locations that are overlapping other locations and ones for all non-overlapping locations that have speed estimates in all four adjacent cells.

Examples

```
# creating convergence/divergence patterns
Mat1 <- matrix(rep(0,9*9), nrow = 9)
Mat1[3, c(4, 6)] <- 1
Mat1[7, c(4, 6)] <- 1
Mat1[c(4, 6), 3] <- 1
Mat1[c(4, 6), 7] <- 1
Mat1

Mat2 <- matrix(rep(0,9*9), nrow = 9)
Mat2[2, c(4, 6)] <- 1
Mat2[8, c(4, 6)] <- 1
Mat2[c(4, 6), 2] <- 1
Mat2[c(4, 6), 8] <- 1
Mat2

Mat1 <- cbind(Mat1, Mat1)
Mat1 <- rbind(Mat1, Mat1)

Mat2 <- cbind(Mat2, Mat2)
Mat2 <- rbind(Mat2, Mat2)

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

# Detecting a divergence
(VFdf1 <- DispField(rast1, rast2, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf1 <- PatternDetect(VFdf1))
```

```
(subdf1 <- RooksNeighCt(patdf1))
# The last call should print a data table with four rows, each with a one under
# the column header of IndPatternCt
```

RooksNeighFind *Classify Rook's Neighbours Comprising Spread Patterns in Vector Fields*

Description

After running the PatternDetect function, this function enables classification of neighbour cells. Because the pattern detect function classifies central cells according to the patterns of vector direction in their Rook's neighbourhood, the neighbouring grid locations that comprise the pattern are not labeled. This is remedied by the RooksNeighFind function which classifies neighbour cells around focal grids classified with one of the four patterns that the PatternDetect function is able to recognize. The function returns a data frame similar to the input data frame with a column appended. In the appended column, neighbours surrounding focal cells labeled with a particular pattern will be labeled as follows: neighbours of the divergence pattern are labeled with a one, neighbours of the convergence pattern are labeled with a two, neighbours of the partial divergence pattern are labeled with a three, and neighbours of the partial convergence pattern are labeled with a four. In cases where neighbours are shared, the priority order from lowest to highest is four for partial convergence to one for divergence. Thus, a neighbour that is shared between a focal grid classified as a convergence and a nearby focal grid classified as a divergence will be labeled with a one instead of a two.

Usage

```
RooksNeighFind(vfdf)
```

Arguments

vfdf	A data frame as returned by PatternDetect

Value

A data frame similar to vfdf except that it includes an additional column called NeighType as described above.

Examples

```
# creating convergence/divergence patterns
Mat1 <- matrix(rep(0,9*9), nrow = 9)
Mat1[3, c(4, 6)] <- 1
Mat1[7, c(4, 6)] <- 1
Mat1[c(4, 6), 3] <- 1
Mat1[c(4, 6), 7] <- 1
Mat1

Mat2 <- matrix(rep(0,9*9), nrow = 9)
```

```
Mat2[2, c(4, 6)] <- 1
Mat2[8, c(4, 6)] <- 1
Mat2[c(4, 6), 2] <- 1
Mat2[c(4, 6), 8] <- 1
Mat2

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

# Detecting a divergence
(VFdf1 <- DispField(rast1, rast2, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf1 <- PatternDetect(VFdf1))
(neighdf1 <- RooksNeighFind(patdf1))
# Rook's neighbour grids are labeled with a one.

# Detecting a convergence
(VFdf2 <- DispField(rast2, rast1, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf2 <- PatternDetect(VFdf2))
(neighdf2 <- RooksNeighFind(patdf2))
# Rook's neighbour grids are labeled with a two.
```

`RotationDetect` *Detect Rotating Patterns in Vector Fields*

Description

Detect patterns in vector fields represented on a grid by looking in the rook's neighbourhood of each grid cell. This function is analogous to `PatternDetect`, except that it detects rotational patterns. Four patterns are detected: clockwise rotation when rotation in all four neighbbour grids appears clockwise, counter clockwise rotation when rotation in all four neighbour grids appears counter-clockwise, and partial clockwise and counter-clockwise rotation, when all but one of the four adjacent neighbour cells has vectors that indicate rotation. For all of the patterns above a subpattern is specified as convergence when all of the vectors in the four adjacent grids point towards the focal cell or a divergence when all of the vectors in the four adjacent grids point away from the focal cell.

Usage

```
RotationDetect(vfdf)
```

Arguments

`vfdf`	A data frame as returned by `DispField`, `DispFieldST`, or `DispFieldSTall` with at least five rows (more is better)

Value

A data frame as returned by DispField, DispFieldST, or DispFieldSTall, with three additional columns. The first additional column is called Pattern in which the patterns around each focal cell are categorized as clockwise, counter-clockwise, partial clockwise, partial counter-clockwise, or NA. The second additional column, called SubPattern, indicates whether all arrows point towards (convergence) or away (divergence) from the focal cell. The third additional column is called PatternCt, which contains a one if all four neighbourhood grid cells contain displacement estimates, and a NA otherwise.

Examples

```
# creating rotation patterns
Mat1 <- matrix(rep(0,9*9), nrow = 9)
Mat1[c(1:3), 4] <- 1
Mat1[c(7:9), 6] <- 1
Mat1[4, c(7:9)] <- 1
Mat1[6, c(1:3)] <- 1
Mat1

Mat2 <- matrix(rep(0,9*9), nrow = 9)
Mat2[c(1:3), 5] <- 1
Mat2[c(7:9), 5] <- 1
Mat2[5, c(7:9)] <- 1
Mat2[5, c(1:3)] <- 1
Mat2

# rasterizing
rast1 <- terra::rast(Mat1)
terra::plot(rast1)
rast2 <- terra::rast(Mat2)
terra::plot(rast2)

# Detecting clockwise rotation
(VFdf1 <- DispField(rast1, rast2, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf1 <- RotationDetect(VFdf1))

# Detecting counter-clockwise rotation
(VFdf2 <- DispField(rast2, rast1, factv1 = 3, facth1 = 3, restricted = TRUE))
(patdf2 <- RotationDetect(VFdf2))
```

SimData *Simulated movement data*

Description

Data based on a partial differential equation were simulated using the ReacTran R package (see details).

Usage

```
data(SimData)
```

Format

A data-frame with 40804 rows and 8 columns.

xcoord in arbitrary units

ycoord in arbitrary units

t1 concentration in arbitrary units at t = 1

t2 concentration in arbitrary units at t = 2

t3 concentration in arbitrary units at t = 3

t4 concentration in arbitrary units at t = 4

t5 concentration in arbitrary units at t = 5

t6 concentration in arbitrary units at t = 6

Details

The simulation algorithm uses a finite differencing scheme with backwards differencing. The model used for simulation is a reaction diffusion-advection equation in which the advection term is variable in space but diffusion and reactions are constant in space see convection-diffusion equation for an example.

The parameters used in the general partial differntial equation in the link above are

D = 0.01 per squared spatial unit

R = 0.5 per unit time

v (advection is variable in space): in the upper left quadrant of the square domain v = (0.2, 0); in the upper right quadrant v = (0, -0.2); in the lower right quadrant v = (-0.2, 0); in the lower right quadrant v = (0, 0.2). Obviously v is discontinous at the quadrant boundaries, which causes some interesting model behaviour that is limited by considering only the first six time steps such that the bulk of the concentration in each quadrant does not cross a quadrant boundary.

The intial condition at time = 0 is a concentration of one unit per arbitrary unit of volume in the central cell of each quadrant.

External boundary conditions are zero-gradient (reflecting).

The data are formatted such that they can easily be converted to a raster stack using ICvectorfields::RastStackData(SimData).

SubgridMoransI *Compute statistics for subgrids*

Description

Functions that facilitate calculation of statistics at the sub-grid level. These may be useful for drivers of movement speed or direction if used in tandem with DispField, DispFieldST, or DispFieldSTall.

Usage

```
SubgridMoransI(inputrast1, factv1, facth1, rad1 = 1)

SubgridStats(inputrast1, factv1, facth1, statistic = "var")
```

Arguments

inputrast1	a raster as produced by terra::rast
factv1	an odd integer for the vertical dimension of sub-grids
facth1	an odd integer for the horizontal dimension of sub-grids
rad1	an integer indicating the neighbourhood radius for Moran's I statistic calculations in rows/columns. Any cell within a distance of rad1 cells of the focal cell is considered to be in its neighbourhood.
statistic	desired output statistic: It should be one of "mean", "var", or "sum". Default setting is var.

Details

Note that when using radius to define the neighbourhood in Moran's I calculations, a radius of one corresponds to the rook's neibhourhood. Values that are NA or Inf are not included in calculations of the Moran's I statistic nor in any of the other statistics that can be computed.

Value

A data frame is returned with the following column names: rowcent, colcent, frowmin, frowmax, fcolmin, fcolmax, and a column for the output statistic.

See Also

DispStats and DispMoransIfor functions that compute statistics at presumed source or sink locations in each region of interest.

Examples

```
(TestMat <- matrix(c(1, 0, 1, 0, 1,
                     0, 1, 0, 1, 0,
                     1, 0, 1, 0, 1,
                     0, 1, 0, 1, 0,
                     1, 0, 1, 0, 1),
                   nrow = 5))

TestRast <- terra::rast(TestMat)
terra::plot(TestRast)

SubgridMoransI(TestRast, factv1 = 5, facth1 = 5, rad1 = 1)
# using rad1 = 1 is equivalent to using the rooks neighbourhood
# and so the output should be -1.

(TestMat <- matrix(c(1, 0, 1, 0, 1,
                     0, 1, 0, 1, 0,
                     1, 0, 1, 0, 1,
                     0, 1, 0, 1, 0,
                     1, 0, 1, 0, 1),
                   nrow = 5))

TestRast <- terra::rast(TestMat)
terra::plot(TestRast)

SubgridStats(TestRast, factv1 = 5, facth1 = 5, statistic = "mean")
SubgridStats(TestRast, factv1 = 5, facth1 = 5, statistic = "var")
SubgridStats(TestRast, factv1 = 5, facth1 = 5, statistic = "sum")
```

Xcov2D *Cross-covariance in two spatial dimensions*

Description

This function efficiently computes two dimensional cross-covariance of two equal dimensioned matrices of real numbers using efficient discrete fast Fourier trasforms.

Usage

```
Xcov2D(mat1, mat2)
```

Arguments

mat1	a real valued matrix
mat2	a real valued matrix of equal dimension to mat1

Details

The algorithm first pads each matrix with zeros so that the outer edges of the matrices do not interact with one another due to the circular nature of the discrete fast Fourier transform. Cross-covariance calculations require computation of the complex conjugate of one of the two imput matrices. Assuming all of it's elements are real, computing the complex conjugate is equivalent to flipping the matrix in the horizontal and vertical directions. Then to compute cross-covariance, the first matrix is convolved with the flipped second matrix as described in the convolution theorem.

This function is called by the main functions that compute displacement fields and vector fields and is included here primarily for demonstration purposes. Specifically, the method for computing the magnitude and direction of shifts is demonstrated in the examples.

The shift that produces the maximum cross-covariance between the two input matrices can be obtained by finding the row and column indices associated with the maximum cross-covariance. The shift in each direction is obtained by subracting one plus the half the dimension of the output matrix (the same for rows and columns) from the row and column values that are associated with the maximum cross-covariance as demonstrated in the examples below. Note that shifts to the right and up are denoted with positive numbers and shifts to the left and down are denoted by negative numbers. This is contrary to some conventions but efficient for producing vector fields. For more details on cross-covariance see cross-correlation.

Value

a real valued matrix showing cross-covariance in each direction

Examples

```
matrix(c(1:6, rep(0, 3)), nrow = 3); matrix(c(rep(0, 3), 1:6), nrow = 3)
dim(Xcov2D(matrix(c(1:6, rep(0, 3)), nrow = 3),
    matrix(c(rep(0, 3), 1:6), nrow = 3)))
ICvectorfields::GetRowCol(
    which.max(Xcov2D(matrix(c(1:6, rep(0, 3)), nrow = 3),
        matrix(c(rep(0, 3), 1:6), nrow = 3))),
    dim1 = dim(Xcov2D(matrix(c(1:6, rep(0, 3)), nrow = 3),
        matrix(c(rep(0, 3), 1:6), nrow = 3)))[1],
    dim2 = dim(Xcov2D(matrix(c(1:6, rep(0, 3)), nrow = 3),
        matrix(c(rep(0, 3), 1:6), nrow = 3)))[2]
        )
# This implies that the shift is 6 - (10/2 + 1) in the vertical
# direction and 7 - (10/2 + 1) in the horizonatal direction.
```

Index