Package ‘imagingPC’

June 24, 2019

Type Package

Title Analysis of Imaging Mass Spectrometry Data using a Process Convolution Approach

Version 0.1.0

Author Cameron Miller

Maintainer Cameron Miller <millercs@musc.edu>

Description Analysis of imaging data collected on a regular grid using a process convolution (PC) approach. In the discrete PC approach we employ, a zero-centered latent process is convolved with a smoothing kernel function in order to account for spatial information. For computational efficiency with large imaging datasets, this package implements a semivariogram-based approach to estimate and fix the smoothing kernel function. In the context of a Bayesian mixed models framework, this package writes and fits models to estimate the latent process at a limited set of locations called support sites while also incorporating covariates of interest. Furthermore, this package incorporates the PC approach into left-censored models and marginalized two-part models to account for different zero-generating processes. This package was designed for imaging mass spectrometry (IMS) data, but the methods can be extended to imaging data collected over a regular grid.

License GPL-2

Encoding UTF-8

LazyData true

Imports geoR (>= 1.7.5.2.1),
nimble (>= 0.6.12),
plyr (>= 1.8.4),
coda (>= 0.19.1),
ggplot2 (>= 3.0.0),
gridExtra (>= 2.3),
grid (>= 3.5.0),
cowplot (>= 0.9.4)

RoxygenNote 6.1.1

Suggests knitr,
rmarkdown,
testthat

VignetteBuilder knitr

R topics documented:

`assignIDs` *Assign ID numbers to samples*

Description

`assignIDs` attempts to assign ID numbers to contiguous regions.

Usage

```
assignIDs(data, xCoord = "x", yCoord = "y",
  neighborhood = "tertiary")
```

Arguments

`data`	The dataset containing x- and y-coordinates.
`xCoord`	A character string specifying the name of the x-coordinate variable.
`yCoord`	A character string specifying the name of the y-coordinate variable.
`neighborhood`	A character string specifying the neighborhood structure to be considered. The options are "primary", "secondary", and "tertiary".

Value

The dataset appended with a variable called "assignID". This variable provides the assigned IDs as a numeric variable.

Examples

```
data("TAMdata")
TAMdata <- assignIDs(TAMdata)
```

`chooseStructures` *Choose support structure(s)*

Description

The `chooseStructures` function selects support structures for every level of the spatial variable (`spatialVar` argument) specified in the `estRange` function. If no spatial variable is provided, then only a single set of support structures is chosen, one structure for each sample. See the **Details** section for more information on how the structures are selected.

Usage

```
chooseStructures(rangeObj, cDist = 0.1, sdWithin = 1,
  defaultStructure = "nextHighest", sdDefault = 1,
  thresholdNumSup = 0.5, sdElim = 2, noZeroRangeStructs = TRUE)
```

Arguments

`rangeObj`	An object of class `rangeList`. This object is a list that contains the estimated range parameter(s). This is obtained by a call to the `estRange` function.
`cDist`	A distance expressing how far beyond the limits of the data that the support sites should be placed. In the `rScale` function, the data are rescaled such that data locations are a distance of 1 apart. Therefore, a cDist of 0.5 is half the distance between data points.
`sdWithin`	The maximum allowed distance between any data point and its nearest support site (in terms of the number of standard deviations of the smoothing kernel). The default is 1, meaning that data must be within one standard deviation of the smoothing kernel of a support site. Smaller values of sdWithin lead to demser support structures. This value should never be set higher than 1.
`defaultStructure`	The default structure if the number of support points exceeds the proportion threshold of data set by `thresholdNumSup`. The options are '5x' or 'nextHighest'.
`sdDefault`	The default standard deviation of the smoothing kernel. The default standard deviation is used if (1) the number of support sites exceeds that allowed by `thresholdNumSup` or (2) the recommended standard deviation of the smoothing kernel exceeds the largest distance between data points. `sdDefault` is a value mutliplied by the maximum observed distance between any data point and the nearest support site. For an `sdDefault` of 1, the maximum distance is mutliplied by 1, and that value is used for the standard deviation of the smoothing kernel when a default support structure is used.
`thresholdNumSup`	A threshold that limits the number of support sites allowed for computational efficiency. The threshold is a proportion of the observations. A threshold of 0.5 limits the number of allowed support sites to 50% of the number of observations.
`sdElim`	A threshold for eliminating support sites (in terms of the standard deviation of the smoothing kernel). Support sites are removed if they are not within the specified number of standard deviations of a data point. This is used to remove support sites for irregularly shaped samples.

`noZeroRangeStructs`
: A TRUE/FALSE argument specifying how to handle instances in which the estimated range parameter is 0. If `noZeroRangeStructs=TRUE` and the estimated range parameter is 0, then no support structure will be used for the corresponding data. If a spatial variable was provided, then no support structure will be used for the data in the corresponding level of that variable. If no spatial variable was provided, then a support structure will not be used at all. Instead, a raster-level intercept will be incorporated to account for extra noise. If `noZeroRanges=FALSE` and the estimated range parameter is 0, then a default support structure will be used.

Details

The process convolution approach uses points called support sites that help to account for underlying spatial structure in data. Collections of support sites are called support structures. How many and where to place support sites were questions that drove the research leading to the creation of this function.

The `chooseStructures` function uses the procedure detailed below. If a spatial variable was provided in the `estRange` function, then the procedure is performed for every level of the spaital variable.

The procedure starts by building all of the fixed support structures. The fixed support structures are a group of five support structures ranging from five to twelve support sites. The minimum distance between each data point and the closest support site is measured for all observations. Support sites that are more than (`sdElim`)*(recommended standard deviation of smoothing kernel) are removed. The total number of support sites is counted across all samples, and each structure is checked to see if all data are within (`sdWithin`)*(recommended standard deviation of smoothing kernel).

Once the fixed support structures are created, an iterative loop is used to generate the alternating support structures. These are support structures in which the rows of support sites have alternating numbers of support sites. For each support, the support structure is built and support sites more than (`sdElim`)*(recommended standard deviation of smoothing kernel) are removed. The support structure is then checked to see if (1) all the data are within (`sdWithin`)*(recommended standard deviation of smoothing kernel) and (2) the number of support sites across all samples has exceeded (`thresholdNumSup`)*(total number of observations). If either criteria is met, the iteratie loop stops. The alternating structures are compared to the fixed structures, and the support structure with the fewest support sites in which all of the data are within (sdWithin)*(recommended standard deviation of smoothing kernel) is chosen as the support structure.

Value

An object of class `structureList` that includes the information below.

`structure` A list of information about the support structures. If a spatial variable is provided, then there will be a list for each level of that variable. The structure list includes `recommendStruct` (the recommended support structure), `recommendSd` (the recommended standard deviation of the smoothing kernel), `nSupportReduced` (the number of support sites after removing those more than `sdElim` standard deviations away from a data point), `coordsU` (a data frame of the coordinates of the support sites, by sample), `coordsData` (a data frame of the coordinates of the data), `defaultStatus` (a character string informing the user if a default support structure was chosen), and `nRowSupport` (the number of rows of support sites for alternating support structures).

`data` The dataset, appended with new x- and y-coordinates.

`subjectVar` A character string denoting the subject variable.

`sampleVar` A character string denoting the sample variable.

`spatialVar` A character string denoting the spatial variable.

`outcome` A character string denoting the outcome of interest.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
```

`createAUCData` *Calculate the area under the curve (AUC) across m/z values on the raster level.*

Description

`createAUCData` calculates the AUC for sets of m/z values.

Usage

```
createAUCData(data, mzThreshold = 0.3, usedMergeIMSFiles = TRUE)
```

Arguments

`data`	A dataset containing columns of m/z measurements.
`mzThreshold`	An m/z threshold that determines which m/z values are combined when calculating the AUC. See `Details` for more information on how this works.

Details

This function is designed to work with data exported from SCiLS Lab software. When using SCiLS make sure to export the m/z values as columns, meaning every column should represent a separate m/z value, and every column name is the m/z value When the data are imported into R, R will add an 'X' in front of each number. Do not remove the 'X' in front as this function will automatically do that.

The `createAUCData` first searches through the m/z values and determines which to combine based on the `mzThreshold`. The function goes one-by-one through the m/z values and determines which other fragments are within the `mzThreshold` Those within the mzThreshold are considered to represent the same fragment and are used to calculate the AUC.

The `mzThreshold` must be smaller than the smallest m/z distance between m/z values representing different fragments. For example, if the highest m/z representing fragment 1 is 2466.005859, and the smallest m/z representing fragment 2 is 2466.878418, then the threshold must be less than 0.872559 (2466.878418-2466.005859=0.872559) to correctly distinguish between the two fragments.

As a result of the way m/z values are combined, this function should be used for data in which the m/z distances defining each fragment are less than the m/z distances between fragments. This is often the case when the data that have gone through the peak picking process. If entire spectra for spot-level data are submitted to this function, all m/z values will be combined.

Value

A new dataset with the AUC calculations. Any columns with column names that do not start with an "X" and then a nubmer will be kept. All original m/z columns will be removed.

`createPCData` *Create the information for running models*

Description

The `createPCData` function generates all the information needed to fit a process convolution model in a Bayesian setting.

Usage

```
createPCData(structureObj, covariates, covariateTypes, covariateLevels,
  trimData = FALSE)
```

Arguments

`structureObj`	An object of class `structureList`. This object is the result of using the `chooseStructures` function.
`covariates`	A concatenated character string specifying the covariates for which modeling information will be generated. Not all covariates have to be included in the model that will be fit, so it is a good idea to include all covariates that may be of interest.
`covariateTypes`	A concatenated character string specifying the type of variable for each covariate given in the covariates argument. The three options are "binary", "continuous", and "categorical". Every variable in the covariates argument must have a corresponding covariate type, where the nth covariate type corresponds to the nth covariate in the covariates argument.
`covariateLevels`	A concatenated character string specifying the level at which each covariates exists. The options are "subject", "sample", and "raster". As an example, for a study examining differences between tumor and non-tumor samples, the level would be 'sample' since the covariate changes between samples. A level must be provided for every covariate given in the argument covariates.
`trimData`	A TRUE/FALSE argument specifying whether or not the data should be trimmed. If trimData=TRUE, the data will be subset to only include the variables required to fit a model.

Value

A list containing the model and other information supplied to the `createPCData` function.

`data` A data frame containing the data

`nSubjs` The number of subjects

`nSamps` The number of samples

`cNSampsPerSubj` A cumulative vector of the number of samples per subject. If no subject variable was provided to the rScale function then this will be NULL.

`cNRastPerSamp` A cumulative vector of the number of rasters per sample.

`totalRasters` A numeric value for the total number of rasters.

`covs` A list of lists of covariate information. For each variable in the covariates argument, a list of covariate information is created that includes elements `covariate` (the name of covariate), `type` (the type of covariate given in the covariateTypes argument), `level` (the level of the covariate given in the covariateLevels argument), `info` (the data corresponding to the covariate, given as data frames for subject-level and sample-level covariates and vectors for raster-level covariates), and `mapping` (a data frame mapping the given covariate values to new ones that are used in modeling).

`KMat` A matrix of density values generated from the smoothing kernel function. The matrix has rows equal to the number of rows in the dataset and columns equal to the maximum number of support sites for any of the samples. Missing cells indicate that the support site corresponding to that column was removed for the sample corresponding to that data row.

`rastersPerVar` A data frame showing the number of rasters, per level of the spatial variable, for each sample. The data frame also shows the cumulative number of rasters. If no spatial variable is given then this is NULL.

`nSupportSites` The number of support sites per sample. If no spatial variable was given then this is a vector. If a spatial variable is given then this is a data frame showing the number of support sites per sample, for each level of the spatial variable.

`nObs` A data frame giving the number of rasters per sample.

`gT0SupportSites` If no spatial variable is provided, then this is NULL. If a spatial variable is given the this is a list of vectors, one for each level of the spatial variable, where each vector gives the samples numbers with more than one support site for the corresponding level of the spatial variable. If no spatial variable is given then this is NULL.

`nVarLevels` The number of levels of the spatial variable. If no spatial variable is given then this is NULL.

`subjectVar` A character string specifying the subject variable.

`sampleVar` A character string specifying the sample variable.

`spatialVar` A character string specifying the spatial variable.

`covariates` A concatenated character string of the covariate names.

`covariateTypes` A concatenated character string of the covariate types (binary, categorical, continuous) corresponding to the covariates.

`covariateLevels` A concatenated character string of the covariate levels (subject, sample, raster) corresponding to the covariates.

`outcome` A character string specifying the name of the variable to be modeled.

`recStructures` An indicator where 0 means no structure was chosen (if estimmated range=0) and 1 means a structure was chosen.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
```

```
PCdat <- createPCData(structs, trimData = FALSE,
                      covariates = c("secondary", "TAM", "secTAM"),
                      covariateTypes = c("binary", "binary", "binary"),
                      covariateLevels = c("sample", "raster", "raster"))
```

estRange *Estimate the range parameter*

Description

The `estRange` function is a wrapper for the `variog` and `variofit` functions in the geoR package. `estRange` calculates the semivariance at every observed distance within samples in the rescaled data and then fits those estimates to a covariance model.

Usage

```
estRange(rScaleObj, outcome, spatialVar = NULL, semivEst = "modulus",
  logTransform = TRUE, covarianceModel = "gaussian")
```

Arguments

`rScaleObj`	An object of class `rScaleList`. This object is the result of using the `rScale` function.
`outcome`	A character string specifying the outcome of interest. This is the variable that will later be modeled.
`spatialVar`	An optional character string specifying a binary or categorical variable. If a variable is input, the range will be estimated for all levels of that variable.
`semivEst`	The form of the semivariance estimator. The options are 'classical' and 'modulus'. The classical estimator is the method of moments etimator, and the modulus estimator is robust estimator from Hawkins and Cressie.
`logTransform`	A TRUE/FALSE variable indicating whether or not the outcome should be log-transformed. Imaging mass spectrometry data are typically lognormally distributed, and so the default is TRUE.
`covarianceModel`	A character string specifying the form of the covariance model. The only current option is 'gaussian'.

Value

A list of class `rangeList` containing the estimated range and other information supplied to the `estRange` function.

`data` A data frame containing the data.

`subjectVar` A character string denoting the subject variable.

`sampleVar` A character string denoting the sample variable.

`spatialVar` A character string denoting the spatial variable.

`outcome` A character string denoting the outcome of interest.

`estRange` A single value or vector of values representing the estimated range parameter. If no spatial variable is given then this will be a single value. If a spatial variable is given then this will be a vector of values, one for each level of the spatial variable. Each estimated range will be named for its corresponding spatial variable level.

`estSig2` A single value or vector of values representing the estimated variance parameter σ^2. The parameter σ^2 is used to calculate the covariance function. For more information, see the `cov.spatial` function in the `geoR` package. If no spatial variable is given then this will be a single value. If a spatial variable is given then this will be a vector of values, one for each level of the spatial variable. Each estimated range will be named for its corresponding spatial variable level.

`semivarFit` The empirical variogram. This is a result of a call to the `variog` function in the `geoR` package. If no spatial variable is provided, then this is a single object. If a spatial variable is provided, then this is a list of objects, one object for every level of the spatial variable.

`covModelFit` The fitted covariance model. This is a result of a call to the `variofit` function in the `geoR` package. If no spatial variable is provided, then this is a single object. If a spatial variable is provided, then this is a list of objects, one object for every level of the spatial variable.

References

Ribeiro, Jr., PJ and Diggle, PJ. 2018. geoR: Analysis of Geostatistical Data. R package version 1.7-2.1.

Cressie, N and Hawkins, DM. 1980. Robust estimation of the variogram: I. *Journal of the International Association for Mathematical Geology*, 12(2):115-125.

Examples

```
data("TAMdata")
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
```

`mergeIMSFiles` *Merge spot-level data and region spots from SCiLS*

Description

`mergeIMSFiles` merges the two files needed to create spot-level data with coordinates. The first is the spot-level data without coordinates, and the second is the region spots with coordinates. We will use the terms spots and rasters interchangeably. Though data are collected over circular areas dependent on the laser diameter, the rasterization process makes data representation easier.

Usage

```
mergeIMSFiles(spotData, regionSpots, trimTops = TRUE)
```

Arguments

`spotData`	A dataset containing the spot-level data in which the columns represent m/z values, and the rows represent spots.
`regionSpots`	A dataset containing the region spots. This should be an exported file that only contains the spot identifiers and the x- and y-coordinates.
`trimTops`	A TRUE/FALSE variable indicating whether or not the top rows of each dataset should be trimmed to remove metadata before merging. If metadata exists in the top rows then trimTops should be set to TRUE to remove it before proceeding to the next step. Otherwise the metadata will ineterfere with downstream steps.

Value

A merged data frame with spot-level data and corresponding coordinates.

`plot.PCResults` *Plot the MCMC iterations for each model coefficient.*

Description

`plot.PCResults` generates the MCMC iteration plots for model coefficients after running a PC model using `runPCModel`.

Usage

```
## S3 method for class 'PCResults'
plot(object)
```

Arguments

`object`	An object of class `PCResults`. This is the result of running a model using `runPCModel`.

Value

A plot of the MCMC chains for each model coefficient.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
PCdat <- createPCData(structs, trimData = FALSE,
                      covariates = c("secondary", "TAM", "secTAM"),
                      covariateTypes = c("binary", "binary", "binary"),
                      covariateLevels = c("sample", "raster", "raster"))
PCmod <- writePCModel(PCdat, multiSampsPerSubj = TRUE, typeOfZero = "censored")
PCresults <- runPCModel(modelObj = PCmod, PCDataObj = PCdat, slideVar='slide',
                        monitorCoefOnly = FALSE,
                        nBurnin = 15000, nIter = 40000, nThin = 25)
plot(PCresults)
```

plot.rangeList *Plot the semivariance estimates overlaid with the covariance model*

Description

`plot.rangeList` uses the S3 plot methods in the `geoR` package to plot the semivariance estimates and then overlay the fitted covariance model.

Usage

```
## S3 method for class 'rangeList'
plot(object)
```

Arguments

`object`	An object of class "rangeList".

Value

A plot or plots or the semivariance estimates overlaid with the fitted covariance model. If there was no spatial variable provided in the `estRange` function then only a single plot is produced. This is equivalent to the plots produced by `geoR`. However, if a spatial variable was provided, then `plot.rangeList` produces a plot for every level of that variable.

References

Cressie, N and Hawkins, DM. 1980. Robust estimation of the variogram: I. *Journal of the International Association for Mathematical Geology*, 12(2):115-125.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
plot(rangs)
```

plotIDs *Plot the assigned ID numbers*

Description

plotIDs plots the samples and overlays the sample IDs.

Usage

```
plotIDs(data, xCoord = "x", yCoord = "y", IDVar = "assignID",
  textSize = 4, TMA = FALSE)
```

Arguments

data	The dataset containing x- and y-coordinates and a sample identifier.
xCoord	A character string specifying the name of the x-coordinate variable.
yCoord	A character string specifying the name of the y-coordinate variable.
IDVar	A character string speciffying the sample ID variable.
textSize	The size of the sample IDs on the plot.
TMA	A TRUE/FALSE variable specifying whether or not the data came from a tissue microarray (TMA). This is only used to determine how to color the samples. If TMA=TRUE, then the samples are colored using a continuous color scale. If TMA=FALSE, then the samples are colored using a discrete color scale.

Value

A plot of the tissue samples colored according to the sample ID and overlaid with the ID.

Examples

```
data("TAMdata")
TAMdata <- assignIDs(TAMdata)
plotIDs(TAMdata)
```

plotPCStructure *Overlay the tissue samples with corresponding support structures*

Description

plotPCStructure plots the tissue sample rasters and overlays them with the support structure(s) used in downstream modeling.

Usage

```
plotPCStructure(structureObj, removeBackground = TRUE,
  marginProp = 0.1, titleSize = 11, supSiteSize = 3,
  supSiteLegendSize = 5, rastLegendSize = 30,
  rasterColors = c("gray75", "gray20"), supSiteColors = c("royalblue",
  "red"), plotCols = NULL)
```

Arguments

`structureObj`	An object of class `structureList`.
`removeBackground`	A TRUE/FALSE argument specifying if the background, including the x- and y-axes, should be removed.
`marginProp`	An argument specifying the size of the margins. It is a proportion of the largest distance from the origin to the outermost support sites. This is used to help separate plots.
`titleSize`	The font size of each title specifying the sample number.
`supSiteSize`	The size of the support point dots.
`supSiteLegendSize`	The size of the support sites in the legend.
`rastLegendSize`	The size of the rasters in the legend.
`rasterColors`	The color of the rasters. This is a concatenated character string that must have at least two elements. Each element specifies an accepted color in R. The `plotPCStructure` function uses the `colorRampPalette` function, which generates smooth transitions between the colors provided. If the number of colors provided is equal to the number of levels of the spatial variable, then the plotted colors will be the same as those provided. However, if the number of colors provided does not equal the number of levels of the spatial variable, then the plotted colors will be interpolated from those provided.
`supSiteColors`	The color of the support sites. This is a concatenated character string that must have at least two elements. The colors are generated in the same way as the `rasterColors`.

Value

A gridded plot of all tissue samples. If a spatial variable was provided in the `estRange` function, then each sample will be colored by the levels of that variable. Using a separate set of colors, the support sites will also be colored according to the levels of the spatial variable. If no spatial variable was provided, then both the sample rasters and the support sites will each be represented by a single color. The last cell in the grid of plots is the legend, showing which colors are used to represent the rasters and support sites.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
plotPCStructure(structs, supSiteSize = 1.5, marginProp = 0.25)
```

plotRangeObj — *Plot a range object.*

Description

plotIDs the semivariance estimates, covariance model, and number of pairs of observations used to estimate semivariance.

Usage

```
plotRangeObj(rangeObj)
```

Arguments

rangeObj	An object of class rangeList. This object is a list that contains the estimated range parameter(s). This is obtained by a call to the estRange function.

Value

A set of plots displaying the semivariance estimates, fitted covariance model, and number of pairs of observations used to estimate semivariance. If no spatial variable was provided to the estRange function, then there will be two plots. The first, on top, will show the semivariance estimates and the fitted covariance model. The second, on the bottom, will show the number of data pairs used to the estimate the semivariance at the corresponding distance. If a spatial variable is supplied to the estRange function, then there will be two plots for every level of the spatial variable.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
plotRangeObj(rangs)
```

rScale — *Rescale the data*

Description

rScale rescales the coordinates of the data so that the distance between observations is 1. It also centers the set of coordinates for each sample at the origin.

Usage

```
rScale(data, subjectVar = NULL, sampleVar, xCoord, yCoord)
```

Arguments

`data`	The dataset to be rescaled. The dataset must be a data frame.
`subjectVar`	A character string specifying the subject variable, if it exists. This should be specified for paired data only, meaning data in which there is more than one sample for at least one subject.
`sampleVar`	A character string specifying the sample variable. This should be a variable with unique values for each sample.
`xCoord`	A character string specifying the name of the x-coordinate variable.
`yCoord`	A character string specifying the name of the y-coordinate variable.

Value

A list including the rescaled dataset and the names of the subject and sample variables. The subject and sample variables are carried forward to subsequent functions so that they only have to be input once.

`data` The dataset, appended with new x- and y-coordinates to be used in further calculations.

`subjectVar` A character string denoting the subject variable.

`sampleVar` A character string denoting the sample variable.

Examples

```
data("TAMdata")
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
```

`runPCModel` *Run the model*

Description

The `runPCModel` function runs the model created by the `writePCModel` function, using the information generated by the `createPCData` function.

Usage

```
runPCModel(modelObj, PCDataObj, nIter = 50000, nBurnin = 10000,
  nThin = 5, slideVar = NULL, keepModel = TRUE,
  keepModelObj = FALSE, monitorCoefOnly = TRUE, returnSample = TRUE,
  returnAllSummary = FALSE, justReturnData = FALSE,
  convergenceTol = 1.1, maxRuns = 5, iterIncrement = 500,
  iterWindow = 1000, sliceSamplers = FALSE)
```

Arguments

`modelObj`	An object of class `PCModelList`. This object is the result of using the `writePCModel` function.
`PCDataObj`	An object of class `PCDataList`. This object is the result of using the `createPCData` function.
`nIter`	An integer specifying he total number of iterations for the MCMC, pre-thinning. This includes the burn-in.
`nBurnin`	An integer specifying the number of iterations (pre-thinning) to be discarded.
`nThin`	An integer specifying the thinning interval.
`slideVar`	An optional argument specifying a slide variable. This option is only useful for data with zeros and when those zeros are assumed to be censored observations. In this case, the censoring threshold can vary by each run of the mass spectrometer. When this argument is used, the censoring threshold will be allowed to vary by slide.
`keepModel`	A TRUE/FALSE argument specifying whether or not to keep the model (found in the modelObj).
`keepModelObj`	A TRUE/FALSE argument specifying whether the compiled NIMBLE model object should be kept after running the model. This object can be large, but it is required for restarting the MCMC.
`monitorCoefOnly`	A TRUE/FALSE argument stating whether to monitor the the model coefficients only (TRUE) or all stochastic nodes (FALSE). Regardless of the value of this argument, if any categorical covariates are included in the model, the differences between all subgroups of the categorical covariate(s) will be monitored in the MCMC with assigned nodes.
`returnSample`	A TRUE/FALSE argument specifying whether to return the entire MCMC sample. If this argument is FALSE, only the summary measures will be returned.
`returnAllSummary`	A TRUE/FALSE argument specifying whether to return summary measures for all monitored nodes.
`justReturnData`	A TRUE/FALSE argument. If justReturnData=TRUE, then no MCMC will be run. Instead, the data and constants lists required to run the model will be returned. This is useful for modifying the model to personal specifications.
`convergenceTol`	A numeric value indicating the highest acceptable value for the Brooks-Gelman-Rubin (BGR) statistic to consider model nodes converged.
`maxRuns`	A numeric value indicating the maximum number of times a model will be restarted to try to achieve convergence. After running the MCMC for nIter iterations, the model is checked for convergence. A model is considered converged if the model coefficients converge according to the BGR statistic. For the marginalized two-part model, only the coefficients for the marginalized part are used. If the model does not converge, then the MCMC will be restarted and run for another iterIncrement iterations (post-thinning). Convergence will then be rechecked. The maxRuns argument specifies how many times the MCMC can be restarted after the initial run of nIter iterations.
`iterWindow`	A numeric value specifying the post-thinning number of iterations (per chain) that will be used to make inference, if the model does not converge with the first nIter iterations.

`sliceSamplers` A TRUE/FALSE argument specifying whether or not to use slice samplers for all stochastic nodes. If `sliceSamplers=TRUE` then NIMBLE's onlySlice option in the `configureMCMC` is set to TRUE. If `sliceSamplers=FALSE` then NIMBLE's default MCMC settings are used. Using slice samplers is useful for reducing autocorrelation in the MCMC, though it significantly increases computation time.

Value

A list of class `PCResults` containing the results of running the MCMC. The list will contain additional objects and information specified by the arguments.

- `results` A matrix containing summary statistics (mean and the 2.5 coefficient. For ease of reading, the names of the model coefficients (beta1, beta2,...) are replaced with the names of their corresponding covariates.
- `model` The model specified from the `writePCModel` function. The model is output if `keepModel=TRUE`.
- `modelObj` The compiled NIMBLE model object after running until covergence or maxRuns is reached. This object is output only if `keepModelObj=TRUE`. This object can be large (>100MB) for even small imaging mass spectrometry datasets, so some thought should be put into deciding whether or not to keep this object.
- `fullSummary` A matrix of summary information for all nodes in the model, This will be returned if `returnAllSummary=TRUE`.
- `sample` An `mcmc.list` object. See the `coda` package for more details on such objects. This is a list of two matrices, one for each chain. Each matrix has rows equal to the number of thinned iterations and columns equal to the number of nodes monitored. The sample is returned if `returnSample=TRUE`.
- `dataList` The data list used to create a compiled NIMBLE model. For all data this list will include the matrix generated from the smoothing kernel function. For data with zeros (censored and true), the model is specified using the zeros trick. In this case `dataList` also has a vector of zeros.
- `constantsList` The list of constants used to create a compiled NIMBLE model. This list is typically composed of covariates and other vectors needed to navigate the covariate information. However, for data with zeros (censored and true), the model is specified using the zeros trick, and so the outcome data is also included in this list.
- `convergenceTol` The limit on the BGR statistic used to determine convergence. The default is 1.10, so if the BGR values for all model coefficients are <=1.10, then the model is considered converged.

References

de Valpine, P., D. Turek, C.J. Paciorek, C. Anderson-Bergman, D. Temple Lang, and R. Bodik. 2017. Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*. 26: 403-413.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
```

```
structs <- chooseStructures(rangs)
PCdat <- createPCData(structs, trimData = FALSE,
                      covariates = c("secondary", "TAM", "secTAM"),
                      covariateTypes = c("binary", "binary", "binary"),
                      covariateLevels = c("sample", "raster", "raster"))
PCmod <- writePCModel(PCdat, multiSampsPerSubj = TRUE, typeOfZero = "censored")
PCresults <- runPCModel(modelObj = PCmod, PCDataObj = PCdat, slideVar='slide',
                        monitorCoefOnly = FALSE,
                        nBurnin = 15000, nIter = 40000, nThin = 25)
```

---

summary.PCResults *Summarize PCResults object*

Description

summary.PCResults provides basic results for an object of class "PCResults".

Usage

```
## S3 method for class 'PCResults'
summary(object)
```

Arguments

object	An object of class "PCResults". This is the result of a call to the runPCModel function.

Value

The summary.PCResults function prints the matrix of summary measures labeled "results" in the object. This matrix contains summary information for the model coefficients.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
PCdat <- createPCData(structs, trimData = FALSE,
                      covariates = c("secondary", "TAM", "secTAM"),
                      covariateTypes = c("binary", "binary", "binary"),
                      covariateLevels = c("sample", "raster", "raster"))
PCmod <- writePCModel(PCdat, multiSampsPerSubj = TRUE, typeOfZero = "censored")
PCresults <- runPCModel(modelObj = PCmod, PCDataObj = PCdat, slideVar='slide',
                        monitorCoefOnly = FALSE,
                        nBurnin = 15000, nIter = 40000, nThin = 25)
summary(PCresults)
```

`summary.rangeList` *Summarize rangeList object*

Description

`summary.rangeList` provides basic results for an object of class `rangeList`.

Usage

```
## S3 method for class 'rangeList'
summary(object)
```

Arguments

`object`	An object of class `rangeList`. This is the result of a call to the `estRange` function.

Value

The `summary.rangeList` prints the estimated range for every level of the spatial variable provided to the `estRange` function. If no spatial varaible was provided, then only one estimated range is printed.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
summary(rangs)
```

`summary.structureList` *Summarize structureList object*

Description

`summary.structureList` provides basic results for an object of class `structureList`.

Usage

```
## S3 method for class 'structureList'
summary(object)
```

Arguments

`object`	An object of class `structureList`. This is the result of a call to the `chooseStructures` function.

Value

For every level a spatial variable (provided in the `estRange` function), the `summary.rangeList` prints the chosen support structure, the total number of support points across all samples after removing support points, and the number of samples. If no spatial variable was provided, only one line of information is printed.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
summary(structs)
```

`TAMdata` *N-glycan abundance measures in primary mammary and secondary lung tumor tissues.*

Description

A dataset containing area under the curve (AUC) abundance measures for 50 N-glycans.

Usage

`TAMdata`

Format

A dataframe with 18,154 rows and 59 columns.

Spot.index A numeric identifier for each spot or raster.

x The x-coordinate of each spot.

y The y-coordinate of each spot.

subject A numeric subject identifier.

ROI A numeric identifier for the region of interest (ROI)

slide A numeric identifier indicating samples that were run on the same slide.

secondary A sample-level binary variable that takes on a value of 0 if the sample was resected from primary mammary tissue and a value of 1 if the sample was resected from secondary lung tissue.

TAM A spot-level binary variable indicating whether the spot stained postive (TAM=1) or negative (TAM=0) for tumor-associated macrophages (TAMs).

meanTAM A spot-level numeric variable on a 0-1 scale that indicates the proportion of pixels (from a corresponding scanned image) within the bounds of the corresponding spot that stained positive for TAMs.

X771.auc The relative abundance, as measured by AUC, of a fragment with an m/z of approximately 771.

X__.auc The relative abundance, as measured by AUC, of a fragment with an m/z given by __.

writePCModel *Write the model*

Description

The `writePCModel` function writes a NIMBLE model that incorporates the process convolution approach.

Usage

```
writePCModel(PCDataObj, typeOfZero = "censored", covariates = NULL,
  covariatesForBinary = NULL, coefPrior = "sdunif",
  multiSampsPerSubj = NULL, errorVarianceLevel = "sample",
  latentVarianceLevel = "sample", addAssignNode = NULL,
  assignNodeLevel = NULL)
```

Arguments

`PCDataObj`	An object of class `PCDataList`. This object is the result of using the `createPCData` function.
`typeOfZero`	A character string specifying the type of zero that will be assumed. The two options are "censored" (default) and "true". If there are no zeros for the outcome of interest then this argument is superfluous.
`covariates`	A concatenated character string specifying the covariates that will be included in the model. When there are no zeros in the data, or when zeros are assumed to be censored, there is only a single set of covariates. When the zeros are assumed to be true zeros, there are two model parts, a binary and a marginalized. The covariates specified in this argument will be used in the marginalized part of the model.
`covariatesForBinary`	A concatenated character string specifying covariates that will be included in the binary part of the marginalized two part model. This argument will only be used if there are zeros in the data and `typeOfZero='true'`. If `typeOfZero='true'`, and this argument is left empty, then the covariates for the binary part of the model will be assumed to be the same as those for the marginalized part of the model.
`coefPrior`	A character string specifying the prior that should be assumed for model coefficients. The options are "sdunif" (default), "dnorm", and "Cauchy". See details for more information.
`multiSampsPerSubj`	A TRUE/FALSE variable indicating whether or not there are mutliple samples per subject.
`errorVarianceLevel`	A character string specifying the level at which the error variance should be estimated. The options are "sample" (default), "subject", and "overall". As an example, if `errorVarianceLevel="sample"` then a separate error variance is estimated for each sample.
`latentVarianceLevel`	A character string specifying the level at which the latent variance should be estimated. The options are "sample" (default), "subject", and "overall".

Details

For the model coefficients, three possible priors are allowed.

sdunif For all model coefficients, including the intercept(s), the prior is given by

$$\beta \sim \mathrm{N}(0, \sigma_\beta^2); \sigma_\beta \sim \mathrm{U}(0, 10).$$

dnorm For all model coefficients, including the intercept(s), the prior is given by

$$\beta \sim \mathrm{N}(0, 0.00001).$$

Cauchy The Cauchy priors are specified according to the recommendations of Gelman et al. (input citation). While Gelman used the Cauchy priors in the context of logistic regression, imaging mass spectrometry data is often lognormally distributed, and on the log scale parameter spaces may need to be constrained in a similar way. For intercepts, the prior is

$$\beta_0 \sim \mathrm{Cauchy}(0, \mathrm{scale} = 10).$$

For other model coefficients the prior is

$$\beta \sim \mathrm{Cauchy}(0, \mathrm{scale} = 2.5).$$

Value

A list containing the model and other information supplied to the `createPCData` function.

`model` A model object created using the `link[nimble]{nimbleCode}` function.

`covariates` A concatenated character string of the covariate names.

`covariatesForBinary` A concatenated character string of the covariate names for the binary part of the marginalized two-part model.

`coefPrior` A character string specifying the assumed prior for the model coefficients.

`multiSampsPerSubj` A TRUE/FALSE variable indicating whether there are multiple samples per subject.

`errorVarianceLevel` A character string specifying the level (overall, subject, or sample) at which the error variance will be estimated.

`latentVarianceLevel` A character string specifying the level (overall, subject, or sample) at which the latent variance will be estimated.

`typeOfZero` A character string specifying the assumption made about the type of zeros in the data.

References

de Valpine, P., D. Turek, C.J. Paciorek, C. Anderson-Bergman, D. Temple Lang, and R. Bodik. 2017. Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*. 26: 403-413.

Examples

```
data("TAMdata")
# The dataset is trimmed only for the speed of the example
TAMdata <- TAMdata[TAMdata$subject < 3, ]
TAMdata <- rScale(TAMdata, subjectVar = 'subject', sampleVar = 'ROI',
                  xCoord = 'x', yCoord = 'y')
rangs <- estRange(TAMdata, outcome = 'X1282.auc', spatialVar = 'TAM',
                  semivEst = 'modulus', logTransform = TRUE)
structs <- chooseStructures(rangs)
PCdat <- createPCData(structs, trimData = FALSE,
                      covariates = c("secondary", "TAM", "secTAM"),
                      covariateTypes = c("binary", "binary", "binary"),
                      covariateLevels = c("sample", "raster", "raster"))
PCmod <- writePCModel(PCdat, multiSampsPerSubj = TRUE, typeOfZero = "censored")
```

Index