Package 'BayesX'

August 24, 2019

Type Package

Title R Utilities Accompanying the Software Package BayesX

Version 0.3-1

Date 2019-08-23

Description Functions for exploring and visualising estimation results
obtained with BayesX, a free software for estimating structured additive
regression models (<http://www.BayesX.org>). In addition,
functions that allow to read, write and manipulate map objects that are required in spatial
analyses performed with BayesX.

Depends shapefiles

Imports sp, maptools, colorspace, coda, splines, methods

Suggests spdep, akima, gpclib, rgeos

LazyLoad yes

License GPL-2 | GPL-3

NeedsCompilation no

Author Nikolaus Umlauf [aut, cre],
Thomas Kneib [aut],
Nadja Klein [aut],
Felix Heinzl [ctb],
Andreas Brezger [ctb],
Daniel Sabanes Bove [ctb]

Maintainer Nikolaus Umlauf <Nikolaus.Umlauf@uibk.ac.at>

Repository CRAN

Date/Publication 2019-08-24 07:50:02 UTC

R topics documented:

`BayesX-package` *R Utilities Accompanying the Software Package BayesX*

Description

This package provides functionality for exploring and visualising estimation results obtained with the software package **BayesX** for structured additive regression. It also provides functions that allow to read, write and manipulate map objects that are required in spatial analyses performed with **BayesX**.

Author(s)

Nikolaus Umlauf, Thomas Kneib, Nadja Klein, Felix Heinzl, Andreas Brezger, Daniel Sabanes Bove

References

Belitz C, Brezger A, Kneib T, Lang S (2011). **BayesX** - Software for Bayesian Inference in Structured Additive Regression Models. Version 2.0.1. URL `http://www.BayesX.org`.

add.neighbor *Add Neighborhood Relations*

Description

Adds a neighborhhod relationship between two given regions to a map object in graph format.

Usage

```
add.neighbor(map, region1, region2)
```

Arguments

map	Map object in graph format that should be modified.
region1, region2	Names of the regions that should be connected as neighbors.

Value

Returns an adjacency matrix that represents the neighborhood structure of map plus the new neighborhood relation in graph format.

Author(s)

Felix Heinzl, Thomas Kneib

See Also

get.neighbor,delete.neighbor,read.gra,write.grabnd2gra

Examples

```
germany <- read.gra(system.file("examples/germany.gra", package="BayesX"))
get.neighbor(germany, c("1001", "7339"))
germany <- add.neighbor(germany, "7339", "1001")
get.neighbor(germany, c("1001", "7339"))
```

bnd2gra *Convert Boundary Format to Graph Format*

Description

Converts a map in boundary format to a map in graph format.

Usage

```
bnd2gra(map)
```

Arguments

map	Map in boundary format that should be converted.

Value

Returns an adjacency matrix that represents the neighborhood structure of the map object in graph format.

Author(s)

Felix Heinzl, Thomas Kneib

References

BayesX Reference Manual. Available from http://www.bayesx.org

See Also

read.bnd,read.gra,write.bnd,write.gra

Examples

```
tanzania.bnd <- read.bnd(system.file("examples/tanzania.bnd", package="BayesX"))
tanzania.gra <- bnd2gra(tanzania.bnd)
```

createxymap *Create map objects for some points and a given distance*

Description

Creates a map object from a list of coordinates by treating observations within a certain distance as neighbors. The resulting neighborhood structure is stored in a map object in graph format while a map in boundary format is created to enable visualisation.

Usage

```
createxymap(x, y, districts=NULL, p=2, max.dist)
```

Arguments

x	Vector of x-coordinates of underlying points
y	Vector of y-coordinates of underlying points
districts	Either NULL or a vector of names for labeling points. If districts=NULL, points are labelled by index.
p	Any p-norm with p>=1 can be chosen as the distance measure with the Euclidian distance (p=2) being the default. For p=Inf, the maximum of coordinates is used. Except for p=Inf, calculations can fail for huge p.
max.dist	Value which determines the neighborship. Points whose distance is smaller or equal than max.dist are considered as neighbors.

Value

List of two elements: map object in graph format and map object in boundary format.

Author(s)

Felix Heinzl, Thomas Kneib

See Also

read.gra,read.bnd,drawmap

Examples

```
x <- c(3,3,5,9.5,11,11)
y <- c(2,4,4,6,4.5,5)
xymap <- createxymap(x, y, districts=c("A","B","C","D","E","F"), max.dist=2)
xymap$gra
drawmap(map=xymap$bnd)
```

`delete.neighbor` *Delete Neighborhood Relations*

Description

Adds the neighborhhod relationship between two given regions from a map object in graph format.

Usage

```
delete.neighbor(map, region1, region2)
```

Arguments

`map`	Map object in graph format that should be modified.
`region1, region2`	Names of the regions that should no longer be regarded as neighbors.

Value

Returns an adjacency matrix that represents the neighborhood structure of `map` minus the deleted neighborhood relation in graph format.

Author(s)

Felix Heinzl, Thomas Kneib

See Also

`get.neighbor`,`add.neighbor`,`read.gra`,`write.grabnd2gra`

Examples

```
germany <- read.gra(system.file("examples/germany.gra", package="BayesX"))
get.neighbor(germany, c("7339"))
germany <- delete.neighbor(germany, "7339", "7141")
get.neighbor(germany, c("7339"))
```

drawmap *Drawing Geographical Information*

Description

Visualises variables that are spatially aligned according to a given map object. Each of the regions in a map will be coloured accoring to the value of the variable.

Usage

```
drawmap(data, map, regionvar=2, plotvar=3, limits, cols="hcl", nrcolors=100,
        swapcolors=FALSE, pcat=FALSE,
        hcl.par=list(h=c(120, 0), c=60, l=c(45,60), power=1.2),
        hsv.par=list(s=1, v=1), legend=TRUE, drawnames=FALSE, cex.names=0.7,
        cex.legend=0.7, mar.min=2, density=15, ...)
```

Arguments

data	Either the name of a file or a data frame containing the variables to be visualised. If missing, the map will be visualised without superposition of any further information
map	Map object containing the required boundary information (as obtained by a call to read.bnd
regionvar	Defines the variable specifying the geographical regions. Either the name of a variable in data or the index of the corresponding column.
plotvar	Defines the variable that should be visualised. Either the name of a variable in data or the index of the corresponding column.
limits	Restricts (or extends) the coloring scheme to a range of values.
cols	Color scheme to be employed. Could be either a vector of colors or one out of the following pre-defined schemes: hcl, hsv, grey
nrcolors	Number of colors (only meaningful when using one of the pre-defined colour schemes).
swapcolors	Reverse the order of colors (works also with user-specified colours but will be most usefule with the pre-defined schemes).
pcat	Option for the visualisation of posterior probabilities. In this case, a three-colour scheme representing significantly positive, insignificant and significantly negative values.
hcl.par	Parameters for the hcl colour scheme (see the documentation of diverge_hcl in package vcd for details).
hsv.par	Parameters for the hsv colour scheme (see the documentation of hsv for details).
legend	Should a legend be added to the figure?
drawnames	Adds the name of each region as a text label to the plot. In most cases the result will be confusing but may be useful when checking the validity of a map.

cex.names	Magnification to be used for the names (if drawnames=TRUE).
cex.legend	Magnification to be used for the legend.
mar.min	Controls the definition of boundaries. Could be either NULL for individual settings of mar or a value which defines mar as follows: The boundaries will be calculated according to the height to width ratio of the map with minimal boundary mar.min.
density	Regions without data will be visualised with diagonal stripes. density defines how dense the stripes should be.
...	Further arguments to be passed to the plot calls that visualise the region boundaries (probably not useful at all).

Author(s)

Felix Heinzl, Thomas Kneib, Andreas Brezger

See Also

read.bnd

Examples

```
germany <- read.bnd(system.file("examples/germany.bnd", package="BayesX"))
drawmap(map=germany)
drawmap(map=germany, drawnames=TRUE)

res <- read.table(system.file("examples/spatial_f_regions_spatial.res",
                              package="BayesX"), header=TRUE)

drawmap(res, map=germany)
drawmap(res, map=germany, limits=c(-2,4))
drawmap(res, map=germany, regionvar="regions", plotvar="pmed")
drawmap(res, map=germany, legend=FALSE)
drawmap(res, map=germany, legend=FALSE, main="spatial effect")

drawmap(res, map=germany, cols="hsv")
drawmap(res, map=germany, swapcolors=TRUE, cols="hsv")
drawmap(res, map=germany, cols="grey")
drawmap(res, map=germany,
        cols=c('darkgreen','green','yellow','orange','red','darkred'))

drawmap(res, map=germany, pcat=TRUE, cols="hcl")
drawmap(res, map=germany, pcat=TRUE, cols="hsv")
drawmap(res, map=germany, pcat=TRUE, cols="grey")

drawmap(res, map=germany, nrcolors=10, cols="hcl")
drawmap(res, map=germany, nrcolors=10, cols="hsv")
drawmap(res, map=germany, nrcolors=10, cols="grey")

drawmap(res, map=germany, cols="hcl",
        hcl.par=list(h=c(0,120), c=60, l=c(45,90), power=1.2))
```

```
drawmap(res, map=germany, cols="hcl",
        hcl.par=list(h=c(300,120), c=60, l=c(45,90), power=1.2))
drawmap(res, map=germany, cols="hcl",
        hcl.par=list(h=c(40,260), c=60, l=c(45,90), power=1.2))
drawmap(res, map=germany, cols="hsv", hsv.par=list(s=0.7, v=0.7))
```

`extractSamples` *Extract MCMC samples from a BayesX results directory*

Description

This is a convenience function to extract samples from a BayesX results directory, which processes the log file to e.g. convert the spline coefficients samples to function values samples.

Usage

```
extractSamples(directoryWithBasename,
               logfile = file.path(dirname(directoryWithBasename), "log.txt"))
```

Arguments

`directoryWithBasename`	The BayesX results directory with basename for the files (e.g. "results/test", if this was specified as outfile in BayesX for the bayesreg object)
`logfile`	The log file of the MCMC run, defaults to log.txt in the results directory.

Value

Returns a list with the extracted samples of effects and deviances as well as the prediction data.frame:

`<function name>`	for P-Splines, Random Walks and spatial effects: a list with mcmc objects 'functionSamples' and 'varianceSamples' containing the respective effects/function and variance parameter samples.
`FixedEffects`	an mcmc object of all fixed simple parametric effects
`RandomEffects`	if there is at least one random effect in the model, this is a list, with elements in the first hierarchy being the group ID names, and elements in the second hierarchy being the names of the covariates. The leafs are the mcmc objects 'functionSamples' and 'varianceSamples', as for the other non-fixed terms
`Deviance`	an mcmc object with the (unstandardized and saturated) deviance
`means`	if the option `predictmu` was used, this mcmc object contains the mean samples
`scale`	an mcmc object with the possible scale parameter samples
`lassoCoefficients`	an mcmc object with the possible lasso regression parameter samples
`ridgeCoefficients`	an mcmc object with the possible ridge regression parameter samples

`PredictMeans` data.frame corresponding to the possible predictmean file in the BayesX directory

Additionally, entries for possibly remaining lasso or ridge variance parameters etc. are included in the return list.

Warning

You should be sure that only one MCMC run is saved in the given results directory in order to get sensible results out of this function.

Author(s)

Daniel Sabanes Bove, with contributions by Fabian Scheipl

Examples

```
## get the samples
samples <- extractSamples(file.path(system.file("examples/samples", package="BayesX"),
                                    "res"))
str(samples)

## check deviance convergence
plot(samples$Deviance)

## fixed parametric effects
plot(samples$FixedEffects)

## nonparametric effects:

## handy plot function to get means and pointwise credible intervals
nonpPlot <- function(samplesMatrix,
                     ...)
{
    x <- as.numeric(colnames(samplesMatrix))

    yMeans <- colMeans(samplesMatrix)
    yCredible <- t(apply(samplesMatrix,
                         MARGIN=2,
                         FUN=quantile,
                         prob=c(0.025, 0.975),
                         na.rm=TRUE))

    matplot(x, cbind(yMeans, yCredible),
            type="l",
            lty=c(1, 2, 2),
            lwd=c(2, 1, 1),
            col=c(1, 2, 2),
            ...)
}

nonpPlot(samples$f_x1$functionSamples,
```

```
         xlab=expression(x[1]),
         ylab=expression(hat(f)(x[1])))
nonpPlot(samples$f_x2$functionSamples,
         xlab=expression(x[2]),
         ylab=expression(hat(f)(x[2])))

## spatial effect
tanzania <- read.bnd(file=system.file("examples/tanzania.bnd", package="BayesX"))
drawmap(map=tanzania,
        data=
        with(samples$f_district,
             data.frame(name=colnames(functionSamples),
                        estimate=colMeans(functionSamples))),
        regionvar="name",
        plotvar="estimate")
```

---

fuse *Combine Regions*

Description

Combines a list of several regions of a map object in boundary format into a single region.

Usage

```
fuse(map, regions, name)
```

Arguments

map	Map object in boundary format that should be modified.
regions	Vector of regions to be combined
name	Name that should be given to the region arising from fusing the specified regions.

Value

Map object in boundary format with the specified regions combined.

Author(s)

Nadja Klein

See Also

read.bnd,write.bnd

Examples

```
## Not run: require("gpclib")
library("maptools")
gpclibPermit()
map <- read.bnd(system.file("examples/germany9301.bnd", package="BayesX"))
drawmap(map=map, drawnames=TRUE)

#vector of regions to be combined
regions <- c("1056","1060","1061")
#new name of combined region
newname <- "1"
newmap <- fuse(map,regions,newname)
drawmap(map=newmap,drawnames=TRUE)

#vector of regions to be combined
germany <- read.bnd(system.file("examples/germany.bnd", package="BayesX"))
drawmap(map=germany, drawnames=TRUE)
regions <- c("9371","9373","9374","9471","9472","9474","9574")
#new name of combined region
newname <- "1"
newmap <- fuse(germany,regions,newname)
drawmap(map=newmap,drawnames=TRUE)
## End(Not run)
```

`get.centroids` *Compute Centroids of Polygons*

Description

Computes all areas and centroids of the regions of a given map in boundary format.

Usage

```
get.centroids(map)
```

Arguments

`map`	Map object in boundary format.

Value

Matrix of area and centroids.

Author(s)

Felix Heinzl, Thomas Kneib

Examples

```
germany <- read.bnd(system.file("examples/germany.bnd", package="BayesX"))
centroids <- get.centroids(germany)
centroids[1:10,]

plot(c(2100,3700),c(6800,8500),type="n", xlab="", ylab="")
for(i in 1:10){
   polygon(germany[[i]])
   region <- attr(germany,"names")[i]
   text(x=centroids[i,2]+50, y=centroids[i,3]+30, region, cex=0.7)
}
points(centroids[1:10,2:3], col='red', pch=16)
```

---

get.neighbor *Obtain Neighbors of Given Regions*

Description

Extracts the neighbors of a number of regions from a map in graph format.

Usage

```
get.neighbor(map, regions)
```

Arguments

map	Map object in graph format.
regions	Vector of names of regions for which the neighbors should be axtracted.

Value

A list of vectors containing the neighbors of the elements in regions.

Author(s)

Felix Heinzl, Thomas Kneib

See Also

add.neighbor,delete.neighbor

Examples

```
germany <- read.gra(system.file("examples/germany.gra", package="BayesX"))
get.neighbor(germany, "1001")
get.neighbor(germany, c("1001", "7339"))
```

hpd *Computing Highest Posterior Density (HPD) Intervals*

Description

Compute approximate HPD intervals out of MCMC-samples in BayesX

Usage

```
hpd(data, alpha = 0.05, ...)
hpd.coda(data, alpha = 0.05)
```

Arguments

data	Either the name of a file or a data frame containing the sample.
alpha	A numeric scalar in the interval (0,1) such that 1 - alpha is the target probability content of the intervals.. The default is alpha = 0.05.
...	Further parameters to be passed to the internal call of optim and integrate.

Details

hpd computes the HPD interval based on a kernel density estimate of the samples. hpd.coda computes the HPD interval with the function HPDinterval available in package coda.

Author(s)

Nadja Klein

Examples

```
res <- read.table(system.file("examples/nonparametric_f_x_pspline_sample.raw",
  package="BayesX"), header = TRUE)
hpd(res)
hpd.coda(res)
```

Interface between nb and gra format *Convert nb and gra format into each other*

Description

Convert neighborhood structure objects of class "nb" from R-package spdep to graph objects of class "gra" from R-package BayesX and vice versa.

Usage

```
nb2gra(nbObject)
gra2nb(graObject)
```

Arguments

`nbObject`	neighborhood structure object of class "nb"
`graObject`	graph object of class "gra"

Value

Equivalent object in the other format.

Author(s)

Daniel Sabanes Bove

See Also

`sp2bnd`, `bnd2sp` for conversion between the geographical information formats and `read.gra`, `write.gra` for the interface to the BayesX files.

Examples

```
## first nb to gra:
library(spdep)
library(maptools)
columbus <- readShapePoly(system.file("etc/shapes/columbus.shp", package="spdep")[1])
colNb <- poly2nb(columbus)
## ... here manual editing is possible ...
## then export to graph format
colGra <- nb2gra(colNb)

## and save in BayesX file
graFile <- tempfile()
write.gra(colGra, file=graFile)

## now back from gra to nb:
colGra <- read.gra(graFile)
newColNb <- gra2nb(colGra)
newColNb
## compare this with the original
colNb
## only the call attribute does not match (which is OK):
all.equal(newColNb, colNb,
          check.attributes=FALSE)
attr(newColNb, "call")
attr(colNb, "call")
```

`Interface between sp and bnd format`

Convert sp and bnd format into each other

Description

Convert geographical information objects of class "SpatialPolygons" (or specializations) from R-package sp to objects of class "bnd" from R-package BayesX and vice versa.

Usage

```
sp2bnd(spObject, regionNames, height2width, epsilon)
bnd2sp(bndObject)
```

Arguments

`spObject`	object of class "SpatialPolygons" (or specializations)
`regionNames`	character vector of region names (parallel to the Polygons list in spObject), defaults to the IDs
`height2width`	ratio of total height to width, defaults to the bounding box values
`epsilon`	how much can two polygons differ (in maximum squared Euclidean distance) and still match each other?, defaults to machine precision
`bndObject`	object of class "bnd"

Value

Equivalent object in the other format.

Author(s)

Daniel Sabanes Bove

See Also

`nb2gra`, `gra2nb` for conversion between the neighborhood structure formats and `read.bnd`, `write.bnd` for the interface to the BayesX files.

Examples

```
## bnd to sp:
germany <- read.bnd(system.file("examples/germany2001.bnd", package="BayesX"))
spGermany <- bnd2sp(germany)

## plot the result together with the neighborhood graph
library(sp)
plot(spGermany)
library(spdep)
```

```
nbGermany <- poly2nb(spGermany)
plot(nbGermany, coords=coordinates(spGermany), add=TRUE)

## example with one region inside another
spExample <- spGermany[c("7211", "7235"), ]
plot(spExample)
plot(poly2nb(spExample), coords=coordinates(spExample), add=TRUE)

## now back from sp to bnd:
bndGermany <- sp2bnd(spGermany)
drawmap(map=bndGermany)

## compare names and number of polygons
stopifnot(identical(names(bndGermany),
                    names(germany)),
          identical(length(bndGermany),
                    length(germany)))

## compare contains-relations
surrounding <- attr(bndGermany, "surrounding")
whichInner <- which(sapply(surrounding, length) > 0L)
bndContainsData <- data.frame(inner=names(bndGermany)[whichInner],
                              outer=unlist(surrounding))

surrounding <- attr(germany, "surrounding")
whichInner <- which(sapply(surrounding, length) > 0L)
originalContainsData <- data.frame(inner=names(germany)[whichInner],
                                   outer=unlist(surrounding))

stopifnot(all(bndContainsData[order(bndContainsData$inner), ] ==
              originalContainsData[order(originalContainsData$inner), ]))
```

`plotautocor` *Computing and Plotting Autocorrelation Functions*

Description

Computes and plot autocorrelation functions for samples obtained with MCMC in BayesX

Usage

```
plotautocor(data, ask = TRUE, lag.max=100, ...)
```

Arguments

`data`	Either the name of a file or a data frame containing the sample.
`ask`	`plotautocor` will plot separate autocorrelation functions for each parameter. If `ask=TRUE`, the user will be prompted before showing the next plot.
`lag.max`	Maximum number of lags to be considered.
`...`	Further parameters to be passed to the internal call of `plot` such as `ylim`, etc.

Author(s)

Felix Heinzl, Thomas Kneib

Examples

```
res <- read.table(system.file("examples/nonparametric_f_x_pspline_sample.raw",
                              package="BayesX"), header=TRUE)
plotautocor(res)
plotautocor(res, lag.max=50)
```

---

plotnonp *Plotting Nonparametric Function Estimates*

Description

Plots nonparametric function estimates obtained from BayesX

Usage

```
plotnonp(data, x = 2, y = c(3, 4, 5, 7, 8), ylim = NULL,
         lty = c(1, 2, 3, 2, 3), cols = rep(1, length(y)), month, year, step=12,
         xlab, ylab, ...)
```

Arguments

`data`	Either the name of a file or a data frame containing the estimation results.
`x`	Defines the x-axis in the plot. Either the name of a variable in `data` or the index of the corresponding column.
`y`	Defines the variables to be plotted against `x`. May be either a vector of names of variables in `data` or the corresponding indices. The default choice corresponds to the point estimate plus two confidence bands.
`ylim`	Since `plotnonp` plots multiple y-variables, it automatically determines the appropriate `ylim` to make all curves visible. Argument `ylim` allows to override this default behaviour with fixed values.
`lty`	Vector of line types used for plotting (must have the same length as `y`). The default corresponds to solid lines for the point estimate and dashed and dotted lines for the confidence bands.
`cols`	Vector of colors used for plotting (must have the same length as `y`). Default are black lines.
`month, year, step`	Provide specific annotation for plotting estimation results for temporal variables. `month` and `year` define the minimum time point whereas step specifies the type of temporal data with `step=4`, `step=2` and `step=1` corresponding to quartely, half yearly and yearly data.
`xlab, ylab`	`plotnonp` constructs default labels that can be overwritten by these arguments
`...`	Further arguments to be passed to the interval call of `plot` such as `type`, etc.

Author(s)

Felix Heinzl, Andreas Brezger and Thomas Kneib

See Also

drawmap,plotautocor,plotsample,plotsurf

Examples

```
res <- read.table(system.file("examples/nonparametric_f_x_pspline.res",
                              package="BayesX"), header=TRUE)
plotnonp(res)
plotnonp(res, x="x")
plotnonp(res, x="x", y="pmean")
plotnonp(res, x="x", y="pmed")
plotnonp(res, x="x", y="pmed", ylim=c(-2,2))
plotnonp(res, x="x", y=c("pmean", "pqu10", "pqu90"), lty=c(1,1,1),
         col=c("red","blue","blue"))
plotnonp(res, xlab="some variable", ylab="f(some variable)",
         main="Nonlinear effect of some variable", sub="penalised spline")

res <- read.table(system.file("examples/nonparametric2_f_time_pspline.res",
                              package="BayesX"), header=TRUE)
plotnonp(res)
plotnonp(res, month=1, year=1980, step=12)

res <- res[1:18,]
plotnonp(res, month=1, year=1980, step=12)
```

---

plotsample *Plotting Sampling Paths*

Description

Plots sampling paths obtained with MCMC-sampling in BayesX

Usage

```
plotsample(data, ask = TRUE, ...)
plotsample.coda(data, ask = TRUE, ...)
```

Arguments

data	Either the name of a file or a data frame containing the sample.
ask	plotsample will plot separate sampling paths for each parameter. If ask=TRUE, the user will be prompted before showing the next plot.
...	Further parameters to be passed to the internal call of plot such as ylim, etc.

Details

plotsample simply plots sampling paths while plotsampe.coda makes use of the plotting facilities available in package coda.

Author(s)

Felix Heinzl, Andreas Brezger, Thomas Kneib

See Also

drawmap,plotautocor,plotnonp,plotsurf,

Examples

```
res <- read.table(system.file("examples/nonparametric_f_x_pspline_sample.raw",
                              package="BayesX"), header=TRUE)
plotsample(res)
```

---

plotsurf *Visualise Surface Estimates*

Description

Visualises surface estimates obtained with BayesX.

Usage

```
plotsurf(data, x=2, y=3, z=4, mode=1, ticktype="detailed",
         expand=0.75, d=100, theta=-30, phi=25, ...)
akimaPermitStatus()
akimaPermit()
```

Arguments

data	Either the name of a file or a data frame containing the estimation results.
x	Defines the x-axis in the plot. Either the name of a variable in data or the index of the corresponding column.
y	Defines the y-axis in the plot. Either the name of a variable in data or the index of the corresponding column.
z	Defines the z-axis in the plot. Either the name of a variable in data or the index of the corresponding column.
mode	plotsurf is mostly a convenient interface to the functions persp (mode=1), image (mode=2) and contour (mode=3).
ticktype, expand, d, theta, phi	Overwrite the default behaviour of persp
...	Further parameteres that are parsed to the internal call to persp, image or contour

Details

The akimaPermit function is used to choose to permit the use of akima if installed, and akimaPermitStatus reports its status. The licence for akima is not Free or Open Source and explicitly forbids commercial use. See the akima licence file for details.

Author(s)

Felix Heinzl, Thomas Kneib

See Also

drawmap,plotautocor,plotsample,plotnonp

Examples

```
res <- read.table(system.file("examples/surface_f_x1_x2_pspline.res",
                              package="BayesX"), header=TRUE)

# call akimaPermit to allow use of akima despite its licence that restricts
# usage to non-commercial purposes
akimaPermit()
plotsurf(res)
plotsurf(res, mode=2)
plotsurf(res, mode=3)

plotsurf(res, x="x1", y="x2", z="pmed")

plotsurf(res, ticktype="simple")

plotsurf(res, main="3D-Plot", xlab="myx", ylab="myy", zlab="f(myx,myy)")
```

---

read.bnd *Read Geographical Information in Boundary Format*

Description

Reads the geographical information provided in a file in boundary format (see Ch. 5 of the BayesX Reference Manual) and stores it in a map object.

Usage

```
read.bnd(file, sorted=FALSE)
```

Arguments

file	Name of the boundary file to be read.
sorted	Should the regions be ordered by the numbers speciying the region names (sorted=TRUE)?

Value

Returns a list of polygons that form the map. Additional attributes are

surrounding	Parallel list where for each polygon, the name of a possible surrounding region is saved.
height2width	Ratio between height and width of the map. Allows customised drawing and storage in files by specifying the appropriate height and width.
class	Indicates whether the map is stored in boundary format (bnd) or graph format (gra). Maps returned by read.bnd are of class bnd

Author(s)

Daniel Sabanes Bove, Felix Heinzl, Thomas Kneib, Andreas Brezger

References

BayesX Reference Manual. Available from http://www.bayesx.org/

See Also

write.bnd,drawmap,read.gra,write.gra

Examples

```
germany <- read.bnd(system.file("examples/germany.bnd", package="BayesX"))
drawmap(map=germany)
attributes(germany)

germany <- read.bnd(system.file("examples/germany2001.bnd", package="BayesX"))
drawmap(map=germany)
attributes(germany)
```

---

read.gra *Read Geographical Information in Graph Format*

Description

Reads the geographical information provided in a file in graph format (see Ch. 5 of the BayesX Reference Manual) and stores it in a map object.

Usage

```
read.gra(file, sorted=FALSE)
```

Arguments

file	Name of the graph file to be read.
sorted	Should the regions be ordered by the numbers speciying the region names (sorted=TRUE)?

Value

Returns an adjacency matrix that represents the neighborhood structure defined in the graph file. Additional attributes are

`dim`	Dimension of the (square) adjacency matrix.
`dimnames`	List of region names corresponding to rows and columns of the adjacency matrix.
`class`	Indicates whether the map is stored in boundary format (bnd) or graph format (gra). Maps returned by `read.gra` are of class `gra`

Author(s)

Thomas Kneib, Felix Heinzl

References

BayesX Reference Manual. Available from http://www.bayesx.org/

See Also

write.gra,read.bnd,write.bnd,get.neighbor,add.neighbor,delete.neighbor

Examples

```
germany <- read.gra(system.file("examples/germany.gra", package="BayesX"))
attributes(germany)
```

---

shp2bnd *convert a shape-file into a boundary object*

Description

Converts the geographical information provided in a shape-file into a boundary object (see Ch. 5 of the Reference Manual)

Usage

```
shp2bnd(shpname, regionnames, check.is.in = TRUE)
```

Arguments

`shpname`	Base filename of the shape-file (including path)
`regionnames`	Either a vector of region names or the name of the variable in the dbf-file representing these names
`check.is.in`	Test whether some regions are surrounded by other regions (FALSE speeds up the execution time but may result in a corrupted bnd-file)

Value

Returns a boundary object, i.e. a list of polygons that form the map. See read.bnd for more information on the format.

Author(s)

Felix Heinzl, Daniel Sabanes Bove, Thomas Kneib with contributions by Michael Hoehle and Frank Sagerer

References

BayesX Reference Manual. Available from http://www.bayesx.org

See Also

write.bnd,drawmap,read.bnd

Examples

```
## read shapefile into bnd object
shpName <- sub(pattern="(.*)\\.dbf", replacement="\\1",
               x=system.file("examples/northamerica_adm0.dbf",
                             package="BayesX"))
north <- shp2bnd(shpname=shpName, regionnames="COUNTRY")

## draw the map
drawmap(map=north)

## compare with shipped bnd file
shippedBnd <- read.bnd(system.file("examples/northamerica.bnd", package="BayesX"))
stopifnot(all.equal(north, shippedBnd))
```

---

smooth.bnd *Round Boundary Information*

Description

Rounds the boundary information in a map object in boundary format to a specified precision.

Usage

```
smooth.bnd(map, digits = 2, scale = 1)
```

Arguments

map	Map object in boundary format that should be modified.
digits	Number of digits to round to.
scale	Scaling factor that should be applied for rounding. For example, with scale=0.1 all polygons are divided by 10 before rounding.

Value

Map object in boundary format rounded to the specified precision.

Author(s)

Felix Heinzl, Thomas Kneib

See Also

read.bnd,write.bnd

---

write.bnd *Saving Maps in Boundary Format*

Description

Writes the information of a map object to a file (in boundary format)

Usage

```
write.bnd(map, file, replace = FALSE)
```

Arguments

map	Map object ot be saved (should be in boundary format).
file	Name of the file to write to
replace	Should an existing file be overwritten with the new version?

Author(s)

Thomas Kneib, Felix Heinzl

References

BayesX Reference Manual. Available from http://www.bayesx.org

See Also

write.gra,read.gra,read.bnd

write.gra *Saving Maps in Graph Format*

Description

Writes the information of a map object to a file (in graph format)

Usage

```
write.gra(map, file, replace = FALSE)
```

Arguments

map	Map object ot be saved (should be in graph format, see bnd2gra for the conversion of boundary format to graph format).
file	Name of the file to write to
replace	Should an existing file be overwritten with the new version?

Author(s)

Thomas Kneib, Felix Heinzl

References

BayesX Reference Manual. Available from http://www.bayesx.org

See Also

write.bnd,read.gra,read.bnd

Index