Package 'MRSea'

March 13, 2017

Title Marine Renewables Strategic environmental assessment

Description Examines animal survey data for signs of changes in animal abundance and distribution following marine renewables development. The functions of this package can be used to analyse segmented line transect data and nearshore vantage point data. The package includes functions for fitting spatially adaptive one and 2D smoothers using SALSA and CReSS. Non-parametric bootstrapping is available to estimate uncertainty. Several model assessment tools are also available. Recent updates include the direct estimation of robust standard errors, given a panel structure.

Version 0.99

Date 2017-03-10

Author Lindesay Scott-Hayward <lass@st-and.ac.uk>, Cornelia Oedekoven, Monique Mackenzie, Cameron Walker <cameron.walker@auckland.ac.nz>

Maintainer Lindesay Scott-Hayward <lass@st-and.ac.uk>

Depends R (>= 3.3.1)

Imports calibrate (>= 1.7.2),
car (>= 2.0-20),
fields (>= 7.1),
ggplot2 (>= 1.0.0),
Matrix (>= 1.1-4),
mgcv (>= 1.8-0),
mrds (>= 2.1.6),
mvtnorm (>= 1.0-0),
splines (>= 3.1.1)

License GPL-2

LazyData true

Note Scott-Hayward LAS, Oedekoven CS, Mackenzie ML and CG Walker (2017). ``MRSea package (version 0.99): Statistical Modelling of bird and cetacean distributions in offshore renewables development areas". University of St. Andrews

URL http://creem2.st-and.ac.uk/software.aspx

RoxygenNote 6.0.1

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

acffunc . . . 3
AICh . . . 4
anova.gamMRSea . . . 4
bootstrap.orig.data . . . 5
checkfactorlevelcounts . . . 6
choose.radii . . . 6
create.bootcount.data . . . 7
create.bootstrap.data . . . 8
create.count.data . . . 9
create.NHAT . . . 10
dis.data.de . . . 10
dis.data.no . . . 11
dis.data.re . . . 12
do.bootstrap.cress . . . 12
do.bootstrap.cress.robust . . . 15
do.bootstrap.gam . . . 17
drop.step_2d . . . 19
exchange.step_2d . . . 19
fit.thinPlate_2d . . . 19
gamMRSea . . . 20
generateNoise . . . 20
getCVids . . . 21
getCV_CReSS . . . 22
getDifferences . . . 23
getDispersion . . . 24
getEmpDistribution . . . 24
getKnotgrid . . . 25
getPlotdimensions . . . 26
getPvalues . . . 27
getRadiiChoices . . . 28
improve.step_2d . . . 29
knotgrid.ns . . . 29
knotgrid.off . . . 29
LocalRadialFunction . . . 30
make.gamMRSea . . . 31
makeBootCIs . . . 31
makeDists . . . 32
makesplineParams . . . 33
MRSea . . . 34
ns.data.de . . . 34
ns.data.no . . . 35
ns.data.re . . . 35
ns.predict.data.de . . . 36
ns.predict.data.no . . . 37
ns.predict.data.re . . . 37
plotacf . . . 38
plotCumRes . . . 38
plotRunsProfile . . . 39
predict.cress . . . 40
predict.data.de . . . 42

predict.data.no . . . 43
predict.data.re . . . 43
predict.gamMRSea . . . 44
QICb . . . 45
qzibinom . . . 45
return.reg.spline.fit . . . 46
return.reg.spline.fit.2d . . . 47
runACF . . . 47
runDiagnostics . . . 48
runInfluence . . . 49
runPartialPlots . . . 50
runSALSA1D . . . 51
runSALSA2D . . . 54
runsTest . . . 56
summary.gamMRSea . . . 58
thinModels . . . 59
timeInfluenceCheck . . . 60
which.bin . . . 60

Index **62**

`acffunc` *calculate correlation for residuals by block*

Description

calculate correlation for residuals by block

Usage

```
acffunc(block, model, suppress.printout = FALSE)
```

Arguments

`block`	Vector of blocks that identify data points that are correlated
`model`	Fitted model object (glm or gam)
`suppress.printout`	(Default: `FALSE`. Logical stating whether to show a printout of block numbers to assess progress. 'FALSE' will show printout.

AICh *Function to calculate AICh (Hardin and Hilbe 2013)*

Description

Function to calculate AICh (Hardin and Hilbe 2013)

Usage

```
AICh(model)
```

Arguments

model	model object of class glm.

Author(s)

Lindesay Scott-Hayward, University of St Andrews

anova.gamMRSea *Anova Tables for* gamMRSea *Models*

Description

Calculates type-III analysis-of-variance tables for model objects produced by gamMRSea (in the MRSea package). Wald chisquare tests are calculated by default although, F-tests may be specified.

Usage

```
## S3 method for class 'gamMRSea'
anova(object, varshortnames = NULL, panelid = NULL,
  test = "Wald")
```

Arguments

object	A gamMRSea model object
varshortnames	(default = NULL). Character vector denoting the short names to use for any smooth terms. May already be specified as part of the model object.
panelid	vector of length of the data used in object. Specified if robust standard errors are to be used.
test	(default='wald'). May also specify "F".

Value

An object of class "anova".

Examples

```
# load data
data(ns.data.re)
ns.data.re$foldid<-getCVids(ns.data.re, folds=5)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)
anova(model)
```

`bootstrap.orig.data` *Obtaining a data frame of bootstrapped data using resamples*

Description

This function extracts the records corresponding to each resample from the original distance data and pastes them together in a new data frame which is returned.

Usage

```
bootstrap.orig.data(orig.data, resample, new.resamples, resamples.no)
```

Arguments

`orig.data`	Original data to be bootstrapped
`resample`	Specifies the resampling unit for bootstrapping, default is `transect.id`. Must match a column name in `orig.data` exactly
`new.resamples`	String of resampled units from `data[,"resample"]`. Created by `create.bootstrap.data()`
`resamples.no`	Length of new.resamples

Value

Returns bootstrapped data. Internal function called by function `create.bootstrap.data`.

Examples

```
data(dis.data.re)
resample<-"transect.id"
samples<-unique(dis.data.re[,resample])
resamples.no<-length(samples)
new.resamples<-sample(samples,resamples.no,replace=TRUE)
bootstrap.data<-bootstrap.orig.data(dis.data.re,resample,new.resamples,resamples.no)
```

checkfactorlevelcounts *Factor level response check This function checks that there are some non-zero counts in each level of each factor variable for consideration in a model*

Description

Factor level response check

This function checks that there are some non-zero counts in each level of each factor variable for consideration in a model

Usage

```
checkfactorlevelcounts(factorlist, data, response)
```

Arguments

factorlist	Vector of factor variables specified in model. Specified so that a check can be made that there are non-zero counts in all levels of each factor.
data	Data frame containing columns of covariates listed in factorlist. Column names must match with names in factorlist
response	A vector of response values

Examples

```
# load data
data(ns.data.re)

checkfactorlevelcounts(factorlist=c('floodebb', 'impact'), ns.data.re,
     ns.data.re$birds)
```

choose.radii *Function to choose the radii for the CReSS local radial basis function*

Description

Function to choose the radii for the CReSS local radial basis function

Usage

```
choose.radii(currentFit, indices, radiusIndices, radii, out.lm, dists, aR,
  baseModel, fitnessMeasure, response, models, interactionTerm, data, initDisp)
```

`create.bootcount.data` *Aggregate bootstrapped distance data into count data*

Description

This function creates a new data set where `dis.data` is aggregated for each visit to a segment. For bootstrapped data, the column with the ids for visits to a segment is `segment.id2` which is created by `create.bootstrap.data` using the default for argument `rename`. The sum of the estimated number of individuals for each segment from `dis.data$NHAT` is given in the column `NHAT` in the new data. All other columns from the observation layer should be discarded. This is achieved by specifying the columns that should be retained using the argument `column.numbers`. Generally, all columns from the segment and higher levels should be kept. If the default is used, `column.numbers=NULL`, the columns `distance`, `object`, `size`, `distbegin` and `distend` from the observation level are automatically discarded. Note that for those columns from the observation layer that are kept, only the first recorded value will be transferred.

Usage

```
create.bootcount.data(dis.data, column.numbers = NULL)
```

Arguments

`dis.data`	Data frame containing distance data (one row for each detection). Expects a column `NHAT`, i.e. size of detection divided by its probability of detection (see `create.NHAT`) and that and that ids in `segment.id2` are unique regardless of what resampled transect they belong to.
`column.numbers`	Optional argument: vector of integers indicating which columns other than `NHAT` from `dis.data` should be retained in the returned data.

Value

This function returns bootstrapped count data that is suited for second stage count modelling of distance sampling data. The data includes the columns `NHAT` and `area` which are the response and the offset required by functions concerned with second stage modelling from this package.

Examples

```
data(dis.data.re)
# bootstrap data without stratification
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
            meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)

bootstrap.data<-create.bootstrap.data(dis.data.re)

bootcount.data<-create.bootcount.data(bootstrap.data)
```

`create.bootstrap.data` *Create bootstrap data for non-parametric bootstrapping*

Description

This function creates one realisation of bootstrapped data based on `dis.data`. The default resampling unit is `transect.id` which may be modified using the argument `resample`.

Usage

```
create.bootstrap.data(dis.data, resample = "transect.id",
  rename = "segment.id", stratum = NULL)
```

Arguments

`dis.data`	Original data to be bootstrapped. Requires a column that matches argument `resample` exactly.
`resample`	Specifies the resampling unit for bootstrapping, default is `transect.id`. Must match a column name in `dis.data` exactly
`rename`	A vector of column names for which a new column needs to be created for the bootstrapped data. This defaults to `segment.id` for line transects, however others might be added A new column with new ids will automatically be created for the column listed in `resample`
`stratum`	The column name in `dis.data` that identifies the different strata. The default `NULL` returns un-stratified bootstrap data. If stratum is specified, this requires a column in `dis.data` that matches argument `stratum` exactly

Value

Returns one realisation of bootstrapped distance data. Note that a new column (in addition to those listed under argument `rename`) is created. If the default for `resample` is used, a column with new unique ids called `transect.id2`. Note that a new column is created with renamed bootstrap resamples to preserve the number of unique bootstrap resamples. If the default for `resample` is used, i.e. `transect.id`, this new column is called `transect.id2`. In addition, a new column `segment.id2` is created which is required for other bootstrap functions.

Examples

```
data(dis.data.re)
# run distance analysis to create NHATS
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
            meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)

# bootstrap data without stratification
bootstrap.data<-create.bootstrap.data(dis.data.re)
# boostrap data with stratification (here by survey which is composed of
# season and impact)
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
bootstrap.data.str<-create.bootstrap.data(dis.data.re, stratum = "survey.id")
```

`create.count.data` *Aggregate distance data into count data*

Description

This function creates a new data set where `dis.data` is aggregated for each visit to a segment (`segment.id`). The sum of the estimated number of individuals for each segment from `dis.data$NHAT` is given in the column `NHAT` in the new data. Only columns from the segment or higher layers should be carried over into `count.data` from `dis.data`. Use argument `column.numbers` to identify these.

Usage

```
create.count.data(dis.data, column.numbers = NULL)
```

Arguments

`dis.data`	Data frame containing distance data (one row for each detection). Expects a column `NHAT`, i.e. size of detection divided by its probability of detection (see `create.NHAT`) and that ids in `segment.id` are unique regardless of what transect they belong to
`column.numbers`	Optional argument: vector of integers indicating which columns other than `NHAT` from `dis.data` should be retained in the returned data. Generally all columns from the segment and higher levels should be kept while those from the observation level should be discarded. If the default is used, `column.numbers=NULL`, the columns `distance`, `object`, `size`, `distbegin` and `distend` from the observation level are automatically discarded. Note that for those columns from the observation layer that are kept, only the first recorded value will be transferred.

Value

This function returns count data that is suited for second stage count modelling of distance sampling data. The data includes the columns `NHAT` and `area` which are the response and the offset required by functions concerned with second stage modelling from this package.

Examples

```
data(dis.data.re)
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
           meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)
count.data<-create.count.data(dis.data.re)
```

create.NHAT *Estimated number of individuals for each detection*

Description

This function creates a new column in data which contains the estimated number of animals for each detection. This is the number of observed individuals divided by their probability of detection using MCDS methods (size/detection probability). In the case that no size column is given in dis.data, it is assumed that detections were made of individuals and size is set to 1 for all detections. The values for size and NHAT are set to zero in case the distance was larger than the truncation distance w specified in det.fct.object. In addition, a new column area is created which is used as the offset in the second stage count model (segment length * (truncation distance/1000) * 2). The truncation distance is divided by 1000 to convert it from metres to km. It is assumed that the segment data represents two-sided surveys. In case the survey was one-sided, this column needs to be divided by 2 after the call to this function.

Usage

```
create.NHAT(data, ddf.obj)
```

Arguments

data	distance data object used with det.fct to estimate probabilities of detection
ddf.obj	detection function object created by ddf

Examples

```
data(dis.data.re)
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re,method="ds",
     meta.data= list(width=250,binned=FALSE))
dis.data<-create.NHAT(dis.data.re,result)
```

dis.data.de *Line transect data with decrease post-impact*

Description

A simulated dataset containing the observed perpendicular distances, the effort data and other variables of segmented line transect data. The variables are as follows:

Format

A data frame with 10759 rows and 12 variables

Details

- `transect.id` Identifier for the individual visits to the transects
- `transect.label` Labels for transects
- `season` Numerical indicator for the four different seasons
- `impact` Numerical indicator for before (0) and after (1) impact
- `segment.id` Identifier for individual visits to the segment
- `segment.label` Label for segments
- `length` Length of segment in km
- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `depth` Depth in m
- `object` Id for detected object
- `distance` Perpendicular distance from the line

`dis.data.no` *Line transect data with no post-impact consequence*

Description

A simulated dataset containing the observed perpendicular distances, the effort data and other variables of segmented line transect data. The variables are as follows:

Format

A data frame with 10771 rows and 12 variables

Details

- `transect.id` Identifier for the individual visits to the transects
- `transect.label` Labels for transects
- `season` Numerical indicator for the four different seasons
- `impact` Numerical indicator for before (0) and after (1) impact
- `segment.id` Identifier for individual visits to the segment
- `segment.label` Label for segments
- `length` Length of segment in km
- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `depth` Depth in m
- `object` Id for detected object
- `distance` Perpendicular distance from the line

dis.data.re — *Line transect data with redistribution post-impact*

Description

A simulated dataset containing the observed perpendicular distances, the effort data and other variables of segmented line transect data. The variables are as follows:

Format

A data frame with 10951 rows and 12 variables

Details

- `transect.id` Identifier for the individual visits to the transects
- `transect.label` Labels for transects
- `season` Numerical indicator for the four different seasons
- `impact` Numerical indicator for before (0) and after (1) impact
- `segment.id` Identifier for individual visits to the segment
- `segment.label` Label for segments
- `length` Length of segment in km
- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `depth` Depth in m
- `object` Id for detected object
- `distance` Perpendicular distance from the line

do.bootstrap.cress — *Bootstrapping function without model selection using CReSS/SALSA for fitting the second stage count model*

Description

This fuction performs a specified number of bootstrapping iterations using CReSS/SALSA for fitting the second stage count model. See below for details.

Usage

```
do.bootstrap.cress(orig.data, predict.data, ddf.obj = NULL, model.obj,
  splineParams, g2k, resample = "transect.id", rename = "segment.id",
  stratum = NULL, B, name = NULL, save.data = FALSE, nhats = FALSE,
  seed = 12345, nCores = 1)
```

Arguments

orig.data	The original data. In case ddf.obj is specified, this should be the original distance data. In case ddf.obj is NULL, it should have the format equivalent to count.data where each record represents the summed up counts at the segments.
predict.data	The prediction grid data
ddf.obj	The ddf object created for the best fitting detection model. Defaults to NULL for when nodetection function object available.
model.obj	The best fitting CReSS model for the original count data. Should be geeglm or a Poisson/Binomial GLM (not quasi).
splineParams	The object describing the parameters for fitting the one and two dimensional splines
g2k	(N x k) matrix of distances between all prediction points (N) and all knot points (k)
resample	Specifies the resampling unit for bootstrapping, default is transect.id. Must match a column name in dis.data exactly
rename	A vector of column names for which a new column needs to be created for the bootstrapped data. This defaults to segment.id for line transects (which is required for create.bootcount.data), others might be added. A new column with new ids will automatically be created for the column listed in resample. In case of nearshore data, this argument is ignored.
stratum	The column name in orig.data that identifies the different strata. The default NULL returns un-stratified bootstrap data. In case of nearshore data, this argument is ignored.
B	Number of bootstrap iterations
name	Analysis name. Required to avoid overwriting previous bootstrap results. This name is added at the beginning of "predictionboot.RData" when saving bootstrap predictions.
save.data	If TRUE, all created bootstrap data will be saved as an RData object in the working directory at each iteration, defaults to FALSE
nhats	(default = FALSE). If you have calculated bootstrap NHATS because there is no simple ddf object then a matrix of these may be fed into the function. The number of columns of data should >= B. The rows must be equal to those in orig.data and d2k and *must* be in matching order.
seed	Set the seed for the bootstrap sampling process.
nCores	Set the number of computer cores for the bootstrap process to use (default = 1). The more cores the faster the proces but be wary of over using the cores on your computer. If nCores > (number of computer cores - 2), the function defaults to nCores = (number of computer cores - 2). Note: On a Mac computer the parallel code does not compute so use nCores=1.

Details

In case of distance sampling data, the following steps are performed for each iteration:

- the original data is bootstrapped

- a detection function is fitted to the bootstrapped data

- a count model is fitted to the bootstrapped data

- coefficients are resampled from a multivariate normal distribution defined by MLE and COV from count model

- predictions are made to the prediction data using the resampled coefficients

In case of count data, the following steps are performed for each iteration:

- coefficients are resampled from a multivariate normal distribution defined by MLE and COV from the best fitting count model

- predictions are made to the prediction data using the resampled coefficients

Value

The function returns a matrix of bootstrap predictions. The number of rows is equal to the number of rows in predict.data. The number of columns is equal to B. The matrix may be very large and so is stored directly into the working directory as a workspace object: '"name"predictionboot.RObj'. The object inside is called bootPreds.

Examples

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# offshore redistribution data
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
data(dis.data.re)
data(predict.data.re)
data(knotgrid.off)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# distance sampling
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
        meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)
count.data<-create.count.data(dis.data.re)

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# spatial modelling
splineParams<-makesplineParams(data=count.data, varlist=c('depth'))
#set some input info for SALSA
count.data$response<- count.data$NHAT
# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(count.data$x.pos, count.data$y.pos), na.omit(knotgrid.off))
# choose sequence of radii
r_seq<-getRadiiChoices(8,distMats$dataDist)
# set initial model without the spatial term
initialModel<- glm(response ~ as.factor(season) + as.factor(impact) + offset(log(area)),
                family='quasipoisson', data=count.data)
# make parameter set for running salsa2d
salsa2dlist<-list(fitnessMeasure = 'QICb', knotgrid = knotgrid.off,
                 knotdim=c(26,14), startKnots=4, minKnots=4,
                maxKnots=20, r_seq=r_seq, gap=4000, interactionTerm="as.factor(impact)")
salsa2dOutput_k6<-runSALSA2D(initialModel, salsa2dlist, d2k=distMats$dataDist,
                   k2k=distMats$knotDist, splineParams=splineParams)

splineParams<-salsa2dOutput_k6$splineParams
# specify parameters for local radial function:
radiusIndices <- splineParams[[1]]$radiusIndices
dists <- splineParams[[1]]$dist
radii <- splineParams[[1]]$radii
```

```
aR <- splineParams[[1]]$invInd[splineParams[[1]]$knotPos]
count.data$blockid<-paste(count.data$transect.id, count.data$season, count.data$impact, sep='')
# Re-fit the chosen model as a GEE (based on SALSA knot placement) and GEE p-values
geeModel<- geeglm(formula(salsa2dOutput_k6$bestModel), data=count.data, family=poisson, id=blockid)
dists<-makeDists(cbind(predict.data.re$x.pos, predict.data.re$y.pos), na.omit(knotgrid.off),
       knotmat=FALSE)$dataDist

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# bootstrapping
do.bootstrap.cress(dis.data.re, predict.data.re, ddf.obj=result, geeModel, splineParams,
            g2k=dists, resample='transect.id', rename='segment.id', stratum='survey.id',
              B=4, name="cress", save.data=FALSE, nhats=FALSE, nCores=1)
load("cresspredictionboot.RData") # loading the bootstrap predictions into the workspace
# look at the first 6 lines of the bootstrap predictions (on the scale of the response)
head(bootPreds)

## Not run:
# In parallel (Note: windows machines only)
require(parallel)
do.bootstrap.cress(dis.data.re, predict.data.re, ddf.obj=result, geeModel, splineParams,
             g2k=dists, resample='transect.id', rename='segment.id', stratum='survey.id',
                B=4, name="cress", save.data=FALSE, nhats=FALSE, nCores=4)
load("cresspredictionboot.RData") # loading the bootstrap predictions into the workspace
# look at the first 6 lines of the bootstrap predictions (on the scale of the response)
head(bootPreds)

## End(Not run)

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# nearshore redistribution data
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
## Not run:
do.bootstrap.cress(ns.data.re, ns.predict.data.re, ddf.obj=NULL, geeModel, splineParams,
             g2k=dists, resample='transect.id', rename='segment.id', stratum=NULL,
             B=2, name="cress", save.data=FALSE, nhats=FALSE)
load("cresspredictionboot.RData") # loading the predictions into the workspace
# look at the first 6 lines of the bootstrap predictions (on the scale of the response)
head(bootPreds)
## End(Not run)
```

`do.bootstrap.cress.robust` *Bootstrapping function without model selection for a model of class 'gamMRSea'*

Description

This fuction performs a specified number of bootstrapping iterations using CReSS/SALSA for fitting the count model. See below for details.

Usage

```
do.bootstrap.cress.robust(model.obj, predictionGrid, splineParams = NULL,
```

```
g2k = NULL, B, robust = T, name = NULL, seed = 12345, nCores = 1,
cat.message = TRUE)
```

Arguments

model.obj	The best fitting CReSS model for the original count data. Should be geeglm or a Poisson/Binomial GLM (not quasi).
predictionGrid	The prediction grid data
splineParams	The object describing the parameters for fitting the one and two dimensional splines
g2k	(N x k) matrix of distances between all prediction points (N) and all knot points (k)
B	Number of bootstrap iterations
name	Analysis name. Required to avoid overwriting previous bootstrap results. This name is added at the beginning of "predictionboot.RData" when saving bootstrap predictions.
seed	Set the seed for the bootstrap sampling process.
nCores	Set the number of computer cores for the bootstrap process to use (default = 1). The more cores the faster the proces but be wary of over using the cores on your computer. If nCores > (number of computer cores - 2), the function defaults to nCores = (number of computer cores - 2). Note: On a Mac computer the parallel code does not compute so use nCores=1.
rename	A vector of column names for which a new column needs to be created for the bootstrapped data. This defaults to segment.id for line transects (which is required for create.bootcount.data), others might be added. A new column with new ids will automatically be created for the column listed in resample. In case of nearshore data, this argument is ignored.

Details

The following steps are performed for each iteration:

- coefficients are resampled from a multivariate normal distribution defined by MLE and COV from the best fitting count model

- predictions are made to the prediction data using the resampled coefficients

Value

The function returns a matrix of bootstrap predictions. The number of rows is equal to the number of rows in predictionGrid. The number of columns is equal to B. The matrix may be very large and so is stored directly into the working directory as a workspace object: '"name"predictionboot.RObj'. The object inside is called bootPreds.

`do.bootstrap.gam` *Bootstrapping function without model selection using* `gam` *as the second stage count model*

Description

This fuction performs a specified number of bootstrapping iterations using gams for fitting the second stage count model. See below for details.

Usage

```
do.bootstrap.gam(orig.data, predict.data, ddf.obj = NULL, model.obj,
  resample = "transect.id", rename = "segment.id", stratum = NULL, B,
  name = NULL, save.data = FALSE, nhats = NULL)
```

Arguments

`orig.data`	The original data. In case `ddf.obj` is specified, this should be the original distance data. In case `ddf.obj` is `NULL`, it should have the format equivalent to `count.data` where each record represents the summed up counts at the segments.
`predict.data`	The prediction grid data
`ddf.obj`	The ddf object created for the best fitting detection model. Defaults to `NULL` for nearshore data.
`model.obj`	The best fitting `gam` model for the original count data
`resample`	Specifies the resampling unit for bootstrapping, default is `transect.id`. Must match a column name in `dis.data` exactly
`rename`	A vector of column names for which a new column needs to be created for the bootstrapped data. This defaults to `segment.id` for line transects (which is required for `create.bootcount.data`), others might be added. A new column with new ids will automatically be created for the column listed in `resample`. In case of nearshore data, this argument is ignored.
`stratum`	The column name in `orig.data` that identifies the different strata. The default `NULL` returns un-stratified bootstrap data. In case of nearshore data, this argument is ignored.
`B`	Number of bootstrap iterations
`name`	Analysis name. Required to avoid overwriting previous bootstrap results. This name is added at the beginning of "predictionboot.RData" when saving bootstrap predictions.
`save.data`	If TRUE, all created bootstrap data will be saved as an RData object in the working directory at each iteration, defaults to FALSE
`nhats`	(default = FALSE). If you have calculated bootstrap NHATS because there is no simple ddf object then a matrix of these may be fed into the function. The number of columns of data should >= B. The rows must be equal to those in `orig.data` and `d2k` and *must* be in matching order.

Details

In case of distance sampling data, the following steps are performed for each iteration:

- the original data is bootstrapped

- a detection function is fitted to the bootstrapped data

- a count model is fitted to the bootstrapped data

- coefficients are resampled from a multivariate normal distribution defined by MLE and COV from count model

- predictions are made to the prediction data using the resampled coefficients

In case of count data, the following steps are performed for each iteration

- coefficients are resampled from a multivariate normal distribution defined by MLE and COV from the best fitting count model

- predictions are made to the prediction data using the resampled coefficients

Value

The function returns a matrix of bootstrap predictions. The number of rows is equal to the number of rows in predict.data. The number of columns is equal to B. The matrix may be very large and so is stored directly into the working directory as a workspace object: '"name"predictionboot.RObj'. The object inside is called bootPreds.

Examples

```
# offshore redistribution data
data(dis.data.re)
data(predict.data.re)
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
            meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)
count.data<-create.count.data(dis.data.re)
require(mgcv)
gam.2<-gam(NHAT~as.factor(impact)+s(x.pos,y.pos,by=as.factor(impact))+offset(log(area)),
           data=count.data,family=quasipoisson)
do.bootstrap.gam(dis.data.re,predict.data.re,ddf.obj=result,model.obj=gam.2,resample="transect.id",
            rename="segment.id",stratum='survey.id',1,name='gam',save.data=FALSE,nhats=NULL)
load("gampredictionboot.RData") # loading the predictions into the workspace
# look at the first 6 lines of the predictions on the response scale
head(bootPreds)


## Not run: # nearshore redistribution data
data(ns.data.re)
data(ns.predict.data.re)
require(mgcv)
gam.ns2=gam(birds~as.factor(impact)+s(x.pos,y.pos,by=as.factor(impact))+offset(log(area)),
         data=ns.data.re,family=quasipoisson)
do.bootstrap.gam(ns.data.re,ns.predict.data.re,ddf.obj=NULL,model.obj=gam.ns2,resample=NULL,
                rename=NULL,stratum=NULL,1,name='ns.gam',save.data=FALSE,nhats=NULL)
# load the replicate predictions into the workspace
load("ns.gampredictionboot.RData")
# look at the first 6 lines of the predictions on the response scale
head(bootPreds)
```

```
## End(Not run)
```

`drop.step_2d` *Function that tries dropping knots to find an improvement in fit*

Description

Function that tries dropping knots to find an improvement in fit

Usage

```
drop.step_2d(radii, invInd, dists, explData, response, explanatory,
  maxIterations, fitnessMeasure, point, knotPoint, position, aR, BIC, track,
  out.lm, improveDrop, minKnots, tol = 0, baseModel, radiusIndices, models,
  interactionTerm, data, initDisp)
```

Author(s)

Cameron Walker, Department of Enginering Science, University of Auckland.

`exchange.step_2d` *Function for exchanging knot locations and re-fitting model to find best one*

Description

Function for exchanging knot locations and re-fitting model to find best one

Usage

```
exchange.step_2d(gap, knotDist, radii, invInd, dists, explData, response,
  explanatory, maxIterations, fitnessMeasure, point, knotPoint, position, aR,
  BIC, track, out.lm, improveEx, maxKnots, tol = 0, baseModel, radiusIndices,
  models, interactionTerm, data, initDisp)
```

Author(s)

Cameron Walker, Department of Enginering Science, University of Auckland.

`fit.thinPlate_2d` *Function to fit a local radial basis function (CReSS) as a two dimensional smooth*

Description

Function to fit a local radial basis function (CReSS) as a two dimensional smooth

Usage

```
fit.thinPlate_2d(fitnessMeasure, dists, aR, radii, baseModel, radiusIndices,
  models, currentFit, interactionTerm, data, initDisp)
```

gamMRSea *gamMRSea model function*

Description

See glm for details. A splineParams object may be specified as part of the model object.

Usage

```
gamMRSea(formula, family = gaussian, data, weights, subset, na.action,
  start = NULL, etastart, mustart, offset, control = list(...),
  model = TRUE, method = "glm.fit", x = FALSE, y = TRUE,
  contrasts = NULL, splineParams = NULL, ...)
```

Author(s)

LAS Scott-Hayward, University of St Andrews

generateNoise *Function to generate noisy data*

Description

The function generates a random sample from poisson, overdispersed poisson, binomial and zero inflated binomial samples.

Usage

```
generateNoise(n, response, family, ...)
```

Arguments

n	number of simulations to generate
response	vector of 'true' means to genereate from
family	one of poisson, binomial or zibinomial
...	Other parameters required for the family specified

Details

An additional parameter for the Poisson distribution is the dispersion parameter, specified by d= The additional parameters for the Binomial distribution can be found in rbinom The zibinomial family requires the VGAM library to generate zero inflated binomial data. Additional parameters can be found in the help for rzibinom.

Author(s)

LAS Scott-Hayward, University of St Andrews

Examples

```
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)

simData<-generateNoise(n=500, response=fitted(model), family='poisson')
```

`getCVids` *IDs for running cross validation*

Description

This function creates a string of integers which will be used for pointing to the right subsets of data for cross validation of regression objects

Usage

```
getCVids(data, folds, block = NULL, seed = 1234)
```

Arguments

`data`	data used in regression model
`folds`	integer number of validation data sets
`block`	column in data indicating the blocking structure for cross-validation (if `block` = NULL, individual observations will be used as blocks)
`seed`	integer number used to set the seed of the fold generation. By default this is set to '1234'.

Details

The function returns a random sequence of 1:folds of the same length as observations in data. It is called by other functions, e.g. `getCV_CReSS`.

Author(s)

LAS Scott-Hayward, University of St Andrews

Examples

```
# load data
data(ns.data.re)

CVids<-getCVids(ns.data.re, 5)
```

getCV_CReSS *Calculate cross-validation score for a CReSS type model*

Description

Calculate cross-validation score for a CReSS type model

Usage

```
getCV_CReSS(datain, baseModel, splineParams = NULL, vector = FALSE)
```

Arguments

datain	Data frame containing columns of covariates contained in baseModel.
baseModel	'glm' or 'gamMRSea' model object
splineParams	list object containing information for fitting one and two dimensional splines. See makesplineParams for more details.
vector	Logical indicating whether the vector of scores is returned (TRUE) or if the mean score is returned (FALSE).

Details

There must be a column in the data called foldid, which can be created using getCVids. This column defines the folds of data for the CV calculation. If this is not provided, by default random allocation is given to 5 folds.

The cost function for this CV is a mean squared error.

Author(s)

LAS Scott-Hayward (University of St Andrews)

Examples

```
# load data
data(ns.data.re)
ns.data.re$foldid<-getCVids(ns.data.re, folds=5)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)

# calculate CV
getCV_CReSS(ns.data.re, model)
```

`getDifferences` *Identify any significant differences between predicted data before an impact event and predicted data after an impact event*

Description

Identify any significant differences between predicted data before an impact event and predicted data after an impact event

Usage

```
getDifferences(beforePreds, afterPreds, quants = c(0.025, 0.975))
```

Arguments

`beforePreds`	Matrix of bootstrap predictions (n x B) to each grid cell before impact (same length and order as `afterPreds`)
`afterPreds`	Matrix of bootstrap predictions (n x B) to each grid cell after impact (same length and order as `beforePreds`)
`quants`	(`default = =c(.025,.975)`) Quantile for significance.

Details

This function finds the differences for every predicted grid cell for every bootstrap replicate. Quantiles are used to determine whether each difference is significantly different from zero and if so, in what direction.

Value

A list is returned consisting of

`mediandiff`	Vector of the median difference for each grid cell
`lowerci`	Vector of the lower 2.5% difference for each grid cell
`upperci`	Vector of the upper 97.5% difference for each grid cell
`significanceMarker`	Vector of significance. 0: not significant, 1: significant and positive, -1: significant and negative

Examples

```
 ## Not run:
getDifferences(beforePreds, afterPreds)
## End(Not run)
```

getDispersion *dispersion parameter*

Description

This function calculates the dispersion parameter for Normal, Binomial, Poisson and Gamma distributions

Usage

```
getDispersion(model)
```

Arguments

`model`

Details

some details

Value

single number of dispersion parameter estimation

Author(s)

LAS Scott-Hayward

getEmpDistribution *Function to generate the empirical distribution of the runs test statistic, given some data and a model.*

Description

Function to generate the empirical distribution of the runs test statistic, given some data and a model.

Usage

```
getEmpDistribution(n.sim, simData, model, data, plot = FALSE,
  returnDist = TRUE, dots = TRUE)
```

Arguments

`n.sim`	The number of simulated sets of data.
`simData`	A matrix, where each column is a set of data simulated under independence, with rows the same length as the data used for the model.
`model`	a glm or gamMRSea model object
`data`	data set used to fit the model.

plot	logical flag. If TRUE, a plot is made showing the 5% critical values for the empirical distribution vs the N(0,1) distribution. Default is 'FALSE'
returnDist	logical flag. Do you want the distribution of test statistics returned ('TRUE') or just the 5% critical values ('FALSE')
dots	(Default: TRUE. Logical stating whether to show a printout of block numebers to assess progress. 'TRUE' will print dots into the workspace.

Author(s)

LAS Scott-Hayward, University of St Andrews

Examples

```
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)

simData<-generateNoise(n=500, response=fitted(model), family='poisson')
empdist<-getEmpDistribution(500, simData, model, data=ns.data.re, plot=FALSE,
                returnDist=TRUE,dots=FALSE)
```

---

getKnotgrid *Generate a grid of knot locations to run SALSA2D.*

Description

Generate a grid of knot locations to run SALSA2D.

Usage

```
getKnotgrid(coordData, numKnots = 300, plot = TRUE)
```

Arguments

coordData	nx2 matrix or data frame of coordinates representing data locations
numKnots	(default = 300)). The user may choose how many legal knot locations are available (only if more than 400)
plot	(default = TRUE). Logical stating whether a plot showing the legal knot positions is given.

Details

SALSA2D requires a grid of knot locations to determine the best locations. Illegal knot positions (those not close to data) are kept as a row in the data frame of locations but given c(NA, NA) to avoid a knot considered.

Value

A (10000 x 2) matrix of knot locations. Any row with an illegal knot location (e.g. not near data) contains c(NA, NA)

Examples

```
 ## Not run:
data(dis.data.re)
# bootstrap data without stratification
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
require(mrds)
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
            meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)

knotgrid<-getKnotgrid(cbind(dis.data.re$x.pos, dis.data.re$y.pos))
## End(Not run)
```

`getPlotdimensions` *find the plotting dimensions for quilt.plot when using a regular grid*

Description

find the plotting dimensions for quilt.plot when using a regular grid

Usage

```
getPlotdimensions(x.pos, y.pos, segmentWidth, segmentLength)
```

Arguments

`x.pos`	Vector of x-coordinates in dataset
`y.pos`	Vector of y-coordinates in dataset
`segmentWidth`	Width of each grid cell of data (in same units as `x.pos`)
`segmentLength`	Length of each grid cell of data (in same units as `y.pos`)

Examples

```
#' # load data
data(ns.data.re)

getPlotdimensions(ns.data.re$x.pos, ns.data.re$y.pos, segmentWidth=500, segmentLength=500)
```

getPvalues *Calculate marginal p-values from a* model.

Description

An ANOVA is fitted repeatedly with each covariate being the last so that the output is marginal. varlist and factorlist are optional and shorten the variable names in the output.

Usage

```
getPvalues(model, varlist = NULL, factorlist = NULL)
```

Arguments

model	Fitted model object of class gee.
varlist	(default =NULL). Vector of covariate names (continous covariates only) used to make the output table names shorter. Useful if spline parameters are specified in the model.
factorlist	(default =NULL). Vector of covariate names (factor covariates only) used to make the output table names shorter. Useful if spline parameters are specified in the model.

Value

Print out table of each variable and its associated marginal p-value.

Examples

```
# load data
data(ns.data.re)

# make blocking structure
ns.data.re$blockid<-paste(ns.data.re$GridCode, ns.data.re$Year, ns.data.re$MonthOfYear,
                    ns.data.re$DayOfMonth, sep='')
ns.data.re$blockid<-as.factor(ns.data.re$blockid)

initialModel<- geeglm(birds ~ as.factor(floodebb) + as.factor(impact) + observationhour + x.pos +
              y.pos + offset(log(area)), family='poisson',data=ns.data.re, id=blockid)

getPvalues(initialModel, varlist=c('observationhour', 'x.pos', 'y.pos'),
            factorlist=c('floodebb', 'impact'))

getPvalues(initialModel)
```

`getRadiiChoices` *Function for obtaining a sequence of range parameters for the CReSS smoother*

Description

Function for obtaining a sequence of range parameters for the CReSS smoother

Usage

```
getRadiiChoices(numberofradii = 10, distMatrix)
```

Arguments

`numberofradii`	The number of range parameters for SALSA to use when fitting the CReSS smooth. The default is 8. Remember, the more parameters the longer SALSA will take to find a suitable one for each knot location.
`distMatrix`	Matrix of distances between data locations and knot locations (n x k). May be Euclidean or geodesic distances. Euclidean distances created using `makeDists`.

Details

The range parameter determines the range of the influence of each knot. Small numbers indicate local influence and large ones, global influence.

Value

This function returns a vector containing a sequence of range parameters.

References

Scott-Hayward, L.; M. Mackenzie, C.Donovan, C.Walker and E.Ashe. Complex Region Spatial Smoother (CReSS). Journal of computational and Graphical Statistics. 2013. DOI: 10.1080/10618600.2012.762920

Examples

```
# load data
data(ns.data.re)
# load knot grid data
data(knotgrid.ns)

# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(ns.data.re$x.pos, ns.data.re$y.pos), na.omit(knotgrid.ns))

# choose sequence of radii
r_seq<-getRadiiChoices(8, distMats$dataDist)
```

improve.step_2d — *Function to move knots to neighbours to see if there is any improvement in fit*

Description

Function to move knots to neighbours to see if there is any improvement in fit

Usage

```
improve.step_2d(gap, knotDist, radii, invInd, dists, gridResp, grid, explData,
  xVals, yVals, num, response, explanatory, maxIterations, fitnessMeasure,
  point, knotPoint, position, aR, BIC, track, out.lm, improveNudge, tol = 0,
  baseModel, radiusIndices, models, interactionTerm, data, initDisp)
```

Author(s)

Cameron Walker, Department of Enginering Science, University of Auckland.

knotgrid.ns — *Knot grid data for nearshore example*

Description

Knot grid data for nearshore example

knotgrid.off — *Knot grid data for offshore example*

Description

Knot grid data for offshore example

`LocalRadialFunction` *Function for creating an Gaussian basis function for a spatial smooth using the CReSS method.*

Description

This function calculates a local radial Gausiian basis matrix for use in `runSALSA2D`.

Usage

```
LocalRadialFunction(radiusIndices, dists, radii, aR)
```

Arguments

`radiusIndices`	Vector of length startKnots identifying which radii (splineParams[[1]]$radii) will be used to initialise the model
`dists`	Matrix of distances between data locations and knot locations (n x k). May be Euclidean or geodesic distances.
`radii`	Sequence of range parameters for the CReSS basis from local (small) to global (large). Determines the range of the influence of each knot.
`aR`	Index of knot locations. The index contains numbers selected by SALSA from 1 to the number of legal knot locations `na.omit(knotgrid)`. Used to specify which columns of `dists` should be used to construct the basis matrix.

Details

Calculate a local radial basis matrix for use in `runSALSA2D`. The distance matrix input may be Euclidean or geodesic distances.

Value

Returns a basis matrix with one column for each knot in `aR` and one row for every observation (i.e. same number of rows as `dists`)

Examples

```
# load data
data(ns.data.re)
# load knot grid data
data(knotgrid.ns)

splineParams<-makesplineParams(data=ns.data.re, varlist=c('observationhour'))

#set some input info for SALSA
ns.data.re$response<- ns.data.re$birds

# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(ns.data.re$x.pos, ns.data.re$y.pos), na.omit(knotgrid.ns), knotmat=FALSE)

# choose sequence of radii
r_seq<-getRadiiChoices(8, distMats$dataDist)
```

```
# using the fourth radius and picking 5 knots
basis<-LocalRadialFunction(radiusIndices=rep(4, 5), dists=distMats$dataDist, radii = r_seq,
        aR=c(3, 10, 15, 28, 31))
```

`make.gamMRSea` *Function to make model of class* `gamMRSea`

Description

Function to allow a model of class gamMRSea to be updated to include a panel structure or short-names to make summary and anova outputs more readable. Function to update an 'lm' or 'glm' model to class 'gamMRSea'.

Usage

```
make.gamMRSea(model, panelid = NULL, splineParams = NULL,
  varshortnames = NULL, gamMRSea = FALSE)
```

Arguments

model	model object of class glm or gamMRSea
panelid	vector of length of the data containing the panel identification for each row of data
splineParams	MRSea based list object
varshortnames	vector containing the short names for each variable. These are used in summary and anova
gamMRSea	logical stating whether the call of the model should be changed to 'gamMRSea' from 'glm'

Author(s)

LAS Scott-Hayward, University of St Andrews

`makeBootCIs` *Calculate percentile confidence intervals from a matrix of bootstrapped predictions*

Description

Calculate percentile confidence intervals from a matrix of bootstrapped predictions

Usage

```
makeBootCIs(preds, quants = c(0.025, 0.975))
```

Arguments

preds	matrix of bootstrap predictions where each column is a bootstrap realisation
quants	(default = c(0.025, 0.975). Vector of length two of quantiles.

Examples

```
## Not run:
makeBootCIs(bootPreds)

## End(Not run)
```

---

makeDists — *Make Euclidean distance matrices for use in CReSS and SALSA model frameworks*

Description

This function makes two Euclidean distance matrices. One for the distances between all spatial observations and all spatial knot locations. The other, if specified, is the distances between knot locations.

Usage

```
makeDists(datacoords, knotcoords, knotmat = TRUE)
```

Arguments

datacoords	Coordinates of the data locations
knotcoords	Coordinates of the legal knot locations
knotmat	(default=TRUE). Should a matrix of knot-knot distances be created

Details

The data-knot matrix is used in the CReSS basis and the knot-knot matrix is used in SALSA to determine where a nearest knot to 'move' should be.

Examples

```
# load data
data(ns.data.re)
# load knot grid data
data(knotgrid.ns)

# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(ns.data.re$x.pos, ns.data.re$y.pos), na.omit(knotgrid.ns))
```

makesplineParams *Constructing an object of spline parameters*

Description

This function makes a list object containing all of the information to fit splines to continuous data.

Usage

```
makesplineParams(data, varlist, predictionData = NULL, degree = NULL)
```

Arguments

data	Data frame containing columns of covariates listed in varlist. Column names must match with names in varlist
varlist	Vector of variable names for the covariates of interest
predictionData	Data frame containing columns of covariates listed in varlist. Column names must match with those in varlist. This parameter is used to find the maximum range of covariates between the data and prediction data. If predictionData is NULL then the range of the data is used.
degree	Vector specifying the degree of the spline. If unspecified, degree 2 is stored.

Details

The information is stored in list slots [[2]] and onward (slot [[1]] is reserved for a spatial term). Specifically:

covar. Name of covariate.

explanatory. Vector of covariate data.

knots. Knot(s) for spline fitting. This function initialises with a knot at the mean covariate value.

bd. This specifies the boundary knots. If predictionData is NULL then this is the range of the covariate data. Otherwise, the boundary knots are the maximum combined range of the data and prediction data.

degree. The degree of a B-spline. This function retuns 2 by default.

See runSALSA2D for details on the spatial slot ([[1]])

Examples

```
# load data
data(ns.data.re)
# load prediction data
data(ns.predict.data.re)

splineParams<- makesplineParams(ns.data.re, varlist=c('observationhour', 'DayOfMonth'),
                predictionData=ns.predict.data.re)
```

MRSea *MRSea*

Description

MRSea

ns.data.de *Nearshore data with decrease post-impact*

Description

A simulated dataset containing the observed counts, the effort data and other variables of grid data. The variables are as follows:

Format

A data frame with 27798 rows and 12 variables

Details

- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `area` area surveyed in the gridcell in km squared
- `floodebb` 3 level factor covariate for tides
- `observationhour` hour of observation
- `GridCode` identifier for the different grids that were surveyed
- `Year` Year of the survey
- `DavOfMonth` Day of the survey
- `MonthOfYear` Month of the survey
- `impact` numerical indicator for before (0) and after (1) impact
- `birds` observed number of birds
- `cellid` identifier for the individual records

ns.data.no *Nearshore data with no effect of impact*

Description

A simulated dataset containing the observed counts, the effort data and other variables of grid data. The variables are as follows:

Format

A data frame with 27798 rows and 12 variables

Details

- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `area` area surveyed in the gridcell in km squared
- `floodebb` 3 level factor covariate for tides
- `observationhour` hour of observation
- `GridCode` identifier for the different grids that were surveyed
- `Year` Year of the survey
- `DavOfMonth` Day of the survey
- `MonthOfYear` Month of the survey
- `impact` numerical indicator for before (0) and after (1) impact
- `birds` observed number of birds
- `cellid` identifier for the individual records

ns.data.re *Nearshore data with redistribution post-impact*

Description

A simulated dataset containing the observed counts, the effort data and other variables of grid data. The variables are as follows:

Format

A data frame with 27798 rows and 12 variables

Details

- x.pos spatial location in the horizontal axis in UTMs
- y.pos spatial location in the vertical axis in UTMs
- area area surveyed in the gridcell in km squared
- floodebb 3 level factor covariate for tides
- observationhour hour of observation
- GridCode identifier for the different grids that were surveyed
- Year Year of the survey
- DavOfMonth Day of the survey
- MonthOfYear Month of the survey
- impact numerical indicator for before (0) and after (1) impact
- birds observed number of birds
- cellid identifier for the individual records

---

ns.predict.data.de *Prediction grid data for nearshore post-impact decrease*

Description

A simulated prediction dataset containing the true counts, the effort data and other variables of grid data. The variables are as follows:

Format

A data frame with 27798 rows and 11 variables

Details

- x.pos spatial location in the horizontal axis in UTMs
- y.pos spatial location in the vertical axis in UTMs
- area Area surveyed in the gridcell in km squared
- floodebb 3 level factor covariate for tide state
- observationhour hour of observation
- GridCode identifier for the different grids that were surveyed
- Year Year of the survey
- DavOfMonth Day of the survey
- MonthOfYear Month of the survey
- impact numerical indicator for before (0) and after (1) impact
- birds true density of birds

ns.predict.data.no *Prediction grid data for nearshore no post-impact consequence*

Description

A simulated prediction dataset containing the true counts, the effort data and other variables of grid data. The variables are as follows:

Format

A data frame with 27798 rows and 11 variables

Details

- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `area` Area surveyed in the gridcell in km squared
- `floodebb` 3 level factor covariate for tide state
- `observationhour` hour of observation
- `GridCode` identifier for the different grids that were surveyed
- `Year` Year of the survey
- `DavOfMonth` Day of the survey
- `MonthOfYear` Month of the survey
- `impact` numerical indicator for before (0) and after (1) impact
- `birds` true density of birds

ns.predict.data.re *Prediction grid data for nearshore post-impact redistribution*

Description

A simulated prediction dataset containing the true counts, the effort data and other variables of grid data. The variables are as follows:

Format

A data frame with 27798 rows and 11 variables

Details

- x.pos spatial location in the horizontal axis in UTMs
- y.pos spatial location in the vertical axis in UTMs
- area Area surveyed in the gridcell in km squared
- floodebb 3 level factor covariate for tide state
- observationhour hour of observation
- GridCode identifier for the different grids that were surveyed
- Year Year of the survey
- DavOfMonth Day of the survey
- MonthOfYear Month of the survey
- impact numerical indicator for before (0) and after (1) impact
- birds true density of birds

---

plotacf *run functions to create acf matrix and plot the results*

Description

run functions to create acf matrix and plot the results

Usage

```
plotacf(acfmat)
```

Arguments

acfmat	Matrix of output from acffunc (blocks x max block length).

---

plotCumRes *Calculate cumulative residuals and plot.*

Description

The output is plots of cumulative residuals.

Usage

```
plotCumRes(model, varlist, d2k = NULL, splineParams = NULL, label = "",
  save = FALSE)
```

Arguments

model	Fitted model object (glm or gam)
varlist	Vector of covariate names (continous covariates only)
d2k	(default=NULL). Distance matrix of data to knot points. Used only if there is a LocalRadialFunction smooth in the model formula
splineParams	(default =NULL) List object containing output from runSALSA/runSALSA2D required for updating model. Used only if there is a LocalRadialFunction smooth in the model formula. See makesplineParams for details of this object.
label	Label printed at the end of the plot name to identify it if save=TRUE.
save	(default=FALSE). Logical stating whether plot should be saved into working directory.

Value

Cumulative residual plots are returned for residuals ordered by each covariate in varlist, predicted value and index of observations (temporally). The blue dots are the residuals The black line is the line of cumulative residual. On the covariate plots (those in varlist) the grey line indicates what we would expect from a well fitted covariate. i.e. one that is fitted with excessive knots.

Note: if the covariate is discrete in nature (like the example below), there will be a lot of overplotting of residuals.

Examples

```
# load data
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
           family='quasipoisson', data=ns.data.re)

plotCumRes(model, varlist=c('observationhour'))
```

---

plotRunsProfile *Calculate runs test and plot profile plot. The output is a plot of runs profiles (with p-value to indicate level of correlation)*

Description

Calculate runs test and plot profile plot. The output is a plot of runs profiles (with p-value to indicate level of correlation)

Usage

```
plotRunsProfile(model, varlist, label = "", save = FALSE)
```

Arguments

model	Fitted model object (glm or gam)
varlist	Vector of covariate names (continous covariates only)
label	Label printed at the end of the plot name to identify it when save=TRUE.
save	(default=FALSE). Logical stating whether plot should be saved into working directory.

Value

Runs profile plots are returned for residuals ordered by each covariate in varlist, predicted value and index of observations (temporally).

The black line is the line of sequences of positive or negative residuals. The vertical lines are the change between a sequence of positive to negative residuals (or vice versa).

The p-values are from a runs.test and indicate whether there is correlation in the residuals ($p<0.05$) or independence ($p>0.05$). The test statistic determines the type of correlation (positive/negative) and the result printed at the bottom of the figure.

Note: if the covariate is discrete in nature (like the example below), there will be a lot of overplotting of runs. Some jittering occurs at each discrete value (for covariates with <= 25 unique values).

Examples

```
# load data
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
            family='quasipoisson', data=ns.data.re)

plotRunsProfile(model, varlist=c('observationhour'))
```

---

predict.cress — *Function for making predictions for a model containing a CReSS basis (two dimensional local smooth).*

Description

This function calculates vector of predictions on the scale of the response or link.

Usage

```
## S3 method for class 'cress'
predict(predict.data, splineParams, g2k, model,
  type = "response", coeff = NULL)
```

Arguments

predict.data	Data frame of covariate values to make predictions to
splineParams	spline parameter object that describes the fitting of 2D and 1D splines in the model object
g2k	Matrix of distances between prediction locations and knot locations (n x k). May be Euclidean or geodesic distances.
model	Object from a GEE or GLM model
type	Type of predictions required. (default='response', may also use 'link'.
coeff	Vector of coefficients (default = NULL). To be used when bootstrapping and sampling coefficients from a distribution e.g. in do.bootstrap.cress.

Details

Calculate predictions for a model whilst centering the CReSS bases in the same way as the fitted model. Note, if there is an offset in the model it must be called 'area'.

Value

Returns a vector of predictions on either the response or link scale

Examples

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# offshore redistribution data
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
data(dis.data.re)
data(predict.data.re)
data(knotgrid.off)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# distance sampling
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
        meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)
count.data<-create.count.data(dis.data.re)

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# spatial modelling
splineParams<-makesplineParams(data=count.data, varlist=c('depth'))
#set some input info for SALSA
count.data$response<- count.data$NHAT
# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(count.data$x.pos, count.data$y.pos), na.omit(knotgrid.off))
# choose sequence of radii
r_seq<-getRadiiChoices(8,distMats$dataDist)
# set initial model without the spatial term
initialModel<- glm(response ~ as.factor(season) + as.factor(impact) + offset(log(area)),
                family='quasipoisson', data=count.data)
# make parameter set for running salsa2d
salsa2dlist<-list(fitnessMeasure = 'QICb', knotgrid = knotgrid.off,
                knotdim=c(26,14), startKnots=4, minKnots=4,
                maxKnots=20, r_seq=r_seq, gap=4000, interactionTerm="as.factor(impact)")
salsa2dOutput_k6<-runSALSA2D(initialModel, salsa2dlist, d2k=distMats$dataDist,
```

```
                    k2k=distMats$knotDist, splineParams=splineParams)

splineParams<-salsa2dOutput_k6$splineParams
# specify parameters for local radial function:
radiusIndices <- splineParams[[1]]$radiusIndices
dists <- splineParams[[1]]$dist
radii <- splineParams[[1]]$radii
aR <- splineParams[[1]]$invInd[splineParams[[1]]$knotPos]
count.data$blockid<-paste(count.data$transect.id, count.data$season, count.data$impact, sep='')
# Re-fit the chosen model as a GEE (based on SALSA knot placement) and GEE p-values
geeModel<- geeglm(formula(salsa2dOutput_k6$bestModel), data=count.data, family=poisson, id=blockid)
dists<-makeDists(cbind(predict.data.re$x.pos, predict.data.re$y.pos), na.omit(knotgrid.off),
       knotmat=FALSE)$dataDist

# make predictions on response scale
preds<-predict.cress(predict.data.re, splineParams, dists, geeModel)
```

`predict.data.de` *Prediction grid data for post-impact decrease*

Description

A simulated dataset containing the true number of birds, the effort data and other variables of prediction grid data. The variables are as follows:

Format

A data frame with 37928 rows and 8 variables

Details

- `area` area surveyed in the gridcell in km squared
- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `depth` depth in m
- `segment.id` Identifier for individual visits to the segment
- `season` Numerical indicator for the four different seasons
- `impact` Numerical indicator for before (0) and after (1) impact
- `truth` number of birds

predict.data.no *Prediction grid data for no post-impact consequence*

Description

A simulated dataset containing the true number of birds, the effort data and other variables of prediction grid data. The variables are as follows:

Format

A data frame with 37928 rows and 8 variables

Details

- `area` area surveyed in the gridcell in km squared
- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `depth` depth in m
- `segment.id` Identifier for individual visits to the segment
- `season` Numerical indicator for the four different seasons
- `impact` Numerical indicator for before (0) and after (1) impact
- `truth` number of birds

predict.data.re *Prediction grid data for post-impact redistribution*

Description

A simulated dataset containing the true number of birds, the effort data and other variables of prediction grid data. The variables are as follows:

Format

A data frame with 37928 rows and 8 variables

Details

- `area` area surveyed in the gridcell in km squared
- `x.pos` spatial location in the horizontal axis in UTMs
- `y.pos` spatial location in the vertical axis in UTMs
- `depth` depth in m
- `segment.id` Identifier for individual visits to the segment
- `season` Numerical indicator for the four different seasons
- `impact` Numerical indicator for before (0) and after (1) impact
- `truth` number of birds

`predict.gamMRSea` *Function for making predictions for a model containing a CReSS basis (two dimensional local smooth).*

Description

This function calculates vector of predictions on the scale of the response or link.

Usage

```
## S3 method for class 'gamMRSea'
predict(predict.data, g2k = NULL, model,
  type = "response", coeff = NULL)
```

Arguments

`predict.data`	Data frame of covariate values to make predictions to
`g2k`	Matrix of distances between prediction locations and knot locations (n x k). May be Euclidean or geodesic distances.
`model`	Object from a GEE or GLM model
`type`	Type of predictions required. (default='response', may also use 'link').
`coeff`	Vector of coefficients (default = NULL). To be used when bootstrapping and sampling coefficients from a distribution e.g. in `do.bootstrap.cress`.

Details

Calculate predictions for a model whilst centering the CReSS bases in the same way as the fitted model. Note, if there is an offset in the model it must be called 'area'.

Value

Returns a vector of predictions on either the response or link scale

Examples

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# offshore redistribution data
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
data(dis.data.re)
data(predict.data.re)
data(knotgrid.off)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# distance sampling
dis.data.re$survey.id<-paste(dis.data.re$season,dis.data.re$impact,sep="")
result<-ddf(dsmodel=~mcds(key="hn", formula=~1), data=dis.data.re, method="ds",
        meta.data=list(width=250))
dis.data.re<-create.NHAT(dis.data.re,result)
count.data<-create.count.data(dis.data.re)

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# spatial modelling
```

```
splineParams<-makesplineParams(data=count.data, varlist=c('depth'))
#set some input info for SALSA
count.data$response<- count.data$NHAT
# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(count.data$x.pos, count.data$y.pos), na.omit(knotgrid.off))
# choose sequence of radii
r_seq<-getRadiiChoices(8,distMats$dataDist)
# set initial model without the spatial term
initialModel<- glm(response ~ as.factor(season) + as.factor(impact) + offset(log(area)),
                family='quasipoisson', data=count.data)
# make parameter set for running salsa2d
salsa2dlist<-list(fitnessMeasure = 'QICb', knotgrid = knotgrid.off,
                 knotdim=c(26,14), startKnots=4, minKnots=4,
                maxKnots=20, r_seq=r_seq, gap=4000, interactionTerm="as.factor(impact)")
salsa2dOutput_k6<-runSALSA2D(initialModel, salsa2dlist, d2k=distMats$dataDist,
                  k2k=distMats$knotDist, splineParams=splineParams)


# make predictions on response scale
preds<-predict.gamMRSea(predict.data.re, dists, salsa2dOutput_k6$bestModel)
```

`QICb` *Function to calculate QICb*

Description

Function to calculate QICb

Usage

```
QICb(model)
```

Arguments

`model`	Model of class geeglm

Author(s)

Lindesay Scott-Hayward, University of St Andrews

`qzibinom` *qzibinom function from the VGAM package*

Description

qzibinom function from the VGAM package

Usage

```
qzibinom(p, size, prob, pstr0 = 0)
```

Author(s)

VGAM package

`return.reg.spline.fit` *Code for adaptively spacing knots for a given covariate.*

Description

Code for adaptively spacing knots for a given covariate.

Usage

```
return.reg.spline.fit(response, explanatory, degree, minKnots, maxKnots,
  startKnots, gap, winHalfWidth, fitnessMeasure = "BIC",
  maxIterations = 100, initialise = TRUE, initialKnots = NULL,
  baseModel = NULL, bd, spl, interactionTerm = interactionTerm,
  suppress.printout = FALSE)
```

Arguments

`response`	vector of response data for the modelling process
`explanatory`	vector of covariate to find knots for
`degree`	degree of the spline to be used
`minKnots`	minimum number of knots to fit
`maxKnots`	maximum number of knots to fit
`startKnots`	number of equally spaced knots to start with (between minKnots and maxKnots)
`gap`	minimum gap between knots (in unit of measurement of `explanatory`)
`winHalfWidth`	Half-width of window used to calculate region with biggest average residual magnitude
`fitnessMeasure`	(default=BIC). Measure used to evaluate the fit. Other options are AIC, AICc, BIC, QAIC, QAICc, QICb (Quasi-Likelihood Information Criterion with log(n) penalty)
`maxIterations`	exchange/improve heuristic will terminate after `maxIterations` if still running
`initialise`	(default = TRUE). Logical stating whether or not to start with equally spaced knots (TRUE) or user specified locations (FALSE)
`initialKnots`	If `initialise=FALSE` then the start locations for the knots are specified in `initialKnots`
`baseModel`	starting model for SALSA to use. Must not contain the covariate in `explanatory`
`bd`	the x-coordinate of the boundary knots of `explanatory`
`spl`	"bs" uses b-spline, "cc" uses cyclic cubic, "ns" uses natural cubic spline for fitting smooth to `explanatory`

Author(s)

Cameron Walker, Department of Enginering Science, University of Auckland.

return.reg.spline.fit.2d *Wrapper function for running SALSA2D*

Description

Wrapper function for running SALSA2D

Usage

```
return.reg.spline.fit.2d(splineParams, startKnots, winHalfWidth,
  fitnessMeasure = "BIC", maxIterations = 10, tol = 0, baseModel = NULL,
  radiusIndices = NULL, initialise = TRUE, initialKnots = NULL,
  interactionTerm = NULL, knot.seed = 10, suppress.printout = FALSE)
```

Author(s)

Cameron Walker, Department of Enginering Science, University of Auckland.

runACF *run functions to create acf matrix and plot the results*

Description

run functions to create acf matrix and plot the results

Usage

```
runACF(block, model, store = FALSE, save = F, suppress.printout = FALSE)
```

Arguments

block	Vector of blocks that identify data points that are correlated
model	Fitted model object (glm or gam)
store	(default=F). Logical stating whether a list of the matrix of correlations is stored (output from acffunc.)
save	(default=FALSE). Logical stating whether plot should be saved into working directory.
suppress.printout	(Default: FALSE. Logical stating whether to show a printout of block numbers to assess progress. 'FALSE' will show printout.

Value

Plot of lag vs correlation. Each grey line is the correlation for each individual block in block. The red line is the mean values for each lag.

If store=TRUE then the matrix of correlations (nblocks x length_max_block) is returned and plotacf may be used to plot the acf.

Author(s)

LAS Scott-Hayward, University of St Andrews

Examples

```
# load data
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
          family='quasipoisson', data=ns.data.re)

ns.data.re$blockid<-paste(ns.data.re$GridCode, ns.data.re$Year, ns.data.re$MonthOfYear,
                    ns.data.re$DayOfMonth, sep='')
ns.data.re$blockid<-as.factor(ns.data.re$blockid)

runACF(ns.data.re$blockid, model, suppress.printout=TRUE)
```

---

runDiagnostics *functions to create observed vs fitted and fitted vs scaled pearsons residual plots*

Description

functions to create observed vs fitted and fitted vs scaled pearsons residual plots

Usage

```
runDiagnostics(model, plotting = "b", save = FALSE)
```

Arguments

model	Fitted model object (glm or gam)
plotting	Plotting options (default='b'). b: returns both plots, f: returns observed vs fitted only and r: returns scale pearsons residual plot only.
save	(default=FALSE). Logical stating whether plot should be saved into working directory.

Value

Two plots:

Observed vs Fitted
: Plot of observed vs fitted with concordence correlation and marginal R-squared printed in the plot title.

Fitted vs scaled Pearsons residuals
: The red line is a locally weighted least squares regression line of all of the residuals.

Examples

```
# load data
data(ns.data.re)

model<-glm(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
           family='quasipoisson', data=ns.data.re)

runDiagnostics(model)
```

`runInfluence` *Assessing the influece of each correlated block on both the precision of the parameter estimates (COVRATIO statistics) and the sensitivity of model predictions (PRESS statistics).*

Description

Assessing the influece of each correlated block on both the precision of the parameter estimates (COVRATIO statistics) and the sensitivity of model predictions (PRESS statistics).

Usage

```
runInfluence(model, id, d2k = NULL, splineParams = NULL, save = FALSE)
```

Arguments

`model`	Fitted model object (glm, gamMRSea or gam)
`id`	blocking structure
`d2k`	(`default=NULL`). (n x k) Matrix of distances between all data points in `model` and all valid knot locations.
`splineParams`	(`default=NULL`). List object containng output from runSALSA (e.g. knot locations for continuous covariates). See `makesplineParams` for more details of this object.
`save`	(`default=FALSE`). Logical stating whether plot should be saved into working directory.

Details

Always run `timeInfluenceCheck` first to see how long it will take to produce the plots.

Value

Two plots one each for COVRATIO and PRESS statistics, giving the influence of each block on precision of the parameter estimates and the sensitivity of model predictions. List object:

`influenceData`	List of `blocks`, COVRATIO statistics and PRESS statistics used for making the plot of PRESS and COVRATIO statistics.
`influencePoints`	Row id of blocks in `influenceData` that lie outside the 95% quantile of COVRATIO statistics and above the 95% quantile of PRESS statistics.

Examples

```
# load data
data(ns.data.re)

ns.data.re$blockid<-paste(ns.data.re$GridCode, ns.data.re$Year, ns.data.re$MonthOfYear,
                    ns.data.re$DayOfMonth, sep='')
ns.data.re$blockid<-as.factor(ns.data.re$blockid)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)

timeInfluenceCheck(model, ns.data.re$blockid)

## Not run:
# **WARNING** this example takes a long time
influences<-runInfluence(model, ns.data.re$blockid)

## End(Not run)
```

`runPartialPlots` *Plot partial plots for each of the variables listed in* `factorlist.in` *or* `varlist.in`.

Description

Plot partial plots for each of the variables listed in `factorlist.in` or `varlist.in`.

Usage

```
runPartialPlots(model, data, factorlist.in = NULL, varlist.in = NULL,
  showKnots = FALSE, type = "response", partial.resid = FALSE,
  save = FALSE, savedata = F, label = NULL)
```

Arguments

`model`	Fitted model object (glm or gam)
`data`	Data frame of data information used to fit `model`
`factorlist.in`	(`default=NULL`). Vector or names of factor variables
`varlist.in`	(`default=NULL`). Vector of names of continuous variables
`showKnots`	(`default=FALSE`). Logical stating whether knot locations should be plotted.
`type`	(`default='responss'`). Character stating whether to return partial plots on the scale of the link function or the response.
`partial.resid`	(`default=FALSE`). Logical stating whether to include partial residuals on the plot.
`save`	(`default=FALSE`). Logical stating whether plot should be saved into working directory.
`savedata`	(`default=FALSE`). Logical stating whether the data used to make the plots should be saved into the working directory. The object is called PartialData_'variablename'.RData
`label`	(`default=NULL`). This enables the user to specify a character label for the plots saved to the working directory. This may also be used to specify an alternative directory.

Value

Partial plots, one for each covariate in `factorlist.in` and `varlist.in`

Author(s)

LAS Scott-Hayward, University of St Andrews

Examples

```
#' # load data
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
          family='quasipoisson', data=ns.data.re)

runPartialPlots(model, ns.data.re, factorlist.in=c('floodebb', 'impact'),
                varlist.in=c('observationhour'))
runPartialPlots(model, ns.data.re, factorlist.in=c('floodebb', 'impact'),
                varlist.in=c('observationhour'), type='link')
runPartialPlots(model, ns.data.re, factorlist.in=c('floodebb', 'impact'),
                varlist.in=c('observationhour'), partial.resid=TRUE)
```

---

runSALSA1D *Running SALSA for continuous one-dimensional covariates.*

Description

This function finds spatially adaptive knot locations for one or more continuous one-dimensional covariates.

Usage

```
runSALSA1D(initialModel, salsa1dlist, varlist, factorlist = NULL,
  predictionData = NULL, varlist_cyclicSplines = NULL,
  splineParams = NULL, datain, removal = FALSE, panelid = NULL,
  suppress.printout = FALSE)
```

Arguments

`initialModel`	The best fitting `CReSS` model with no continuous covariates specified. This must be a model of class `glm`.
`salsa1dlist`	Vector of objects required for `runSALSA1D`: `fitnessMeasure`, `minKnots_1d`, `maxKnots_1d`, `startKnots_1d` `degree`, `maxIterations` `gap`.
`varlist`	Vector of variable names for the covariates required for knot selection
`factorlist`	vector of factor variables specified in `initialModel`. Specified so that a check can be made that there are non-zero counts in all levels of each factor. Uses the function `checkfactorlevelcounts`. Default setting is NULL.
`predictionData`	The data for which predictions are to be made. Column names must correspond to the data in `initialModel`. If predictionData is not specified (`NULL`), then the range of the data is used to create the smooth terms.

varlist_cyclicSplines
: Vector of variable names for covariates to be modelled with cyclic cubic splines. This must be a subset of varlist.The default is NULL

splineParams
: List object containing information for fitting splines to the covariates in varlist. If not specified (NULL) this object is created and returned. See makesplineParams for details.

datain
: Data used to fit the initial Model.

removal
: (Default: FALSE). Logical stating whether a selection procedure should be done to choose smooth, linear or removal of covariates. If FALSE all covariates are returned and smooth. If TRUE then cross-validation is used to make model selection choices. The folds are specified by a column in the dataset called foldid.

panelid
: Vector denoting the panel identifier for each data point (if robust standard errors are to be calculated). Defaults to data order index if not given.

suppress.printout
: (Default: FALSE. Logical stating whether to show the analysis printout.

Details

There must be columns called response (response variable) and foldid (for cross-validation calculation) in the data used in the initial model to be fitted. If the data is proportion, then there should be two columns called successess and failures.

The object salsa1dlist contains parameters for the runSALSA1D function.

fitnessMeasure. The criterion for selecting the 'best' model. Available options: AIC, AIC_c, BIC, QIC_b.

minKnots_1d. Minimum number of knots to be tried.

maxKnots_1d. Maximum number of knots to be tried.

startKnots_1d. Starting number of knots (spaced at quantiles of the data).

degree. The degree of the B-spline. Does not need to be specified if splineParams is a parameter in runSALSA1D.

maxIterations.The exchange/improve steps will terminate after maxIterations if still running.

gaps. The minimum gap between knots (in unit of measurement of explanatory), usually set to zero.

minKnots_1d, maxKnots_1d, startKnots_1d and gaps are vectors the same length as varlist. This enables differing values of these parameters for each covariate.

The initial model contains all the factor level covariates and any covariates of interest that are not specified in the varlist argument of runSALSA1D

Note: The algorithm may remove variables in varlist but not the variables in factorlist. If there is no better model than with a knot at the mean, the output will include that covariate with a knot at the mean. The best model with a given smooth term is tested both against a model with the term as linear or removed. Cross-Validation is used in the selection process.

Value

A list object is returned containing 4 elements:

bestModel
: A model object of class gam.MRSea from the best model fitted

`modelFits1D` A list object with an element for each new term fitted to the model. The first element is a model fitted with a knot at the mean for each of the covariates (startmodel) in `varlist`. Within the first element, the current fit and formula of the start model.

The second element is the result of SALSA on the first term in `varlist`. Within this element:

- `term`: term of interest
- `kept`: Statement of whether the term is kept in the model (yes- initial knots, yes - new knots, yes -linear or no)
- `basemodelformula`: the resulting model formula. If `kept=yes` or `kept=linear` then the term of interest is included in the model otherwise it is removed.
- `knotSelected`: the knots chosen for the term of interest (NA if term removed or linear)
- `baseModelFits`: fit statistics for the resulting formula
- `modelfits`: fit statistics for the model with the term included (same as resulting formula if `kept=yes`)

This continues till all covariates in `varlist` have been through SALSA.

`splineParams` The updated spline parameter object, with the new (if chosen) knot locations for each covariate in `varlist`

`fitstat` The final fit statistic of `bestModel`. The type of statistic was specified in `salsa1dlist`.

`keptvarlist` The covariates from `varlist` that have been retained in the model

Author(s)

Lindesay Scott-Hayward

References

Walker, C.; M. Mackenzie, C. Donovan and M. O'Sullivan. SALSA - a Spatially Adaptive Local Smoothing Algorithm. Journal of Statistical Computation and Simulation, 81(2):179-191, 2010

Examples

```
# load data
data(ns.data.re)
# load prediction data
data(ns.predict.data.re)

varlist=c('observationhour', 'DayOfMonth')

# make column with foldid for cross validation calculation
ns.data.re$foldid<-getCVids(ns.data.re, folds=5)

# set initial model without the spline terms in there
# (so all other non-spline terms)
ns.data.re$response<- ns.data.re$birds
initialModel<- glm(response ~ as.factor(floodebb) + as.factor(impact) + offset(log(area)),
                   family='quasipoisson',data=ns.data.re)

#set some input info for SALSA
salsa1dlist<-list(fitnessMeasure = 'QBIC', minKnots_1d=c(2,2), maxKnots_1d = c(5, 5),
                  startKnots_1d = c(2,2), degree=c(2,2), maxIterations = 10, gaps=c(1,1))
```

```
# run SALSA
salsa1dOutput<-runSALSA1D(initialModel, salsa1dlist, varlist=varlist,
               factorlist=c('floodebb', 'impact'),
               ns.predict.data.re, splineParams=NULL, datain=ns.data.re, removal=TRUE)
```

---

runSALSA2D *Running SALSA for a spatial smooth with a CReSS basis*

Description

This function fits a spatially adaptive two dimensional smooth of spatial coordinates with knot number and location selected by SALSA.

Usage

```
runSALSA2D(model, salsa2dlist, d2k, k2k, splineParams = NULL, chooserad = F,
  panels = NULL, suppress.printout = FALSE)
```

Arguments

model	A model with no spatial smooth
salsa2dlist	Vector of objects required for runSALSA2D: fitnessMeasure, knotgrid, startKnots, minKnots, codemaxKnots, r_seq, gap, interactionTerm.
d2k	(n x k) Matrix of distances between all data points in model and all valid knot locations specified in knotgrid
k2k	(k x k) Matrix of distances between all valid knot locations specified in knotgrid
splineParams	(default =NULL) List object containng output from runSALSA (e.g. knot locations for continuous covariates)
chooserad	logical flag. If FALSE (default) then the range parameter of the basis is chosen after the knot location and number. If TRUE, the range is assessed at every iteration of a knot move/add/drop.
panels	Vector denoting the panel identifier for each data point (if robust standard errors are to be calculated). Defaults to data order index if not given.
suppress.printout	(Default: FALSE. Logical stating whether to show the analysis printout.

Details

There must be a column called response in the data, which is the response variable used in the initial model to be fitted.

The object salsa2dlist contains parameters for the runSALSA2D function.

fitnessMeasure. The criterion for selecting the 'best' model. Available options: AIC, AIC_c, BIC, QIC_b.

knotgrid. A grid of legal knot locations. Must be a regular grid with c(NA, NA) for rows with an illegal knot. An illegal knot position may be outside the study region or on land for a marine species for example. May be made using getKnotgrid.

knotdim. The dimensions of the knot grid as a vector. (x, y). If knotgrid is made using getKnotgrid then the default dimensions are c(100, 100).

startknots. Starting number of knots (initialised as spaced filled locations).

minKnots. Minimum number of knots to be tried.

maxKnots. Maximum number of knots to be tried.

r_seq. Sequence of range parameters for the CReSS basis from local (small) to global (large). Determines the range of the influence of each knot. Sequence made using getRadiiChoices.

gap. The minimum gap between knots (in unit of measurement of coordinates). interactionTerm. Specifies which term in baseModel the spatial smooth will interact with. If NULL no interaction term is fitted.

Value

The spline paramater object that is returned as part of the model object now contains a list in the first element (previously reserved for the spatial component). This list contains the objects required for the SALSA2D fitting process:

knotDist	Matrix of knot to knot distances (k x k). May be Euclidean or geodesic distances. Must be square and the same dimensions as nrows(na.omit(knotgrid)). Created using makeDists.
radii	Sequence of range parameters for the CReSS basis from local (small) to global (large). Determines the range of the influence of each knot.
dist	Matrix of distances between data locations and knot locations (n x k). May be Euclidean or geodesic distances. Euclidean distances created using makeDists.
gridresp	The first column of knotgrid.
grid	Index of knotgrid locations. Should be same length as knotgrid but with x=integer values from 1 to number of unique x-locations and y= integer values from 1 to number of unique y-locations.
datacoords	Coordinates of the data locations
response	Vector of response data for the modelling process
knotgrid	Grid of legal knot locations. Must be a regular grid with c(NA, NA) for rows with an illegal knot.
minKnots	Minimum number of knots to be tried.
maxKnots	Maximum number of knots to be tried.
gap	Minimum gap between knots (in unit of measurement of datacoords)
radiusIndices	Vector of length startKnots identifying which radii (splineParams[[1]]$radii) will be used for each knot location (splineParams[[1]]$knotPos)
knotPos	Index of knot locations. The index identifies which knots (i.e. which rows) from knotgrid were selected by SALSA
invInd	This is a vector of length the number of rows of knotgrid. It is used to translate between knotgrid (used in SALSA) and na.omit(knotgrid) (used in dist and LocalRadialFunction).

Author(s)

Lindesay Scott-Hayward (University of St Andrews), Cameron Walker (University of Auckland)

References

Scott-Hayward, L.; M. Mackenzie, C.Donovan, C.Walker and E.Ashe. Complex Region Spatial Smoother (CReSS). Journal of computational and Graphical Statistics. 2013. doi: 10.1080/10618600.2012.762920

Scott-Hayward, L.. Novel Methods for species distribution mapping including spatial models in complex regions: Chapter 5 for SALSA2D methods. PhD Thesis, University of St Andrews. 2013

Examples

```
# load data
data(ns.data.re)
# load prediction data
data(ns.predict.data.re)
# load knot grid data
data(knotgrid.ns)

splineParams<-makesplineParams(data=ns.data.re, varlist=c('observationhour'))

#set some input info for SALSA
ns.data.re$response<- ns.data.re$birds

# make distance matrices for datatoknots and knottoknots
distMats<-makeDists(cbind(ns.data.re$x.pos, ns.data.re$y.pos), na.omit(knotgrid.ns))

# choose sequence of radii
r_seq<-getRadiiChoices(8, distMats$dataDist)

# set initial model without the spatial term
# (so all other non-spline terms)
initialModel<- glm(response ~ as.factor(floodebb) + as.factor(impact) + offset(log(area)),
                   family='quasipoisson', data=ns.data.re)

# make parameter set for running salsa2d
# I have chosen a gap parameter of 1000 (in metres) to speed up the process.
# Note that this means there cannot be two knots within 1000m of each other.

salsa2dlist<-list(fitnessMeasure = 'QICb', knotgrid = knotgrid.ns, knotdim = c(7, 9),
                  startKnots=6, minKnots=4, maxKnots=20, r_seq=r_seq, gap=1000,
                  interactionTerm="as.factor(impact)")

salsa2dOutput_k6<-runSALSA2D(initialModel, salsa2dlist, d2k=distMats$dataDist,
                             k2k=distMats$knotDist, splineParams=splineParams)
```

`runsTest` *Runs Test for Randomness*

Description

This function performs the runs test for randomness. Users can choose whether to plot the correlation graph or not, and whether to test against two-sided, negative or positive correlation. NAs from the data are omitted. An empirical distribution may be used for the distribution of the test statistic under the null hypothesis of independence.

Usage

```
runsTest(y, plot.it = FALSE, alternative = c("two.sided",
  "positive.correlated", "negative.correlated"), emp.distribution = NULL)
```

Arguments

y	a numeric vector of data values
plot.it	logical flag. If 'TRUE' then the graph will be plotted. If 'FALSE', then it is not plotted.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "negative.correlated" or "positive.correlated"
emp.distribution	vector containing the empirical distribution of test statistics under the nyll hypothesis. Generated using getEmpDistribution.

Details

On the graph observations which are less than the sample median are represented by letter "A" in red color, and observations which are greater or equal to the sample median are represented by letter "B" in blue color.

Value

A list with the following components.

statistic the value of the standardized Runs statistic. p.value the p-value for the test. data.name a character string giving the names of the data. alternative a character string describing the alternative hypothesis.

References

Mendenhall, W (1982), Statistics for Management and Economics, 4th Ed., 801-807, Duxbury Press, Boston.

J.L. Gastwirth; Y.R. Gel, W. L. Hui, V. Lyubchich, W. Miao and K. Noguchi (2015). lawstat: Tools for Biostatistics, Public Policy, and Law. R package version 3.0

Examples

```
data(ns.data.re)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)

runsTest(residuals(model))

## Empirical distribution:

simData<-generateNoise(n=500, response=fitted(model), family='poisson')

empdist<-getEmpDistribution(500, simData, model, data=ns.data.re, plot=FALSE,
                            returnDist=TRUE,dots=FALSE)
runsTest(residuals(model), emp.distribution=empdist)
```

summary.gamMRSea *Summarising model fits from models fitted using the MRSea package.*

Description

Summarising model fits from models fitted using the MRSea package.

Usage

```
## S3 method for class 'gamMRSea'
summary(object, dispersion = NULL, varshortnames = NULL,
  ...)
```

Arguments

object	an object of class "gamMRSea", usually, a result of a call from the MRSea package.
dispersion	the dispersion parameter for the family used. Either a single numerical value or NULL (the default), when it is inferred from object (see 'Details').
varshortnames	vector stating the short versions of the covariate names if required.
...	further arguments passed to or from other methods.
x	an object of class "summary.gamMRSea", usually, a result of a call to summary.gamMRSea.
correlation	logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed.
digits	the number of significant digits to use when printing.
symbolic.cor	logical. If TRUE, print the correlations in a symbolic form (see symnum) rather than as numbers.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.

Details

print.summary.gamMRSea tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives 'significance stars' if signif.stars is TRUE. The coefficients component of the result gives the estimated coefficients and their estimated standard errors (raw and robust), together with their ratio (from robust s.e.). The third column gives the robust standard errors calculated using the sandwich estimator. If no correlation is present, the second and third columns are the same as the sandwich estimator is not used when data points are independent. The fourth column is labelled Wald and gives the Wald test statistic, based on the robust standard errors. The fifth column gives the two-tailed p-value corresponding to the Wald test ().

Aliased coefficients are omitted in the returned object but restored by the print method.

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print summary(object)$correlation directly.

summary.gamMRSea returns an object of class "summary.gamMRSea", a list with components call the component from object. family the component from object. deviance the component from object. contrasts the component from object. df.residual the component from object. null.deviance the component from object. df.null the component from object. deviance.resid the deviance residuals: see residuals.glm. coefficients the matrix of coefficients, standard errors, z-values and p-values.

Aliased coefficients are omitted. aliased named logical vector showing if the original coefficients are aliased. dispersion either the supplied argument or the inferred/estimated dispersion if the latter is NULL. df a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of coefficients (including aliased ones). cov.unscaled the unscaled (dispersion = 1) estimated covariance matrix of the estimated coefficients. cov.scaled ditto, scaled by dispersion. correlation (only if correlation is true.) The estimated correlations of the estimated coefficients. symbolic.cor (only if correlation is true.) The value of the argument symbolic.cor.

Note

Code adapted from `summary.glm`

Author(s)

Lindesay Scott-Hayward, Univeristy of St Andrews.

Examples

```
# load data
data(ns.data.re)
ns.data.re$foldid<-getCVids(ns.data.re, folds=5)

model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)
summary(model)
```

---

thinModels *function to thin the number of models*

Description

function to thin the number of models

Usage

```
thinModels(models)
```

Author(s)

Cameron Walker, Department of Enginering Science, University of Auckland.

timeInfluenceCheck *Timing check to see how long it will take to run* runInfluence.

Description

Timing check to see how long it will take to run runInfluence.

Usage

```
timeInfluenceCheck(model, id, d2k = NULL, splineParams = NULL)
```

Arguments

model	Fitted model object (glm, gamMRSea or gam)
id	blocking structure
d2k	(default=NULL). (n x k) Matrix of distances between all data points in model and all valid knot locations.
splineParams	(default=NULL). List object containng output from runSALSA (e.g. knot locations for continuous covariates). See makesplineParams for more details of this object.

Examples

```
# load data
data(ns.data.re)

ns.data.re$blockid<-paste(ns.data.re$GridCode, ns.data.re$Year, ns.data.re$MonthOfYear,
                    ns.data.re$DayOfMonth, sep='')
ns.data.re$blockid<-as.factor(ns.data.re$blockid)
model<-gamMRSea(birds ~ observationhour + as.factor(floodebb) + as.factor(impact),
              family='poisson', data=ns.data.re)

timeInfluenceCheck(model, ns.data.re$blockid)
```

which.bin *Determining the distance bin*

Description

For a vector of perpendicular (or radial) distances, this function determines which distance bin it belongs to (given the input of cut points) and adds the beginning and end points of the respective distance bins in new colunns in dis.data called "distbegin" and "distend".

Usage

```
which.bin(dis.data, cutpoints)
```

Arguments

`dis.data`	A data frame with distance data for which perpendicular (or radial) distances are recorded in the `distance` column
`cutpoints`	A vector of cut points of the intervals (this function is not set up to deal with left-truncation)

Details

If a value in `dis.data$distance` matches a cut point in `cutpoints` exactly, the value of `dis.data.re$distance` will be attributed to the bin that is closer to the line/point unless the value of `dis.data.re$distance` is 0.

E.g. if `cutpoints=c(0,1,2,3)`, `dis.data$distance=2` will be attributed to interval 2 (and not 3).

Value

The `dis.data` data frame to which columns "distbegin" and "distend" were added giving the beginning and end cutpoints of the bin that the respective `dis.data$distance` belongs to.

Index

∗Topic **datasets**
dis.data.de, 10
dis.data.no, 11
dis.data.re, 12
ns.data.de, 34
ns.data.no, 35
ns.data.re, 35
ns.predict.data.de, 36
ns.predict.data.no, 37
ns.predict.data.re, 37
predict.data.de, 42
predict.data.no, 43
predict.data.re, 43

acffunc, 3
AICh, 4
anova.gamMRSea, 4

bootstrap.orig.data, 5

checkfactorlevelcounts, 6
choose.radii, 6
create.bootcount.data, 7
create.bootstrap.data, 8
create.count.data, 9
create.NHAT, 10

dis.data.de, 10
dis.data.no, 11
dis.data.re, 12
do.bootstrap.cress, 12
do.bootstrap.cress.robust, 15
do.bootstrap.gam, 17
drop.step_2d, 19

exchange.step_2d, 19

fit.thinPlate_2d, 19

gamMRSea, 20
generateNoise, 20
getCV_CReSS, *21*, 22
getCVids, 21, *22*
getDifferences, 23
getDispersion, 24
getEmpDistribution, 24, *57*
getKnotgrid, 25, *54*, *55*
getPlotdimensions, 26
getPvalues, 27
getRadiiChoices, 28, *55*
glm, *20*

improve.step_2d, 29

knotgrid.ns, 29
knotgrid.off, 29

LocalRadialFunction, 30, *39*

make.gamMRSea, 31
makeBootCIs, 31
makeDists, *28*, 32, *55*
makesplineParams, *22*, 33, *39*, *49*, *52*, *60*
MRSea, 34
MRSea-package *(*MRSea*)*, 34

ns.data.de, 34
ns.data.no, 35
ns.data.re, 35
ns.predict.data.de, 36
ns.predict.data.no, 37
ns.predict.data.re, 37

plotacf, 38
plotCumRes, 38
plotRunsProfile, 39
predict.cress, 40
predict.data.de, 42
predict.data.no, 43
predict.data.re, 43
predict.gamMRSea, 44

QICb, 45
qzibinom, 45

rbinom, *20*
return.reg.spline.fit, 46
return.reg.spline.fit.2d, 47
runACF, 47
runDiagnostics, 48

runInfluence, 49
runPartialPlots, 50
runs.test, *40*
runSALSA1D, 51
runSALSA2D, *30*, *33*, 54
runsTest, 56
rzibinom, *20*

summary.gamMRSea, 58

thinModels, 59
timeInfluenceCheck, *49*, 60

which.bin, 60