

Package ‘hzar’

July 2, 2014

Type Package

Title Hybrid Zone Analysis using R

Version 0.2-5

Date 2013-09-02

Author Graham Derryberry

Maintainer Graham Derryberry <asterion@alum.mit.edu>

Description A collection of tools for modeling the shape of 1 dimensional clines.

License GPL (>= 2)

LazyLoad yes

Depends R (>= 2.10.0), MCMCpack, foreach, coda

NeedsCompilation no

Repository CRAN

Date/Publication 2013-09-23 18:03:53

R topics documented:

hzar-package	3
hzar.AIC.default	4
hzar.AIC.hzar.obsDataGroup	6
hzar.chain.doSeq	7
hzar.copyModelLabels	8
hzar.cov.rect	9
hzar.dataGroup.add	10
hzar.dataGroup.null	11
hzar.dBernoulli.LL	12
hzar.doCLTData1DRaw	13
hzar.doFit	14

hzar.doFit.multi	15
hzar.doMolecularData1DPops	16
hzar.doMorphoSets	17
hzar.eval.clineLL	18
hzar.extract.obsData	19
hzar.extract.old.model.gen	20
hzar.first.fitRequest.old.ML	21
hzar.gen.cline	22
hzar.get.ML.cline	23
hzar.getCredCut	23
hzar.getCredParam	24
hzar.getCredParamRed	25
hzar.getLLCutParam	25
hzar.make.cline	26
hzar.make.clineLLfunc.old.ML	27
hzar.make.dataGroup	28
hzar.make.fitRequest	29
hzar.make.fzCline	31
hzar.make.LLfunc.null	32
hzar.make.mcmcParam	33
hzar.make.obsDataGroup	34
hzar.makeCline1DFreq	35
hzar.makeTraitObsData	38
hzar.map.dms2deg	38
hzar.map.greatCircleDistance	39
hzar.map.latLong.dms	41
hzar.map.latLongSites	42
hzar.mcmc.bindLL	43
hzar.meta.init	44
hzar.model.addBoxReq	45
hzar.multiFitRequest	47
hzar.next.fitRequest	48
hzar.overPlot.fzCline	49
hzar.plot.cline	50
hzar.plot.fzCline	51
hzar.plot.obsData	52
hzar.profile.dataGroup	53
hzar.qScores	54
hzar.sameModel	55
hzar.yLabel	56
manakinLocations	56
manakinMolecular	57
manakinMorphological	58

hzar-package

Hybrid Zone Analysis using R

Description

A collection of tools for modeling the shape of 1 dimensional clines.

Details

Package: hzar
Type: Package
Version: 0.2-5
Date: 2013-09-02
License: GPL (>= 2)
LazyLoad: yes

Author(s)

Graham Derryberry Maintainer: Graham Derryberry <asterion@alum.mit.edu>

References

Brumfield, R. T., R. W. Jernigan, D. B. McDonald, and M. J. Braun. 2001. Evolutionary implications of divergent clines in an avian (*Manacus*: Aves) hybrid zone. *Evolution* 55:2070-2087.

Gay, L., P.-A. Crochet, D. A. Bell, and T. Lenormand. 2008. Comparing clines on molecular and phenotypic traits in hybrid zones: a window on tension zone models. *Evolution* 62:2789-2806.

Szymura, J., and N. H. Barton. 1986. Genetic analysis of a hybrid zone between the fire-bellied toads, *Bombina bombina* and *B. variegata*, near Cracow in southern Poland. *Evolution* 40:1141-1159.

Szymura, J., and N. H. Barton. 1991. The genetic structure of the hybrid zone between the fire-bellied toads *Bombina bombina* and *B. variegata*: comparisons between transects and between loci. *Evolution* 45:237-261.

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
hzar.plot.obsData(mknAdaA);
mknAdaAmodel <-
```

```

hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600);
mknAdaAmodelFitR <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                               mknAdaA,
                               verbose=FALSE);
mknAdaAmodelFitR$mcmcParam$chainLength <- 2e3;
mknAdaAmodelFitR$mcmcParam$burnin <- 5e2;
mknAdaAmodelFit <- hzar.doFit(mknAdaAmodelFitR)
plot(hzar.mcmc.bindLL(mknAdaAmodelFit))
mknAdaAmodelData <-
  hzar.dataGroup.add(mknAdaAmodelFit);
## Not run:
mknAdaAmodelData <-
  hzar.dataGroup.add(
    mknAdaAmodelData,
    hzar.chain.doSeq(hzar.next.fitRequest(mknAdaAmodelFit)));
hzar.plot.cline(mknAdaAmodelData);
hzar.plot.fzCline(mknAdaAmodelData);

## End(Not run)
print(hzar.getLLCutParam(mknAdaAmodelData, c("center", "width")));
mknAdaAmodelNull <- hzar.dataGroup.null(mknAdaA);
mknAdaAdGs <- list(clineModel = mknAdaAmodelData,
                  nullModel = mknAdaAmodelNull);
mknAdaAoDG <- hzar.make.obsDataGroup(mknAdaAdGs);
mknAdaAoDG <- hzar.copyModelLabels(mknAdaAdGs, mknAdaAoDG);
hzar.plot.cline(mknAdaAoDG);
print(hzar.AICc.hzar.obsDataGroup(mknAdaAoDG));

```

hzar.AIC.default *Calculate the AIC score.*

Description

Calculate the AIC or the corrected AIC (AICc) for the given likelihood, number of parameters and number of observations.

Extracts the parameters as needed when passed the correct hzar object.

Usage

```

hzar.AIC.default(maxLL, param.count)
hzar.AICc.default(maxLL, param.count, nObs)
hzar.AIC.hzar.cline(cline)
hzar.AICc.hzar.cline(cline, nObs)
hzar.AIC.hzar.dataGroup(dataGroup)
hzar.AICc.hzar.dataGroup(dataGroup)

```

Arguments

maxLL	The maximum log likelihood value.
param.count	The number of free parameters, also known as the number of degrees of freedom.
nObs	The number of samples observed.
cline	A hzar.cline object.
dataGroup	A hzar.dataGroup object.

Details

The formula for AIC used is $2 * (\text{param.count} - \text{maxLL})$.

The formula for AICc used is: $\text{AIC} + 2 * \text{param.count} * (\text{param.count} + 1) / (\text{nObs} - \text{param.count} - 1)$

Value

The AIC or AICc score calculated.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[AIC hzar.AIC.hzar.obsDataGroup](#)

Examples

```
print(hzar.AIC.default(-8,3))
print(hzar.AICc.default(-8,3,30))

data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$aDa.A,
                             manakinMolecular$aDa.nSamples);
hzar.plot.obsData(mknAdaA);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed",tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel,-30,600);
mknAdaAmodelFitR <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                              mknAdaA,
                              verbose=FALSE);
print(hzar.AIC.hzar.dataGroup(hzar.fit2DataGroup(mknAdaAmodelFitR)))

mknAdaAcline <- hzar.gen.cline(list(center=300,width=10),
```

```

                                mknAdaAmodelFitR);
print(hzar.AIC.hzar.cline(mknAdaAcline));

```

```

hzar.AIC.hzar.obsDataGroup
                                Generate an AIC score table.

```

Description

Calculate the AIC or corrected AIC score table for the given hzar.obsDataGroup object. There will be one score generated for each model associated with this object.

Usage

```

hzar.AIC.hzar.obsDataGroup(obsDataGroup, label = "AIC",
                             show.count = FALSE, show.param = FALSE)
hzar.AICc.hzar.obsDataGroup(obsDataGroup, label = "AICc",
                              show.count = FALSE, show.param = FALSE)

```

Arguments

obsDataGroup	The hzar.obsDataGroup object for which to generate the score table.
label	The name to use for the score column.
show.count	Include an addition column with a count of the number of free parameters.
show.param	Currently does nothing. Include additional columns with the parameter values of the maximum likelihood model for each score.

Value

A data frame with at least one column, with the label specified above.

If the models in the obsDataGroup have names, then the rownames of the data frame are the models' names.

If show.count is TRUE, the result will have one additional column with the label "count". For each score this column will have the number of free parameters.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[AIC hzar.AIC.default](#)

Examples

```
##TODO
```

hzar.chain.doSeq	<i>Repeatedly run the optimizer in series to tune the optimizer.</i>
------------------	--

Description

hzar.chain.doSeq cyclically calls [hzar.doFit](#) and [hzar.next.fitRequest](#) in order to optimize the covariance matrix driving mcmc process, [MCMCmetrop1R](#).

Usage

```
hzar.chain.doSeq(hzar.request,  
                 count = 3,  
                 collapse = FALSE,  
                 announce.complete = "Chain Complete")
```

Arguments

hzar.request	The hzar.fitRequest object to use initially.
count	How many iterations to perform.
collapse	If TRUE, if all iterations succeed return a single hzar.fitRequest object with all of the results concatenated instead of a list of hzar.fitRequest objects.
announce.complete	Diagnostics message to print when the chain has completed.

Details

For each iteration, [hzar.doFit](#) is called using `hzar.request` and the result is added to the results list.

For the second and all subsequent iterations, `hzar.request` is generated by [hzar.next.fitRequest](#) using the results of the previous iteration. If [hzar.next.fitRequest](#) fails, the results list is returned.

When `count` iterations are performed and if `collapse` is TRUE, the results list is reduced to a single [hzar.fitRequest](#) object with the covariance matrix set to the last matrix used. All other fields are set as if the concatenated results were the results of a single call to [hzar.doFit](#).

Value

Either a list of all successful [hzar.fitRequest](#) objects, or, if `collapse` is TRUE, a single [hzar.fitRequest](#) object with all of the results concatenated.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.doFit](#) [MCMCmetrop1R](#) [hzar.next.fitRequest](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600)
mknAdaAmodelFit <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                               mknAdaA,
                               verbose=FALSE);
mknAdaAmodelFit$mcmcParam$chainLength <- 1e3;
mknAdaAmodelFit$mcmcParam$burnin <- 50;
mknAdaAmodelFit$mcmcParam$thin <- 10;
str(hzar.chain.doSeq(mknAdaAmodelFit, count=2));
```

`hzar.copyModelLabels` *Copy names from one hzar object to another.*

Description

Set the [names](#) of the list of [hzar.dataGroup](#) objects contained in a [hzar.obsDataGroup](#) object using the names from either a named list of [hzar.dataGroup](#) objects or another [hzar.obsDataGroup](#) object.

Usage

```
hzar.copyModelLabels(group1, group2)
```

Arguments

`group1` An object from which to extract model labels.
`group2` A [hzar.obsDataGroup](#) into which the model labels need to be inserted.

Value

The updated `group2`.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.dataGroup](#) [hzar.obsDataGroup](#) [names](#)

hzar.cov.rect

Generate a covariance matrix for the cline optimizer.

Description

These methods are intended to generate covariance matrices suitable for use with MCMCmetrop1R.

Usage

```
hzar.cov.rect(clineLLfunc, param.lower, param.upper,
              pDiv = 11, random = 0, passCenter = FALSE)
hzar.cov.mcmc(clineLLfunc, mcmcRaw,
              pDiv = 15, random = 10000, passCenter = FALSE)
```

Arguments

clineLLfunc	The log likelihood function of the parameters.
param.lower	The minimum boundary of the region of parameter space to consider.
param.upper	The maximum boundary of the region of parameter space to consider.
pDiv	If generating a covariance matrix using a lattice, the lattice should have this many points on an edge.
random	Use random number of points drawn from a uniform likelihood space to generate the covariance matrix. If 0, use a lattice to generate the covariance matrix.
passCenter	Should weighted mean of the parameter space be returned.
mcmcRaw	A mcmc object used to refine the covariance matrix.

Details

This method is adaptive, refining the pDiv and random parameters until either it can generate a useable matrix without too high of a memory cost, or random > 1e9 (use a 1 billion or more samples).

Value

A square matrix with a width equal to the number of free parameters.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[MCMCmetrop1R](#)

hzar.dataGroup.add *Prepare optimizer output for analysis.*

Description

Intended to group multiple fits of the same model and the same observation data into a single object. As it works with [hzar.fitRequest](#) objects, it is simpler to use than [hzar.make.dataGroup](#).

Usage

```
hzar.dataGroup.add(dataGroup, fitRequestL = list(), doPar = FALSE)
hzar.fit2DataGroup(fitRequest, doPar = FALSE)
```

Arguments

dataGroup	A single hzar.dataGroup object to update. If <code>fitRequestL</code> is a list of length 0, this argument may also be a list of hzar.fitRequest or hzar.dataGroup objects.
fitRequestL	A hzar.fitRequest or hzar.dataGroup object, a list of such objects, or a deep list of such objects.
fitRequest	A single hzar.fitRequest object. A hzar.dataGroup object may also be used.
doPar	This argument is passed to hzar.eval.clineLL .

Value

A [hzar.dataGroup](#) object.

Note

A deep list of T is a list of length greater > 1 that contains only deep lists of T or objects of class T.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.dataGroup](#) [hzar.obsDataGroup](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$sada.A,
                             manakinMolecular$sada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600)
mknAdaAmodelFit <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                               mknAdaA,
                               verbose=FALSE);
mknAdaAmodelFit$mcmcParam$chainLength <- 1e3;
mknAdaAmodelFit$mcmcParam$burnin <- 5e2;
mknAdaAFit <- hzar.doFit(mknAdaAmodelFit);
str(hzar.fit2DataGroup(mknAdaAFit));
```

hzar.dataGroup.null	<i>Datagroup placeholder for the null model (frequency independent of location)</i>
---------------------	---

Description

Generates a [hzar.dataGroup](#) object representing a fit of the null model to a [hzar.obsData](#) object.

Usage

```
hzar.dataGroup.null(obsData)
```

Arguments

obsData The [hzar.obsData](#) object for which to generate a [hzar.dataGroup](#) object.

Value

A [hzar.dataGroup](#) object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.dataGroup](#) [hzar.obsData](#) [hzar.make.LLfunc.null](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
mkn.AdaA.null <- hzar.dataGroup.null(mknAdaA)
hzar.plot.cline(mkn.AdaA.null);
```

hzar.dBernoulli.LL *Transformations of Scalar Data into Bernoulli Trials*

Description

This method operates on a collection of sampled scalar values and the sample location site factor. It calculates a score for each "cut" value that can split the samples into unique groups, and either returns those scores, the cut value with the best score, or a table of the frequencies of the sample values that are less than the cut value with the best score.

Usage

```
hzar.dBernoulli.LL(values, locations, getMax = FALSE, getProbs = FALSE)
```

Arguments

values	The sample values to use.
locations	The factor grouping sample values by location.
getMax	Should this method return the best cut value?
getProbs	Should this method return the table of frequencies of sample values which are less than the best cut value?

Details

The score for a cut value is the information of learning the location of a sample conditioned on the knowledge of the whether the sample value is greater or less than the cut value. This score indicates how little a cut value distinguishes between localities.

The best scoring cut value is the one that determines the greatest amount of information about the sample location and therefore it is the one with lowest score.

Value

Either a vector of scores, a single cut value, or a table of frequencies of the sample values that are less than the cut value with the best score.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.doMorphoSets](#) [hzar.obsData](#)

`hzar.doCLTData1DRaw` *Create a `hzar.obsData` object using a table of individual traits.*

Description

Create a [hzar.obsData](#) object using a table of individual traits.

Usage

```
hzar.doCLTData1DRaw(distance, traitValue)
hzar.doNormalData1DRaw(site.dist, traitSite, traitValue)
hzar.mapSiteDist(siteID, distance)
```

Arguments

<code>distance</code>	The distance of the sampling site. For <code>hzar.doCLTData1DRaw</code> , samples at the same distance are treated as being from the same sampling site.
<code>traitValue</code>	The value of the trait of the individual sampled.
<code>traitSite</code>	The id of site where the individual was found.
<code>site.dist</code>	A named vector mapping site id codes to the distance of the sampling site. The function <code>hzar.mapSiteDist</code> returns a suitable vector.
<code>siteID</code>	The list of id codes associated with the sampling site. This list should be identical in length to <code>distance</code> , each entry must be unique, and the order of the sites referenced must be identical for <code>distance</code> and <code>siteID</code> .

Details

For `hzar.doCLTData1DRaw`:

If for any locality, there is only a small number of samples taken, warnings will be issued.

If at any locality, the sample variance is 0, a warning is issued, and additional variance is included by estimating the amount of variance ignored due to measurement error.

For `hzar.doNormalData1DRaw`:

Use the helper function `hzar.mapSiteDist` to generate `site.dist`.

The [hzar.obsData](#) object created is meant for use with the models constructed using [hzar.makeCline1DNormal](#).

Value

A [hzar.obsData](#) object, using the site distances and sample means and variances as calculated from the values given.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.obsData](#)

hzar.doFit

Run the optimizer.

Description

Run the optimizer using the parameters listed in the [hzar.fitRequest](#) given.

Usage

```
hzar.doFit(fitRequest)
```

Arguments

fitRequest The [hzar.fitRequest](#) object to be processed

Value

An updated [hzar.fitRequest](#) object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest MCMCmetrop1R](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$sada.A,
                             manakinMolecular$sada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600)
mknAdaAmodelFit <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel,
                              mknAdaA,
```

```

                                verbose=FALSE);
mknAdaAmodelFit$mcmcParam$chainLength <- 5e3;
mknAdaAmodelFit$mcmcParam$burnin <- 1e2;
str(hzar.doFit(mknAdaAmodelFit));

```

hzar.doFit.multi *Run hzar fit commands on a list of hzar.fitRequest objects*

Description

These methods simplify repeated calling of `hzar.doFit` or `hzar.chain.doSeq` while taking advantage of `%dopar%` if requested.

Usage

```

hzar.doFit.multi(mFitR, doPar = FALSE, inOrder = TRUE)
hzar.doChain.multi(mFitR, doPar = FALSE, inOrder = TRUE, ...)

```

Arguments

<code>mFitR</code>	Provide a list of <code>hzar.fitRequest</code> objects. Use <code>hzar.multiFitRequest</code> to ensure independent seeds and to request independent chains.
<code>doPar</code>	Use <code>%dopar%</code> ?
<code>inOrder</code>	Should the results be returned in order? If FALSE, the results are returned in the order of completion. See <code>foreach</code> for more information.
<code>...</code>	Additional arguments to pass to <code>hzar.chain.doSeq</code>

Value

A list of the fitted `hzar.fitRequest` objects.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

`hzar.chain.doSeq` `hzar.doFit` `hzar.fitRequest` `hzar.multiFitRequest` `%dopar%` `foreach`

Examples

```

##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

```

```
hzar.doMolecularData1DPops
```

Generate a hzar.obsData object using summary data about each locality

Description

Creates a hzar.obsData object using the observations given. The likelihood function used is chosen based on the method called.

Usage

```
hzar.doMolecularData1DPops(distance, pObs, nEff,
                           siteID=paste("P", 1:length(distance), sep=""),
                           ylim=extendrange(c(0, 1)))
hzar.doCLTData1DPops(distance, muObs, varObs, nEff)
hzar.doNormalData1DPops(distance, muObs, varObs, nEff,
                         siteID=paste("P", 1:length(distance), sep=""),
                         ylim=NULL)
```

Arguments

All arguments should be of the same length.

The distance of each locality. If the same distance is given multiple times, then multiple localities are assumed to be at that distance.

pObs	The observed frequency at each locality.
nEff	The effective number of samples observed at each locality.
muObs	The mean trait value observed at each site.
varObs	The trait variance observed at each site.
ylim	The ylim to use when plotting the observed data.
siteID	The identifier to use for each sampling site.

Details

For hzar.doCLTData1DPops, varObs must not be less than zero, and should be greater than zero. If equal to zero, the method will attempt to estimate the number of significant digits in the observed trait value, and use that to calculate additional variance due to measurement error at each site, and add that variance to the observed variance for each site.

Value

A hzar.obsData object with the following structure.

frame	A data.frame composed of the arguments.
model.LL	A function of one argument that returns a log likelihood. The argument is a function of distance that estimates either frequency or trait value as appropriate.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

References

Szymura, J., and N. H. Barton. 1986. Genetic analysis of a hybrid zone between the fire-bellied toads, *Bombina bombina* and *B. variegata*, near Cracow in southern Poland. *Evolution* 40:1141-1159.

Szymura, J., and N. H. Barton. 1991. The genetic structure of the hybrid zone between the fire-bellied toads *Bombina bombina* and *B. variegata*: comparisons between transects and between loci. *Evolution* 45:237-261.

See Also

[manakinMolecular](#) [hzar.plot.obsData](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
print(str(mknAdaA));
hzar.plot.obsData(mknAdaA);
```

hzar.doMorphoSets	<i>Make hzar.obsData objects from scalar observations using reference tables</i>
-------------------	--

Description

Perform a Bernoulli transform on a table of scalar traits of sampled individuals while using a separate table of localities.

Usage

```
hzar.doMorphoSets(traitNames, tDist, tLocCol, tDDistCol, tValues, tVLocCol)
```

Arguments

traitNames	The columns of tValues to transform.
tDist	A data.frame with a column of locality identifiers and a column of locality distances. See manakinLocations for an example.
tLocCol	The name of the column of tDist with locality identifiers.
tDDistCol	The name of the column of tDist with locality distances.

tValues	A data.frame of observed scalar traits of individuals. See manakinMorphological for an example.
tVLocCol	The name of the column of tValues with locality identifiers where the sample was taken.

Value

A list of [hzar.obsData](#) objects. The values of traitNames are used as names. The Bernoulli likelihood function is for each [hzar.obsData](#) object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[manakinLocations](#) [manakinMorphological](#) [hzar.plot.obsData](#)

Examples

```
data(manakinMorphological);
data(manakinLocations);
mkn <-
  hzar.doMorphoSets("beard.length",
                    tDist=manakinLocations,
                    tDLocCol="LocalityID",
                    tDDistCol="distance",
                    tValues=manakinMorphological,
                    tVLocCol="Locality")
print(str(mkn));
hzar.plot.obsData(mkn$beard.length);
```

`hzar.eval.clineLL` *Calculate the Log Likelihoods of the table of parameters provided.*

Description

Using the likelihood function and the table of parameter values provided, calculate the likelihood of each row of parameter values.

Usage

```
hzar.eval.clineLL(data, llFunc, doPar = FALSE)
```

Arguments

data	A data.frame of the free parameter values. Each column name should match the corresponding parameter name.
llFunc	The log likelihood function to use.
doPar	If TRUE, use %dopar% to iterate over the rows of data.

Value

A data.frame with a single column (model.LL) containing the log likelihoods for each row of data.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[%dopar%](#)

hzar.extract.obsData *Extract the observation data used by the optimizer.*

Description

Most hzar objects have at least an indirectly if not directly associated hzar.obsData object. This function returns that hzar.obsData object.

Usage

```
hzar.extract.obsData(fitRequest)
```

Arguments

fitRequest A [hzar.dataGroup](#), [hzar.obsDataGroup](#), [hzar.fitRequest](#) or [hzar.obsData](#) object, or a likelihood function generated by hzar.

Value

The associated hzar.obsData object.

Note

This function if passed a hzar.obsData objects returns the same object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

Examples

```
##TODO
```

`hzar.extract.old.model.gen`*Extract information about the clineMetaModel used.*

Description

These methods extract the `$func` and `$req` methods from the `clineMetaModel` object used initially. This includes modifications specified by the `$parameterTypes` item.

Usage

```
hzar.extract.old.model.gen(fitRequest)
hzar.extract.old.model.req(fitRequest)
```

Arguments

`fitRequest` A `hzar.fitRequest` or a `hzar.dataGroup` object, or a log likelihood function generated by `hzar`.

Value

A method which takes the cline parameters as arguments, with the fixed parameters set to default values.

The result of that method is boolean for `hzar.extract.old.model.req` and a function of distance for `hzar.extract.old.model.gen`.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.dataGroup](#) [clineMetaModel](#) [hzar.first.fitRequest.old.ML](#)

Examples

```
data(manakinMolecular);
ASdata <-
  hzar.doMolecularData1DPops(distance=manakinMolecular$distance,
                             pObs=manakinMolecular$ak2.A,
                             nEff=manakinMolecular$ak2.nSamples);
ASclineM <- hzar.makeCline1DFreq(data=ASdata,scaling="none", tails="none");
ASclineM$func;
ASclineM$req;
ASfitA <- hzar.first.fitRequest.old.ML(ASclineM,ASdata)
hzar.extract.old.model.gen(ASfitA)
hzar.extract.old.model.req(ASfitA)
```

`hzar.first.fitRequest.old.ML`*Generate a ML based hzar.fitRequest using a meta model structure.*

Description

This method generates a `hzar.fitRequest` object suitable for `hzar.doFit`.

Usage

```
hzar.first.fitRequest.old.ML(model, obsData, verbose = TRUE)
hzar.first.fitRequest.gC(gModel, obsData, verbose = TRUE)
```

Arguments

<code>model</code>	A <code>clineMetaModel</code> object.
<code>gModel</code>	A <code>clineMetaModel</code> object generated by <code>hzar.makeCline1DNormal</code> .
<code>obsData</code>	The <code>hzar.obsData</code> object to which the meta model is to be fit.
<code>verbose</code>	Should <code>MCMCmetrop1R</code> be verbose?

Value

A `hzar.fitRequest` object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.obsData](#) [clineMetaModel](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodelFit <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                              mknAdaA,
                              verbose=FALSE);
mknAdaAmodelFit$mcmcParam$chainLength <- 1e5;
str(mknAdaAmodelFit);
```

hzar.gen.cline	<i>Make a hzar.cline object using the given parameters and model.</i>
----------------	---

Description

Make a [hzar.cline](#) object using the given parameters and model.

Usage

```
hzar.gen.cline(free.parameters, fitRequest)
```

Arguments

free.parameters	A named list of free parameter values.
fitRequest	An object referring to the model that should be used, such as a hzar.fitRequest or hzar.dataGroup object.

Value

The [hzar.cline](#) object requested.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.cline](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
hzar.plot.obsData(mknAdaA);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600);
mknAdaAmodelFitR <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                              mknAdaA,
                              verbose=FALSE);
mknAdaAcline <- hzar.gen.cline(list(center=300,width=10),
                              mknAdaAmodelFitR);
```

```
str(mknAdaAcline);  
hzar.plot.cline(mknAdaAmodelFitR);  
hzar.plot.cline(mknAdaAcline,add=TRUE);
```

hzar.get.ML.cline *Extract the maximum likelihood cline.*

Description

A method for retrieving the fitted cline object (a [hzar.cline](#) object) with the maximum likelihood calculated from a fitted cline model (a [hzar.dataGroup](#) object or a successful [hzar.fitRequest](#))

Usage

```
hzar.get.ML.cline(fitRequest)
```

Arguments

fitRequest A [hzar.dataGroup](#) object or a successful [hzar.fitRequest](#) object.

Value

A [hzar.cline](#) object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.gen.cline](#)

hzar.getCredCut *Get the minimum log likelihood of sufficient credibility.*

Description

This method first calculates the cumulative distribution using the sampled likelihoods and then selects the greatest likelihood that has no more than rejectionPercent of the cumulative likelihood distribution less than that likelihood.

Usage

```
hzar.getCredCut(dataGroup, rejectionPercent = 0.05)
```

Arguments

dataGroup The [hzar.dataGroup](#) object to analyze.
rejectionPercent The proportion of the cumulative likelihood distribution to reject.

Value

The greatest likelihood that has no more than rejectionPercent of the cumulative likelihood distribution less than that likelihood.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.dataGroup](#)

`hzar.getCredParam` *Select a credible subset of a collection of parameters.*

Description

Select the subset of the generated samples with a likelihood greater than the result of [hzar.getCredCut](#).

Usage

```
hzar.getCredParam(dataGroup, rejectionPercent = 0.05)
```

Arguments

dataGroup The [hzar.dataGroup](#) object to analyze.
rejectionPercent The proportion of the cumulative likelihood distribution to reject.

Value

A [data.frame](#) of the subset of the generated samples.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[data.frame](#) [hzar.getCredCut](#) [hzar.dataGroup](#)

hzar.getCredParamRed *Create a 95% credibility hzar.fzCline object*

Description

Generate a [hzar.fzCline](#) of the parameter subset selected from a [hzar.dataGroup](#) by [hzar.getCredParam](#) with a rejectionPercent of 0.05.

Usage

```
hzar.getCredParamRed(dataGroup)
```

Arguments

dataGroup The [hzar.dataGroup](#) to analyze.

Value

A [hzar.fzCline](#) object for the samples selected by [hzar.getCredParam](#).

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.make.fzCline](#) [hzar.getCredParam](#) [hzar.dataGroup](#) [hzar.fzCline](#)

hzar.getLLCutParam *Get the region of parameter space close to the maximum likelihood*

Description

This function returns the range of parameter values that are within two log likelihood units of the maximum likelihood for a provided character vector of parameters.

Usage

```
hzar.getLLCutParam(dataGroups, params, cutValue = 2)
```

Arguments

dataGroups Either a [hzar.dataGroup](#) object, or a list of [hzar.dataGroup](#) objects.
params The parameters to report.
cutValue The number of log likelihood units to retain.

Value

A `data.frame`, with 2 columns for each parameter requested, containing the maximum and minimum parameter value within `cutValue` log likelihood units of the maximum likelihood observed.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.dataGroup](#)

Examples

```
##TODO
```

```
hzar.make.cline      Make a hzar.cline object.
```

Description

This method creates a `hzar.cline` object, which describes a cline model with specific parameter values. A log likelihood of the parameters can be assigned to this object. See [hzar.gen.cline](#) for a simpler method which just requires the free parameters and a cline model reference.

Usage

```
hzar.make.cline(free.parameters, parameters, func, LL, isValid = is.function(func))
```

Arguments

<code>free.parameters</code>	The optimized parameter values for this cline.
<code>parameters</code>	All of the parameter values for this cline.
<code>func</code>	The estimator function for this cline.
<code>LL</code>	The log likelihood of this cline.
<code>isValid</code>	Is this cline valid?

Value

A `hzar.cline` object. A list with the values:

<code>param.free</code>	See <code>free.parameters</code> above
<code>param.all</code>	See <code>parameters</code> above
<code>clineFunc</code>	See <code>func</code> above
<code>logLike</code>	See <code>LL</code> above
<code>isValid</code>	See <code>isValid</code> above

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.gen.cline](#)

hzar.make.clineLLfunc.old.ML

Generate a Log Likelihood method for the cline model.

Description

A low-level method for assembling a cline likelihood method. See [hzar.first.fitRequest.old.ML](#) for an easier to use method for working with clines.

Using the arguments given, assemble either a maximum likelihood (ML) or bayesian (bayes) function with a single argument theta. The result is suitable for [MCMCmetrop1R](#).

Usage

```
hzar.make.clineLLfunc.old.ML(param.free.names, param.fixed,
  param.check.func, meta.cline.func, model.LL, LLrejectedModel = -1e+08)
hzar.make.clineLLfunc.old.bayes(param.free.names, param.fixed,
  param.check.func, meta.cline.func, model.LL, prior.LL,
  LLrejectedModel = -1e+08)
```

Arguments

param.free.names	The names of the free parameters.
param.fixed	A named list of the fixed parameters.
param.check.func	A boolean function of the parameters, which returns true if the model described is valid.
meta.cline.func	A function of the parameters, which returns a function of distance estimating frequency.
model.LL	A function which takes one argument and returns a log likelihood. The argument is a function of distance estimating frequency.
LLrejectedModel	A finite log likelihood of an invalid model.
prior.LL	A function of the parameters, which returns the log likelihood of those parameters.

Value

A function with a single argument `theta` that returns a finite log likelihood.
`Theta` is a named list of the free parameters for a cline.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[MCMCmetrop1R](#) [hzar.first.fitRequest.old.ML](#)

Examples

```
##TODO
```

```
hzar.make.dataGroup Make a hzar.dataGroup object with given parameters.
```

Description

Make a `hzar.dataGroup` object with given parameters. This method only needs the `mcmc` chain to encapsulate and the log likelihood function generated by [hzar.make.clineLLfunc.old.ML](#) or [hzar.make.clineLLfunc.old.bayes](#).

Use [hzar.dataGroup.add](#) and [hzar.fit2DataGroup](#) instead of this method as they operate directly on a [hzar.fitRequest](#) object.

Usage

```
hzar.make.dataGroup(data.mcmc,
                    llFunc,
                    ML.cline = NULL,
                    doPar = FALSE,
                    data.LL = hzar.eval.clineLL(llFunc = llFunc,
                                                data = data.mcmc,
                                                doPar = doPar),
                    data.param = as.data.frame(data.mcmc),
                    obsData = hzar.extract.obsData(llFunc))
```

Arguments

<code>data.mcmc</code>	The <code>mcmc</code> chain to encapsulate.
<code>llFunc</code>	The log likelihood function of the model.
<code>ML.cline</code>	The hzar.cline of maximum likelihood. Automatically calculated from <code>data.mcmc</code> if <code>NULL</code> .
<code>doPar</code>	Argument passed to hzar.eval.clineLL .

data.LL	The log likelihood of each row of data.mcmc.
data.param	data.mcmc as a data.frame.
obsData	The hzar.obsData object backing the hzar.dataGroup created.

Value

Object of class hzar.dataGroup

llFunc	llFunc from above.
data.mcmc	data.mcmc from above.
data.param	data.mcmc as a data.frame.
data.LL	The log likelihood of each row of data.param.
ML.cline	See hzar.get.ML.cline .
obsData	The hzar.obsData object backing llFunc.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.eval.clineLL](#) [hzar.extract.obsData](#) [hzar.dataGroup.add](#) [hzar.fit2DataGroup](#)

hzar.make.fitRequest *Generate a hzar.fitRequest with the specified attributes.*

Description

This method is meant to generate an arbitrary hzar.fitRequest object to pass to [hzar.doFit](#). Note that this method does not check its arguments for validity. Use [hzar.first.fitRequest.old.ML](#) or [hzar.next.fitRequest](#) for a more convenient method for generating hzar.fitRequest objects.

Usage

```
hzar.make.fitRequest(modelParameters,
                    covMatrix,
                    clineLLfunc,
                    mcmcParameters,
                    mcmcRaw = NULL,
                    fit.run = FALSE,
                    fit.success = FALSE)
```

Arguments

<code>modelParameters</code>	A named list of at least length 2. See details for format.
<code>covMatrix</code>	A covariance matrix for the gaussian proposal distribution as described in MCMCmetrop1R . Should be a square matrix of dimension equal to the number of free parameters, which can be generated by hzar.cov.rect or hzar.cov.mcmc . Although NULL is acceptable, it is extremely likely that <code>hzar.doFit</code> will crash.
<code>clineLLfunc</code>	A function of theta that returns a log likelihood. It is best to use the results of hzar.make.clineLLfunc.old.ML , hzar.make.clineLLfunc.old.bayes , or hzar.make.LLfunc.null .
<code>mcmcParameters</code>	The parameters controlling the operation of the mcmc process. Use the results from hzar.make.mcmcParam .
<code>mcmcRaw</code>	The mcmc object created by a successful run of MCMCmetrop1R . Useful if you wish to import old mcmc objects for use with hzar.
<code>fit.run</code>	Has this particular request been run? Note that this does not update automatically; the result of hzar.doFit is a new object with <code>mcmcRaw</code> , <code>fit.run</code> , and <code>fit.success</code> updated as appropriate. Primarily affects the behavior of hzar.next.fitRequest .
<code>fit.success</code>	Has this particular request been run successfully? Note that this does not update automatically; the result of hzar.doFit is a new object with <code>mcmcRaw</code> , <code>fit.run</code> , and <code>fit.success</code> updated as appropriate. Primarily affects the behavior of hzar.next.fitRequest .

Details

For `modelParameters`, the list must have the following entries:

- `$initA` named list with the initial values of the free parameters. Used as `theta.init` in [MCMCmetrop1R](#).
- `$tuneA` named list with the tuning values of the free parameters. Used as `tune` in [MCMCmetrop1R](#).

Value

A `hzar.fitRequest` object.

A list with values:

<code>\$modelParam</code>	<code>modelParameters</code> from above.
<code>\$cM</code>	<code>covMatrix</code> from above.
<code>\$llFunc</code>	<code>clineLLfunc</code> from above.
<code>\$mcmcParam</code>	<code>mcmcParameters</code> from above.
<code>\$mcmcRaw</code>	<code>mcmcRaw</code> from above.

and with attributes:

<code>"fit.run"</code>	<code>fit.run</code> from above.
<code>"fit.success"</code>	<code>fit.success</code> from above.

Note

Although `modelParameters` only needs to contain `$init` and `$tune`, `modelParam` will sometimes contain `$fixed`, `$lower` and `$upper`. These are artifacts of `hzar.make.clineLLfunc.old.ML` and can be safely ignored.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.first.fitRequest.old.ML](#) [hzar.next.fitRequest](#) [hzar.cov.mcmc](#) [hzar.cov.rect](#) [hzar.make.clineLLfunc.old.ML](#)
[hzar.make.clineLLfunc.old.bayes](#) [hzar.make.LLfunc.null](#) [hzar.make.mcmcParam](#) [MCMCmetrop1R](#)

<code>hzar.make.fzCline</code>	<i>Create a <code>hzar.fzCline</code> object</i>
--------------------------------	--

Description

Compile a list of [hzar.cline](#) objects into a `hzar.fzCline` object.

Usage

```
hzar.make.fzCline(clineList)
```

Arguments

`clineList` A list of clines.

Value

A `hzar.fzCline` object.

`clines` `clineList`

`listFuncInt` A function of a scalar `x`

`fzCline` A function of a numeric series over which `clineList` is evaluated and the series of maximum and minimum values is returned.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

hzar.make.LLfunc.null *Likelihood function for the null model*

Description

Likelihood function for the null model (frequency independent of location).

Usage

```
hzar.make.LLfunc.null(obsData,  
                      model.LL = obsData$model.LL,  
                      LLrejectedModel = -1e+08)
```

Arguments

obsData A [hzar.obsData](#) object.

model.LL The likelihood function for the observed data.

LLrejectedModel A finite likelihood to return when the likelihood calculated is $-\text{INF}$.

Value

A function of $p\text{Est}$, that returns the likelihood that $p\text{Est}$ is the mean frequency at all observed localities.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.obsData](#)

Examples

```
##TODO
```

hzar.make.mcmcParam *Set the parameters controlling the optimizer environment.*

Description

Defines the general parameters controlling an MCMC process. This includes the burnin, chainLength, diagnostic output, and the random number generator seed, as used by the [MCMCmetrop1R](#) optimizer.

Usage

```
hzar.make.mcmcParam(chainLength, burnin, verbosity, thin,
  seedStreamChannel = 1, useSeedStream = TRUE, mersenneSeed = 12345,
  lecuyerSeed = rep(12345, 6))
```

Arguments

chainLength	Defines how many generations to run the MCMC process after completing burnin.
burnin	Defines how many states to generate and discard at the beginning of the chain.
verbosity	If 0, output nothing while mcmc process is running. If positive, print the model state every verbosity generations after burn-in.
thin	Keep only states whose number of generations after burn-in is evenly divisible by thin.
seedStreamChannel	Set the channel number used in the lecuyer random number generator.
useSeedStream	If TRUE, use the lecuyer random number generator in the MCMC process. If FALSE, use the mersenne twister random generator in the MCMC process.
mersenneSeed	Sets the seed value for mersenne twister. Expects a numeric of length 1.
lecuyerSeed	Sets the seed value for lecuyer random number generator. Expects a numeric of length 6.

Value

A list of 5 values:

chainLength	The value of the mcmc argument for MCMCmetrop1R .
burnin	The value of the burnin argument for MCMCmetrop1R .
verbosity	The value of the verbose argument for MCMCmetrop1R .
thin	The value of the thin argument for MCMCmetrop1R .
seed	The value of the seed argument for MCMCmetrop1R .

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[MCMCmetrop1R](#)

Examples

```
##TODO
```

```
hzar.make.obsDataGroup
```

Collect optimizer output from multiple models for analysis.

Description

This method collects optimizer output based on the same [hzar.obsData](#) object. It automatically creates [hzar.dataGroup](#) objects as needed, as well as combining any objects which use the same model as determined by [hzar.sameModel](#).

Usage

```
hzar.make.obsDataGroup(dataGroups, obsDataGroup = NULL)
```

Arguments

<code>dataGroups</code>	A list of hzar.dataGroup or hzar.fitRequest objects to include.
<code>obsDataGroup</code>	The hzar.obsDataGroup to which <code>dataGroups</code> will be added. If NULL, an empty hzar.obsDataGroup will be added.

Value

A [hzar.obsDataGroup](#) object.

<code>data.groups</code>	A list of hzar.dataGroup objects, each with a unique meta models. See hzar.sameModel .
<code>obsData</code>	The hzar.obsData object shared by <code>data.groups</code>

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.sameModel](#) [hzar.obsData](#) [hzar.dataGroup](#)

Examples

```

data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$aDa.A,
                             manakinMolecular$aDa.nSamples);
hzar.plot.obsData(mknAdaA);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed",tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel,-30,600);
mknAdaAmodelFitR <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                               mknAdaA,
                               verbose=FALSE);
mknAdaAmodelFitR$mcmcParam$chainLength <- 5e3;
mknAdaAmodelFitR$mcmcParam$burnin <- 5e2;
mknAdaAmodelFit <- hzar.doFit(mknAdaAmodelFitR)
mknAdaAmodelData <-
  hzar.dataGroup.add(mknAdaAmodelFit);
## Not run:
mknAdaAmodelFitL <-
  hzar.chain.doSeq(hzar.next.fitRequest(mknAdaAmodelFit), count=3);

## End(Not run)
mknAdaAmodelNull <- hzar.dataGroup.null(mknAdaA);
mknAdaAdGs <- list(clineModel = mknAdaAmodelData,
                  nullModel = mknAdaAmodelNull);
mknAdaAoDG <- hzar.make.obsDataGroup(mknAdaAmodelFit);
## Not run: mknAdaAoDG <- hzar.make.obsDataGroup(mknAdaAmodelFitL,mknAdaAoDG);
mknAdaAoDG <- hzar.make.obsDataGroup(mknAdaAmodelNull,mknAdaAoDG);

mknAdaAoDG <- hzar.copyModelLabels(mknAdaAdGs,mknAdaAoDG);
hzar.plot.cline(mknAdaAoDG);
print(hzar.AICc.hzar.obsDataGroup(mknAdaAoDG));

```

hzar.makeCline1DFreq *Make a cline model with the requested attributes.*

Description

Constructs a `clineMetaModel` object for use with `hzar.first.fitRequest.old.ML`. Said object can be further tailored to the specific model desired, or can be used as-is.

Usage

```

hzar.makeCline1DFreq(data = NULL, scaling = "none", tails = "none",
                    direction = NULL)

```

```

hzar.makeCline1DCLT(data = NULL, scaling = "free", tails = "none",
  direction = NULL)
hzar.makeCline1DNormal(data, tails = "none")

```

Arguments

<code>data</code>	A hzar.obsData object, used to determine cline direction and estimate initial values.
<code>scaling</code>	Can be one of three strings: <ul style="list-style-type: none"> • "none" A model with fixed minimum value 0 and maximum value 1 is desired. • "fixed" A model with minimum and maximum values fixed to the minimum and maximum observed mean values of <code>data</code> is desired. • "free" A model with the minimum and maximum value as free parameters is desired.
<code>tails</code>	Can be one of five strings: <ul style="list-style-type: none"> • "none" A model with no exponential tails is desired • "right" A model with just one exponential tail on the right is desired. • "left" A model with just one exponential tail on the left is desired. • "mirror" A model with two exponential tails mirrored about the cline center is desired. • "both" A model with two tails with independent parameters is desired.
<code>direction</code>	Can be one of three values: <ul style="list-style-type: none"> • <code>NULL</code> Determine direction using data • "ascending" A model whose estimates increase as the site distance increases is desired. • "descending" A model whose estimates decrease as the site distance increases is desired.

Details

The `clineMetaModel` object returned by `hzar.makeCline1DNormal` has a slightly different structure, due to the complexity of the normal cline model. Use [hzar.first.fitRequest.gC](#) instead of [hzar.first.fitRequest.old.ML](#) to construct the [hzar.fitRequest](#) object needed for [hzar.doFit](#).

Value

A `clineMetaModel` object, which is a list with the following 4 components:

<code>req</code>	A boolean function of the model parameters <code>w</code>
<code>prior</code>	Description of 'comp1'
<code>func</code>	Description of 'comp1'
<code>parameterTypes</code>	A list of <code>clineParameter</code> objects, named with the parameter names. A <code>clineParameter</code> object structure:

- Components:
- valThe initial or fixed value.
- wThe parameter tuning.
- Attributes:
- "param"The parameter name.
- "fixed"TRUE if the parameter is fixed.
- "limit.lower"The parameter minimum finite value.
- "limit.upper"The parameter maximum finite value.
- "realBTWN01"The parameter is restricted to between 0 and 1.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

References

Gay, L., P.-A. Crochet, D. A. Bell, and T. Lenormand. 2008. Comparing clines on molecular and phenotypic traits in hybrid zones: a window on tension zone models. *Evolution* 62:2789-2806.

Szymura, J., and N. H. Barton. 1986. Genetic analysis of a hybrid zone between the fire-bellied toads, *Bombina bombina* and *B. variegata*, near Cracow in southern Poland. *Evolution* 40:1141-1159.

Szymura, J., and N. H. Barton. 1991. The genetic structure of the hybrid zone between the fire-bellied toads *Bombina bombina* and *B. variegata*: comparisons between transects and between loci. *Evolution* 45:237-261.

See Also

[hzar.obsData](#) [hzar.first.fitRequest.old.ML](#) [hzar.first.fitRequest.gC](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
str(mknAdaAmodel);
```

hzar.makeTraitObsData *Compile and transform raw scalar data*

Description

Compile and transform raw scalar data into a frequency based [hzar.obsData](#) object.

Usage

```
hzar.makeTraitObsData(distOfLocation, locationOfValue, values)
```

Arguments

distOfLocation A vector mapping locality ids to distances of each locality.
locationOfValue
A vector of locality ids of each sample.
values A vector of trait values for each sample.

Value

A [hzar.obsData](#) object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.obsData](#) [hzar.doMorphoSets](#)

Examples

```
##TODO
```

hzar.map.dms2deg *Convert degrees, minutes, seconds, direction to degrees.*

Description

This method converts angular measurements from degree, minute, second, direction notation to decimal notation.

Usage

```
hzar.map.dms2deg(deg, min, sec, dir)
```

Arguments

deg	The degrees of the angular measurement.
min	The minutes of the angular measurement.
sec	The seconds of the angular measurement.
dir	A character vector indicating the direction of the angular measurement. The case insensitive values "N", "S", "E", "W", "West", "North", "South" and "East" are all valid, any others will result in an error.

Value

A numeric vector of angular measurements.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.map.latLong.dms](#) [hzar.map.latLongSites.dms](#)

Examples

```
hzar.map.dms2deg(9, 24, 0, "N")  
hzar.map.dms2deg(82, 33, 50, "W")
```

hzar.map.greatCircleDistance

The distance between two points on the Earth's surface.

Description

The distance along a great circle between two points on a spheroid approximation of the Earth's surface.

Usage

```
hzar.map.greatCircleDistance(lat1, long1, lat2, long2, units = "Km", degrees = TRUE)
```

Arguments

lat1	The latitude of point 1.
long1	The longitude of point 1.
lat2	The latitude of point 2.
long2	The longitude of point 2.
units	The units of distances returned. Only the case sensitive values "Km" for kilometers, "miles" for miles, and "nautical" for nautical miles are valid.
degrees	Are the latitude and longitude in degrees?

Details

The Lambert formula is the approximation used to calculate the distance, due to its high accuracy and robustness.

```
Geometry of Earth R = 6371.009 #Earth radius in kilometers earthSphd.r = 298.257223563 #WGS84
earthSphd.ep= (2*earthSphd.r -1)/(earthSphd.r-1)^2
```

```
dLat=p2$lat.rad-p1$lat.rad; dLong=p2$long.rad-p1$long.rad; dLong=ifelse( dLong>pi, dLong-2*pi, ifelse(
mLat=(p2$lat.rad+p1$lat.rad)/2;
```

```
reLat1=atan((earthSphd.r -1)*tan(p1$lat.rad) /earthSphd.r ) reLat2=atan((earthSphd.r -1)*tan(p2$lat.rad) /earthSphd.r )
cenNum=sqrt((cos(reLat2)*sin(dLong))^2+(cos(reLat1)*sin(reLat2)-cos(reLat2)*sin(reLat1)*cos(dLong))^2)
cenDen=sin(reLat1)*sin(reLat2)+cos(reLat2)*cos(reLat1)*cos(dLong); central <- atan2(cenNum,cenDen);
lFP <- (reLat1+ reLat2)/2 ; lFQ <- (-reLat1+ reLat2)/2 ; lFX <- (central-sin(central))*sin(lFP)^2*cos(central)
lFY <- (central+sin(central))*cos(lFP)^2*sin(lFQ)^2/sin(central/2)^2;
```

```
lambertFormulaeD <- ifelse(central==0,0,R*(central-(lFX+lFY)/(2*earthSphd.r)));
```

Value

The great circle distance between points 1 and 2.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

References

Lambert formula, published on the web somewhere.

Examples

```
hzar.map.greatCircleDistance(89.5,60,89.5,390)
# 28.87587
hzar.map.greatCircleDistance(-89.5,-90,89.5,90)
# 19981.56
hzar.map.greatCircleDistance(-89.5,90,89.5,90)
# 19869.99
hzar.map.greatCircleDistance(0,90,89.5,90)
# 9934.996
hzar.map.greatCircleDistance(0,90,0,-90)
```



```
# 20015.12
hzar.map.greatCircleDistance(0,90,0,180)
# 10007.56
```

hzar.map.latLong.dms *Convert D-M-S C strings to degrees*

Description

Translate a character vector of typical map coordinate(s) to a matrix of numeric values.

Usage

```
hzar.map.latLong.dms(coordinates)
```

Arguments

`coordinates` A character vector, with each string containing one or more latitude / longitude measurements, separated by the cardinal direction of the measurement (N/S for latitude, E/W for longitude).

Value

A NxM matrix of numeric values, where N is the number of strings in `coordinates` and M is the greatest number of measurements in a single string. Each row contains the measurements from each string in `coordinates`, filling one column from left to right for each measurement. The remaining columns (if any) in a row have the value NA.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.map.dms2deg](#)

Examples

```
# A single value:
hzar.map.latLong.dms(c("9-52 N"))
# A 1:2 matrix:
hzar.map.latLong.dms(c("9-52 N 82 W"))
# A 2:2 matrix with NA for the upper right corner:
hzar.map.latLong.dms(c("9-52-34 E" ,"9-52 N 82 W"))
```

hzar.map.latLongSites *Generate a table of site locations.*

Description

Methods for processing tables of site location: Generate a table of latitude and longitude of site locations from either precalculated values or GPS coordinates. Generate a table of distances from an included site, using a table of latitude and longitude of site locations.

Usage

```
hzar.map.latLongSites(siteIDs, site.lat, site.long, degrees = TRUE)
hzar.map.latLongSites.dms(siteIDs, coordinates)
hzar.map.distanceFromSite(latLongSites, site0, units = "Km")
```

Arguments

siteIDs	A character vector used to identify each site.
site.lat	A numeric vector of site longitudes.
site.long	A numeric vector of site longitudes.
degrees	Are site.lat and site.long in degrees? If FALSE, site.lat and site.long should be in radians.
coordinates	A character vector the same length as siteIDs. Each value must contain the latitude and longitude in DMS format, such as: 9-22 N 82-34-50 W
latLongSites	The result of either the hzar.map.latLongSites or hzar.map.latLongSites.dms method.
site0	The ID string for the site place at 0.
units	The units of distances returned. Only the case sensitive values "Km" for kilometers, "miles" for miles, and "nautical" for nautical miles are valid.

Value

A [data.frame](#): For all methods:

site	A character vector used to identify each site.
------	--

For hzar.map.latLongSites and hzar.map.latLongSites.dms:

lat.rad	The site latitude in radians.
long.rad	The site longitude in radians.
lat.deg	The site latitude in degrees.
long.deg	The site longitude in degrees.

For hzar.map.distanceFromSite:

distance	The distance to each site from a common origin.
----------	---

Note

Distances in `hzar.map.distanceFromSite` are calculated using the method `hzar.map.greatCircleDistance`.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.map.greatCircleDistance](#)

Examples

```
a=hzar.map.latLongSites(c("Norway"),60.4,11)
b=hzar.map.latLongSites.dms(c("Norway","Sweden"),c("60-24 N 11 E","58 N 15 E"))
hzar.map.distanceFromSite(b,"Norway")
```

<code>hzar.mcmc.bindLL</code>	<i>Generate a mcmc object with sampled parameters and log likelihoods.</i>
-------------------------------	--

Description

This function returns the mcmc data with an added a log likelihood column.

Usage

```
hzar.mcmc.bindLL(fitRequest,
                 dataGroup = hzar.fit2DataGroup(fitRequest),
                 mcmcData =
                 if(inherits(fitRequest,"hzar.fitRequest")){
                   mcmc(fitRequest$mcmcRaw,
                        thin=fitRequest$mcmcParam$thin,
                        start=1+fitRequest$mcmcParam$burnin);
                 }else{
                   as.mcmc(dataGroup$data.mcmc)},
                 llData = dataGroup$data.LL,
                 t0 = start(mcmcData),
                 tF = thin(mcmcData))
```

Arguments

<code>fitRequest</code>	The hzar.fitRequest or hzar.dataGroup object to use.
<code>dataGroup</code>	The hzar.dataGroup object to use.
<code>mcmcData</code>	The mcmc object with the series parameter values.
<code>llData</code>	The series of log likelihoods of the parameter values.
<code>t0</code>	The <code>start.mcmc</code> attribute of the result.
<code>tF</code>	The <code>thin.mcmc</code> attribute of the result.

Value

A `mcmc` object, with columns for each free parameter and the log likelihood of each row.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.dataGroup](#) [mcmc](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodelFit <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                              mknAdaA,
                              verbose=FALSE);
mknAdaAmodelFit$mcmcParam$chainLength <- 5e3;
mknAdaAmodelFit$mcmcParam$burnin <- 5e2;
plot(hzar.mcmc.bindLL(hzar.doFit(mknAdaAmodelFit)));
```

hzar.meta.init

Observe and Alter the model parameters in the clineMetaModel

Description

This is a collection of methods to get or set attributes of the various model parameters.

Usage

```
hzar.meta.init(x)
hzar.meta.init(x) <- value
hzar.meta.tune(x)
hzar.meta.tune(x) <- value
hzar.meta.fix(x)
hzar.meta.fix(x) <- value
hzar.meta.lower(x)
hzar.meta.lower(x) <- value
hzar.meta.upper(x)
hzar.meta.upper(x) <- value
```

Arguments

x	The clineMetaModel to use.
value	The new value or values to set.

Value

Returns a list, with one numeric or boolean value per cline parameter.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[clineMetaModel](#)

`hzar.model.addBoxReq` *Add parameter restriction clauses to cline model*

Description

The intent of these methods is to assist the optimizer in exploring the model parameter space by instructing it to ignore models that are not interesting. For example, if all of the sampled localities are in a region 100km wide, then a cline width of 110km is probably not interesting. A cline width of 500km in that scenario would definitely not be interesting at all.

Usage

```
hzar.model.addBoxReq(meta.model, low, high)
hzar.model.addCenterRange(meta.model, low, high)
hzar.model.addMaxCenter(meta.model, maxValue)
hzar.model.addMinCenter(meta.model, minValue)
hzar.model.addMaxDelta(meta.model, maxValue)
hzar.model.addMaxWidth(meta.model, maxValue)
hzar.model.addMaxVariance(meta.model, maxValue)
hzar.model.addNormalBox(meta.model, left, right, bottom, top)
hzar.model.addMuRange(meta.model, low, high)
```

Arguments

meta.model	The clineMetaModel object to modify.
minValue	The smallest value to consider.
maxValue	The greatest value to consider.
left	The leftmost location to consider.
right	The rightmost location to consider.

bottom	The least trait value to consider.
top	The greatest trait value to consider.
	The following arguments specifier a range in distances, with ascending values from left to right.
low	The leftmost location to consider.
high	The rightmost location to consider.

Details

The three center methods only add requirements to the center parameter. Likewise, `hzar.model.addMaxWidth` only adds a maximum width requirement. In contrast, `hzar.model.addMaxDelta` adds a maximum value requirement to any and all delta parameters present in `meta.model`.

`hzar.model.addBoxReq` adds requirements to any and all of the parameters `center`, `width`, `deltaM`, `deltaL`, and `deltaR`. The center requirements are the same as calling `hzar.model.addCenterRange(meta.model, low, high)`. The remaining parameters are required to have a maximum value of `high-low`.

Value

The modified `clineMetaModel` object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[clineMetaModel](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$sada.A,
                             manakinMolecular$sada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");
mknAdaAmodelB <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600);
mknAdaAmodel$req;
mknAdaAmodelB$req;
```

hzar.multiFitRequest *Collect hzar.fitRequest objects to simplify automation*

Description

This method manipulates a list of `hzar.fitRequest` objects to ensure each object has an independent seed. If requested, it can replicate each `hzar.fitRequest` to create independent chains.

Usage

```
hzar.multiFitRequest(fitL, each = 1, baseSeed = c(1234, 2345, 3456,
4567, 5678, 6789, 7890, 8901, 9012, 123), rotateSeed = TRUE, baseChannel
= 50, adjChannel = 50, skip = 0)
```

Arguments

<code>fitL</code>	A single <code>hzar.fitRequest</code> objects or a list of <code>hzar.fitRequest</code> objects
<code>each</code>	How many times to replicate each <code>hzar.fitRequest</code> object.
<code>baseSeed</code>	The pool of values from which to draw seeds. If NULL, do not change the seed. If <code>rotateSeed</code> is TRUE this pool is automatically reduced to unique values.
<code>rotateSeed</code>	If TRUE, a unique set of six values is drawn from <code>baseSeed</code> . If FALSE, the first 6 values of <code>baseSeed</code> are used. The method <code>rep</code> is used to expand <code>baseSeed</code> to 6 values if needed.
<code>baseChannel</code>	The initial stream channel to set for each element of <code>fitL</code> . The method <code>rep</code> is used to expand <code>baseChannel</code> to the length of <code>fitL</code> if needed. If NULL, use the original stream channel from each element of <code>fitL</code> .
<code>adjChannel</code>	Amount to increment the stream channel
<code>skip</code>	Assume <code>skip</code> sets of unique values have already been drawn from <code>baseSeed</code> .

Details

This method assumes that the user wishes to generate independent chains unless instructed otherwise.

By default, this method will use a unique seed for each element of `fitL` and increment the stream channel for each replication of each element of `fitL`.

If `rotateSeed` is TRUE, `baseSeed` is numeric, and `adjChannel` is not numeric, then this method will use a unique seed for every element of the result.

If `rotateSeed` is FALSE, `baseChannel` is a numeric of length 1, `adjChannel` is numeric, and each is greater than one, then this method will increment the stream channel for every element of the result.

Value

Returns a list of `hzar.fitRequest` object, suitable for `hzar.doFit.multi`.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.doFit.multi](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

`hzar.next.fitRequest` *Generate a new fitRequest using data from another fitRequest.*

Description

The method is the glue for parallel runs of sequential chains. It returns a ready to run [hzar.fitRequest](#) object based on the [hzar.fitRequest](#) supplied. If `oldFitRequest` had already been successfully run, this method's result will be dependent on the prior run. If not, this method's result will be a request for an independent run.

Usage

```
hzar.next.fitRequest(oldFitRequest)
```

Arguments

`oldFitRequest` A [hzar.fitRequest](#) object.

Details

If `oldFitRequest` describes a successful run, a new covariance matrix is generated, `modelParam$init` is updated to the covariance matrix center, and the lecuyer seed channel incremented by 1. If the mersenne twister was used previously, the lecuyer random number generator is requested on channel 2.

If `oldFitRequest` does not describe a successful run, everything is copied, except the lecuyer seed channel incremented by 10 to prevent overlapping. If the mersenne twister was used previously, the lecuyer random number generator is requested on channel 11.

Value

A [hzar.fitRequest](#) object.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.fitRequest](#) [hzar.chain.doSeq](#) [hzar.cov.mcmc](#)

Examples

```
data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$ada.A,
                             manakinMolecular$ada.nSamples);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed", tails="none");

mknAdaAmodelFit <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                              mknAdaA,
                              verbose=FALSE);

mknAdaAmodelFit$mcmcParam$chainLength <- 1e4;
str(hzar.next.fitRequest(mknAdaAmodelFit))
## Not run:
mknAdaAinitialFit <- hzar.doFit(mknAdaAmodelFit);
str(hzar.next.fitRequest(mknAdaAinitialFit))

## End(Not run)
```

`hzar.overPlot.fzCline` *Plot a cline region for multiple models and / or loci.*

Description

Generates and plots multiple `hzar.fzCline` objects on the same graph, using shading lines to identify each cline region.

Usage

```
hzar.overPlot.fzCline(dataGroupSet,
                     fzClineSet = sapply(dataGroupSet,
                                         hzar.getCredParamRed,
                                         simplify = FALSE),
                     type = "p",
                     fzDens = 8,
                     fzShadeAngle =
                       ((1:length(dataGroupSet)) * 180)
                       %/% (1 + length(dataGroupSet)),
                     ...)
```

Arguments

dataGroupSet	The list of <code>hzar.dataGroup</code> objects to generate <code>hzar.fzCline</code> objects for plotting.
fzClineSet	The list of <code>hzar.fzCline</code> objects to plot.
type	Passed to <code>hzar.plot.obsData</code> .
fzDens	Density of the shading lines. See <code>polygon</code> .
fzShadeAngle	Angle of the shading lines. See <code>polygon</code> .
...	Additional parameters to pass to <code>hzar.plot.fzCline</code> .

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

`polygon` `hzar.getCredParamRed` `hzar.plot.fzCline` `hzar.dataGroup`

<code>hzar.plot.cline</code>	<i>Generate a plot of the cline.</i>
------------------------------	--------------------------------------

Description

Plots a line representing the expected frequency versus distance for the given object. For `hzar.dataGroup` and `hzar.obsDataGroup` objects, plots the observed data backing the model. For `hzar.obsDataGroup` objects, plots the maximum likelihood cline for each model.

Usage

```
hzar.plot.cline(cline, add = FALSE, ylim=FALSE, ...)
```

Arguments

cline	A <code>hzar.cline</code> , <code>hzar.dataGroup</code> or <code>hzar.obsDataGroup</code> object.
add	Add to an existing plot if TRUE.
ylim	Graphical parameter passed to <code>plot</code> . If FALSE, determine ylim from cline if needed.
...	Arguments to be passed to methods, such as graphical parameters (see <code>plot</code>).

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

`plot`

Examples

```

data(manakinMolecular);
mknAdaA <-
  hzar.doMolecularData1DPops(manakinMolecular$distance,
                             manakinMolecular$aada.A,
                             manakinMolecular$aada.nSamples);
hzar.plot.obsData(mknAdaA);
mknAdaAmodel <-
  hzar.makeCline1DFreq(mknAdaA, scaling="fixed",tails="none");
mknAdaAmodel <-
  hzar.model.addBoxReq(mknAdaAmodel, -30, 600);
mknAdaAmodelFitR <-
  hzar.first.fitRequest.old.ML(model=mknAdaAmodel ,
                              mknAdaA,
                              verbose=FALSE);
mknAdaAcline <- hzar.gen.cline(list(center=300,width=10),
                              mknAdaAmodelFitR);
hzar.plot.cline(mknAdaAmodelFitR);
hzar.plot.cline(mknAdaAcline,add=TRUE);

```

`hzar.plot.fzCline` *Plot the 95% credible cline region for the given locus model.*

Description

Plots the maximum likelihood cline and observed frequency data over a the associated fuzzy cline region. The default region is the 95% credible cline region.

Usage

```

hzar.plot.fzCline(dataGroup,
                 fzCline = hzar.getCredParamRed(dataGroup),
                 type = "p", pch = "+",
                 col = "black", fzCol = "gray", ...)

```

Arguments

<code>dataGroup</code>	The <code>hzar.dataGroup</code> object for which to generate a fuzzy cline. Defaults to a 95% credible interval region.
<code>fzCline</code>	The <code>hzar.fzCline</code> object to plot.
<code>type</code>	The type parameter to pass to <code>hzar.plot.obsData</code> .
<code>pch</code>	The plotting character to pass to <code>hzar.plot.obsData</code> .
<code>col</code>	The color to plot the maximum likelihood cline and the observed frequencies.
<code>fzCol</code>	The color to fill the fuzzy cline region with.
<code>...</code>	Additional parameters to pass to the initial call to plot.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.getCredParamRed](#) [hzar.make.fzCline](#) [plot](#) [hzar.plot.obsData](#) [hzar.plot.cline](#)

Examples

```
##TODO
```

<code>hzar.plot.obsData</code>	<i>Generate a plot of the observed data points.</i>
--------------------------------	---

Description

Plots the associated observed frequency versus distance for a variety of hzar objects.

Usage

```
hzar.plot.obsData(x, type = "p", pch = "+",  
                 xlab = "Distance", ylab = hzar.yLabel(x),  
                 add = FALSE, ylim=FALSE, ...)
```

Arguments

<code>x</code>	The object from which to extract the observed data to plot.
<code>type</code>	The plot type for the scatter plot.
<code>pch</code>	The mark to use to plot the data points.
<code>xlab</code>	The x axis label.
<code>ylab</code>	The y axis label.
<code>add</code>	Draw on an existing plot.
<code>ylim</code>	Graphical parameter passed to plot . If FALSE, determine ylim from x as needed.
<code>...</code>	Additional parameters to pass to plot() .

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[plot](#) [hzar.extract.obsData](#)

`hzar.profile.dataGroup`*Generate a likelihood profile for a single parameter*

Description

TODO

Usage

```
hzar.profile.dataGroup(dG, parameter, pVals = NULL, pDivs = NULL, nDiv =  
20, appeture = NULL, doPar = FALSE, ...)
```

Arguments

<code>dG</code>	A hzar.dataGroup of the fitted model
<code>parameter</code>	
<code>pVals</code>	
<code>pDivs</code>	
<code>nDiv</code>	
<code>appeture</code>	
<code>doPar</code>	
<code>...</code>	Arguments to pass to hzar.multiFitRequest

Details

This method does not actually do the fitting of the likelihood profile. See the example for a useable workflow.

Value

A list of [hzar.fitRequest](#) objects to be fitted.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

References

There is probably a reference to how to do this somewhere...

See Also

[hzar.multiFitRequest](#) [hzar.dataGroup](#) [hzar.fitRequest](#) [hzar.doFit.multi](#) [hzar.make.obsDataGroup](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

hzar.qScores	<i>Calculate credibility intervals.</i>
--------------	---

Description

Calculate the values with an estimated cumulative likelihood equal to probs for the weighted sampled distribution.

Usage

```
hzar.qScores(x, wt, probs = c(0, 0.25, 0.5, 0.75, 1))
hzar.qScores.dataGroup(dataGroup, probs = c(0.025, 0.5, 0.975))
hzar.qScores.obsDataGroup(oDG, probs = c(0.025, 0.5, 0.975))
```

Arguments

x	The series of values to analyze.
wt	The log of the weight applied to each value.
probs	The cumulative probability values for which to calculate intervals.
dataGroup	The hzar.dataGroup to analyze.
oDG	The hzar.obsDataGroup to analyze.

Details

`hzar.qScores.dataGroup` generates intervals for all of the free parameters.

`hzar.qScores.obsDataGroup` generates intervals for the cline model with the best AICc score.

Value

For `hzar.qScores`, the values with an estimated cumulative likelihood equal to probs.

For both `hzar.qScores.dataGroup` and `hzar.qScores.obsDataGroup`, a `data.frame` with one column "q" for probs, and one additional column for each free parameter with the values returned by `hzar.qScores` for probs, given the parameter samples and likelihoods.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

See Also

[hzar.dataGroup](#) [hzar.obsDataGroup](#) [hzar.AICc.default](#)

Examples

```
##TODO
```

hzar.sameModel	<i>Do the hzar objects share the same model or data?</i>
----------------	--

Description

Test hzar objects for identical associated cline models or identical hzar.obsData objects.

Usage

```
hzar.sameModel(fitA, fitB)
hzar.sameObsData(fitA, fitB)
```

Arguments

fitA	An object to compare.
fitB	An object to compare.

Value

FALSE if the objects can not be compared (either does not have an associated model or observed data).

For hzar.sameModel:

TRUE if the model associated with fitA is identical (same equation, parameters, free parameters, and fixed parameter values).

FALSE otherwise.

For hzar.sameObsData:

TRUE if the `[["frame"]]` of the hzar.obsData object associated with fitA is identical to the `[["frame"]]` of the hzar.obsData object associated with fitB.

FALSE otherwise.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

hzar.yLabel	<i>Suggest a y axis label</i>
-------------	-------------------------------

Description

Suggests a y axis label based on the hzar object passed.

Usage

```
hzar.yLabel(x)
```

Arguments

x An object.

Value

A character vector.

Author(s)

Graham Derryberry <asterion@alum.mit.edu>

manakinLocations	<i>Distance from locality A for each locality.</i>
------------------	--

Description

Distance from locality A for each locality sampled across the Manakin Cline.

Usage

```
data(manakinLocations)
```

Format

A data frame with 12 observations on the following 2 variables.

LocalityID a factor with levels A B C D E F G H I J K L

distance a numeric vector

Source

Brumfield, R. T., R. W. Jernigan, D. B. McDonald, and M. J. Braun. 2001. Evolutionary implications of divergent clines in an avian (Manacus: Aves) hybrid zone. *Evolution* 55:2070-2087.

Examples

```
data(manakinLocations)
print(manakinLocations)
## maybe str(manakinLocations) ; plot(manakinLocations) ...
```

manakinMolecular *Molecular data samples for multiple loci accross the Manakin Cline.*

Description

Allele frequencies for multiple loci from localitys sampled in the Manakin Cline.

Usage

```
data(manakinMolecular)
```

Format

A data frame with 11 observations on the following 30 variables.

locationID ID code for the locality, a factor with levels B C D E F G H I J K L

distance The distance from locality A

ada.A The frequency of allele A of locus Ada

ada.B The frequency of allele B of locus Ada

ada.nSamples The number of alleles sampled from locus Ada

ak2.A The frequency of allele A of locus Ak2

ak2.B The frequency of allele B of locus Ak2

ak2.nSamples The number of alleles sampled from locus Ak2

gsr.A The frequency of allele A of locus Gsr

gsr.B The frequency of allele B of locus Gsr

gsr.C The frequency of allele C of locus Gsr

gsr.D The frequency of allele D of locus Gsr

gsr.E The frequency of allele E of locus Gsr

gsr.nSamples The number of alleles sampled from locus Gsr

pgm2.A The frequency of allele A of locus Pgm2

pgm2.B The frequency of allele B of locus Pgm2

pgm2.C The frequency of allele C of locus Pgm2

pgm2.D The frequency of allele D of locus Pgm2

pgm2.nSamples The number of alleles sampled from locus Pgm2

15.A The frequency of allele A of locus L5

15.B The frequency of allele B of locus L5

15.nSamples The number of alleles sampled from locus L5
 pscn3.A The frequency of allele A of locus Pscn3
 pscn3.B The frequency of allele B of locus Pscn3
 pscn3.nSamples The number of alleles sampled from locus Pscn3
 mtDNA.A The frequency of allele A of mitochondrial DNA
 mtDNA.B The frequency of allele B of mitochondrial DNA
 mtDNA.nSamples The number of alleles sampled from mitochondrial DNA
 geneticHybridIndex.mu The mean genetic hybrid index value
 geneticHybridIndex.sigma The standard deviation of the genetic hybrid index value

Source

Brumfield, R. T., R. W. Jernigan, D. B. McDonald, and M. J. Braun. 2001. Evolutionary implications of divergent clines in an avian (*Manacus*: Aves) hybrid zone. *Evolution* 55:2070-2087.

Examples

```
data(manakinMolecular)
str(manakinMolecular) ;
```

manakinMorphological *Manakin observed Morphological Traits*

Description

Morphological Traits observations of individuals sampled from the Manakin Cline.

Usage

```
data(manakinMorphological)
```

Format

A data frame with 165 observations on the following 7 variables.

Locality The id code of the locality, a factor with levels A B C D E F G H I J K L

ID The ID code of the individual sampled

Name The locality name, a factor with levels Chiriqui_Grande Costa_Rica Quebrada_Pastores
 Rio_Changuinola Rio_Oeste Rio_Robalo Rio_Sixaola Rio_Teribe Rio_Uyama Soberania
 Tierra_Oscura Valiente_Peninsula

collar.color The collar color of the individual sampled

belly.color The belly color of the individual sampled

epaulet.width The width of the epaulet of the individual sampled

beard.length The length of the beard of the individual sampled

Source

Brumfield, R. T., R. W. Jernigan, D. B. McDonald, and M. J. Braun. 2001. Evolutionary implications of divergent clines in an avian (*Manacus*: Aves) hybrid zone. *Evolution* 55:2070-2087.

Examples

```
data(manakinMorphological)
str(manakinMorphological)
```

Index

- *Topic **datasets**
 - manakinLocations, [56](#)
 - manakinMolecular, [57](#)
 - manakinMorphological, [58](#)
- *Topic **package**
 - hzar-package, [3](#)
- %dopar%, [15](#), [18](#), [19](#)
- AIC, [5](#), [6](#)
- clineMetaModel, [20](#), [21](#), [45](#), [46](#)
- clineMetaModel (hzar.makeCline1DFreq), [35](#)
- data.frame, [18](#), [24](#), [26](#), [42](#)
- foreach, [15](#)
- hzar (hzar-package), [3](#)
- hzar-package, [3](#)
- hzar.AIC.default, [4](#), [6](#)
- hzar.AIC.hzar.cline (hzar.AIC.default), [4](#)
- hzar.AIC.hzar.dataGroup (hzar.AIC.default), [4](#)
- hzar.AIC.hzar.obsDataGroup, [5](#), [6](#)
- hzar.AICc.default, [55](#)
- hzar.AICc.default (hzar.AIC.default), [4](#)
- hzar.AICc.hzar.cline (hzar.AIC.default), [4](#)
- hzar.AICc.hzar.dataGroup (hzar.AIC.default), [4](#)
- hzar.AICc.hzar.obsDataGroup (hzar.AIC.hzar.obsDataGroup), [6](#)
- hzar.chain.doSeq, [7](#), [15](#), [49](#)
- hzar.cline, [5](#), [22](#), [23](#), [28](#), [31](#)
- hzar.cline (hzar.make.cline), [26](#)
- hzar.copyModelLabels, [8](#)
- hzar.cov.mcmc, [30](#), [31](#), [49](#)
- hzar.cov.mcmc (hzar.cov.rect), [9](#)
- hzar.cov.rect, [9](#), [30](#), [31](#)
- hzar.dataGroup, [5](#), [8–11](#), [19](#), [20](#), [22–26](#), [34](#), [43](#), [44](#), [50](#), [53–55](#)
- hzar.dataGroup (hzar.make.dataGroup), [28](#)
- hzar.dataGroup.add, [10](#), [28](#), [29](#)
- hzar.dataGroup.null, [11](#)
- hzar.dBernoulli.LL, [12](#)
- hzar.doChain.multi (hzar.doFit.multi), [15](#)
- hzar.doCLTData1DPops (hzar.doMolecularData1DPops), [16](#)
- hzar.doCLTData1DRaw, [13](#)
- hzar.doFit, [7](#), [8](#), [14](#), [15](#), [21](#), [29](#), [30](#), [36](#)
- hzar.doFit.multi, [15](#), [47](#), [48](#), [53](#)
- hzar.doMolecularData1DPops, [16](#)
- hzar.doMorphoSets, [13](#), [17](#), [38](#)
- hzar.doNormalData1DPops (hzar.doMolecularData1DPops), [16](#)
- hzar.doNormalData1DRaw (hzar.doCLTData1DRaw), [13](#)
- hzar.eval.clineLL, [10](#), [18](#), [28](#), [29](#)
- hzar.extract.obsData, [19](#), [29](#), [52](#)
- hzar.extract.old.model.gen, [20](#)
- hzar.extract.old.model.req (hzar.extract.old.model.gen), [20](#)
- hzar.first.fitRequest.gC, [36](#), [37](#)
- hzar.first.fitRequest.gC (hzar.first.fitRequest.old.ML), [21](#)
- hzar.first.fitRequest.old.ML, [20](#), [21](#), [27–29](#), [31](#), [35–37](#)
- hzar.fit2DataGroup, [28](#), [29](#)
- hzar.fit2DataGroup (hzar.dataGroup.add), [10](#)
- hzar.fitRequest, [7](#), [8](#), [10](#), [14](#), [15](#), [19–23](#), [28](#), [34](#), [36](#), [43](#), [44](#), [47–49](#), [53](#)
- hzar.fitRequest (hzar.make.fitRequest),

- 29
- hzar.fzCline, 25, 50
- hzar.fzCline (hzar.make.fzCline), 31
- hzar.gen.cline, 22, 23, 26, 27
- hzar.get.ML.cline, 23, 29
- hzar.getCredCut, 23, 24
- hzar.getCredParam, 24, 25
- hzar.getCredParamRed, 25, 50, 52
- hzar.getLLCutParam, 25
- hzar.make.cline, 26
- hzar.make.clineLLfunc.old.bayes, 28, 30, 31
- hzar.make.clineLLfunc.old.bayes (hzar.make.clineLLfunc.old.ML), 27
- hzar.make.clineLLfunc.old.ML, 27, 28, 30, 31
- hzar.make.dataGroup, 10, 28
- hzar.make.fitRequest, 29
- hzar.make.fzCline, 25, 31, 52
- hzar.make.LLfunc.null, 11, 30, 31, 32
- hzar.make.mcmcParam, 30, 31, 33
- hzar.make.obsDataGroup, 34, 53
- hzar.makeCline1DCLT (hzar.makeCline1DFreq), 35
- hzar.makeCline1DFreq, 35
- hzar.makeCline1DNormal, 13, 21
- hzar.makeCline1DNormal (hzar.makeCline1DFreq), 35
- hzar.makeTraitObsData, 38
- hzar.map.distanceFromSite (hzar.map.latLongSites), 42
- hzar.map.dms2deg, 38, 41
- hzar.map.greatCircleDistance, 39, 43
- hzar.map.latLong.dms, 39, 41
- hzar.map.latLongSites, 42
- hzar.map.latLongSites.dms, 39
- hzar.mapSiteDist (hzar.doCLTData1DRow), 13
- hzar.mcmc.bindLL, 43
- hzar.meta.fix (hzar.meta.init), 44
- hzar.meta.fix<- (hzar.meta.init), 44
- hzar.meta.init, 44
- hzar.meta.init<- (hzar.meta.init), 44
- hzar.meta.lower (hzar.meta.init), 44
- hzar.meta.lower<- (hzar.meta.init), 44
- hzar.meta.tune (hzar.meta.init), 44
- hzar.meta.tune<- (hzar.meta.init), 44
- hzar.meta.upper (hzar.meta.init), 44
- hzar.meta.upper<- (hzar.meta.init), 44
- hzar.model.addBoxReq, 45
- hzar.model.addCenterRange (hzar.model.addBoxReq), 45
- hzar.model.addMaxCenter (hzar.model.addBoxReq), 45
- hzar.model.addMaxDelta (hzar.model.addBoxReq), 45
- hzar.model.addMaxVariance (hzar.model.addBoxReq), 45
- hzar.model.addMaxWidth (hzar.model.addBoxReq), 45
- hzar.model.addMinCenter (hzar.model.addBoxReq), 45
- hzar.model.addMuRange (hzar.model.addBoxReq), 45
- hzar.model.addNormalBox (hzar.model.addBoxReq), 45
- hzar.multiFitRequest, 15, 47, 53
- hzar.next.fitRequest, 7, 8, 29–31, 48
- hzar.obsData, 11, 13, 14, 18, 19, 21, 29, 32, 34, 36–38
- hzar.obsData (hzar.doMolecularData1DPops), 16
- hzar.obsDataGroup, 8–10, 19, 54, 55
- hzar.obsDataGroup (hzar.make.obsDataGroup), 34
- hzar.overPlot.fzCline, 49
- hzar.plot.cline, 50, 52
- hzar.plot.fzCline, 50, 51
- hzar.plot.obsData, 17, 18, 50, 52, 52
- hzar.profile.dataGroup, 53
- hzar.qScores, 54
- hzar.sameModel, 34, 55
- hzar.sameObsData (hzar.sameModel), 55
- hzar.yLabel, 56
- manakinLocations, 17, 18, 56
- manakinMolecular, 17, 57
- manakinMorphological, 18, 58
- mcmc, 43, 44
- MCMCmetrop1R, 7, 8, 10, 14, 21, 27, 28, 30, 31, 33, 34
- names, 8, 9
- plot, 50, 52

polygon, [50](#)

rep, [47](#)

start.mcmc, [43](#)

thin.mcmc, [43](#)