Package 'COSTdbe'

July 31, 2009

Title Design-based estimates package for COST.

Version 1.3-3

Date 2009

Author COST Team and various contributors.

Description COSTdbe is a package dealing with Design Based Estimates, i.e. considering two sets of methods, analytical issued from the sampling theory (Cochran, 1977) and resampling making use of the bootstrap (Efron and Tibshirani, 1993).

Depends R(>= 2.8.1), COSTcore, lattice, boot, methods, tcltk

Maintainer David Maxwell <david.maxwell@cefas.co.uk>

License GPL 2 or above

LazyLoad Yes

LazyData Yes

R topics documented:

`InterCatch_output` *Create InterCatch Exchange Format file*

Description

Converts a COST dbeOutput object of numbers at length or age into InterCatch Exchange Format Version 1.4 file

Usage

```
makeICdf(dbeOutput, filename=NA, output.df = TRUE, append = FALSE,CANUMtype = "length",
         Country, Species = NA, ReportingCategory, Usage = NA, DataToFrom = NA,
         SamplesOrigin, InfoFleet = NA, InfoStockCoordinator = NA, InfoGeneral = NA,
         Sex = "N", PlusGroup = -9, unitMeanWeight = "g", unitCANUM = "n", Maturity = NA)

makeICfile(ICdfs, filename, append=FALSE)
```

Arguments

`dbeOutput`	A COST dbeOutput object of numbers by length and/or age
`ICdfs`	list of 3 dataframes corresponding to InterCatch HI, SI and SD record types
`filename`	filename for output, if NA no file is written
`output.df`	logical, should list of dataframes be output
`append`	logical, should output be appended to existing file
`CANUMtype`	”age” for numbers at age, ”lngt” or ”length” for numbers at length
`Country`	The remaining arguments are character elements that match the InterCatch format description and are transferred directly to the output file
`Species`	If species is NA then it is taken from the dbeOutput, but note this is likely to be scientific name and InterCatch expects FAO code

`ReportingCategory`

`Usage`

`DataToFrom`

`SamplesOrigin`

`InfoFleet`
`InfoStockCoordinator`

`InfoGeneral`

`Sex`

`PlusGroup`
`unitMeanWeight`

`unitCANUM`

`Maturity`

Details

makeICdf creates a list of 3 dataframes corresponding to InterCatch HI, SI and SD record types then if filename is not NA makeICfile is called and a CSV file output to filename.

The process can be split into two parts by using makeICdf first, storing the list of dataframes and then calling makeICfile on the list.

We expect the function to develop as InterCatch develops and expect more of the fields to be automatically filled in future versions.

Value

A list of 3 dataframes if output.df = TRUE A csv file if filename is not NA

Author(s)

David Maxwell

References

InterCatch Exchange Format Version 1.4 `http://www.ices.dk/datacentre/InterCatch/IC-ExchFormat1-0\%20Doc1-4.pdf`

See Also

`dbeOutput`

Examples

```
# load example data set
data("LEMexample")
# Setup object for output
LEM.dbeOut.an = dbeObject(desc="Example",species="Microstomus kitt",param="landings",
                          strataDesc=LEM.strat, catchCat="LAN",methodDesc="analytical")

## Run vesselRaise function
LEM.dbeOut.an = vesselRaise.an (csObject = LEM.CScon, clObject = LEM.CLcon, dbeOutp = LEM.dbeOut.an)

# In two steps
# Create list with HI, SI, SD dataframes
LEM.ICdf <- makeICdf (LEM.dbeOut.an, Country = "UKE", Species = "LEM", ReportingCategory = "R",
                      SamplesOrigin = "M")
# edit dataframes in LEM.ICdf if required then write InterCatch file
# Not run
makeICfile (LEM.ICdf, filename ="IC_LEM.csv")

# In one step, straight to file. output.df=F stops dataframes being saved within R
# Not run
makeICdf (LEM.dbeOut.an, filename ="IC_LEM.csv", output.df = FALSE, Country = "UKE",
```

```
Species = "LEM", ReportingCategory = "R", SamplesOrigin = "M")
```

`LEM.objects` *ToDo*

Description

ToDo

Usage

```
LEM.CE
```

Format

ToDo

Details

ToDo

WARNING

This data is provided is for testing only and should not be used for any other purpose.

Source

ToDo

Examples

```
data(LEMexample)
```

`RaiseAge` *Estimation of total numbers-at-age from market sampling*

Description

Estimation of total numbers-at-age from market sampling

Usage

```
RaiseAge(dbeOutput,csObject,clObject,...)
```

Arguments

`dbeOutput`	A *dbeOutput* object.
`csObject`	A *csDataCons* object matching 'dbeOutput' specifications.
`clObject`	Optionnal. A *clDataCons* object matching 'dbeOutput' specifications (only required if type=`"direct"`).
`...`	Further arguments such as:
type	Character. Specification of the raising method : `"p"` (default value), `"fixedK"`, `"propK"`, `"agesK"`, or `"direct"`.
sex	Character. Sex specification, basically `"M"`, `"F"` or `as.character(NA)` (default value) for no restriction.

Details

Method used for estimates and variance calculation of total numbers at age is specified by *type* input parameter. If `"direct"`, calculations are based on a stratified simple random sampling (see *Type1SimpleRandomSampling.doc* file in COSTdbe package's doc folder for methodology), and raising process is made within *clObject* input data. If `"p"`, `"fixedK"`, `"propK"` or `"agesK"`, estimates are similarly computed following Kimura (see *references* section). In these cases, as calculation requires total numbers-at-length estimates, *RaiseLgth* method must be run beforehand to update input *dbeOutput* object with needed information. Contrary to estimates, variances computation differs between these 4 methods. `"fixedK"` is computing within an "age subsample fixed" strategy expectation, `"propK"` within an "age subsample random" strategy, `"agesK"` within a "random sample with ages only" strategy (see Kimura for those methodologies), and finally, `"p"` default method is based on "random sample with ages and lengths" strategy.

Value

An updated object of class *dbeOutput*. Slots nSamp$age & nMeas$age with number of samples and measurements, ageStruc$estim with numbers-at-age estimates, ageVar with the variance of numbers-at-age .

Author(s)

Mathieu Merzereaud & Marcel Machiels

References

D.K. Kimura, Statistical assessment of the age-length key, J. Fish. Res. Board Can. 34 (1977)

See Also

`dbeOutput`, `dbeObject`, `RaiseAgeBoot`, `RaiseLgth`, `csDataCons`, `clDataCons`

Examples

```
data(sole)
#stratification
strD <- strIni(timeStrata="quarter",techStrata="commCat")
#only market sampling data and biological parameters are kept
csObject <- csDataCons(csDataVal(subset(sole.cs,sampType%in%c("M","V"))),strD)
clObject <- clDataCons(clDataVal(sole.cl),strD)
```

```
#initializing the output object
dbeOutput <- dbeObject(species="Solea solea",catchCat="LAN",strataDesc=strD)

# total numbers at length
dbeOutput <- RaiseLgth (dbeOutput, csObject, clObject)

# total numbers at age
dbeOutput <- RaiseAge (dbeOutput, csObject)
```

`RaiseAgeBoot` *Estimation of numbers-at-age from market sampling with bootstrap variance*

Description

RaiseAgeBoot follows the approach of RaiseAge to calculate numbers-at-age by strata from market sampling data. A set of bootstrap estimates is created to calculate the variance. This method requires RaiseLgthBoot to have been run to estimate the length structure and define the number of bootstrap replicates.

Usage

```
RaiseAgeBoot(dbeOutput, csObject, type="p", sex=as.character(NA), bootMethod = "samples", .
```

Arguments

`dbeOutput`	A *dbeOutput* object containing output from RaiseLgthBoot.
`csObject`	A *csDataCons* object matching 'dbeOutput' specifications.
`type`	Allocation strategy used to establish the age-length data, one of "p", "fixedK", "propK" or "agesK" (see *RaiseAge*)
`sex`	Sex
`bootMethod`	"samples" (the default) or "otoliths"
`...`	Further arguments

Details

If bootMethod="samples" then samples in the data are defined by PSUid and SSUid and these samples are selected with replacement to create bootstrap estimates.

If bootMethod="otoliths" then individual age observations within each length class (i.e. row of the ALK) are resampled instead. This makes the assumption that individual otoliths are independent. In other words, that for age-given-length, fish in the same sample are not more similar than fish in different samples.

Value

An updated object of class *dbeOutput*. Slot methodDesc with bootstrap samples or bootstrap otoliths, nSamp$age & nMeas$age with number of samples and measurements, ageStruc$rep with bootstrap replicates for numbers-at-age, iter=0 is assigned the estimates from the original data, ageStruc$estim with the mean of the bootstrap replicates, ageVar with the variance of the bootstrap replicates

Author(s)

David Maxwell & Mathieu Merzereaud

See Also

`dbeOutput`, `dbeObject`, `csDataCons`, `clDataCons`, `RaiseAge`, `RaiseLgthBoot`

Examples

```
data(sole)
strD <- strIni(timeStrata="quarter",techStrata="commCat")
csObject <- csDataCons(csDataVal(subset(sole.cs,sampType%in%c("M","V"))),strD)
clObject <- clDataCons(clDataVal(sole.cl),strD)
dbeOutput <- dbeObject(species="Solea solea",catchCat="LAN",strataDesc=strD)

# Analytical estimates
sol.dbe.an <- RaiseLgth (dbeOutput, csObject, clObject)

# Bootstrap estimates. B set to a very low number of iterations for demonstration only.
# Several thousand iterations recommended for final run, which will take some time.
sol.dbe.boot <- RaiseLgthBoot (dbeOutput, csObject, clObject, B=10)

sol.dbe.boot <- RaiseAgeBoot (dbeOutput = sol.dbe.boot, csObject = csObject, type="p")
```

`RaiseLgth` *Estimation of total numbers-at-length from market sampling*

Description

This method calculates total numbers-at-length by strata from market sampling data.

Usage

```
RaiseLgth(dbeOutput,csObject,clObject,spp,taxon,sex=as.character(NA),...)
```

Arguments

`dbeOutput`	A *dbeOutput* object.
`csObject`	A *csDataCons* object matching 'dbeOutput' specifications.
`clObject`	A *clDataCons* object matching 'dbeOutput' specifications. If not specified, data is only raised to trips.
`spp`	Species, if missing this is set to dbeOutput@species
`taxon`	Taxon, if missing this is set to dbeOutput@species
`sex`	Sex
`...`	Further arguments

Value

An updated object of class *dbeOutput*. Slots nSamp$len & nMeas$len with number of samples and measurements, totalW$estim with total weight, lenStruc$estim with numbers-at-length estimates, lenVar with the variance of numbers-at-length.

Author(s)

Mathieu Merzereaud

See Also

dbeOutput, dbeObject, RaiseAge, csDataCons, clDataCons

Examples

```
data(sole)
#stratification
strD <- strIni(timeStrata="quarter",techStrata="commCat")
#only market sampling data and biological parameters are kept
csObject <- csDataCons(csDataVal(subset(sole.cs,sampType%in%c("M","V"))),strD)
clObject <- clDataCons(clDataVal(sole.cl),strD)
#initializing the output object
dbeOutput <- dbeObject(species="Solea solea",catchCat="LAN",strataDesc=strD)

# total numbers at length
dbeOutput <- RaiseLgth (dbeOutput, csObject, clObject)
```

`RaiseLgthBoot` *Estimation of numbers-at-length from market sampling with bootstrap variance*

Description

RaiseLgthBoot follows the approach of RaiseLgth to calculate numbers-at-length by strata from market sampling data. The data are resampled with replacement to create B bootstrap estimates that are used to calculate the variance.

Usage

```
RaiseLgthBoot(dbeOutput,csObject,clObject,spp,taxon,sex=as.character(NA),B ,...)
```

Arguments

dbeOutput	A *dbeOutput* object.
csObject	A *csDataCons* object matching 'dbeOutput' specifications.
clObject	A *clDataCons* object matching 'dbeOutput' specifications.
spp	Species, if missing this is set to dbeOutput@species
taxon	Taxon, if missing this is set to dbeOutput@species
sex	Sex
B	Number of bootstrap interations
...	Further arguments

Value

An updated object of class *dbeOutput*. Slots nSamp$len & nMeas$len with number of samples and measurements, methodDesc with "bootstrap", totalW$estim with total weight, lenStruc$rep & totalN$rep with bootstrap replicates for numbers-at-length and total numbers, iter=0 is assigned the estimates from the original data, lenStruc$estim & totalN$estim with the mean of the bootstrap replicates, lenVar & totalNvar with the variance of the bootstrap replicates

Author(s)

David Maxwell & Mathieu Merzereaud

See Also

`dbeOutput`, `dbeObject`, `csDataCons`, `clDataCons`, `RaiseLgth`

Examples

```
data(sole)
strD <- strIni(timeStrata="quarter",techStrata="commCat")
csObject <- csDataCons(csDataVal(subset(sole.cs,sampType%in%c("M","V"))),strD)
clObject <- clDataCons(clDataVal(sole.cl),strD)
dbeOutput <- dbeObject(species="Solea solea",catchCat="LAN",strataDesc=strD)

# Analytical estimates
sol.dbe.an <- RaiseLgth (dbeOutput, csObject, clObject)

# Bootstrap estimates. B set to a very low number of iterations for demonstration only.
# Several thousand iterations recommended for final run, which will take some time.
sol.dbe.boot <- RaiseLgthBoot (dbeOutput, csObject, clObject, B=10)
```

`alkLgthRec` *Method for managing gaps in age-length keys*

Description

This function proposes various methods to solve alk gaps problems inherent to a 'csDataCons' object. Input object is updated according to chosen method(s) (grouped/recoded length classes, addition of 'virtual' individuals,...)

Usage

```
alkLgthRec(object,type="stepIncr",value,preview=FALSE,postview=TRUE,update=FALSE,...)
```

Arguments

`object`	A *csDataCons* object with ca information.
`type`	Character for chosen method. Values are :
"stepIncr"	Default parameter. Length class step is increased to specified *value* parameter (default value=10)

"fillMiss"	All gaps (with size <= value) are filled out with the sum of surrounding recorded classes (default value=1)
"sFillMiss"	The 'value' empty classe(s) prior to first recorded length class is filled out with the latter (default value=1)
"lFillMiss"	The 'value' empty classe(s) following last recorded length class is filled out with the latter (default value=1)
`value`	Numerical parameter for chosen method (see 'type').
`preview`	Logical. If `TRUE`, original age length key is displayed.
`postview`	Logical. If `TRUE`, new age length key is displayed.
`update`	Logical. If `TRUE`, 'csDataCons' object is updated in accordance with chosen method, and then returned. If `FALSE`, descriptive elements about updated alk are returned (see 'values'), but input object remains unchanged.
`...`	Further arguments, and particularly a *start* numerical parameter specifying the first considered length class when recoding (only useful for 'type="stepIncr"'). Default value is the minimum aged length class in *ca* table.

Value

If `update=FALSE`, returned elements are : `$alk` is the raw resulting age-length key, `$propMiss` are short statistics about gaps (see 'propMissLgthCons' method), `$lgthCls` is a description of length classes recoding for 'stepIncr', 'sExtrGrp' and 'lExtrGrp' methods and `$addIndTab` is a description of added virtual individuals for other methods.

Author(s)

Mathieu Merzereaud

See Also

`viewGapsAlkCons`, `propMissLgthCons`

Examples

```
data(sole)
  #restriction to "27.7.d" & "27.7.e" areas
csRaw <- subset(sole.cs,area%in%c("27.7.d","27.7.e"),table="hh")
  #consolidation process
conSole.cs <- csDataCons(csDataVal(csRaw),strIni(timeStrata="quarter",spaceStrata="area"))

res1 <- alkLgthRec(conSole.cs,type="stepIncr",value=20)
names(res1)

res1$missProp #updated statistics about missing length classes

res1$lgthCls #updated Length Classes
#if it's allright, consolidated object can be updated --> conSole.cs1
conSole.cs1 <- alkLgthRec(conSole.cs,type="stepIncr",value=20,postview=FALSE,update=TRUE)
```

`bpBoot` *Bootstrap estimates of biological parameters*

Description

This method implements a non-parametric bootstrap procedure to estimate empirical estimates of weight-at-length, maturity-at-length, sex-ratio-at-length, weight-at-age, maturity-at-age, sex-ratio-at-age and their associated variances. It requires a csDataCons object and a dbeOutput object.

Usage

```
bpBoot(dbeOutput,object,mat.scale=list(immature=c(0,1),mature=c(2:8)),
       sample.boot=FALSE,nboot=1000,...)
```

Arguments

`dbeOutput`	A *dbeOutput* object.
`object`	A *csDataCons* object.
`mat.scale`	List containing the maturity stages for immature and mature individuals. The codes must match with maturity scale of *csDataCons* object.
`sample.boot`	Logical value. If FALSE, the resampling unit is the individual. If TRUE, the resampling unit is the sample.
`nboot`	Single positive integer indicating the number of bootstrap replicates.
`...`	Any additional arguments.

Details

This method uses the non-parametric bootstrap techniques to estimate the variation in biological parameters (mean weight, maturity and sex-ratio) based on the original data provided at the slot ca of a consolidated COSTobject.

It is assumed that the biological sampling is representative of the catches. If the number of samples by defined strata is lower than ten, variance estimates should not be considered reliable.

The results of this method are included as components of the object of class "dbeObject".

Author(s)

Paz Sampedro

References

B. Efron and R. Tibshirani (1993). *An Introduction to the Bootstrap.* Chapman & Hall.

See Also

`dbeOutput,dbeOutput,dbeOutput`

Examples

```
## Mean weight-at-length/age and variance for IFREMER sole data
## Estimates are made by quarter and area
#
#data (sole)
#sole.str <- strIni(timeStrata="quarter",spaceStrata="area")
#sole.cons <- csDataCons(csDataVal(sole.cs),sole.str)
#sole.dbeOutput <- dbeObject(desc="Results of design based estimates",species="Solea solea",
#                            param="weight",strataDesc=sole.str,methodDesc="Bootstrap")
#sole.Weight <- bpBoot(sole.dbeOutput,sole.cons, sample.boot=TRUE)
```

`bpEstim` *Analytical estimates of biological parameters*

Description

This method implements analytical estimates of empirical weight-at-length/age, maturity-at-length/age, sex-ratio-at-length/age and variances. The needed parameters are from input 'dbeOutput' slots. Estimates-at-age can be calculated by injecting the length distribution resultin from 3 different sampling data : *ca* table or *hl* table from input consolidated object, or total number-at-length from an input *dbeOuput* object. In the latter case, *totalN* and *totalW* slots are also inserted in output *dbeOutput* object.

Usage

```
bpEstim(dbeOutput,object,dbeLD,adjust=TRUE,immature.scale=1,...)
```

Arguments

`dbeOutput`	A *dbeOutput* object.
`object`	A *csDataCons* object.
`dbeLD`	Optionnal. A *dbeOutput* object with total numbers-at-length data (possibly resulting from *RaiseLgth* method). If missing, injected length distribution for estimates-at-age calculation is created within input consolidated object. In that case, the source table is specified with *adjust* parameter.
`adjust`	Logical. Only useful if *dbeLD* parameter is missing. If FALSE, length distribution in *object*'s CA table is supposed to be representative of the catch (all calculations are made within CA). If TRUE (default value), estimates-at-age are calculated by injecting *object*'s HL information.
`immature.scale`	Numeric or character. Specifies the value(s) in *matStage* field (from ca table in *object*) for which the individuals are defined as immature.
`...`	Further arguments.

Author(s)

Mathieu Merzereaud

See Also

`dbeOutput,dbeOutput`

Examples

```
data(sole)
#stratification object
strDef <- strIni(timeStrata="quarter",spaceStrata="area")
#consolidated object
object <- csDataCons(csDataVal(sole.cs),strDef)
#dbeOutput initial object with needed parameters
dbeOutput <- dbeObject(desc="My object",species="Solea solea",param="weight",
                       strataDesc=strDef,methodDesc="analytical")

lWeight <- bpEstim(dbeOutput,object)
```

---

dbeCalc *CI and CV calculation*

Description

Method for calculating coefficients of variation or confidence intervals from 'dbeOutput' object estimates. Input object can optionally be updated with resulting table.

Usage

```
dbeCalc(object,type="CI",vrbl="l",probs=c(0.025,0.975),replicates=FALSE,update=TRUE,...)
```

Arguments

`object`	A *dbeOutput* object.
`type`	Character. "CI" for confidence interval calculation, or "CV" for coefficients of variation.
`vrbl`	Character specifying 'dbeOutput' estimates on which calculation is applied : "l" for length structure, "a" for age structure, "n" for total number estimates, "w" for total weight estimates.
`probs`	Numeric vector of probabilities with values in [0,1]. Defines CI bounds (relevant only if `type="CI"`). See *quantile*.
`replicates`	Logical. If `TRUE`, calculation is made from @...$rep elements ; if `FALSE`, @...$estim and @...Var 'dbeOutput' data are used.
`update`	Logical. If `TRUE`, updated 'dbeOutput' object is returned ; if `FALSE`, only resulting dataframe is returned
`...`	Further arguments used as '*quantile* method input parameter (if `type="CI"` and besides `probs` parameter).

Details

If calculation is made from replicates (see *replicates* parameter), confidence interval is estimated using *quantile* fonction with *probs* and ... parameters. If calculation is made from estimates, normal distribution of total estimates is assumed to compute confidence intervals. Possible resulting negative bounds are automatically replaced by 0 in output object.

Author(s)

Mathieu Merzereaud

See Also

dbeOutput, dbePlot, quantile

Examples

```
data(sole)
#stratification
strD <- strIni(timeStrata="quarter",techStrata="commCat")
#only market sampling data and biological parameters are kept
csObject <- csDataCons(csDataVal(subset(sole.cs,sampType%in%c("M","V"))),strD)
clObject <- clDataCons(clDataVal(sole.cl),strD)
#initializing the output object
dbeOutput <- dbeObject(species="Solea solea",catchCat="LAN",strataDesc=strD)

# total numbers at length
dbeOutput <- RaiseLgth (dbeOutput, csObject, clObject)
#object is updated with cv-at-length
dbeOutput <- dbeCalc(dbeOutput,type="CV")
```

---

dbeCorrPlot — *Levelplot of a correlation matrix between estimates at age (replicates)*

Description

Method for plotting an image of the correlation matrix between estimates at age, from *dbeOutput* 'ageStruc' slot replicates.

Usage

```
dbeCorrPlot(object,dispKey=TRUE,...)
```

Arguments

object	A *dbeOutput* object with replicates information in 'ageStruc' slot.
dispKey	Logical. If TRUE, a legend is drawn
...	Further graphical arguments.

Author(s)

Mathieu Merzereaud

See Also

dbeOutput

`dbeObject` *Initialized 'dbeOutput' object creator*

Description

This function returns an initialized *dbeOutput* object that can be used as input object for 'COSTdbe' package methods.

Usage

```
dbeObject(desc, species, catchCat, param, strataDesc, methodDesc, ...)
```

Arguments

`desc`	'desc' slot of returned 'dbeOutput' object.
`species`	'species' slot of returned 'dbeOutput' object.
`catchCat`	'catchCat' slot of returned 'dbeOutput' object.
`param`	'param' slot of returned 'dbeOutput' object.
`strataDesc`	'strataDesc' slot of returned 'dbeOutput' object.
`methodDesc`	'methodDesc' slot of returned 'dbeOutput' object.
`...`	Further arguments.

Author(s)

Mathieu Merzereaud

See Also

`dbeOutput`

Examples

```
obj <- dbeObject(desc="My object",species="Solea solea",catchCat="DIS",methodDesc="analytical")
```

`dbeOutput-class` *Class "dbeOutput"*

Description

Outcome object from *COSTdbe* methods

Slots

slot	**desc**	**elements**	**class**	**description**
`desc`			`character`	Descriptive slot
`species`			`character`	Species
`catchCat`			`character`	Catch category (eg "LAN", "DIS" or "all")
`param`			`character`	Parameter estimated (eg "maturity","sex-ratio",...)
`strataDesc`			`strIni`	Stratification considered
`methodDesc`			`character`	Used method (eg "analytical","bootstrap" or "Bayesian")

All the following slots contain numerics or dataframes that will be generated by COSTdbe methods :

nSamp		`list`	Number of samples
	len	`data.frame`	Number of length samples.
	age	`data.frame`	Number of age samples.
nMes		`list`	Number of individual measured
	len	`data.frame`	Number of individual measured for length data.
	age	`data.frame`	Number of individual measured for age data.
lenStruc		`list`	Estimates of the length structure
	estim	`data.frame`	Final estimates (*length* field added).
	rep	`data.frame`	Resampling replicates (*length* and *iter* fields added).
lenVar		`data.frame`	Estimates of the variance of 'lenStruc'
lenNum		`list`	Precision estimates for length structure
	ci	`data.frame`	Confidence intervals.
	cv	`data.frame`	Coefficients of variation.
	DCRcvIndicator	`numeric`	Weighted global cv.
ageStruc		`list`	Estimates of the age structure
	estim	`data.frame`	Final estimates (*age* field added).
	rep	`data.frame`	Resampling replicates (*age* and *iter* fields added).
ageVar		`data.frame`	Estimates of the variance of 'ageStruc'
ageNum		`list`	Precision estimates for age structure
	ci	`data.frame`	Confidence intervals.
	cv	`data.frame`	Coefficients of variation.
	DCRcvIndicator	`numeric`	Weighted global cv.
totalN		`list`	Estimates of the total number of the parameters
	estim	`data.frame`	Final estimates.
	rep	`data.frame`	Resampling replicates (*iter* field added).
totalNvar		`data.frame`	Estimates of the variance of 'totalN'
totalNnum		`list`	Precision estimates for total numbers
	ci	`data.frame`	Confidence intervals.
	cv	`data.frame`	Coefficients of variation.
	DCRcvIndicator	`numeric`	Weighted global cv.
totalW		`list`	Estimates of the total weight of the parameters
	estim	`data.frame`	Final estimates.
	rep	`data.frame`	Resampling replicates (*iter* field added).
totalWvar		`data.frame`	Estimates of the variance of 'totalW'
totalWnum		`list`	Precision estimates for total weights
	ci	`data.frame`	Confidence intervals.
	cv	`data.frame`	Coefficients of variation.
	DCRcvIndicator	`numeric`	Weighted global cv.

Author(s)

Mathieu Merzereaud

See Also

`totVolume`, `dbePlot`

Examples

```
showClass("dbeOutput")
```

`dbePlot` *Graphical display of 'dbeOutput' final estimates*

Description

Method for plotting final estimates from an input 'dbeOutput' object.

Usage

```
dbePlot(object,elmt,type="bar",Xstratum=NULL,step=NA,dispKey=TRUE,indScale=FALSE,...)
```

Arguments

`object`	A *dbeOutput* object.
`elmt`	Character specifying an element (a dataframe) of *dbeOutput* 'object'. For example, "lenStruc$estim", "ageVar" or "totalNnum$cv". 'rep' elements are not accepted ; see *dbePlotRep*.
`type`	Character specifying the type of the drawn plot. To be chosen between "bar" (default value), "point" and "line".
`Xstratum`	Stratum displayed on x-axis if 'elmt' doesn't point at length or age structure information. To be chosen between "time", "space", "technical" and `NULL` (default value).
`step`	Numeric. If given, empty length or age classes will be considered and displayed, according to specified value.
`dispKey`	Logical. If `TRUE`, a describing key is displayed
`indScale`	Logical. If `TRUE`, y-axis scale is specific to each panel. If `FALSE`, the same limits are used for every panel.
...	Further graphical arguments such as *col, lwd, lty, pch, cex, font, rot,...*

Author(s)

Mathieu Merzereaud

See Also

`dbeOutput`, `dbePlotRep`

Examples

```
data(sole)

#stratification object
strDef <- strIni(timeStrata="quarter",spaceStrata="area")
#consolidated object
object <- csDataCons(csDataVal(sole.cs),strDef)
#dbeOutput initial object with needed parameters
dbeOutput <- dbeObject(desc="My object",species="Solea solea",param="weight",
                       strataDesc=strDef,methodDesc="analytical")

lW <- bpEstim(dbeOutput,object)
```

```
dbePlot(lW,elmt="ageStruc$estim",step=1,ylab="Mean weight (g)")
```

`dbePlotRep` *Graphical display of 'dbeOutput' replicates*

Description

Method for plotting replicates data from an input 'dbeOutput' object (distribution of total estimates replicates, description of length or age structure replicates).

Usage

```
dbePlotRep(object,Slot,probs=c(0.05,0.95),step=NA,origin=TRUE,dispKey=TRUE,
           KurtSkew=FALSE,...)
```

Arguments

`object`	A *dbeOutput* object with replicates information.
`Slot`	A *dbeOutput* slot with a 'rep' element (eg "totalW" or "lenStruc").
`probs`	Numeric with 2 elements (or NA for no display) : quantiles defining displayed confidence intervals for each set of replicates (per stratum and length/age class). Only useful for "lenStruc" or "ageStruc" *Slot* parameter
`step`	Numeric. If given, empty length or age classes will be considered and displayed, according to specified value. Only useful for "lenStruc" or "ageStruc" *Slot* parameter.
`origin`	Logical. Only useful for "lenStruc" or "ageStruc" *Slot* parameter. If `TRUE`, original raw estimates, stored in 'rep' object for `iter` value at 0, are displayed.
`dispKey`	Logical. If `TRUE`, a legend is drawn
`KurtSkew`	Logical. If `TRUE`, skewness and kurtosis values are calculated for each set of replicates (per stratum and length/age class) and as specific plot is displayed. Only useful for "lenStruc" or "ageStruc" *Slot* parameter.
`...`	Further graphical arguments.

Details

This method provides 3 types of graphical outputs. If *Slot* is "totalN" or "totalW", a frequency histogram of total estimates replicates is displayed. If *Slot* is "lenStruc" or "ageStruc", then means at length/age calculated from replicates are showed with confidence interval (if *probs* is given) and raw estimates (if *origin* is TRUE). If *KurtSkew* is TRUE, skewness and kurtosis values at length/age are calculated from replicates, and drawn.

Author(s)

Mathieu Merzereaud

See Also

`dbeOutput`

`plus-dbeOutput` *Addition of 'dbeOutput' objects*

Description

This method implements addition of 'dbeOutput' objects provided by COSTdbe package.

Details

Every dataframe are summed, except *"num"* elements that are set to NA.

Methods

```
signature(e1 = "dbeOutput", e2 = "dbeOutput")
```

`propMissLgthCons` *Proportions of empty length classes in an age-length key*

Description

This method calculates empty length classes proportion in an age-length key obtained from a 'csDataCons' object (ca table). 'pEmpty' element describes the proportion of missing LC per alk within hl table. 'pEmptyExtr' submits the proportion of extrema missing LC (among all missing LC) per alk within hl table.

Usage

```
propMissLgthCons(object,...)
```

Arguments

`object`	A *csDataCons* object.
`...`	Further arguments.

Author(s)

Mathieu Merzereaud

See Also

`alkLgthRec`, `viewGapsAlkCons`, `csDataCons`

Examples

```
data(sole)
  #subset to "27.7.d" & "27.7.e" areas
csSub <- subset(sole.cs,area%in%c("27.7.d","27.7.e"),table="hh")
conSole.cs <- csDataCons(csDataVal(csSub),strIni(spaceStrata="area"))

propMissLgthCons(conSole.cs)
```

rbind2-dbeOutput *rbind2 for 'dbeOutput' objects*

Description

This method implements rbind2 for the 'dbeOutput' class provided by COSTdbe package.

Details

Only dataframes are combined in the output object. Its descriptive slots are the same as *x*'s, and *dcrCVIndicator* elements are set to NA.

Methods

```
signature(x = "dbeOutput", y = "dbeOutput")
```

sampStrDef *Definition of the sampling strategy*

Description

This method defines from 'csData' or 'csDataVal' datasets what is the sampling strategy at haul level (one value per *hh* row) and at trip level (one value per *tr* row).

Usage

```
sampStrDef(x,species,fraction="LAN",fishAct="foCatEu5",sampPar=TRUE,...)
```

Arguments

`x`	A *csData* or *csDataVal* object.
`species`	Character specifying species for which sampling strategy must be defined
`fraction`	Character specifying catch category for which sampling strategy must be defined. To be chosen between "LAN" (default value) for landings and "DIS" for discards.
`fishAct`	Character specifying fishing activity field taken into account for sampling strategy definition. To be chosen between "foCatEu5" (default value), "foCatEu6" and "focatNat".
`sampPar`	Logical. Checks if given species is considered to be automatically sampled (`TRUE`, default value) or not (`FALSE`) during the sampling process.
`...`	Further arguments.

Value

Returned elements are `$sampStrategyTR` that describes the sampling strategy at trip level, and `$sampStrategyHH` for sampling strategy at haul level. Used codification is :

`1`	Sampling for length in fishing trips - Unsorted catch
`2`	Sampling for length in fishing trips - Commercial Categories
`3`	Sampling for length in Commercial categories
`4`	Sampling for age in fishing trips - Unsorted catch
`5`	Sampling for age in fishing trips - Commercial Categories
`6`	Sampling for age in commercial categories
`0`	No sampling or undefined strategy
`9`	Several defined strategies (trip level)

Author(s)

Mathieu Merzereaud

See Also

`csData`, `csDataVal`

Examples

```
data(sole)
sampStrDef(sole.cs,species="Solea solea")
```

`sampledFO` *Sampled haul index*

Description

Method returning from a 'csDataCons' object an index of sampled hauls for a given species and a given fraction.

Usage

```
sampledFO(x,species,fraction="LAN",sampPar=TRUE,...)
```

Arguments

`x`	A *csDataCons* object.
`species`	Given species for which sampling must be considered.
`fraction`	Given catch category for which sampling must be considered. To be chosen between "LAN" (default value) and "DIS".
`sampPar`	logical specifying if given species is considered to be automatically sampled during the sampling process (default value is `TRUE`).
`...`	Further arguments.

Value

A list with 2 vectors of the same length as HH table from input 'csDataCons' object. `$sampWt` is the index for weight sampling, `$sampLg` is for length sampling (numbers). 1 means that the haul is sampled/measured and the species is caught in the fraction, 0 means that the haul is sampled/measured but the species is not caught in the fraction (0-values), and NA means that the haul is not sampled/measured.

Author(s)

Mathieu Merzereaud

See Also

`csDataCons`

Examples

```
data(sole)
x <- csDataCons(csDataVal(sole.cs))
sampledFO(x,species="Solea solea",fraction="LAN")
```

---

stratAggreg *Aggregation of 'dbeOutput' table*

Description

This method is aggregating (summing in case of number or weight estimates, or averaging as to mean parameters) input 'dbeOutput' object tables over chosen stratification(s).

Usage

```
stratAggreg(object,timeStrata=TRUE,spaceStrata=TRUE,techStrata=FALSE,wt="totalW",...)
```

Arguments

`object`	A *dbeOutput* object.
`timeStrata`	Logical. If `TRUE`, aggregation is made over time strata.
`spaceStrata`	Logical. If `TRUE`, aggregation is made over space strata.
`techStrata`	Logical. If `TRUE`, aggregation is made over technical strata.
`wt`	Character. Input *object* slot with which a weighted mean over strata is computed (only if *param* slot is "weight", "maturity" or "sex"). Possible values are "totalW" or "totalN". Check that specified slot is not empty, otherwise resulting table of estimates will also be empty.
`...`	Further arguments

Details

Aggregating process is only applied to these tables : `nSamp`, `nMeas`, `lenVar`, `ageVar`, `totalNvar`, `totalWvar`, and all `estim` and `rep` list elements. Other tables in output object are empty (so, for example, 'dbeCalc' method must be called to calculate and insert *cv* table in this new object). WARNING : summing variances requires some strong probability assumptions within strata, such as uncorrelated variables. In the 'variance at age' case, this can be problematic since the same age-length key is used for every technical strata.

Value

An updated object of class *dbeOutput*.

Author(s)

Mathieu Merzereaud

See Also

`dbeOutput`, `dbeObject`

`tkFillGaps` *Addition of virtual individuals in age-length keys*

Description

This method displays stratified ALKs from a *csDataCons* object in a GUI table that allows to add 'virtual' individuals. Input object is then updated and returned (see *Value* section).

Usage

```
tkFillGaps(object)
```

Arguments

`object`	A *csDataCons* object with ca information.

Details

See *viewGapsAlkCons* for a description of the codification used in displayed ALKs. Interface has been computed with *tcltk* package.

Value

An updated *csDataCons* object. CA table is updated with new individuals (one per line). New *PSUid* and *trpCode* values are then created, and *tr* table is also updated with those new 'virtual trips'.

Author(s)

Mathieu Merzereaud

See Also

csDataCons, viewGapsAlkCons

Examples

```
data(sole)
  #subset to "27.7.d" & "27.7.e" areas
csSub <- subset(sole.cs,area%in%c("27.7.d","27.7.e"),table="hh")
object <- csDataCons(csDataVal(csSub),strIni(spaceStrata="area"))
#obj <- tkFillGaps(object)
#tail(obj)
```

---

totVolume — *Estimation of total volume of discards or/and landings (weight, number or number-at-length)*

Description

Generic function to estimate total volume of discards or/and landings (weight, number or number-at-length) based on various raising methods.

Usage

```
totVolume(dbeOutput,csObject,ceObject,clObject,...)
```

Arguments

dbeOutput	A *dbeOutput* object. All necessary information for calculation process are taken in the first slots (species, catch category,...). See *dbeObject* method for object initialization.
csObject	A *csDataCons* object matching 'dbeOutput' specifications.
ceObject	A *ceDataCons* object matching 'dbeOutput' specifications.
clObject	An optionnal *clDataCons* object matching 'dbeOutput' specifications. If specified, raising is made with ratio-to-landings method.
...	Further arguments such as:
type	Specification of the raising method : "trip" (default value) for raising by trip, "fo" for raising by fishing operations, "fd" for raising by fishing days,"landings" for ratio-to-total landings raising method, and "time" for ratio-to-fishing duration raising method.
val	Estimated parameter. To be chosen between "weight" (default value), "number" and "nAtLength".
sampPar	logical specifying if given species is considered to be automatically sampled during the sampling process (default value is TRUE).
landSpp	character vector describing the species considered in the 'volume of landings' variable if chosen raising method is ratio-to-landings (see 'clObject' description).

Value

An updated object of class *dbeOutput*.

Author(s)

Mathieu Merzereaud

References

Vigneau, J. (2006) *Raising procedures for discards : Sampling theory (Toward agreed methodologies for calculating precision in the discard programmes).* Working document in support of PGCCDBS (Rostock, 2006).

See Also

`dbeOutput`, `dbeObject`, `csDataCons`, `ceDataCons`, `clDataCons`

Examples

```
data(sole)

#consolidated datasets are built
strDef <- strIni(timeStrata="quarter",techStrata="foCatEu5")
csObject <- csDataCons(csDataVal(sole.cs),strDef)
clObject <- clDataCons(clDataVal(sole.cl),strDef)
ceObject <- ceDataCons(ceDataVal(sole.ce),strDef)

#dbeOutput initial object
obj <- dbeObject(desc="My object",species="Solea solea",catchCat="DIS",strataDesc=strDef,
                 methodDesc="analytical")

##raising by trip
newObj <- totVolume(obj,csObject,ceObject)
newObj
```

`vesselRaise` *Estimate numbers-at-age and numbers-at-length*

Description

Estimate numbers-at-age and numbers-at-length from market sampling data with Sampling strategy: Length + ALK, Commercial categories within Trips Sampling source: Auction/Market/Harbour Landed fraction vesselRaise.boot provides variance estimates by bootstrapping the age and length samples. vesselRaise.an provides analytical variance estimates derived from the proportions in the ALK and variation in numbers per kg in the length samples.

Usage

```
vesselRaise.an(csObject, clObject, dbeOutp, age.plus = -1)
vesselRaise.boot(csObject, clObject, dbeOutp, B, age.plus = -1, bootMethod = "samples")
```

Arguments

`csObject`	A *csDataCons* object of sampling data
`clObject`	A *clDataCons* object of landings data
`dbeOutp`	A *dbeOutput* object that provides sample information and stores the function's results.
`B`	numeric, length=1. Number of bootstrap iterations
`age.plus`	numeric, length=1. All ages greater than `age.plus` are set to `age.plus` to form a plus group
`bootMethod`	only acceptable value is "samples" to bootstrap samples.

Details

Estimates by length are provided for each combination of the space, time and technical strata specified in dbeOutp. Estimates by age are provided for each combination of the space and time strata because the biological data (ca table) does not include technical stratification. Estimation for mixed species landings is not implemented; the species in the biological data must match the taxon field in the landings data. Similarly, estimates are not given by sex if unsexed length data and sexed age data are used. Either use combined-sex length and age data or single-sex length and age data. If clDataCons includes adjustments to the landed weight these are applied as: Total Landings = Official Landings * Multiplier + Unallocated Catch + Misallocated Catch If the multiplier field is NA it is assumed to be one, if unallocated catch and misallocated catch are NA they are assumed to be zero.

Value

The following slots of dbeOutp are filled:

`methodDesc`	Method of estimation
`nSamp`	Number of length samples by technical strata and number of age samples where technical is output as "AgeSamples"
`nMes`	Number of length measurements by technical strata and number of age measurements where technical is output as "Otoliths"
`lenStruc`	Numbers-at-length. For vesselRaise.boot bootstrap replicates are also given and iter=0 contains estimates from the original data.
`lenVar`	Variance of numbers-at-length.
`ageStruc`	Numbers-at-age. For vesselRaise.boot bootstrap replicates are also given and iter=0 contains estimates from the original data.
`ageVar`	Variance of numbers-at-age.
`totalN`	Total numbers. For vesselRaise.boot bootstrap replicates are also given and iter=0 contains estimates from the original data.
`totalNvar`	Variance of total numbers.
`totalW`	Total landed weight (kg)

Note

You should check the number of samples and age-length data are suitable before calculating estimates. vesselRaise.an stops if there are length classes with no age information. vesselRaise.boot fills in gaps that are 1 or 2 length groups wide with ages from surrounding length groups, see alkLgthRec(type="fillMiss", value=2).

Author(s)

David Maxwell

References

http://wwz.ifremer.fr/cost

See Also

dbeOutput, dbeOutput, dbeOutput, dbeOutput, dbeOutput

Examples

```
# load example data set
data("LEMexample")
# Setup objects for output
# LEM.strat was defined when creating consolidated object
LEM.dbeOut = dbeObject(desc="Example",species="Microstomus kitt",param="landings",
                       strataDesc=LEM.strat, catchCat="LAN",methodDesc="bootstrap samples")
LEM.dbeOut.an = dbeObject(desc="Example",species="Microstomus kitt",param="landings",
                          strataDesc=LEM.strat, catchCat="LAN",methodDesc="analytical")

# Run vesselRaise.boot function. B set to a very low number of iterations for demonstration only.
# example data set is very small so it generates warnings about the number of samples.
LEM.dbeOut = vesselRaise.boot (csObject = LEM.CScon, clObject = LEM.CLcon,
                               dbeOutp = LEM.dbeOut, B = 3 )
dbePlotRep(LEM.dbeOut,"lenStruc") # outputs to 2 pages turn graph history on, to see the first page

# Run vesselRaise.an function
LEM.dbeOut.an = vesselRaise.an (csObject = LEM.CScon, clObject = LEM.CLcon, dbeOutp = LEM.dbeOut.an)
# CV (see function dbeCalc for full implementation of calculating CV and confidence intervals)
100*sqrt(LEM.dbeOut.an@ageVar$value)/LEM.dbeOut.an@ageStruc$estim$value
```

---

viewGapsAlkCons *Calculation and display of age-length keys with gaps underlining*

Description

This method calculates and underlines empty length classes in an age-length key from a given 'csDataCons' object (ca table). '.' shows length classes that are not recorded in hl for the specified stratum. '-' shows length classes that are recorded in hl but not in ca, for the specified stratum.

Usage

```
viewGapsAlkCons(object,...)
```

Arguments

object	A *csDataCons* object.
...	Further arguments.

Author(s)

Mathieu Merzereaud

See Also

alkLgthRec, propMissLgthCons, csDataCons

Examples

```
data(sole)
  #subset to "27.7.d" & "27.7.e" areas
csSub <- subset(sole.cs,area%in%c("27.7.d","27.7.e"),table="hh")
conSole.cs <- csDataCons(csDataVal(csSub),strIni(spaceStrata="area"))

viewGapsAlkCons(conSole.cs)
```

Index