# Package: RDBEScore (via r-universe)

May 15, 2026

**Type** Package

**Title** Functions for the ICES Regional Database and Estimation System (RDBES)

**Version** 0.3.5

**Author** c( person(given = ``David", family = ``Currie", role = c(``aut"), comment = c(ORCID = ``0000-0002-3523-6895")), person(given = ``Richard", family = ``Meitern", role = c(``aut"), email = ``richard.meitern@ut.ee", comment = c(ORCID = ``0000-0002-2600-3002")), person(given = ``Nuno", family = ``Prista", role = c(``aut"), email = ``nuno.prista@slu.se", comment = c(ORCID = ``0000-0002-5145-7241")), person(given = ``Nicholas", family = ``Carey", role = c(``aut"), email = ``nicholas.carey@gov.scot"), person(given = ``Petri", family = ``Sarvamaa", role = c(``aut"), email = ``Petri.Sarvamaa@luke.fi"), person(given = ``Kirsten", family = ``Birch Håkansson", role = c(``aut"), email = ``kih@aqua.dtu.dk"), person(given = ``Karolina", family = ``Molla Gazi", role = c(``aut"), email = ``karolina.mollagazi@wur.nl"), person(given = ``Julia", family = ``Wischnewski", role = c(``aut"), email = ``julia.wischnewski@thuenen.de"), person(given = ``Ana Cláudia", family = ``Fernandes", role = c(``aut"), email = ``acfernandes@ipma.pt"), person(given = ``Katarzyna", family = ``Krakówka", role = c(``aut"), email = ``kkrakowka@mir.gdynia.pl"), person(given = ``Marta", family = ``Szymańska", role = c(``aut"), email = ``msuska@mir.gdynia.pl"), person(given = ``Nicolas", family = ``Goñi", role = c(``aut"), email = ``nicolas.goni@luke.fi"), person(given = ``Annica", family = ``de Groote", role = c(``ctb"), email = ``annica.isaksson.de.groote@slu.se"), person(given = ``Jonathan", family = ``Ball", role = c(``ctb"), email = ``johnathan.ball@cefas.co.uk"), person(given = ``Jonathan", family = ``Rault", role = c(``ctb"), email = ``Jonathan.Rault@ifremer.fr"), person(given = ``Antti", family = ``Sykkö", role = c(``ctb"), email = ``antti.sykko@luke.fi"), person(given = ``Liz", family = ``Clarke", role = c(``ctb"), email

= ``Liz.Clarke@gov.scot''), person(given = ``Chun'', family = ``Chen'', role = c(``ctb''), email = ``chun.chen@wur.nl''), person(given = ``Hongru'', family = ``Zhai'', role = c(``ctb''), email = ``hongru.zhai@slu.se''), person(given = ``Eros'', family = ``Quesada'', role = c(``ctb''), email = ``eros.quesada@slu.se''), person(given = ``Jonathan'', family = ``Stounberg'', role = c(``ctb''), email = ``jostou@aqua.dtu.dk''), person(given = ``Ana'', family = ``Ribeiro Santos'', role = c(``ctb''), email = ``ana.ribeirosantos@cefas.co.uk''), person(given = ``Jose'', family = ``Castro'', role = c(``ctb''), email = ``jose.castro@ieo.es''), person(given = ``Jessica'', family = ``Craig'', role = c(``ctb''), email = ``Jessica.Craig@gov.scot'') )

**Maintainer** Colin Millar <colin.millar@ices.dk>

**Description** The RDBEScore package provides functions to import and work with fisheries data downloaded from the ICES RDBES database. It also contains functions to perform estimation analysis using the resulting objects.

**URL** https://github.com/ices-tools-dev/RDBEScore, https://rdbes.ices.dk, https://ices-tools-dev.github.io/RDBEScore/

**License** GPL-3 + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**Roxygen** list(markdown = TRUE)

**Suggests** knitr, rmarkdown, testthat

**Depends** R (>= 4.1)

**Imports** dplyr (>= 1.1.3), data.table

**VignetteBuilder** knitr

**Repository** https://ices-tools-dev.r-universe.dev

**Date/Publication** 2025-10-17 12:51:26 UTC

**RemoteUrl** https://github.com/ices-tools-dev/RDBEScore

**RemoteRef** HEAD

**RemoteSha** 9077812f232a2a62afcc52483f54a514d7a57468

# Contents

---

`addCLtoLowerCS` *Add CL data to a lower-level CS table in an RDBESDataObject*

---

## Description

This function adds data from a `CL` table in an `RDBESDataObject` to a BV or FM table. It combines information from the `CS` and `CL` tables and calculates aggregate statistics such as the sum of the specified fields in the `CL` table.

## Usage

```
addCLtoLowerCS(
  rdbes,
  strataListCS,
  strataListCL,
  combineStrata = T,
  lowerHierarchy = "C",
  CLfields = c("CLoffWeight"),
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `rdbes` | An object of class `RDBESDataObject`. This object contains tables from the RDBES data structure, including a `CL` table and `CS` data. |
| `strataListCS` | A named list of filter criteria for subsetting the `CS` data in the `RDBESDataObject`. The names of the list should match column names in any of the `CS` tables. |
| `strataListCL` | A named list of filter criteria for subsetting the `CL` data in the `RDBESDataObject`. The names of the list should match column names in the `CL` table. |

| | |
|---|---|
| combineStrata | Logical, if TRUE, strata in the strataListCS are combined using a vertical bar (\|). |
| lowerHierarchy | A character string specifying the level of the lower hierarchy table to which the CL data will be added. Currently, only "C" is supported ie BV data only. |
| CLfields | A character vector of field names from the CL table that will be summed and added as new columns to the lower-level biological data. |
| verbose | Logical, if TRUE, the function prints informative text. |

## Details

The function first subsets the biological data in the RDBESDataObject based on the criteria in strataListCS. It then retrieves the corresponding CL data based on the criteria in strataListCL, sums the fields specified in CLfields, and adds them as new columns to the biological data. If combineStrata is TRUE, strata columns from the CS data are collapsed using a vertical bar (|). The function currently supports only biological data at the "C" hierarchy level.

## Value

A data.table containing the biological data from the lower hierarchy with added strata information from the CL table and the sum of the specified fields from the CL data.

## See Also

getLowerTableSubsets, upperTblData

## Examples

```
## Not run:
strataListCS <- list(LEarea="27.3.d.28.1",
                     LEmetier6 = "OTM_SPF_16-31_0_0",
                     TEstratumName = month.name[1:3],
                     SAspeCodeFAO = "SPR")
strataListCL <- list(CLarea="27.3.d.28.1",
                     CLquar = 1,
                     CLmetier6 = "OTM_SPF_16-31_0_0",
                     CLspecFAO = "SPR")
biolCL <- addCLtoLowerCS(rdbesObject, strataListCS, strataListCL,
                         combineStrata = TRUE,
                         lowerHierarchy = "C",
                         CLfields = c("CLoffWeight"))

## End(Not run)
```

`applyGenerateProbs` *Generate probabilities missing from RDBES Data*

## Description

Wrapper to generate probabilities. The wrapper calls runChecksOnSelectionAndProbs which main tests need to be passed before probabilities can be calculated. The it calls generateProbs for each sample in each sampling level of the hierarchy.

## Usage

```
applyGenerateProbs(
  x,
  probType,
  overwrite,
  runInitialProbChecks = TRUE,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| `x` | • RDBES data object |
| `probType` | • string. Can be set to "selection" (only selection probabilities are calculated), "inclusion" (only inclusion probabilities are calculated) or "both" (both types of probabilities are calculated) |
| `overwrite` | • if TRUE will overwrite probabilities already existing for SRSWR and SRSWOR |
| `runInitialProbChecks` | • if TRUE runs runChecksOnSelectionAndProbs |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| `strict` | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Value

a list of all the RDBES data tables with probabilites calculated

## See Also

`runChecksOnSelectionAndProbs` `generateProbs`

## Examples

```
# To be added
```

## check_key_column *Check if a column exists in a data table and has unique values*

### Description

This function checks if a specified column exists in a given data table and has unique values. If the column does not exist or has non-unique values, an error is thrown.

### Usage

```
check_key_column(dt, col)
```

### Arguments

| | |
|---|---|
| dt | A data table to check |
| col | A character string specifying the name of the column to check |

### Value

nothing if the column exists and has unique values, otherwise an error is thrown

### Examples

```
## Not run:
  RDBEScore:::check_key_column(H1Example$DE, "DEid")

## End(Not run)
```

## combineRDBESDataObjects *Combine Two RDBES Raw Objects combines 2 RDBESDataObjects into a single RDBESDataObject by merging individual tables one by one*

### Description

Combine Two RDBES Raw Objects combines 2 RDBESDataObjects into a single RDBESDataObject by merging individual tables one by one

### Usage

```
combineRDBESDataObjects(
  RDBESDataObject1,
  RDBESDataObject2,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| RDBESDataObject1 | The first object to combine |
| RDBESDataObject2 | The second object to combine |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Details

When combining RDBESDataObjects from different hierarchies (e.g., H1 and H5), a warning is issued. The resulting combined object will have a mixed hierarchy, which may be structurally and statistically invalid for some analyses. However, such combinations can be useful for fisheries overviews, annual reports, or countries performing broader estimations.

## Value

the combination of RDBESDataObject1 and RDBESDataObject2

## See Also

rbindlist

## Examples

```
## Not run:

myH1RawObject <-
    importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19")
myH5RawObject <-
    importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h5_v_1_19")

myCombinedRawObject <- combineRDBESDataObjects(RDBESDataObject1=myH1RawObject,
                                               RDBESDataObject2=myH5RawObject)

## End(Not run)
```

---

# createDBEPrepObj *Load raw object and create prepared object Function relies on the data being correctly named following established hierarchy*

---

## Description

Load raw object and create prepared object Function relies on the data being correctly named following established hierarchy

### Usage

```
createDBEPrepObj(input, output)
```

### Arguments

| | |
|---|---|
| input | a string pointing towards the input folder |
| output | a string pointing towards the output folder |

### Value

.Rdata files

### Examples

```
## Not run:
input <- "WKRDB-EST2/testData/output/DBErawObj/"
output <- "WKRDB-EST2/subGroup1/personal/John/PreparedOutputs/"

createDBEPrepObj(input = input, output = output)

## End(Not run)
```

---

## createRDBESDataObject *Create an RDBES Data Object*

---

### Description

This function lets you create an RDBES Data object in your current R environment.

### Usage

```
createRDBESDataObject(
  input = NULL,
  listOfFileNames = NULL,
  castToCorrectDataTypes = TRUE,
  verbose = FALSE,
  ...
)
```

### Arguments

| | |
|---|---|
| input | Strings or `list` object. The path to the zip file downloaded from RDBES (or multiple zip files - see details), or path to a folder of `csv` files, or a `list` object in the current environment containing data frames of each table. If `NULL` an empty `RDBESDataObject` is created. |

`listOfFileNames`
: `list` of Strings, Optional. For use with `csv` inputs only, and only required if the csv file names are *not* the default file names used by RDBES when downloading data (for instance if you created the files yourself). The actual file names should be a `list` of the two-letter code for the relevant table e.g. `list("DE" = "DE.csv", "SD" = "SD.csv", etc.)`. If not used then it is assumed the files have the default file names used by the RDBES data download ("Design.csv" etc).

`castToCorrectDataTypes`
: Logical. If `TRUE` then the function will attempt to cast the required columns to the correct data type. If `FALSE` then the column data types will be determined by how the csv files are read in. Default is `TRUE`.

`verbose`
: (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE.

`...`
: Additional parameters forwarded to helper functions used by this function. Most commonly these are forwarded to `validateRDBESDataObject()` during the validation step. Common options:

  - `strict` (logical, default `TRUE`): if `FALSE`, validation issues result in warnings instead of stopping with an error.
  - `verbose` (logical, default `FALSE`): request extra informational output from validation.
  - `Hierarchy` (integer, e.g. `1`, optional; zip inputs only): when the zip file contains multiple hierarchies, selects which hierarchy to import. Note: `checkDataTypes` is controlled by the `castToCorrectDataTypes` argument of this function and should not be supplied via `...`.

## Details

The `input` should be either:

- A `zip` file downloaded from RDBES (or multiple zip files if you want to include or overwrite tables, for example CL and CE data). NOTE: Only the downloaded RDBES data with Table data format with ids is loaded by this function and not the uploaded format.
- A folder containing `csv` files downloaded from RDBES (e.g. the unzipped file), or any set of csv files of the RDBES tables.
- A `list` of data frames in the current environment representing different tables in the hierarchy.
- A `NULL` input will return and empty RDBES data object

**ZIP file inputs** This `input` should be a path to a zip file downloaded from RDBES. Multiple zip files can be entered if you want to include additional tables, for example CL and CE. E.g. 'input = c("path/to/H1.zip", "path/to/CL.zip"). If any tables in the first input are overwritten by other inputs a warning is given. You should not input different hierarchy files; this function will not combine them.

If the zip contains multiple hierarchies (e.g., H1 and H5 within the same archive), you can select which one to import by passing `Hierarchy` via `...`, for example: `Hierarchy = 1`. If `Hierarchy` is not specified and the zip contains multiple hierarchies, an error is raised prompting you to set it.

**CSV file inputs** This `input` should be a path to a folder of `csv` files. These can be the `csv` files downloaded from RDBES (e.g. an unzipped hierarchy), or *any* set of csv files containing RDBES tables. If the files do not have the default RDBES name (e.g. 'Design.csv') the `listOfFileNames` input can by used to specify the file names e.g. `list("DE" = "DE.csv", "SD" = "SD.csv", etc.)`.

**List of data frames inputs** This `input` should be a `list` object containing data frames (or data.tables) for each table in your hierarchy. They should be named with the appropriate 2-letter code (`DE`, `SD`, etc.). Columns within these tables will be renamed to the RDBES model documentation 'R name'. Note if you choose to create an `RDBESDAtaObject` from local data frames these may have not passed the data integrity checks performed when you upload to RDBES!

**NULL inputs** This `input` produces an empty `RDBESDataObject`, i.e. all tables with correct data classes but the tables will be empty.

## Value

A RDBESDataObject

## Examples

```
# Create an empty object
myEmptyRDBESObject <- createRDBESDataObject(input = NULL)
```

---

`createRDBESEstObject` *Creates an RDBESEstObject from RDBES data*

---

## Description

Creates an RDBESEstObject from RDBES data

## Usage

```
createRDBESEstObject(
  rdbesPrepObject,
  hierarchyToUse = NULL,
  stopTable = NULL,
  verbose = FALSE,
  strict = TRUE,
  incDesignVariables = TRUE
)
```

## Arguments

| | |
|---|---|
| `rdbesPrepObject` | The RDBES object that should be used to create an estimation object |
| `hierarchyToUse` | The upper RDBES hierarchy to use |
| `stopTable` | (Optional) The table to stop at in the RDBES hierarchy. If specified, only tables up to and including this table will be included in the resulting RDBESEstObject. The default is NULL, which means all tables in the hierarchy will be included. |

| | |
|---|---|
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |
| incDesignVariables | (Optional) Should the design variables be included? The default is TRUE. |

## Value

An object of class RDBESEstObject ready for use in design based estimation

## Examples

```
myH1EstObj <- createRDBESEstObject(H1Example, 1, "SA")
```

---

# createTableOfRDBESIds *Create a table of RDBES Ids*

---

## Description

examples for now see https://github.com/ices-eg/WK_RDBES/tree/master/WKRDB-EST2/chairs/Nuno

## Usage

```
createTableOfRDBESIds(x, addSAseqNums = TRUE)
```

## Arguments

| | |
|---|---|
| x | RDBESdataObject |
| addSAseqNums | should SAseqNum be included? Default value is TRUE |

## Value

data frame of Ids of all tables in sampling hierarchy

## Examples

```
## Not run:

myH1RawObject <-
  importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19_13")

myTableOfIds<- createTableOfRDBESIds(myH1RawObject)

## End(Not run)
```

`DefaultFileNames` *The default files names when you download data from the RDBES.*

## Description

The default files names when you download data from the RDBES.

## Usage

```
DefaultFileNames
```

## Format

A list containing the default file names for each RDBES table

## Source

https://rdbes.ices.dk/

`designVariables` *A dataset containing the RDBES "design variable" names*

## Description

A dataset containing the RDBES "design variable" names

## Usage

```
designVariables
```

## Format

A vector containing the short R names of the RDBES design variables (without any 2 letter table prefixes) R field name:

**designVariables** The design variable names

## Source

https://sboxrdbes.ices.dk

doBVestimCANUM *Estimate Catch at Number (CANUM) for Biological Variables*

## Description

This function estimates catch at number (CANUM) for a specified biological variable, such as age or length. It aggregates data based on specified columns and generates a "plus group" for the highest value in the defined classes. The function supports grouping by various units (e.g., age, length, weight) and calculates required indices, totals, and proportions for the groups.

## Usage

```
doBVestimCANUM(
  bv,
  addColumns,
  classUnits = "Ageyear",
  classBreaks = 1:8,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| bv | A `data.table` containing biological data, with columns for the biological variable, class units (e.g., `Ageyear`, `Lengthmm`, `Weightg`), and other relevant variables. |
| addColumns | A character vector of additional column names used to group the data for aggregation (e.g., `BVfishId` and other identifiers). |
| classUnits | A character string specifying the class units of the biological variable to use for grouping (e.g., "Ageyear", "Lengthmm", "Weightg"). Default is "Ageyear". |
| classBreaks | A numeric vector specifying the breakpoints for classifying the biological variable. The last value defines the lower bound of the "plus group". Default is `1:8` for age groups. |
| verbose | Logical, if `TRUE`, prints detailed information about the process. Default is `FALSE`. |

## Details

The function performs the following steps:

- Validates the presence of the `classUnits` in the biological variable data.
- Reshapes the input data using `dcast` and groups the biological variable into classes using `cut()`.
- Aggregates mean weights and lengths by the defined classes, along with calculating proportions and indices based on the sample size.
- A "plus group" is created for values exceeding the highest `classBreaks` value.

- Calculates total weights, catch numbers, and performs a sanity check to ensure there are no rounding errors in the final results.

**Mathematical Logic::**

Let:

- $W_{mean}$ be the mean weight for each group.
- $L_{mean}$ be the mean length for each group.
- $n_W$ be the number of weight measurements in each group.
- $N$ be the total number of measurements in the sample.
- $P$ be the proportion of the sample represented by each group.
- $I_W$ be the weight index for each group.
- $S$ be the sum of weight indices across all groups.
- $C$ be the total catch weight.
- $T_W$ be the total weight for each group.
- $C_{num}$ be the total catch number for each group.

The calculations are as follows:

1. Proportion of sample:
$$P = \frac{n_W}{N}$$
2. Weight Index:
$$I_W = P \times \left( \frac{W_{mean}}{1000} \right)$$
3. Sum of Weight Indices:
$$S = \sum I_W$$
4. Total Weight Coefficient:
$$\text{TWCoef} = \frac{C}{S}$$
5. Total Weight per Group:
$$T_W = I_W \times \text{TWCoef}$$
6. Total Catch Number per Group:
$$C_{num} = \frac{T_W}{\left(\frac{W_{mean}}{1000}\right)}$$

## Value

A `data.table` containing the aggregated results, including groupings, calculated means, proportions, indices, and totals for the specified biological variable.

## doDBEestimantionObjUpp

*Generates the DBE estimation object for the upper hierarchy tables*

### Description

Generates the DBE estimation object for the upper hierarchy tables

### Usage

```
doDBEestimantionObjUpp(inputList)
```

### Arguments

| | |
|---|---|
| inputList | All the data tables in a named list. Name should be equal to the short table names e.g. DE, SD, TE, FO. |

### Value

The upper hierarchy tables in the DBE estimation object (DBEestimantionObjUpp)

### Examples

```
## Not run:
H1 <-
readRDS("./WKRDB-EST2/testData/output/DBErawObj/DBErawObj_DK_1966_H1.rds")
H1out <- doDBEestimantionObjUpp(H1)

## End(Not run)
```

## doDBestimation

*Create design-based point and variance estimates from RDBES estimation object (rdbesEstimObj)*

### Description

Create design-based point and variance estimates from RDBES estimation object (rdbesEstimObj)

### Usage

```
doDBestimation(
  x = rdbesEstimObj,
  estimateType = "total",
  pointEstimator = "Unbiased",
  varEstimator = "WRonPSUviaPik",
  stage = 0,
  domainOfinterest = NULL
)
```

### Arguments

| | |
|---|---|
| x | a data.frame (or data.table) in rdbesEstimObj format with value of target variable in column targetValue |
| estimateType | a string with type of estimate. As of now only "total" is defined |
| pointEstimator | a string with type of point estimator. As of now only "Unbiased" is defined |
| varEstimator | a string with type of variance estimator. As of now only "WRonPSUviaPik" is defined |
| stage | a natural number (0,1,..) with sampling stage of estimate. 0 corresponds to DE level. |
| domainOfinterest | list ofdomains of interest (e.g., SAarea). As of now only NULL (=no domain estimate) is defined |

### Value

a list of values for pointEstimate, varEstimate and estimation options

### Examples

```
## Not run:
data(shrimps)
doDBestimation (x = shrimps, estimateType = "total",
pointEstimator = "Unbiased", varEstimator = "WRonPSUviaPsi",stage = 0,
domainOfinterest = NULL )

## End(Not run)
```

---

## doEstimationForAllStrata

*Estimate totals and means, and try to generate samples variances for all strata in an RDBESEstObject*

---

### Description

Estimate totals and means, and try to generate samples variances for all strata in an RDBESEstObject

### Usage

```
doEstimationForAllStrata(RDBESEstObjectForEstim, targetValue, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| RDBESEstObjectForEstim | The RDBESEstObject to generate estimates for |
| targetValue | The field to estimate for, for example "SAsampWtLive" |
| verbose | (Optional) If set to TRUE more detailed text will be printed out by the function. Default is FALSE |

## Value

A data frame containing estimates for all strata

## Examples

```
## Not run:

myH1RawObject <-
  importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19")

# Update our test data with some random sample measurements
myH1RawObject[["SA"]]$SAsampWtLive <-
  round(runif(n = nrow(myH1RawObject[["SA"]]), min = 1, max = 100))

myH1EstObj <- createRDBESEstObject(myH1RawObject, 1)

myStrataEst <- doEstimationForAllStrata(
  RDBESDataObjectForEstim = myH1EstObj,
  targetValue = 'SAsampWtLive'
)

## End(Not run)
```

---

# doEstimationRatio *Estimate Numbers and Mean Values by Length or Age Class*

---

## Description

The function is under development and does not work yet.

## Usage

```
doEstimationRatio(
  RDBESDataObj,
  targetValue = "LengthComp",
  raiseVar = "Weight",
  classUnits = "mm",
  classBreaks = c(100, 300, 10),
  LWparam = NULL,
  lowerAux = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| RDBESDataObj | A validated RDBESDataObject containing hierarchical sampling and biological data. Must include appropriate tables (e.g., CL, CE, SA, FM, or BV) depending on estimation requirements. |

| | |
|---|---|
| targetValue | A character string specifying the type of composition to estimate. Options are "LengthComp" or "AgeComp". |
| raiseVar | The variable used to construct the ratio. |
| classUnits | Units of the class intervals for length or age, typically "mm" for millimeters or "cm" for centimeters. Used in defining class intervals. |
| classBreaks | A numeric vector of three values: minimum value, maximum value, and class width (e.g., c(100, 300, 10)). Defines the class intervals for grouping lengths or ages. |
| LWparam | A numeric vector of length two specifying parameters (a, b) for the weight-length relationship (W = a * L^b). Used if no direct weights are available but lengths are provided. |
| lowerAux | A numeric or character vector referencing a variable in the SA table used as an auxiliary variable for ratio estimation (e.g., sample weights, sub-sample expansion factors). |
| verbose | Logical; if TRUE, detailed messages are printed during processing. |

## Value

A list or data.table containing the estimated numbers at length or age and associated mean values such as weight and length, depending on input and target type.

---

# estim *Generic function for estimation of population total and variance*

---

## Description

Generic function for estimation of population total and variance

## Usage

```
estim(
  y,
  enk,
  enkl,
  method = "SRSWOR",
  estFunction,
  varFunction,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| y | numeric variable to be estimated |
| enk | expected value of k |
| enkl | expected value of k, given l |

| | |
|---|---|
| method | character selection method code e.g SRSWOR |
| estFunction | the function to use to estimate total given parameters y and enk |
| varFunction | the function to use to estimate variance given parameters y,enk and enkl |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

## Value

list of 7 elements including the population mean, total (and their variance), the algorithm name used and the I order inclusion probabilities

## Examples

```
estimMC(c(3, 4, 4, 5), c(4, 4, 4, 4), c(8, 8, 8, 8))
```

---

# estimMC *Multiple Count Estimator for Population Total and Variance*

---

## Description

Multiple Count Estimator for Population Total and Variance

## Usage

```
estimMC(
  y,
  sampled,
  total,
  method = "SRSWOR",
  selProb = NULL,
  incProb = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| y | numeric variable to be estimated |
| sampled | numeric total number of units sampled |
| total | numeric total number of units int the population |
| method | character selection method code e.g SRSWOR |
| selProb | the selection probabilities (if known) |
| incProb | the inclusion probabilities (if known) |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

### Value

list of 7 elements including the population mean, total (and their variance), the algorithm name used and the I order inclusion probabilities

### Examples

```
estimMC(c(3, 4, 4, 5), c(4, 4, 4, 4), c(8, 8, 8, 8))
```

---

## exportEstimationResultsToInterCatchFormat

*Export Estimation Results to InterCatch Exchange Format*

---

### Description

This function transforms the estimation results into the InterCatch format.

### Usage

```
exportEstimationResultsToInterCatchFormat(dataToExport, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `dataToExport` | A data frame containing the estimation results - this should include the output from the doEstimationForAllStrata function and already have the the InterCatch columns present. |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

### Value

A character vector representing the flattened InterCatch exchange format. The vector includes all fields from the HI, SI, and SD components, ordered by their associated keys, and is suitable for writing to an InterCatch-formatted exchange file.

---

## filterAndTidyRDBESDataObject

*Filter and remove orphan records in an RDBESDataObject*

---

### Description

This function filters an RDBESDataObject based on specified fields and values, and can optionally remove any orphan records. The returned object will include all rows which either: a) do not include any of the field names in `fieldsToFilter`, or b) do include the field names and have one of the allowed values in `valuesToFilter`. If `killOrphans` is set to `TRUE`, the function will remove orphaned rows. The default is `TRUE`.

## Usage

```
filterAndTidyRDBESDataObject(
  RDBESDataObjectToFilterAndTidy,
  fieldsToFilter,
  valuesToFilter,
  killOrphans = TRUE,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| RDBESDataObjectToFilterAndTidy | The RDBESDataObject to filter. |
| fieldsToFilter | A vector of the field names you wish to check. |
| valuesToFilter | A vector of the field values you wish to filter for. |
| killOrphans | Controls if orphan rows are removed. Default is TRUE. |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Value

The filtered input object of the same class as RDBESDataObjectToFilterAndTidy.

## Examples

```
## Not run:

myH1RawObject <- createRDBESDataObject(rdbesExtractPath = "tests\\testthat\\h1_v_1_19_13")

# To check how removeBrokenVesselLinks() works
myH1RawObject$VD$VDlenCat[which(myH1RawObject$VD$VDencrVessCode=="VDcode_10")] <- "VL40XX"

myFields <- c("VSencrVessCode", "VDlenCat")
myValues <- c("VDcode_1","VDcode_2", "VDcode_10","VL1518","VL2440")

myFilteredObject <- filterAndTidyRDBESDataObject(myH1RawObject,
  fieldsToFilter = myFields,
  valuesToFilter = myValues,
  killOrphans = TRUE, verboseBrokenVesselLinks = TRUE
)

## End(Not run)
```

`filterRDBESDataObject` *Filter an RDBESDataObject*

## Description

The returned object will include all rows which either: a) do not included any of the field names in `fieldsToFilter`, or b) do include the field names and have one of the allowed values in `valuesToFilter`. If you want to filter for a id field like `DEid`, `FTid` etc, the filtering works only on the table where the id field is its key. For example, if you try to filter on `FOid` it does not look `FOid` in other tables like `FT`, although the field `FOid` exists in `FT` table.

## Usage

```
filterRDBESDataObject(
  RDBESDataObjectToFilter,
  fieldsToFilter,
  valuesToFilter,
  killOrphans = FALSE,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| `RDBESDataObjectToFilter` | The `RDBESDataObject` to filter |
| `fieldsToFilter` | A vector of the field names you wish to check |
| `valuesToFilter` | A vector of the field values you wish to filter for |
| `killOrphans` | Controls if orphan rows are removed. Default is `FALSE`. |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| `strict` | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Details

`killOrphans` allows you to remove orphaned rows if set to `TRUE`. The default is `FALSE`.

## Value

the filtered input object of the same class as `RDBESDataObjectToFilter`

## Examples

```
## Not run:

myH1RawObject <-
  importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19")

myFields <- c("SDctry", "VDctry", "VDflgCtry", "FTarvLoc")
myValues <- c("ZW", "ZWBZH", "ZWVFA")

myFilteredObject <- filterRDBESDataObject(myH1RawObject,
  fieldsToFilter = myFields,
  valuesToFilter = myValues
)

# Inverse filtering (exclude certain values)
# Example: keep all DE rows except those with DEid in `excludedValues`
# Compute the complement of the excluded set using setdiff
allValues <- unique(myH1RawObject$DE$DEid)
excludedValues <- c(5351)
myInverseFiltered <- filterRDBESDataObject(
  myH1RawObject,
  fieldsToFilter = "DEid",
  valuesToFilter = setdiff(allValues, excludedValues)
)

## End(Not run)
```

---

`filterRDBESEstObject` *Filter an RDBESEstObject*

---

## Description

The returned object will include all rows which include the field names and have one of the allowed values in `valuesToFilter`.

## Usage

```
filterRDBESEstObject(
  RDBESEstObjectToFilter,
  fieldsToFilter,
  valuesToFilter,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `RDBESEstObjectToFilter` | The `RDBESEstObject` to filter |
| `fieldsToFilter` | A vector of the field names you wish to check |

| | |
|---|---|
| valuesToFilter | A vector of the field values you wish to filter for |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

## Value

the filtered input object of the same class as RDBESEstObjectToFilter

## Examples

```
## Not run:

 myRawObject <- createRDBESDataObject(input = "tests\\testthat\\h1_v_1_19_26")
 myEstObject <- createRDBESEstObject(myRawObject,1)
 myFilteredEst <- filterRDBESEstObject(myEst,c("BVid"),c(7349207))


## End(Not run)
```

---

# findAndKillOrphans

*This function finds and removed any orphan records in an RDBESDataObject. Normally data that has been downloaded from the RDBES will not contain orphan records - however if the data is subsequently filtered it is possible to introduce orphan records.*

---

## Description

This function finds and removed any orphan records in an RDBESDataObject. Normally data that has been downloaded from the RDBES will not contain orphan records - however if the data is subsequently filtered it is possible to introduce orphan records.

## Usage

```
findAndKillOrphans(objectToCheck, verbose = FALSE, strict = TRUE)
```

## Arguments

| | |
|---|---|
| objectToCheck | an RDBESDataObject. |
| verbose | (Optional) If set to TRUE more detailed text will be printed out by the function. Default is FALSE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Value

an RDBESDataObject with any orphan records removed

## Examples

```
## Not run:

myH1RawObject <-
importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19")
myFields <- c("SDctry","VDctry","VDflgCtry","FTarvLoc")
myValues <- c("ZW","ZWBZH","ZWVFA" )
myFilteredObject <- filterRDBESDataObject(myH1RawObject,
                                          fieldsToFilter = myFields,
                                          valuesToFilter = myValues )
myObjectNoOrphans <- findAndKillOrphans(objectToCheck = myFilteredObject,
                                        verbose = FALSE)

## End(Not run)
```

---

`findOrphansByTable` *Internal function to identify orphan records in a given RDBESDataObject table*

---

## Description

Internal function to identify orphan records in a given RDBESDataObject table

## Usage

```
findOrphansByTable(tableToCheck, objectToCheck, foreignKeyIds, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `tableToCheck` | The two letter code for the table to check |
| `objectToCheck` | An RDBESDataObject |
| `foreignKeyIds` | A vetor of the foreign key field names to check |
| `verbose` | (Optional) If set to TRUE more detailed text will be printed out by the function. Default is TRUE. |

## Value

A data frame with the primary keys of the table checked, the two letter table identifier, and their orphan status.

`fixSLids` *Fixes SLid in SL table (facilitating SS-SL joins).*

## Description

Fixes SLid in SL table (facilitating SS-SL joins).

## Usage

```
fixSLids(RDBESDataObject, verbose = FALSE, validate = TRUE, strict = TRUE)
```

## Arguments

| | |
|---|---|
| `RDBESDataObject` | A valid RDBESDataObject |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| `validate` | (Optional) Should the function validate its input data? The default is TRUE. |
| `strict` | (Optional) If the function validates its input data - should the validation be strict? The default is TRUE. |

## Details

RDBES SL can be seen as a join of two tables - one that identifies the species list in terms of *SLcou * SLinst * SLspeclistName * SLyear * SLcatchFrac* and one that specifies the taxa (*SLcommTaxon * SLsppCode*) in the list. In SS, SLid remits to the 1st taxa in a species list and not - as it would be expected - to the species list itself. This function fixes this by creating a new SLtaxaId variable in SL and assigning all taxa in a species to a single SSid.

## Value

an RDBESDataObject with SL ids reworked

## Examples

```
# To add
```

`generateMissingSSRows` *Generate any missing SS rows. When FOcatchReg=="All" it is expected that SScatchFraction is either "Catch" OR "Lan"+"Dis". In the latter case, if one is missing the other is to be assumed 0. This function generates SS rows for any missing catch fractions.*

## Description

Generate any missing SS rows. When FOcatchReg=="All" it is expected that SScatchFraction is either "Catch" OR "Lan"+"Dis". In the latter case, if one is missing the other is to be assumed 0. This function generates SS rows for any missing catch fractions.

## Usage

```
generateMissingSSRows(
  RDBESDataObject,
  speciesListName,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| `RDBESDataObject` | A valid RDBESDataObject |
| `speciesListName` | The name of the Species List you want to use for any SS rows that are created. |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| `strict` | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Value

A data table of SS data with any missing rows added

## Examples

```
# To follow
```

`generateNAsUsingSL` *Generate NAs in samples using Species List information*

## Description

Generate NAs in samples using Species List information

## Usage

```
generateNAsUsingSL(
  RDBESDataObject,
  targetAphiaId,
  overwriteSampled = TRUE,
  validate = TRUE,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| `RDBESDataObject` | An RDBESDataObject. |
| `targetAphiaId` | a vector of aphiaId. |
| `overwriteSampled` | (Optional) should SAtotalWtMes and SAsampWtMes be set to 0 if spp recorded but absent from SL? The default is TRUE. |
| `validate` | (Optional) Set to TRUE if you want validation to be carried out. The default if TRUE. |
| `verbose` | (Optional) Set to TRUE if you want informative text on validation printed out, or FALSE if you don't. The default is FALSE. |
| `strict` | (Optional) This function can validate its input data - should the validation be strict? The default is TRUE. |

## Value

RDBES data object where SA was complemented with NAs for species not looked for (sensu in SL)

## Examples

```
# To be added
```

generateProbs *Generate vector of selection or inclusion probabilities*

## Description

Generate vector of selection or inclusion probabilities

## Usage

```
generateProbs(x, probType, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| x | RDBES data object |
| probType | "selection" or "inclusion" for selection and inclusion probabilities respectively |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

## Details

When the selection method is SRSWR selection probabilities are calculated as $1/N$ and inclusion probabilities as $1 - (1 - 1/N)^n$. When the selection method is SRSWOR selection probabilities are not currently implemented. Inclusion probabilities are calculated as $n/N$. When the selection method is CENSUS both types of probabilities are set to 1. Probabilities for selection methods UPSWR and UPSWOR are not calculated (they need to be supplied by the user). The same happens with regards to non-probabilistic methods

## Value

A vector or probabilities

## Examples

```
## Not run:
generateProbs(x = Pckg_SDAResources_agstrat_H1[["VS"]], probType = ("inclusion"))
# population size
a<-generateProbs(x = Pckg_SDAResources_agstrat_H1[["VS"]], probType = ("inclusion"))
sum(1/a$VSincProb)
# returns error
generateProbs(x = Pckg_SDAResources_agstrat_H1[["VS"]], probType = ("selection"))


## End(Not run)
```

## generateSSRows *Private function to generate SS rows*

### Description

Private function to generate SS rows

### Usage

```
generateSSRows(FOids, speciesListName, catchFra)
```

### Arguments

| | |
|---|---|
| FOids | Vector of FOids |
| speciesListName | Name of the species list |
| catchFra | The catch fraction to create |

### Value

SS data frame

## generateTestTbls *Generate a List of Related Data Tables*

### Description

Generates a named list of data tables that follow the structure of RDBESDataObject. The tables only have columns required for testing The generate tables

### Usage

```
generateTestTbls(tblNames, prevTbls = list(), ...)
```

### Arguments

| | |
|---|---|
| tblNames | character vector of table names to be created |
| prevTbls | list of data.tables upstream of the generated table. Defaults to empty list |
| ... | Arguments passed on to makeTbl<br>tblName Name of the table<br>rows numeric number of rows per parent record. Defaults to 4.<br>propSamp numeric proportion of how many of total are sampled. This is ignored for "CENSUS". Defaults to 0.5 |

selMeth character selection method used. Defaults to "CENSUS". Others like SRSWR or SRSSWOR can be used as well

stratums character vector of the stratum names to be created. Defaults to c("U"), meaning not stratified.

mean numeric the expected mean of the target variable. The variable is created using rnorm and saved under column ending with "y". Defaults to 5.

## Value

a list of named data.table's

## Examples

```
## Not run:
generateTestTbls(c("A", "B", "C"), selMeth = "SRSWOR")
generateTestTbls(LETTERS[1:5]) # makes 5 tables with method CENSUS

## End(Not run)
```

---

# generateZerosUsingSL *Generate zeros in samples using Species List information*

---

## Description

examples for now see https://github.com/ices-eg/WK_RDBES/tree/master/WKRDB-EST2/chairs/Nuno

## Usage

```
generateZerosUsingSL(x, verbose = FALSE, strict = TRUE)
```

## Arguments

| | |
|---|---|
| x | RDBES data frame |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

## Value

RDBES data frame where SA was complemented with species looked for (sensu in sampling objectives) but not registered in sample

## getEstimForStratum *Private function used by doEstimationForAllStrata to get the estimates*

### Description

Private function used by doEstimationForAllStrata to get the estimates

### Usage

```
getEstimForStratum(x)
```

### Arguments

| | |
|---|---|
| x | The input |

### Value

Data frame with estimated values

## getLinkedDataFromLevel *Retrieve Linked Data Between RDBES Tables at a Specified Level*

### Description

The `getLinkedDataFromLevel` function facilitates the retrieval of linked data between different levels of RDBES tables. Depending on the relative positions of the source and target tables within the `RDBESDataObject`, the function determines whether to traverse "up" or "down" the data hierarchy to obtain the desired linked data.

### Usage

```
getLinkedDataFromLevel(field, values, rdbesTables, level, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `field` | A character string specifying the field name from which to retrieve linked data. The first two characters of this field indicate the source table. |
| `values` | A vector of values corresponding to the specified `field` for which linked data is to be retrieved. |
| `rdbesTables` | An `RDBESDataObject` containing the relevant RDBES tables. This object should include all tables that may be linked based on the provided `field` and `level`. |
| `level` | A character string specifying the target table level from which to retrieve linked data. This must be one of the names within the `rdbesTables` object. |
| `verbose` | Logical flag indicating whether to print detailed information about the data retrieval process. Default is `FALSE`. |

## Value

The subset of the table at the specified `level`.

## Examples

```
## Not run:
# Example 1: Going up in the table hierarchy to retrieve data from the DE table
# Retrieve data from the DE level based on BVid from the BV table
# This returns 1 row from the DE table
getLinkedDataFromLevel("BVid", c(1), H8ExampleEE1, "DE", TRUE)

# Example 2: Going down in the table hierarchy to retrieve data from the SA table
# Retrieve data from the SA level based on DEid from the DE table
# This returns 15 rows from the SA table
getLinkedDataFromLevel("DEid", c(1), H8ExampleEE1, "SA", TRUE)

# Example 3: Going up in the table hierarchy to see the Vessel that caught a specific fish
# Retrieve data from the VS level based on BVfishId from the BV table
getLinkedDataFromLevel("BVfishId", c("410472143", "410472144"), H8ExampleEE1, "VS", TRUE)

## End(Not run)
```

---

# `getLowerTableSubsets` *Get Lower Table from Several Upper Table Fields*

---

## Description

This function takes a list of subsets, a target lower level table name, and a list of tables. It returns a unique data frame containing the rows of the target lower level table that are associated with the given values of the upper table field in each subset. The function can also add the subset values to the result for reference.

## Usage

```
getLowerTableSubsets(
  subsets,
  tblName,
  rdbesTables,
  combineStrata = TRUE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `subsets` | A named list of vectors. Each vector contains values for a specific upper table field. |
| `tblName` | A character string specifying the name of the target lower level table. |
| `rdbesTables` | A RDBESData object containing the tables. |

| | |
|---|---|
| combineStrata | A logical value indicating whether to include the strata information in the result. If TRUE, and if any strata has more than one value, those values are collapsed into a single string and appended to the result with a warning. |
| verbose | A logical value indicating whether to print informative text. |

## Details

The function recursively intersects the rows of the target lower level table that match the values from each subset in the upper tables. It then ensures that only unique rows are returned, based on the ID column of the target table.

## Value

A unique data frame containing the rows of the target lower level table that are associated with the given values of the upper table field in each subset. If combineStrata = TRUE, the result will also include a column for each subset with the corresponding collapsed values.

---

# getMissingSSCatchFraction

*Private function to find which FO rows are not matching SS*

---

## Description

Private function to find which FO rows are not matching SS

## Usage

```
getMissingSSCatchFraction(FOdata, SSdata, catchFra, verbose)
```

## Arguments

| | |
|---|---|
| FOdata | The FOdata |
| SSdata | The SSdata |
| catchFra | The catchfra |
| verbose | verbose or not? |

## Value

Vector of FOids that aren't matching SS rows

`getTablesInRDBESHierarchy` *Returns the tables for a given hierarchy*

## Description

Returns the tables for a given hierarchy

## Usage

```
getTablesInRDBESHierarchy(
  hierarchy,
  includeOptTables = TRUE,
  includeLowHierTables = TRUE,
  includeTablesNotInSampHier = TRUE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `hierarchy` | Integer value between 1 and 13 inclusive |
| `includeOptTables` | Include any optional tables? Default value is TRUE |
| `includeLowHierTables` | Include the lower hierarchy tables? Default value is TRUE |
| `includeTablesNotInSampHier` | Include tables that aren't sampling units in that hierarcy? Default value is TRUE |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

## Value

A vector containing the 2-letter names of the tables in the requested hierarchy

## Examples

```
getTablesInRDBESHierarchy(5)
```

H1Example *A dataset containing test RDBES data for H1 in the RDBESDataObject structure*

## Description

A dataset containing test RDBES data for H1 in the RDBESDataObject structure

## Usage

```
H1Example
```

## Format

A list containing entries required for H1 RDBES data:

**DE** the Design data table

**SD** the Sampling Details data table

**VS** the Vessel Selection data table

**FT** the Fishing Trip data table

**FO** the Fishing Operation data table

**SS** the Species Selection data table

**SA** the Sample data table

**FM** the Frequency Measure data table

**BV** the Biological Variable data table

**VD** the Vessel Details data table

**SL** the Species List data table

**IS** the Individual Species table

H5Example *A dataset containing test RDBES data for H5 in the RDBESDataObject structure*

## Description

A dataset containing test RDBES data for H5 in the RDBESDataObject structure

## Usage

```
H5Example
```

## Format

A list containing entries required for H5 RDBES data:

**DE** the Design data table
**SD** the Sampling Details data table
**FT** the Fishing Trip data table
**OS** the Onshore Event data table
**LE** the Landing Event data table
**SS** the Species Selection data table
**SA** the Sample data table
**FM** the Frequency Measure data table
**BV** the Biological Variable data table
**VD** the Vessel Details data table
**SL** the Species List data table
**IS** the Individual Species table

---

`H7Example` *A dataset containing test RDBES data for H7 in the RDBESDataObject structure*

---

## Description

This dataset does not have passed the RDBES upload checks, hence the object might be somewhat invalid, however it resembles real data from the Estonian Market Sampling for 2022 for 2 species

## Usage

```
H7Example
```

## Format

A list containing entries required for H7 RDBES data:

**DE** the Design data table
**SD** the Sampling Details data table
**OS** the Onshore Sample data table
**LE** the Landing Event data table
**SS** the Species Selection data table
**SA** the Sample data table
**BV** the Biological Variable data table
**SL** the Species List data table
**IS** the Individual Species table

#' @source Richard Meitern @ Estonian Marine Institute, 2025

H8ExampleEE1 *A dataset containing test RDBES data for H8 in the RDBESDataObject structure*

## Description

This dataset does not have passed the RDBES upload checks, hence the object might be somewhat invalid, however it resembles real data from the Estonian Baltic Trawling fleet for 2022 sprat total landings and commercial sampling

## Usage

```
H8ExampleEE1
```

## Format

A list containing entries required for H8 RDBES data:

**DE** the Design data table

**SD** the Sampling Details data table

**TE** the Temporal Event data table

**VS** the Vessel Selection data table

**LE** the Landing Event data table

**SS** the Species Selection data table

**SA** the Sample data table

**BV** the Biological Variable data table

**VD** the Vessel Details data table

**SL** the Species List data table

**IS** the Individual Species

**CL** the Commertial Landing data table

**CE** the Commertial Effort data table

#' @source Richard Meitern @ Estonian Marine Institute, 2025

## icesSpecWoRMS *ICES Species (WoRMS) code list snapshot*

### Description

A dataset containing a copy of the ICES 'Species (WoRMS)' code list. The latest code list can be downloaded from https://vocab.ices.dk/.

### Usage

```
icesSpecWoRMS
```

### Format

A data frame with the following columns:

**CodeTypeGUID** GUID of the code type in ICES Vocabulary (e.g. the 'Species (WoRMS)' list).

**CodeTypeID** Numeric ID of the code type.

**Guid** GUID identifying this code record.

**Id** Numeric ID of this code record.

**Key** AphiaID (numeric key from WoRMS).

**Description** Scientific name.

**LongDescription** (If present) additional description; often not used.

**Modified** Datetime the record was last modified at ICES.

**Deprecated** Logical; whether this code is deprecated at ICES.

**DateDownloaded** Date the snapshot was downloaded, e.g. "2023-10-18".

### Source

https://vocab.ices.dk/

## killOrphans *Internal function to remove orphan records from an RDBESDataObject*

### Description

Internal function to remove orphan records from an RDBESDataObject

### Usage

```
killOrphans(objectToCheck, orphansToRemove)
```

### Arguments

| | |
|---|---|
| objectToCheck | an RDBESDataObject |
| orphansToRemove | The output from the findOrphansByTable function (A data frame with the primary keys of the table checked, the two letter table identifier, and their orphan status.) |

### Value

RDBESDataObject with orphan records removed

---

## listPackageFunctions *Extract Functions and Descriptions from an R Package*

---

### Description

This function extracts the list of functions contained within a specified R package and retrieves a brief description for each function from the package documentation. The description is obtained by parsing the Rd file associated with each function to extract the text from the `\title` field. In addition, the function determines whether each function is exported from the package by comparing against the package's exported names.

### Usage

```
listPackageFunctions(pkg)
```

### Arguments

| | |
|---|---|
| pkg | A character string or an unquoted name specifying the package from which to extract functions. The package must be installed and accessible. |

### Details

The function first accesses the package namespace using `asNamespace` and retrieves all objects using `ls`. It filters these objects to include only functions. For each function, the associated help file is retrieved using `utils::help` and the Rd file is extracted with `utils:::.getHelpFile`. The internal helper function `getRdTitle` is then used to parse the Rd object and extract the text in the `\title` field. Finally, the function assembles the output into a data frame that also includes an indicator of whether each function is exported.

### Value

A data frame with three columns. The `Function` column lists the names of the functions found in the package. The `Description` column contains the brief descriptions extracted from each function's documentation, and the `Exported` column is a logical vector indicating whether the function is exported from the package.

## Examples

```
## Not run:
# Extract functions from the stats package along with their descriptions and export status.
tab <- listPackageFunctions("stats")
print(tab)

## End(Not run)
```

---

`makeTbl` *Generate a Data Table*

---

## Description

Generate a Data Table

## Usage

```
makeTbl(
  tblName,
  prevTbls = list(),
  rows = 4,
  propSamp = 0.5,
  selMeth = "CENSUS",
  stratums = c("U"),
  mean = 5
)
```

## Arguments

| | |
|---|---|
| `tblName` | Name of the table |
| `prevTbls` | list of data.tables upstream of the generated table. Defaults to empty list |
| `rows` | numeric number of rows per parent record. Defaults to 4. |
| `propSamp` | numeric proportion of how many of total are sampled. This is ignored for "CENSUS". Defaults to 0.5 |
| `selMeth` | character selection method used. Defaults to "CENSUS". Others like SRSWR or SRSSWOR can be used as well |
| `stratums` | character vector of the stratum names to be created. Defaults to c("U"), meaning not stratified. |
| `mean` | numeric the expected mean of the target variable. The variable is created using `rnorm` and saved under column ending with "y". Defaults to 5. |

## Value

a data.table

## mapColNamesFieldR *A dataset containing the mapping from database column names to R field names*

### Description

A dataset containing the mapping from database column names to R field names

### Usage

```
mapColNamesFieldR
```

### Format

A data frame containing database field names and their equivalent R field name:

**Table.Prefix** The two letter prefix of the relevent RDBES table

**Field.Name** The database field names

**R.Name** The equivalent R field name

**RDataType** The equivalent R data type (e.g. "integer", "character" etc)

**Type** The Data type in the RDBES documentation (e.g. "Decimal", etc)

**EssentialForEst** Is this column considered essential? ...

### Source

https://sboxrdbes.ices.dk

## newRDBESDataObject *Constructor for RDBESDataObject class*

### Description

Constructor for RDBESDataObject class

### Usage

```
newRDBESDataObject(
  DE = NULL,
  SD = NULL,
  VS = NULL,
  FT = NULL,
  FO = NULL,
  TE = NULL,
  LO = NULL,
```

```
  OS = NULL,
  LE = NULL,
  SS = NULL,
  SA = NULL,
  FM = NULL,
  BV = NULL,
  VD = NULL,
  SL = NULL,
  IS = NULL,
  CL = NULL,
  CE = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| DE | Data table of RDBES DE data or null |
| SD | Data table of RDBES DE data or null |
| VS | Data table of RDBES DE data or null |
| FT | Data table of RDBES DE data or null |
| FO | Data table of RDBES DE data or null |
| TE | Data table of RDBES DE data or null |
| LO | Data table of RDBES DE data or null |
| OS | Data table of RDBES DE data or null |
| LE | Data table of RDBES DE data or null |
| SS | Data table of RDBES DE data or null |
| SA | Data table of RDBES DE data or null |
| FM | Data table of RDBES DE data or null |
| BV | Data table of RDBES DE data or null |
| VD | Data table of RDBES DE data or null |
| SL | Data table of RDBES DE data or null |
| IS | Data table of RDBES DE data or null |
| CL | Data table of RDBES DE data or null |
| CE | Data table of RDBES DE data or null |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

## Value

a named list

Pckg_SDAResources_agstrat_H1 *A RDBESDataObject converted from package SDAResources dataset agstrat*

## Description

This data set is derived from the data(agstrat) used in Lohr examples 3.2 and 3.6. Table VS is stratified with VSstratumName set to agstrat$region, and VSnumberSampled and VSnumberTotal set according to agstrat. VSunitName is set to a combination of original agstrat$county, agstrat$state, agstrat$region and agstrat$agstrat row numbers. Table SA contains the variable measured agstrat$acres92 in SAtotalWeightMeasured, SAsampleWeightMeasured and SAconversionFactorMeasLive set to 1. Table DE, SD, FT and FO are for the most dummy tables inserted to meet RDBES model requirements to be aggregated during estimation tests. Values of mandatory fields have dummy values taken from an onboard programme, with exception of selectionMethod that is set to CENSUS. BV, FM, CL, and CE are not provided. SL and VD are subset to the essential rows.

## Usage

```
Pckg_SDAResources_agstrat_H1
```

## Format

A list containing entries required for H1 RDBES data:

**DE** the Design data table. Contains dummy values with exception of selectionMethod that is set to CENSUS

**SD** the Sampling Details data table. Contains dummy values

**VS** the Vessel Selection data table. Contains core information of data(agstrat), VSstratumName set to agstrat$region, and VSnumberSampled and VSnumberTotal set according to agstrat, VSunitName is set to a combination of original agstrat$county, agstrat$state, agstrat$region and agstrat$agstrat row numbers

**FT** the Fishing Trip data table. Contains dummy values

**FO** the Fishing Operation data table. Contains dummy values

**SS** the Species Selection data table. Contains dummy values

**SA** the Sample data table. Contains the variable measured agstrat$acres92 in SAtotalWeightMeasured, SAsampleWeightMeasured and SAconversionFactorMeasLive set to 1

**FM** the Frequency Measure data table. Not provided

**BV** the Biological Variable data table. Not provided

**VD** the Vessel Details data table. Subset to the essential rows

**SL** the Species List data table. Subset to the essential rows

**IS** the Individual Species table

## Source

https://CRAN.R-project.org/package=SDAResources

`Pckg_survey_apiclus2_H1` *A Multi-Stage RDBESDataObject converted from package survey dataset apiclus2*

## Description

This data set is derived from the Academic Performance Index computed for all California schools based on standardized testing of students. The original data sets contain information for all schools with at least 100 students and for various probability samples of the data. The design is 2-stage cluster sampling with clusters of unequal sizes. An SRS of 40 districts is selected (psus) from the 757 districts in the population and then up to 5 schools (min

1. were selected from each district (ssus).

## Usage

```
Pckg_survey_apiclus2_H1
```

## Format

A list containing entries required for H1 RDBES data:

**DE** the Design data table. Contains 1 DE row with DEstratumName == "Pckg_SDAResources_apiclus2_H1"

**SD** the Sampling Details data table. Contains 1 child SD row

**VS** the Vessel Selection data table. Contains 40 child rows (the 40 districts), VSnumberTotal is 757, VSnumberSampled is 40

**FT** the Fishing Trip data table. Contains 126 child rows (the 126 schools finally observed), each associated to its cluster (dname), FTnumberTotal is the number of schools in district, FTnumberSAmpled is 1...5 schools sampled

**FO** the Fishing Operation data table. Just 1:1 links to the final data (in SA)

**SS** the Species Selection data table. Just 1:1 links to the final data (in SA)

**SA** the Sample data table. SAsampleWeightMeasured is enroll (NB! there are 4 NAs)

**FM** the Frequency Measure data table

**BV** the Biological Variable data table

**VD** the Vessel Details data table

**SL** the Species List data table

**IS** the Individual Species table

## Source

https://CRAN.R-project.org/package=survey

`Pckg_survey_apistrat_H1` *A RDBESDataObject converted from package survey dataset apistrat*

## Description

This data set is a stratified version of the previous "apiclus2" data. It is derived from the Academic Performance Index computed for all California schools based on standardized testing of students. The original data sets contain information for all schools with at least 100 students and for various probability samples of the data. The design is 1-stage cluster sampling with clusters of unequal sizes. An SRS of 200 districts is selected (psus) from the 755 districts in the population. All schools within district are selected (ssus).

## Usage

```
Pckg_survey_apistrat_H1
```

## Format

A list containing entries required for H1 RDBES data:

**DE** the Design data table. Contains 1 DE row

**SD** the Sampling Details data table. Contains 1 child SD row

**VS** the Vessel Selection data table. Contains 200 child rows (the 200 schools finally observed), each associated to its cluster (dname), VSnumberTotalClusters is 755, VSnumberTotal is 50-100 schools sampled

**FT** the Fishing Trip data table. Contains 200 child rows (the 200 schools finally observed), each associated to its cluster (dname), FTnumberTotal is the number of schools in the cluster (census)

**FO** the Fishing Operation data table. Just 1:1 links to the final data (in SA)

**SS** the Species Selection data table. Just 1:1 links to the final data (in SA)

**SA** the Sample data table. SAsampleWeightMeasured is enroll

**FM** the Frequency Measure data table

**BV** the Biological Variable data table

**VD** the Vessel Details data table. Contains 311 child rows

**SL** the Species List data table. Contains 1 child row

**IS** the Individual Species table

## Source

https://CRAN.R-project.org/package=survey

## prepareSubSampleLevelLookup *Private function to get sub-sample level and top-level SAid for SA data*

### Description

Private function to get sub-sample level and top-level SAid for SA data

### Usage

```
prepareSubSampleLevelLookup(SAdata)
```

### Arguments

| | |
|---|---|
| SAdata | The SA data to check |

### Value

A data.table with SAid, topLevelSAid and subSampleLevel

## print.RDBESDataObject *Print method for RDBESDataObject*

### Description

This method prints the hierarchy of the DE data.table (if it exists), and the number of rows for each data.table in the RDBESDataObject that is not NULL. It also provides the sampling method and number sampled and number total for tables where it is applicable. If the RDBESDataObject has a mixed hierarchy, a warning message is printed.

This method sorts the RDBESDataObject based on the hierarchy.

This method returns a list containing the hierarchy of the DE data.table, the number of rows for each data.table in the RDBESDataObject that is not NULL, and a logical value indicating if the hierarchy is not NULL.

### Usage

```
## S3 method for class 'RDBESDataObject'
print(x, ...)

## S3 method for class 'RDBESDataObject'
sort(x, decreasing = TRUE, ...)

## S3 method for class 'RDBESDataObject'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class RDBESDataObject. |
| ... | parameters to underling functions (not used currently) |
| decreasing | should hierarchy tables be the first ones |
| object | An object of class RDBESDataObject. |

### Value

None.

The sorted RDBESDataObject by hierarchy.

A list with three elements:

- hierarchy: The hierarchy of the DE data.table in the RDBESDataObject.
- rows: A named list where the names are the names of the data.tables in the RDBESDataObject and the values are the number of rows in each data.table. NULL values are excluded.
- CS: A logical value indicating if the hierarchy is not NULL.

### Examples

```
# Print the package data object
print(H1Example)
# Sort the package data
sort(H8ExampleEE1)
# Get summary of the package data
summary(H1Example)
```

---

## procRDBESEstObjLowHier

*Private function to process the lower hierarchies when creating the RDBESEstObject*

---

### Description

Private function to process the lower hierarchies when creating the RDBESEstObject

### Usage

```
procRDBESEstObjLowHier(rdbesPrepObject, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| rdbesPrepObject | A prepared RDBESRawObj |
| verbose | logical. Output messages to console. |

### Value

allLower - the FM and BV tables combined

`procRDBESEstObjUppHier` *Private function to process the upper hierarchies when creating the RDBESEstObject*

## Description

Private function to process the upper hierarchies when creating the RDBESEstObject

## Usage

```
procRDBESEstObjUppHier(
  myRDBESEstObj = NULL,
  rdbesPrepObject,
  hierarchyToUse,
  i = 1,
  targetTables,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `myRDBESEstObj` | An RDBESEstObj to add data to |
| `rdbesPrepObject` | A prepared RDBESDataObject |
| `hierarchyToUse` | The hierarchy we are using |
| `i` | Integer to keep track of where we are in the list of tables |
| `targetTables` | The RDBES tables we are interested in |
| `verbose` | logical. Output messages to console. |

## Value

A partial RDBESEstObject with the data from the upper hierarchy

`removeBrokenSpeciesListLinks` *Remove rows which are not pointing to a valid SpecliestListDetails (SL) records i.e.those rows which have a value of SpeciesListName that does not exist in the SL table.*

## Description

Remove rows which are not pointing to a valid SpecliestListDetails (SL) records i.e.those rows which have a value of SpeciesListName that does not exist in the SL table.

### Usage

```
removeBrokenSpeciesListLinks(objectToCheck, verbose = FALSE, strict = TRUE)
```

### Arguments

| | |
|---|---|
| objectToCheck | an RDBESDataObject. |
| verbose | (Optional) If set to TRUE more detailed text will be printed out by the function. Default is TRUE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

### Value

an RDBESDataObject with any records with an invalid SpeciesListName rows removed

### Examples

```
## Not run:

myH1RawObject <-
  importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19")
myFields <- c("SLspeclistName")
myValues <- c("WGRDBES-EST TEST 5 - sprat data")
myFilteredObject <- filterRDBESDataObject(myH1RawObject,
  fieldsToFilter = myFields,
  valuesToFilter = myValues
)
myObjectValidSpeciesListLinks <- removeBrokenSpeciesListLinks(
  objectToCheck = myFilteredObject,
  verbose = FALSE
)

## End(Not run)
```

---

## removeBrokenVesselLinks

*Remove rows which are not pointing to a valid VesselDetails (VD) records i.e. those rows which have a value of VDid that does not exist in the VD table.*

---

### Description

Remove rows which are not pointing to a valid VesselDetails (VD) records i.e. those rows which have a value of VDid that does not exist in the VD table.

### Usage

```
removeBrokenVesselLinks(objectToCheck, verbose = FALSE, strict = TRUE)
```

### Arguments

| | |
|---|---|
| objectToCheck | an RDBESDataObject. |
| verbose | (Optional) If set to TRUE more detailed text will be printed out by the function. Default is TRUE. |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

### Value

an RDBESDataObject with any records with an invalid VDid removed

### Examples

```
## Not run:

myH1RawObject <-
  importRDBESDataCSV(rdbesExtractPath = "tests\\testthat\\h1_v_1_19")
myFields <- c("VDlenCat")
myValues <- c("18-<24")
myFilteredObject <- filterRDBESDataObject(myH1RawObject,
  fieldsToFilter = myFields,
  valuesToFilter = myValues
)
myObjectValidVesselLinks <- removeBrokenVesselLinks(
  objectToCheck = myFilteredObject,
  verbose = FALSE
)

## End(Not run)
```

---

## runChecksOnSelectionAndProbs

*Run basic checks on selection methods and probabilities*

---

### Description

This function runs some basic checks on selection methods and and probabilities of the different sampling tables of a hierarchy. It should be run ahead of generateProbs to secure its correct execution and for that reason it is included in the wrapper applyGenerateProbs.

### Usage

```
runChecksOnSelectionAndProbs(x, verbose = FALSE, strict = TRUE)
```

### Arguments

| | |
|---|---|
| x | • RDBES raw object |
| verbose | • If TRUE prints the issue behind the stop |
| strict | (Optional) This function validates its input data - should the validation be strict? The default is TRUE. |

### Value

nothing

### See Also

applyGenerateProbs generateProbs

examples for now see https://github.com/ices-eg/WK_RDBES/tree/master/WKRDB-EST2/chairs/Nuno

---

## setRDBESDataObjectDataTypes

*For a given RDBESDataObject convert the required columns to the correct data types. (This function can cause an error if we have data in the columns that can't be cast to the desired data type.)*

---

### Description

For a given RDBESDataObject convert the required columns to the correct data types. (This function can cause an error if we have data in the columns that can't be cast to the desired data type.)

### Usage

```
setRDBESDataObjectDataTypes(RDBESDataObjectToConvert)
```

### Arguments

| | |
|---|---|
| RDBESDataObjectToConvert | list - the raw item for conversion |

### Value

An RDBESDataObject with the correct date types for the required fields

`shrimps` *One quarter of sample data from swedish shrimp landings of the SWE_OTB_CRU_32-69_0_0 fishery*

## Description

A dataset of rdbesEstimObj type containing simplified haul-level samples (rows) of shrimp landings (targetValue, in kg) observed onboard using H1 of RDBES with UPWOR on vessels. Data is provided for developing/testing purposes only.

## Usage

```
shrimps
```

## Format

A data frame with 10 rows and 95 variables:

- DEsamplingScheme - Sampling Scheme
- DEyear - Year of data collection
- DEstratumName - Fishery code
- DEhierarchyCorrect - Design Variable of RDBES. More details in RDBES documentation
- DEhierarchy - Design Variable of RDBES. More details in RDBES documentation
- DEsampled - Design Variable of RDBES. More details in RDBES documentation
- DEreasonNotSampled - Design Variable of RDBES. More details in RDBES documentation
- SDcountry - Country that collected the data
- SDinstitution - Institution that collected the data
- su1, su2, su3, su4, su5 - sampling units of RDBES. More details in RDBES documentation
- XXXnumberSampled, ... - Design Variables of RDBES. More details in RDBES documentation
- targetValue - estimate of weight landed in each haul (in kg)
- plus XX other columns

## Source

Nuno Prista @ SLU Aqua, 2022

`shrimpsStrat` *One quarter of sample data from swedish shrimp catches of the SWE_OTB_CRU_32-69_0_0 fishery*

## Description

A dataset of rdbesEstimObj type containing simplified haul-level samples (rows) of shrimp catches (targetValue, in kg) observed onboard using H1 of RDBES with UPWOR on vessels. Catches are divided into three strata (91, 92, 93_94) that correspond to sorting sieves used onboard. Data is provided for developing/testing purposes only.

## Usage

```
shrimpsStrat
```

## Format

A data frame with 10 rows and 95 variables:

- DEsamplingScheme - Sampling Scheme
- DEyear - Year of data collection
- DEstratumName - Fishery code
- DEhierarchyCorrect - Design Variable of RDBES. More details in RDBES documentation
- DEhierarchy - Design Variable of RDBES. More details in RDBES documentation
- DEsampled - Design Variable of RDBES. More details in RDBES documentation
- DEreasonNotSampled - Design Variable of RDBES. More details in RDBES documentation
- SDcountry - Country that collected the data
- SDinstitution - Institution that collected the data
- su1, su2, su3, su4, su5 - sampling units of RDBES. More details in RDBES documentation
- XXXnumberSampled, ... - Design Variables of RDBES. More details in RDBES documentation
- su5stratumName - sieve fraction
- targetValue - estimate of weight fraction in each haul (in kg)
- plus XX other columns

## Source

Nuno Prista @ SLU Aqua, 2022

## tablesInRDBESHierarchies

*The tables required for each RDBES hierarchy.*

### Description

A data frame containing the tables required for each RDBES hierachy

### Usage

```
tablesInRDBESHierarchies
```

### Format

A data frame containing the tables required for each RDBES hierachy.

**hierarchy** the hierachy this applies to H1 to H13

**table** the 2-letter table name

**lowerHierarchy** is this a lower hierarchy table?

**optional** is this table optional within the hierarchy?

**samplingUnit** is this table a sampling unit within the hierarchy?

**sortOrder** the table sort order within the hiaerarchy

### Source

https://github.com/davidcurrie2001/MI_RDBES_ExchangeFiles

## updateSAwithTaxonFromSL

*Function which changes the value of SAspeCode in SA.*

### Description

Function checks the rank of aphia id in both of tables SA and SL, and tries to replace a more accurate rank in SA with a broader rank from SL. There are 3 possible situations:

1. If the aphiaid id in the SA table is more accurate than in the SL table and the aphiaids are from the same kingdom, phylum, class, order, family, or genus then the aphiaid in the SA table is changed to the aphia id from the SL table.
2. If the aphia Ids are from different kingdom, phylum, class, order, family, or genus then the function retains the original SA species code.
3. If the SL table has a more accurate rank than in the SA table, the function also retains the original SA species code.

### Usage

```
updateSAwithTaxonFromSL(
  RDBESDataObject,
  validate = TRUE,
  verbose = FALSE,
  strict = TRUE
)
```

### Arguments

| | |
|---|---|
| RDBESDataObject | An RDBESDataObject. |
| validate | Set to TRUE if you want validation to be carried out. The default if TRUE. |
| verbose | (Optional) Set to TRUE if you want informative text on validation printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) This function can validate its input data - should the validation be strict? The default is TRUE. |

### Value

RDBES data object where species in SA were renaming for species occuring in SL for level of species rank. If in SA is Sprat(126425), in SL Clupeidae (125464) function renameSpeciesSA rename Sprat from SA to Clupeidae. Clupeidae(family rank) is higher rank than Sprat(species rank).

### Examples

```
## Not run:
myObject <- createRDBESDataObject(input = "WGRDBES-EST/personal/Kasia/vignettes/vignetteData")
renameSpeciesSA(RDBESDataObject=myObject,validate,verbose,strict)
## End(Not run)
```

---

## validateRDBESDataObject

*Check Whether an* RDBESDataObject *is in a Valid Format*

---

### Description

Perform basic checks on a object.

### Usage

```
validateRDBESDataObject(
  objectToCheck,
  checkDataTypes = FALSE,
  verbose = FALSE,
```

```
  strict = TRUE
)

checkRDBESDataObject(
  objectToCheck,
  checkDataTypes = FALSE,
  verbose = FALSE,
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| objectToCheck | RDBESDataObject i.e. a list of data.tables |
| checkDataTypes | (Optional) Set to TRUE if you want to check that the data types of the required columns are correct, or FALSE if you don't care. Default value is FALSE. |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) Set to TRUE if you want an error if validation fails, set to FALSE if you want only a warning to be issued. The default is TRUE. |

## Details

Checks if 'objectToCheck' parameter is valid. Returns the parameter if it is valid and otherwise stops on error. It checks the RDBESDataObject if:

- Is this an object of class RDBESDataObject
- Tables don't have column names that aren't allowed
- Tables have all the required column names

It does not check if the data is valid. The RDBES upload system performs an extensive set of checks on the uploaded data.

## Value

Returns objectToCheck

## Examples

```
## Not run:
myH1RawObject <-
importRDBESDataCSV(rdbesExtractPath = "tests/testthat/h1_v_1_19")
validateRDBESDataObject(myH1RawObject)
## End(Not run)
```

validateRDBESDataObjectDataTypes *Checks the data types of the columns in an RDBESDataObject against an expected list of data types. Any differences are returned*

## Description

Checks the data types of the columns in an RDBESDataObject against an expected list of data types. Any differences are returned

## Usage

```
validateRDBESDataObjectDataTypes(objectToCheck)
```

## Arguments

| | |
|---|---|
| objectToCheck | An RDBESDataObject to check |

## Value

A data frame containing any data type differences (an empty data frame if there are no differences)

validateRDBESDataObjectDuplicates *check RDBES Raw Object Content Private function to do some basic checks on the content of the RDBESDataObject (e.g. all required field names are present). Function is only used by checkRDBESDataObject and should only be passed a list of non-null objects*

## Description

check RDBES Raw Object Content Private function to do some basic checks on the content of the RDBESDataObject (e.g. all required field names are present). Function is only used by checkRDBESDataObject and should only be passed a list of non-null objects

## Usage

```
validateRDBESDataObjectDuplicates(
  objectToCheck,
  verbose = FALSE,
  strict = TRUE
)
```

### Arguments

| | |
|---|---|
| objectToCheck | • RDBESDataObject i.e. a list of data.tables |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) Set to TRUE if you want to be sure all columns are present in the data, set to FALSE if you only want to check that essential columns are present. The default is TRUE. |

### Value

list with first element as the object and the second the warnings

---

validateRDBESDataObjectFieldNames

*check RDBES Data Object field names Private function to do some checks on the columns of an RDBESDataObject -*

1. *are all required fields present? 2) are there any extra fields present? It is used by validateRDBESDataObject() and should only be passed a list of non-null objects*

---

### Description

check RDBES Data Object field names Private function to do some checks on the columns of an RDBESDataObject -

1. are all required fields present? 2) are there any extra fields present? It is used by validateRDBESDataObject() and should only be passed a list of non-null objects

### Usage

```
validateRDBESDataObjectFieldNames(
  objectToCheck,
  verbose = FALSE,
  strict = TRUE
)
```

### Arguments

| | |
|---|---|
| objectToCheck | • RDBESDataObject i.e. a list of data.tables |
| verbose | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |
| strict | (Optional) Set to TRUE if you want to be sure all columns are present in the data, set to FALSE if you only want to check that essential columns are present. The default is TRUE. |

### Value

list with first element as a boolean indicating validity and the second element contains any warnings

---

`validateRDBESEstObject` *Check whether an object is a valid RDBESEstObject*

---

### Description

Check whether an object is a valid RDBESEstObject

### Usage

```
validateRDBESEstObject(objectToCheck, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `objectToCheck` | The object to check |
| `verbose` | (Optional) Set to TRUE if you want informative text printed out, or FALSE if you don't. The default is FALSE. |

### Value

Whoever revises this function please specify what it returns here

### Examples

```
## Not run:
myH1RawObject <-
importRDBESDataCSV(rdbesExtractPath = "tests/testthat/h1_v_1_19")
myEStObj <- createRDBESEstObject(myH1RawObject,1)
validateRDBESEstObject(myEStObj)
## End(Not run)
```

---

`wormsAphiaRecord` *A dataset containing aphia records for species found in icesSpecWoRMS*

---

### Description

A dataset containing aphia records for species found in icesSpecWoRMS

### Usage

```
wormsAphiaRecord
```

## Format

A data frame

**AphiaID** E.g. 100684

**url** E.g. "https://www.marinespecies.org/aphia.php?p=taxdetails&id=100684"

**scientificname** E.g. "Cerianthidae"

**authority** E.g. "Milne Edwards & Haime, 1851"

**status** E.g. "accepted"

**unacceptreason** E.g. NA

**taxonRankID** E.g. 140

**rank** E.g. "Family" "Genus" "Species" "Species"

**valid_AphiaID** E.g. 100684

**valid_name** E.g. "Cerianthidae"

**valid_authority** E.g. "Milne Edwards & Haime, 1851"

**parentNameUsageID** E.g. 151646

**kingdom** E.g. "Animalia"

**phylum** E.g. "Cnidaria"

**class** E.g. "Anthozoa"

**order** E.g. "Spirularia"

**family** E.g. "Cerianthidae"

**genus** E.g. NA "Cerianthus"

**citation** E.g. "Molodtsova, T. (2023). World List of Ceriantharia. Cerianthidae Milne Edwards & Haime, 1851. Accessed through: "...

**lsid** internal database identifier

**isMarine** E.g. 1

**isBrackish** E.g. 1

**isFreshwater** E.g. 0

**isTerrestrial** E.g. 0

**isExtinct** E.g. NA

**match_type** E.g. "exact"

**modified** E.g. "2018-01-22T17:48:34.063Z"

**DateDownloaded** E.g. "2023-10-18" ...

## Source

https://www.marinespecies.org/

# Index