# Package 'dsBase'

September 5, 2019

**Title** DataSHIELD server site base functions

**Version** 5.0.0

**Author** DataSHIELD Developers <datashield@newcastle.ac.uk>

**Maintainer** DataSHIELD Developers <datashield@newcastle.ac.uk>

**License** GPL-3

**Description** DataSHIELD server site base functions.

**Imports** RANN,
nlme

**AggregateMethods** alphaPhiDS,
asFactorDS1,
asListDS,
checkNegValueDS,
covDS,
dataFrameSubsetDS1,
densityGridDS,
dimDS,
dimDS,
glmDS1,
glmDS2,
glmSLMADS1,
glmSLMADS2,
heatmapPlotDS,
histogramDS1,
histogramDS2,
isNaDS,
isValidDS,
lengthDS,
lexisDS1,
listDisclosureSettingsDS,
matrixDetDS1,
meanDS,
meanDS,
meanSdGpDS,
meanSdGpDS,

messageDS,
namesDS,
numNaDS,
quantileMeanDS,
rangeDS,
recodeValuesDS1,
rmDS,
scatterPlotDS,
scoreVectDS,
setSeedDS,
table1DDS,
table2DDS,
tapplyDS,
testObjExistsDS,
tTestFDS2,
unListDS,
varDS,
class=base::class,
colnames=base::colnames,
dim=base::dim,
exists=base::exists,
is.character=base::is.character,
is.factor=base::is.factor,
is.list=base::is.list,
is.null=base::is.null,
is.numeric=base::is.numeric,
length=base::length,
levels=base::levels,
ls=base::ls,
NROW=base::NROW

**AssignMethods** asCharacterDS,
asDataMatrixDS,
asFactorDS,
asFactorDS2,
asIntegerDS,
asListDS,
asLogicalDS,
asMatrixDS,
asMatrixDS,
asNumericDS,
BooleDS,
cbindDS,
cDS,
changeRefGroupDS,
dataFrameDS,
dataFrameFillDS,
dataFrameSortDS,
dataFrameSubsetDS2,

listDS,
lexisDS2,
lexisDS3,
matrixDetDS2,
matrixDiagDS,
matrixDimnamesDS,
matrixDS,
matrixInvertDS,
matrixMultDS,
matrixTransposeDS,
mergeDS,
rbindDS,
rBinomDS,
recodeLevelsDS,
recodeValuesDS2,
replaceNaDS,
reShapeDS,
rNormDS,
rowColCalcDS,
rPoisDS,
rUnifDS,
seedDS,
seqDS,
subsetByClassDS,
subsetDS,
tapplyDS.assign,
as.character=base::as.character,
as.null=base::as.null,
as.numeric=base::as.numeric,
c=base::c,
cbind=base::cbind,
complete.cases=stats::complete.cases,
exp=base::exp,
list=base::list,
log=base::log,
rep=base::rep,
sum=base::sum,
unlist=base::unlist

**Options** datashield.privacyLevel=5,
default.nfilter.glm=0.33,
default.nfilter.kNN=3,
default.nfilter.string=80,
default.nfilter.subset=3,
default.nfilter.stringShort=20,
default.nfilter.tab=3,
default.nfilter.noise=0.25,
default.nfilter.levels=0.33

**RoxygenNote** 6.1.1

# R topics documented:

---

`alphaPhiDS` *Computes the parameters alpha and phi*

---

## Description

This function is called by the client function 'ds.gee' to calculate the parameters alpha and phi.

## Usage

```
alphaPhiDS(data, formula, family, clusterID, corstr, startBetas)
```

### Arguments

| | |
|---|---|
| data | the input dataframe which contains the variable specified in the formula. |
| formula | a regression formula. |
| family | an object of class Family. |
| clusterID | the name of the column that holds the cluster IDs. |
| corstr | the correlation structure. |
| startBetas | a character, the starting values concatenated by comma because it is not possible to use 'c()' in aggregate functions. |

### Details

the parameters are calculated according to the correlation structure.

### Value

a list

### Author(s)

Gaye, A.; Jones EM.

---

## asCharacterDS *Coerces an R object into class character*

---

### Description

this function is based on the native R function as.character

### Usage

```
asCharacterDS(x.name)
```

### Arguments

| | |
|---|---|
| x.name | the name of the input object to be coerced to class integer. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.asCharacter |

### Details

See help for function as.character in native R

### Value

the object specified by the <newobj> argument (or its default name <x.name>.char) which is written to the serverside. For further details see help on the clientside function ds.asCharacter

### Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

---

`asDataMatrixDS` *asDataMatrixDS a serverside assign function called by ds.asDataMatrix*

---

### Description

Coerces an R object into a matrix maintaining original class for all columns in data.frames.

### Usage

```
asDataMatrixDS(x.name)
```

### Arguments

| | |
|---|---|
| `x.name` | the name of the input object to be coerced to class data.matrix. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.asDataMatrix |

### Details

This assign function is based on the native R function data.matrix If applied to a data.frame, the native R function as.matrix coverts all columns into character class. In contrast, if applied to a data.frame the native R function data.matrix converts the data.frame to a matrix but maintains all data columns in their original class

### Value

the object specified by the <newobj> argument (or its default name <x.name>.mat) which is written to the serverside. For further details see help on the clientside function ds.asDataMatrix

### Author(s)

Paul Burton for DataSHIELD Development Team

## asFactorDS1 *Determines the levels of the input variable in each single study*

### Description

This function is an aggregate DataSHIELD function that returns the levels of the input variable from each single study to the client-side function.

### Usage

```
asFactorDS1(input.var.name = NULL)
```

### Arguments

`input.var.name` the name of the variable that is to be converted to a factor.

### Details

The function encodes the input vector as factor and returns its levels in ascending order if the levels are numerical or in alphabetical order if the levels are of type character.

### Value

the levels of the input variable.

## asFactorDS2 *Converts a numeric vector into a factor*

### Description

This function is an assign DataSHIELD function that converts a numeric vector into a factor type that presented as a vector or as a matrix with dummy variables.

### Usage

```
asFactorDS2(input.var.name = NULL, all.unique.levels.transmit = NULL,
  fixed.dummy.vars = NULL, baseline.level = NULL)
```

### Arguments

| | |
|---|---|
| `input.var.name` | the name of the variable that is to be converted to a factor. |
| `all.unique.levels.transmit` | the levels that the variable will be transmitted to. |
| `fixed.dummy.vars` | a boolean that determines whether the new object will be represented as a vector or as a matrix of dummy variables indicating the factor level of each data point. If this argyment is set to FALSE (default) then the input variable is converted to a factor and assigned as a vector. If is set to TRUE then the input variable is converted to a factor but assigned as a matrix of dummy variables. |
| `baseline.level` | a number indicating the baseline level to be used in the creation of the matrix of dummy variables. |

### Details

The functions converts the input variable into a factor which is presented as a vector if the `fixed.dummy.vars` is set to FALSE or as a matrix with dummy variables if the `fixed.dummy.vars` is set to TRUE (see the help file of ds.asFactor.b for more details).

### Value

an object of class factor

---

## asIntegerDS *Coerces an R object into class integer*

---

### Description

this function is based on the native R function as.integer

### Usage

```
asIntegerDS(x.name)
```

### Arguments

| | |
|---|---|
| `x.name` | the name of the input object to be coerced to class integer. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.asInteger |

### Details

See help for function as.integer in native R

### Value

the object specified by the <newobj> argument (or its default name <x.name>.int) which is written to the serverside. For further details see help on the clientside function ds.asInteger

### Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

---

`asListDS` *asListDS a serverside aggregate function called by ds.asList*

---

### Description

Coerces an R object into a list

### Usage

```
asListDS(x.name, newobj)
```

### Arguments

| | |
|---|---|
| `x.name` | the name of the input object to be coerced to class data.matrix. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.asList |
| `newobj` | is the object hard assigned '«-' to be the output of the function written to the serverside |

### Details

Unlike most other class coercing functions this is an aggregate function rather than an assign function. This is because the datashield.assign function in opal deals specially with a created object (newobj) if it is of class list. Reconfiguring the function as an aggregate function works around this problem. This aggregate function is based on the native R function as.list and so additional information can be found in the help for as.list

### Value

the object specified by the <newobj> argument (or its default name <x.name>.mat) which is written to the serverside. In addition, two validity messages are returned. The first confirms an output object has been created, the second states its class. The way that as.list coerces objects to list depends on the class of the object, but in general the class of the output object should usually be 'list'

### Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

## asLogicalDS *Coerces an R object into class numeric*

### Description

this function is based on the native R function as.numeric

### Usage

```
asLogicalDS(x.name)
```

### Arguments

| | |
|---|---|
| x.name | the name of the input object to be coerced to class numeric. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.aslogical |

### Details

See help for function as.logical in native R

### Value

the object specified by the <newobj> argument (or its default name <x.name>.logic) which is written to the serverside. For further details see help on the clientside function ds.asLogical

### Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

## asMatrixDS *Coerces an R object into a matrix*

### Description

this function is based on the native R function as.matrix

### Usage

```
asMatrixDS(x.name)
```

### Arguments

| | |
|---|---|
| x.name | the name of the input object to be coerced to class matrix. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.asMatrix |

### Details

See help for function as.matrix in native R

### Value

the object specified by the <newobj> argument (or its default name <x.name>.mat) which is written to the serverside. For further details see help on the clientside function ds.asMatrix

### Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

---

## asNumericDS *Coerces an R object into class numeric*

---

### Description

this function is based on the native R function as.numeric

### Usage

```
asNumericDS(x.name)
```

### Arguments

| | |
|---|---|
| x.name | the name of the input object to be coerced to class numeric. Must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.asNumeric |

### Details

See help for function as.numeric in native R

### Value

the object specified by the <newobj> argument (or its default name <x.name>.num) which is written to the serverside. For further details see help on the clientside function ds.asNumeric

### Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

BooleDS *BooleDS*

## Description

Converts the individual elements of a vector or other object into Boolean indicators.

## Usage

```
BooleDS(V1.name = NULL, V2.name = NULL, Boolean.operator.n = NULL,
  na.assign.text, numeric.output = TRUE)
```

## Arguments

| | |
|---|---|
| V1.name | A character string specifying the name of the vector to which the Boolean operator is to be applied |
| V2.name | A character string specifying the name of the vector or scalar to which <V1> is to be compared. |
| Boolean.operator.n | An integer value (1 to 6) providing a numeric coding for the character string specifying one of six possible Boolean operators: '==', '!=', '>', '>=','<', '<=' that could legally be passed from client to server via DataSHIELD parser |
| na.assign.text | A character string taking values 'NA', '1' or '0'. If 'NA' then any NA values in the input vector remain as NAs in the output vector. If '1' or '0' NA values in the input vector are all converted to 1 or 0 respectively.#' @return the levels of the input variable. |
| numeric.output | a TRUE/FALSE indicator defaulting to TRUE determining whether the final output variable should be of class numeric (1/0) or class logical (TRUE/FALSE). |

## Details

The function converts the input vector into Boolean indicators.

## Author(s)

DataSHIELD Development Team

cbindDS *cbindDS called by ds.cbind c*

## Description

serverside assign function that takes a sequence of vector, matrix or data-frame arguments and combines them by column to produce a matrix.

## Usage

```
cbindDS(x.names.transmit = NULL, colnames.transmit = NULL)
```

## Arguments

`x.names.transmit`
: This is a vector of character strings representing the names of the elemental components to be combined converted into a transmittable format. This argument is fully specified by the <x> argument of ds.cbind

`colnames.transmit`
: This is NULL or a vector of character strings representing forced column names for the output object converted into a transmittable format. This argument is fully specified by the <force.colnames> argument of ds.cbind.

## Details

A sequence of vector, matrix or data-frame arguments is combined column by column to produce a matrix which is written to the serverside. For more details see help for ds.cbind and the native R function cbind.

## Value

the object specified by the <newobj> argument of ds.cbind(or default name <cbind.out>) which is written to the serverside. Just like the cbind function in native R, the output object is of class matrix unless one or more of the input objects is a data.frame in which case the class of the output object is data.frame. As well as writing the output object as <newobj> on the serverside, two validity messages are returned indicating whether <newobj> has been created in each data source and if so whether it is in a valid form. If its form is not valid in at least one study - e.g. because a disclosure trap was tripped and creation of the full output object was blocked - ds.cbind() also returns any studysideMessages that can explain the error in creating the full output object. As well as appearing on the screen at run time,if you wish to see the relevant studysideMessages at a later date you can use the ds.message function. If you type ds.message("<newobj>") it will print out the relevant studysideMessage from any datasource in which there was an error in creating <newobj> and a studysideMessage was saved. If there was no error and <newobj> was created without problems no studysideMessage will have been saved and ds.message("<newobj>") will return the message: "ALL OK: there are no studysideMessage(s) on this datasource".

### Author(s)

Paul Burton for DataSHIELD Development Team

---

## cDS *Concatenates objects into a vector or list*

---

### Description

This function is similar to the R base function 'c'.

### Usage

```
cDS(objs)
```

### Arguments

| | |
|---|---|
| objs | a list which contains the the objects to concatenate. |

### Details

Unlike the R base function 'c' on vector or list of certain length are allowed as output

### Value

a vector or list

### Author(s)

Gaye, A.

---

## changeRefGroupDS *Changes a reference level of a factor*

---

### Description

This function is similar to R function relevel,

### Usage

```
changeRefGroupDS(xvect, ref = NULL, reorderByRef = NULL)
```

### Arguments

| | |
|---|---|
| xvect | a factor vector |
| ref | a character, the reference level |
| reorderByRef | a boolean that tells whether or not the new vector should be ordered by the reference group. |

### Details

In addition to what the R function does, this function allows for the user to re-order the vector, putting the reference group first. If the user chooses the re-order a warning is issued as this can introduce a mismatch of values if the vector is put back into a table that is not reordered in the same way. Such mismatch can render the results of operations on that table invalid.

### Value

a factor of the same length as xvect

### Author(s)

Isaeva, J., Gaye, A.

---

## checkNegValueDS *Checks if a numeric variable has negative values*

---

### Description

this function is only called by the client function `ds.glm`.

### Usage

```
checkNegValueDS(weights)
```

### Arguments

| | |
|---|---|
| `weights` | a numeric vector |

### Details

if a user sets the parameter 'weights' on the client site function `ds.glm` this server side function is called to verify that the 'weights' vector does not have negative values because no negative are allowed in weights.

### Value

a boolean; TRUE if the vector has one or more negative values and FALSE otherwise

### Author(s)

Gaye, A.

corDS *Computes correlation between two or more vectors*

## Description

this function is similar to R function 'cor'

## Usage

```
corDS(x = NULL, y = NULL, use = NULL)
```

## Arguments

| | |
|---|---|
| x | a character, the name of a vector, dataframe or matrix |
| y | (optional) a character, the name of a vector, dataframe or matrix |
| use | a character string giving a method for computing covariances in the presence of missing values. This must be one of the strings: "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". |

## Value

correlation

## Author(s)

Gaye, A.

covDS *Computes the sum of each variable and the sum of products for each pair of variables*

## Description

This function computes the sum of each vector of variable and the sum of the products of each two variables (i.e. the scalar product of each two vectors).

## Usage

```
covDS(x = NULL, y = NULL, use = NULL)
```

### Arguments

| | |
|---|---|
| x | a character, the name of a vector, matrix or dataframe of variable(s) for which the covariance(s) and the correlation(s) is (are) going to calculated for. |
| y | NULL (default) or the name of a vector, matrix or dataframe with compatible dimensions to x. |
| use | a character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "casewise.complete" or "pairwise.complete". If use is set to 'casewise.complete' then any rows with missing values are omitted from the vector, matrix or dataframe before the calculations of the sums. If use is set to 'pairwise.complete' (which is the default case set on the client-side), then the sums of products are computed for each two variables using only the complete pairs of observations on the two variables. |

### Details

computes the sum of each vector of variable and the sum of the products of each two variables

### Value

a list tha includes a matrix with elements the sum of products between each two variables, a matrix with elements the sum of the values of each variable, a matrix with elements the number of complete cases in each pair of variables, a list with the number of missing values in each variable separately (columnwise) and the number of missing values casewise or pairwise depending on the arqument use, and an error message which indicates whether or not the input variables pass the disclosure control (i.e. none of them is dichotomous with a level having less counts than the pre-specified threshold). If any of the input variables does not pass the disclosure control then all the output values are replaced with NAs

### Author(s)

Gaye A., Avraam D., Burton P.

---

## dataFrameDS *dataFrameDS called by ds.dataFrame*

---

### Description

The serverside function that creates a data frame from its elemental components. That is: pre-existing data frames; single variables; and/or matrices

### Usage

```
dataFrameDS(vectors = NULL, r.names = NULL, ch.rows = FALSE,
  ch.names = TRUE, clnames = NULL, strAsFactors = TRUE,
  completeCases = FALSE)
```

## Arguments

| | |
|---|---|
| vectors | a list which contains the elemental components to combine. These correspond to the vector of character strings specified in argument x of the clientside function ds.dataFrame() |
| r.names | NULL or a character vector specifying the names of the rows. Default NULL. |
| ch.rows | logical, if TRUE then the rows are checked for consistency of length and names. Default FALSE. |
| ch.names | logical, if TRUE then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names and are not duplicated. Default TRUE. In fact, the clientside function ensures no duplicated names can be presented to dataFrameDS but this argument is kept to check for other forms of syntactic validity. |
| clnames | a list of characters, the column names of the output data frame. These are generated by the clientside function from the names of vectors, and the column names of data.frames and matrices being combined in producing the output data.frame |
| strAsFactors | logical, if TRUE determines whether character vectors should automatically be converted to factors? Default TRUE. |
| completeCases | logical. If TRUE indicates that only complete cases should be included: any rows with missing values in any component will be excluded. Default FALSE. |

## Details

A data frame is a list of variables all with the same number of rows with unique row names, which is of class 'data.frame'. ds.dataFrame will create a data frame by combining a series of elemental components which may be pre-existing data.frames, matrices or variables. A critical requirement is that the length of all component variables, and the number of rows of the component data.frames or matrices must all be the same. The output data.frame will then have this same number of rows. The serverside function dataFrameDS() calls the native R function data.frame() and several of its arguments are precisely the same as for data.frame(). In consequence, additional information can be sought from the help() for data.frame().

## Value

a dataframe composed of the specified elemental components will be created on the serverside and named according to the <newobj> argument of the clientside function ds.dataFrame()

## Author(s)

DataSHIELD Development Team

## dataFrameFillDS *dataFrameFillDS*

### Description

An assign function called by the clientside ds.dataFrameFill function.

### Usage

```
dataFrameFillDS(df.name, allNames.transmit)
```

### Arguments

| | |
|---|---|
| df.name | a character string representing the name of the input data frame that will be filled with extra columns with missing values if a number of variables is missing from it compared to the data frames of the other studies used in the analysis. |
| allNames.transmit | unique names of all the variables that are included in the input data frames from all the used datasources. |

### Details

This function checks if each study has all the variables compared to the other studies in the analysis. If a study does not have some of the variables, the function generates those variables as vectors of missing values and combines them as columns to the input data frame. Then, the "complete" in terms of the columns dataframe is saved in each server with a name specified by the argument newobj on the clientside.

### Value

Nothing is returned to the client. The generated object is written to the serverside.

### Author(s)

Demetris Avraam for DataSHIELD Development Team

## dataFrameSortDS *dataFrameSortDS called by ds.dataFrameSort*

### Description

The serverside function that sorts a data frame using a specified sort key.

### Usage

```
dataFrameSortDS(df.text = NULL, sort.key.text = NULL,
  sort.descending = FALSE, sort.alphabetic = FALSE,
  sort.numeric = FALSE)
```

### Arguments

| | |
|---|---|
| df.text | a character string providing the name for the data.frame to be sorted. This corresponds to the argument <df.name> in ds.dataFrameSort |
| sort.key.text | a character string providing the name for the sort key. This corresponds to the argument <sort.key.name> in ds.dataFrameSort |
| sort.descending | logical, if TRUE the data.frame will be sorted by the sort key in descending order. Default = FALSE (sort order ascending) |
| sort.alphabetic | logical, if TRUE the sort key is treated as if alphabetic Default=FALSE. |
| sort.numeric | logical, if TRUE the sort key is treated as if numeric Default=FALSE. The arguments <sort.alphabetic> and <sort.numeric> are both derived directly from the corresponding arguments specified in ds.dataFrameSort If both sort.alphabetic and sort.numeric are FALSE, the sort.key will interpreted naturally: as numeric if it is numeric, otherwise as alphabetic ie as if it is a vector of character strings. |

### Details

A data frame is a list of variables all with the same number of rows, which is of class 'data.frame'. For details of numeric and alphabetic sorting and how ds.dataFrameSort/dataFrameSortDS jointly operate, please see help for ds.dataFrameSort.

### Value

the appropriately re-sorted data.frame will be written to the serverside R environmnet as a data.frame named according to the <newobj> argument in ds.dataFrameSortDS (or with default name <df.name>.sorted where <df.name> is the first argument of ds.dataFrameSortDS)

### Author(s)

DataSHIELD Development Team

---

## dataFrameSubsetDS1 *dataFrameSubsetDS1 an aggregate function called by ds.dataFrameSubset*

---

### Description

First serverside function for subsetting a data frame by row or by column.

## Usage

```
dataFrameSubsetDS1(df.name = NULL, V1.name = NULL, V2.name = NULL,
  Boolean.operator.n = NULL, keep.cols = NULL, rm.cols = NULL,
  keep.NAs = NULL)
```

## Arguments

| | |
|---|---|
| df.name | a character string providing the name for the data.frame to be sorted. <df.name> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |
| V1.name | A character string specifying the name of a subsetting vector to which a Boolean operator will be applied to define the subset to be created. <V1.name> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |
| V2.name | A character string specifying the name of the vector or scalar to which the values in the vector specified by the argument <V1.name> is to be compared. <V2.name> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |
| Boolean.operator.n | A character string specifying one of six possible Boolean operators: '==', '!=', '>', '>=', '<', '<=' <Boolean.operator.n> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |
| keep.cols | a numeric vector specifying the numbers of the columns to be kept in the final subset when subsetting by column. For example: keep.cols=c(2:5,7,12) will keep columns 2,3,4,5,7 and 12. <keep.cols> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |
| rm.cols | a numeric vector specifying the numbers of the columns to be removed before creating the final subset when subsetting by column. For example: rm.cols=c(2:5,7,12) will remove columns 2,3,4,5,7 and 12. <rm.cols> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |
| keep.NAs | logical, if TRUE any NAs in the vector holding the final Boolean vector indicating whether a given row should be included in the subset will be converted into 1s and so they will be included in the subset. Such NAs could be caused by NAs in either <V1.name> or <V2.name>. If FALSE or NULL NAs in the final Boolean vector will be converted to 0s and the corresponding row will therefore be excluded from the subset. <keep.NAs> argument generated and passed directly to dataFrameSubsetDS1 by ds.dataFrameSubset |

## Details

A data frame is a list of variables all with the same number of rows, which is of class 'data.frame'. For all details see the help header for ds.dataFrameSubset

## Value

This first serverside function called by ds.dataFrameSubset provides first level traps for a comprehensive series of disclosure risks which can be returned directly to the clientside because dataFrameSubsetDS1 is an aggregate function. The second serverside function called by ds.dataFrameSubset

(dataFrameSubsetDS2) carries out most of the same disclosure tests, but it is an assign function because it writes the subsetted data.frame to the serverside. In consequence, it records error messages as studysideMessages which can only be retrieved using ds.message

## Author(s)

DataSHIELD Development Team

---

`dataFrameSubsetDS2` *dataFrameSubsetDS2 an assign function called by ds.dataFrameSubset*

---

## Description

Second serverside function for subsetting a data frame by row or by column.

## Usage

```
dataFrameSubsetDS2(df.name = NULL, V1.name = NULL, V2.name = NULL,
  Boolean.operator.n = NULL, keep.cols = NULL, rm.cols = NULL,
  keep.NAs = NULL)
```

## Arguments

| | |
|---|---|
| `df.name` | a character string providing the name for the data.frame to be sorted. <df.name> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset |
| `V1.name` | A character string specifying the name of a subsetting vector to which a Boolean operator will be applied to define the subset to be created. <V1.name> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset |
| `V2.name` | A character string specifying the name of the vector or scalar to which the values in the vector specified by the argument <V1.name> is to be compared. <V2.name> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset |
| `Boolean.operator.n` | A character string specifying one of six possible Boolean operators: '==', '!=', '>', '>=', '<', '<=' <Boolean.operator.n> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset |
| `keep.cols` | a numeric vector specifying the numbers of the columns to be kept in the final subset when subsetting by column. For example: keep.cols=c(2:5,7,12) will keep columns 2,3,4,5,7 and 12. <keep.cols> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset |
| `rm.cols` | a numeric vector specifying the numbers of the columns to be removed before creating the final subset when subsetting by column. For example: rm.cols=c(2:5,7,12) will remove columns 2,3,4,5,7 and 12. <rm.cols> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset |

keep.NAs — logical, if TRUE any NAs in the vector holding the final Boolean vector indicating whether a given row should be included in the subset will be converted into 1s and so they will be included in the subset. Such NAs could be caused by NAs in either <V1.name> or <V2.name>. If FALSE or NULL NAs in the final Boolean vector will be converted to 0s and the corresponding row will therefore be excluded from the subset. <keep.NAs> argument generated and passed directly to dataFrameSubsetDS2 by ds.dataFrameSubset

## Details

A data frame is a list of variables all with the same number of rows, which is of class 'data.frame'. For all details see the help header for ds.dataFrameSubset

## Value

the object specified by the <newobj> argument (or default name '<df.name>_subset') initially specified in calling ds.dataFrameSubset. The output object (the required subsetted data.frame called <newobj> is written to the serverside. In addition, two validity messages are returned via ds.dataFrameSubset indicating whether <newobj> has been created in each data source and if so whether it is in a valid form. If its form is not valid in at least one study - e.g. because a disclosure trap was tripped and creation of the full output object was blocked - dataFrameSubsetDS2 (via ds.dataFrame()) also returns any studysideMessages that can explain the error in creating the full output object. As well as appearing on the screen at run time,if you wish to see the relevant studysideMessages at a later date you can use the ds.message function. If you type ds.message("newobj") it will print out the relevant studysideMessage from any datasource in which there was an error in creating <newobj> and a studysideMessage was saved. If there was no error and <newobj> was created without problems no studysideMessage will have been saved and ds.message("newobj") will return the message: "ALL OK: there are no studysideMessage(s) on this datasource".

## Author(s)

DataSHIELD Development Team

---

# densityGridDS *Generates a density grid with or without a priori defined limits*

---

## Description

Generates a density grid that can then be used for heatmap or countour plots.

## Usage

```
densityGridDS(xvect, yvect, limits = FALSE, x.min = NULL,
  x.max = NULL, y.min = NULL, y.max = NULL, numints = 20)
```

### Arguments

| | |
|---|---|
| xvect | a numerical vector |
| yvect | a numerical vector |
| limits | a logical expression for whether or not limits of the density grid are defined by a user. If limits is set to "FALSE", min and max of xvect and yvect are used as a range. If limits is set to "TRUE", limits defined by x.min, x.max, y.min and y.max are used. |
| x.min | a minimum value for the x axis of the grid density object, if needed |
| x.max | a maximum value for the x axis of the grid density object, if needed |
| y.min | a minimum value for the y axis of the grid density object, if needed |
| y.max | a maximum value for the y axis of the grid density object, if needed |
| numints | a number of intervals for the grid density object, by default is 20 |

### Details

Invalid cells (cells with count < to the set filter value for the minimum allowed counts in table cells) are turn to 0.

### Value

a grid density matrix

### Author(s)

Julia Isaeva, Amadou Gaye, Demetris Avraam for DataSHIELD Development Team

---

## dimDS *Returns the dimension of a data frame or matrix*

---

### Description

This function is similar to R function dim.

### Usage

```
dimDS(x)
```

### Arguments

| | |
|---|---|
| x | a string character, the name of a dataframe or matrix |

### Details

The function returns the dimension of the input dataframe or matrix

## Value

the dimension of the input object

## Author(s)

Demetris Avraam, for DataSHIELD Development Team

---

`glmDS1` *glmDS1 called by ds.glm*

---

## Description

This is the first serverside aggregate function called by ds.glm

## Usage

```
glmDS1(formula, family, weights, offset, data)
```

## Arguments

| | |
|---|---|
| `formula` | a glm() formula consistent with R syntax eg U~x+y+Z to regress variables U on x,y and Z |
| `family` | a glm() family consistent with R syntax eg "gaussian", "poisson", "binomial" |
| `weights` | an optional variable providing regression weights |
| `offset` | the offset |
| `data` | an optional character string specifying a data.frame object holding the data to be analysed under the specified model |

## Details

It is an aggregation function that sets up the model structure and creates the starting beta.vector that feeds, via ds.glm, into glmDS2 to enable iterative fitting of the generalized linear model that has been been specified. For more details please see the extensive header for ds.glm.

## Author(s)

Burton PR for DataSHIELD Development Team

glmDS2 *glmDS2 called by ds.glm*

## Description

This is the second serverside aggregate function called by ds.glm.

## Usage

```
glmDS2(formula, family, beta.vect, offset, weights, dataName)
```

## Arguments

| | |
|---|---|
| formula | a glm() formula consistent with R syntax eg U~x+y+Z to regress variables U on x,y and Z |
| family | a glm() family consistent with R syntax eg "gaussian", "poisson", "binomial" |
| beta.vect | a numeric vector created by the clientside function specifying the vector of regression coefficients at the current iteration |
| offset | an optional variable providing a regression offset |
| weights | an optional variable providing regression weights |
| dataName | an optional character string specifying a data.frame object holding the data to be analysed under the specified model same |

## Details

It is an aggregation function that uses the model structure and starting beta.vector constructed by glmDS1 to iteratively fit the generalized linear model that has been been specified. The function glmDS2 also carries out a series of disclosure checks and if the arguments or data fail any of those tests, model construction is blocked and an appropriate serverside error message is created and returned to ds.glm on the clientside. For more details please see the extensive header for ds.glm.

## Author(s)

Burton PR for DataSHIELD Development Team

`glmSLMADS1` *glmSLMADS1 called by ds.glmSLMA*

## Description

This is the first serverside aggregate function called by ds.glmSLMA

## Usage

```
glmSLMADS1(formula, family, weights, offset, data)
```

## Arguments

| | |
|---|---|
| `formula` | a glm() formula consistent with R syntax eg U~x+y+Z to regress variables U on x,y and Z. Fully specified by <formula> argument in ds.glmSLMA |
| `family` | a glm() family consistent with R syntax eg "gaussian", "poisson", "binomial". Fully specified by <family> argument in ds.glmSLMA |
| `weights` | an optional variable name (as a character sting) identifying a vector of prior regression weights. Fully specified by <weights> argument in ds.glmSLMA |
| `offset` | an optional variable name (as a character string) identifying an offset vector. Fully specified by <offset> argument in ds.glmSLMA |
| `data` | an optional character string specifying the name of a data.frame object holding the data to be analysed under the specified model. Fully specified by <dataName> argument in ds.glmSLMA |

## Details

It is an aggregate function that sets up the generalized linear model structure and feeds this structural information via ds.glmSLMA into the call to glmSLMADS2 that enacts fitting of the specified generalized linear model (to completion) in each of the studies to be included in the study-level meta-analysis. For more details please see the extensive header for ds.glmSLMA in DataSHIELD and help on the glm function in native R.

## Value

All quantitative, Boolean, and character objects required by glmSLMADS2 to fit the glm in each study. Also, returns warning flags and associated information enabling DataSHIELD to halt analysis in any given study if a disclosure trap is triggered and to inform the user what trap has been triggered.

## Author(s)

Paul Burton for DataSHIELD Development Team

`glmSLMADS2` *glmSLMADS2 called by ds.glmSLMA*

## Description

This is the second serverside aggregate function called by ds.glmSLMA

## Usage

```
glmSLMADS2(formula, family, offset, weights, dataName)
```

## Arguments

| | |
|---|---|
| `formula` | a glm() formula consistent with R syntax eg U~x+y+Z to regress variables U on x,y and Z. Fully specified by <formula> argument in ds.glmSLMA |
| `family` | a glm() family consistent with R syntax eg "gaussian", "poisson", "binomial". Fully specified by <family> argument in ds.glmSLMA |
| `offset` | an optional variable name (as a character string) identifying an offset vector. Fully specified by <offset> argument in ds.glmSLMA |
| `weights` | an optional variable name (as a character sting) identifying a vector of prior regression weights. Fully specified by <weights> argument in ds.glmSLMA |
| `dataName` | an optional character string specifying the name of a data.frame object holding the data to be analysed under the specified model. Fully specified by <dataName> argument in ds.glmSLMA |

## Details

ds.glmSLMA specifies the structure of a generalized linear model (glm) to be fitted separately on each study. The model is first constructed and subject to preliminary disclosure checking by glmSLMADS1. This aggregate function then returns this output to ds.glmSLMA which processes the information and uses it in a call to glmSLMADS2. This call specifies and fits the required glm in each data source. Unlike glmDS2 (called by the more commonly used generalized linear modelling client-side function ds.glm) the requested model is then fitted to completion on the data in each study rather than iteration by iteration on all studies combined. At the end of this SLMA fitting process glmSLMADS2 returns study-specific parameter estimates and standard errors to the client. These can then be pooled using random effects (or fixed effects) meta-analysis - eg using the metafor package. This mode of model fitting may reasonably be called study level meta-analysis (SLMA) although the analysis is based on estimates and standard errors derived from direct analysis of the individual level data in each study rather than from published study summaries (as is often the case with SLMA of clinical trials etc). Furthermore, unlike common approaches to study-level meta-analysis adopted by large multi-study research consortia (eg in the combined analysis of identical genomic markers across multiple studies), the parallel analyses (in every study) under ds.glmSLMA are controlled entirely from one client. This avoids the time-consuming need to ask each study to run its own analyses and the consequent necessity to request additional work from individual studies if the modelling is to be extended to include analyses not subsumed in the original analytic plan. Additional analyses of this nature may, for example, include analyses based on

interactions between covariates identified as having significant main effects in the original analysis. From a mathematical perspective, the SLMA approach (using ds.glmSLMA) differs fundamentally from the usual approach using ds.glm in that the latter is mathematically equivalent to placing all individual-level data from all sources in one central warehouse and analysing those data as one combined dataset using the conventional glm() function in R. However, although this may sound to be preferable under all circumstances, the SLMA approach actually offers key inferential advantages when there is marked heterogeneity between sources that cannot simply be corrected with fixed effects each reflecting a study or centre-effect. In particular, fixed effects cannot simply be used in this way when there there is heterogeneity in the effect that is of scientific interest. For more details please see the extensive header for ds.glmSLMA in DataSHIELD and help on the glm function in native R.

## Value

All quantitative, Boolean, and character objects required to enable the SLMA pooling of the separate glm models fitted to each study - in particular including the study-specific regression coefficients and their corresponding standard errors. Also, returns warning flags and associated information enabling DataSHIELD to halt analysis and destroy model output from any given study if a disclosure trap is triggered and to inform the user what trap has been triggered.

## Author(s)

Burton PR

---

# heatmapPlotDS *Calculates the coordinates of the centroid of each n nearest neighbours*

---

## Description

This function calculates the coordinates of the centroids for each n nearest neighbours.

## Usage

```
heatmapPlotDS(x, y, k, noise, method.indicator)
```

## Arguments

| | |
|---|---|
| `x` | the name of a numeric vector, the x-variable. |
| `y` | the name of a numeric vector, the y-variable. |
| `k` | the number of the nearest neghbours for which their centroid is calculated if the `method.indicator` is equal to 1 (i.e. deterministic method). |
| `noise` | the percentage of the initial variance that is used as the variance of the embedded noise if the `method.indicator` is equal to 2 (i.e. probabilistic method). |
| `method.indicator` | a number equal to either 1 or 2. If the value is equal to 1 then the 'deterministic' method is used. If the value is set to 2 the 'probabilistic' method is used. |

### Details

The function finds the n-1 nearest neighbours of each data point in a 2-dimensional space. The nearest neighbours are the data points with the minimum Euclidean distances from the point of interest. Each point of interest and its n-1 nearest neighbours are then used for the calculation of the coordinates of the centroid of those n points. Centroid here is referred to the centre of mass, i.e. the x-coordinate of the centroid is the average value of the x-coordinates of the n nearest neighbours and the y-coordinate of the centroid is the average of the y-coordinates of the n nearest neighbours. The coordinates of the centroids return to the client side function and can be used for the plot of non-disclosive graphs (e.g. scatter plots, heatmap plots, contour plots, etc).

### Value

a list with the x and y coordinates of the centroids if the deterministic method is used or the x and y coordinated of the noisy data if the probabilistic method is used.

### Author(s)

Demetris Avraam for DataSHIELD Development Team

---

## histogramDS1 *returns the minimum and the maximum of the input numeric vector*

---

### Description

this function returns the minimum and maximum of the input numeric vector which depends on the argument `method.indicator`. If the method.indicator is set to 1 (i.e. the 'smallCellsRule' is used) the computed minimum and maximum values are multiplied by a very small random number. If the method.indicator is set to 2 (i.e. the 'deteministic' method is used) the function returns the minimum and maximum values of the vector with the scaled centroids. If the method.indicator is set to 3 (i.e. the 'probabilistic' method is used) the function returns the minimum and maximum values of the generated 'noisy' vector.

### Usage

```
histogramDS1(xvect, method.indicator, k, noise)
```

### Arguments

| | |
|---|---|
| `xvect` | the numeric vector for which the histogram is desired. |
| `method.indicator` | a number equal to either 1, 2 or 3 indicating the method of disclosure control that is used for the generation of the histogram. If the value is equal to 1 then the 'smallCellsRule' is used. If the value is equal to 2 then the 'deterministic' method is used. If the value is set to 3 then the 'probabilistic' method is used. |
| `k` | the number of the nearest neghbours for which their centroid is calculated if the `method.indicator` is equal to 2 (i.e. deterministic method). |
| `noise` | the percentage of the initial variance that is used as the variance of the embedded noise if the `method.indicator` is equal to 3 (i.e. probabilistic method). |

## Value

a numeric vector which contains the minimum and the maximum values of the vector

## Author(s)

Amadou Gaye, Demetris Avraam for DataSHIELD Development Team

---

`histogramDS2` *Computes a histogram of the input variable without plotting.*

---

## Description

This function produces the information required to plot a histogram. This is done without allowing for bins (cells) with number of counts less than the pre-specified disclosure control set for the minimum cell size of a table. If a bin has less counts than this threshold then their counts and its density are replaced by a 0 value.

## Usage

```
histogramDS2(xvect, num.breaks, min, max, method.indicator, k, noise)
```

## Arguments

| | |
|---|---|
| `xvect` | the numeric vector for which the histogram is desired. |
| `num.breaks` | the number of breaks that the range of the variable is divided. |
| `min` | a numeric, the lower limit of the distribution. |
| `max` | a numeric, the upper limit of the distribution. |
| `method.indicator` | a number equal to either 1, 2 or 3 indicating the method of disclosure control that is used for the generation of the histogram. If the value is equal to 1 then the 'smallCellsRule' is used. If the value is equal to 2 then the 'deterministic' method is used. If the value is set to 3 then the 'probabilistic' method is used. |
| `k` | the number of the nearest neghbours for which their centroid is calculated if the `method.indicator` is equal to 2 (i.e. deterministic method). |
| `noise` | the percentage of the initial variance that is used as the variance of the embedded noise if the `method.indicator` is equal to 3 (i.e. probabilistic method). |

## Details

Please find more details in the documentation of the clientside ds.histogram function.

## Value

a list with an object of class `histogram` and the number of invalid cells

## Author(s)

Amadou Gaye, Demetris Avraam for DataSHIELD Development Team

## isNaDS *Checks if a vector is empty*

### Description

this function is similar to R function `is.na` but instead of a vector of booleans it returns just one boolean to tell if all the element are missing values.

### Usage

```
isNaDS(xvect)
```

### Arguments

| | |
|---|---|
| `xvect` | a numerical or character vector |

### Value

the integer '1' if the vector contains on NAs and '0' otherwise

### Author(s)

Gaye, A.

## isValidDS *Checks if an input is valid*

### Description

Tells if an object on the server side is valid.

### Usage

```
isValidDS(obj)
```

### Arguments

| | |
|---|---|
| `obj,` | a vector (numeric, integer, factor, character), data.frame or matrix |

### Details

This function checks if an object is valid.

### Value

a boolean, TRUE if input is valid or FALSE if not.

### Author(s)

Gaye, A.

---

`lengthDS` *Returns the length of a vector or list*

---

### Description

This function is similar to R function `length`.

### Usage

```
lengthDS(x)
```

### Arguments

| | |
|---|---|
| `x` | a string character, the name of a vector or list |

### Details

The function returns the length of the input vector or list.

### Value

a numeric, the number of elements of the input vector or list.

### Author(s)

Demetris Avraam, for DataSHIELD Development Team

---

`lexisDS1` *lexisDS1*

---

### Description

The first serverside function called by ds.lexis.

### Usage

```
lexisDS1(exitCol = NULL)
```

### Arguments

| | |
|---|---|
| `exitCol` | a character string specifying the variable holding the time that each individual is censored or fails |

### Details

This is an aggregate function. For more details see the extensive header for ds.lexis.

### Author(s)

Burton PR

---

`lexisDS2` *lexisDS2*

---

### Description

The second serverside function called by ds.lexis.

### Usage

```
lexisDS2(datatext = NULL, intervalWidth, maxmaxtime, idCol, entryCol,
  exitCol, statusCol, vartext = NULL)
```

### Arguments

| | |
|---|---|
| `datatext` | a clientside provided character string specifying the data.frame holding the data set to be expanded |
| `intervalWidth` | a clientside generated character string specifying the width of the survival epochs in the expanded data |
| `maxmaxtime` | a clientside generated object specifying the maximum follow up time in any of the sources |
| `idCol` | a clientside generated character string specifying the variable holding the IDs of indivuals in the data set to be expanded |
| `entryCol` | a clientside specified character string identifying the variable holding the time that each individual starts follow up |
| `exitCol` | a clientside specified character string identifying the variable holding the time that each individual ends follow up (is censored or fails) |
| `statusCol` | a clientside specified character string identifying the variable holding the final censoring status (failed/censored) |
| `vartext` | is a clientside provided vector of character strings denoting the column names of additional variables to include in the final expanded table. If the 'variables' argument is not set (is null) but the 'data' argument is set the full data.frame will be expanded and carried forward |

### Details

This is the assign function which actually creates the expanded dataframe containing surival data for a piecewise exponential regression. lexisDS2 also carries out a series of disclosure checks and if the arguments or data fail any of those tests, creation of the exapanded dataframe is blocked and an appropriate serverside error message is stored. For more details see the extensive header for ds.lexis.

## Author(s)

Burton PR

---

`lexisDS3` *@title lexisDS3*

---

## Description

The third serverside function called by ds.lexis.

## Usage

```
lexisDS3()
```

## Details

This is an assign function that simplifies the returned output from ds.lexis. Specifically, without lexisDS3 the output consists of a table within a list, but lexisDS3 converts this directly into a dataframe. For more details see the extensive header for ds.lexis.

---

`listDisclosureSettingsDS` *listDisclosureSettingsDS*

---

## Description

This serverside function is an aggregate function that is called by the ds.listDisclosureSettings

## Usage

```
listDisclosureSettingsDS()
```

## Details

For more details see the extensive header for ds.listDisclosureSettings

## Author(s)

Paul Burton, Demetris Avraam for DataSHIELD Development Team

## listDS *Coerce objects into a list*

### Description

this function is similar to R function 'list'

### Usage

```
listDS(input = NULL, eltnames = NULL)
```

### Arguments

| | |
|---|---|
| input | a list of objects to coerce into a list |
| eltnames | a character list, the names of the elements in the list. |

### Details

Unlike the R function 'list' it takes also a vector of characters, the names of the elements in the output list.

### Value

a list

### Author(s)

Gaye, A.

## matrixDetDS1 *matrixDetDS aggregate function called by ds.matrixDet.report*

### Description

Calculates the determinant of a square matrix A and returns the output to the clientside

### Usage

```
matrixDetDS1(M1.name = NULL, logarithm)
```

### Arguments

| | |
|---|---|
| M1.name | A character string specifying the name of the matrix for which determinant to be calculated |
| logarithm | logical. Default is FALSE, which returns the determinant itself, TRUE returns the logarithm of the modulus of the determinant. |

## Details

Calculates the determinant of a square matrix (for additional information see help for det function in native R). This operation is only possible if the number of columns and rows of A are the same.

## Value

Output is the determinant of the matrix identified by argument <M1> which is returned to the clientside. For more details see help for ds.matrixDet

## Author(s)

Paul Burton for DataSHIELD Development Team

---

`matrixDetDS2` *matrixDetDS assign function called by ds.matrixDet*

---

## Description

Calculates the determinant of a square matrix A and writes the output to the serverside

## Usage

```
matrixDetDS2(M1.name = NULL, logarithm)
```

## Arguments

| | |
|---|---|
| `M1.name` | A character string specifying the name of the matrix for which determinant to be calculated |
| `logarithm` | logical. Default is FALSE, which returns the determinant itself, TRUE returns the logarithm of the modulus of the determinant. |

## Details

Calculates the determinant of a square matrix (for additional information see help for det function in native R). This operation is only possible if the number of columns and rows of A are the same.

## Value

Output is the determinant of the matrix identified by argument <M1> which is written to the serverside. For more details see help for ds.matrixDet

## Author(s)

Paul Burton for DataSHIELD Development Team

matrixDiagDS *matrixDiagDS assign function called by ds.matrixDiag*

## Description

Extracts the diagonal vector from a square matrix A or creates a diagonal matrix A based on a vector or a scalar value and writes the output to the serverside

## Usage

```
matrixDiagDS(x1.transmit, aim, nrows.transmit)
```

## Arguments

| | |
|---|---|
| x1.transmit | identifies the input matrix or vector. Fully specified by <x1> argument of ds.matrixDiag. For more details see help for ds.matrixDiag. |
| aim | a character string specifying what behaviour is required of the function. Fully specified by <aim> argument of ds.matrixDiag. For more details see help for ds.matrixDiag. |
| nrows.transmit | a scalar value forcing the number of rows and columns in an output matrix.Fully specified by <nrows.scalar> argument of ds.matrixDiag. For more details see help for ds.matrixDiag. |

## Details

For details see help for function ds.matrixDiag.

## Value

Output is the matrix or vector specified by the <newobj> argument (or default name diag_<x1>) which is written to the serverside. For more details see help for ds.matrixDiag.

## Author(s)

Paul Burton for DataSHIELD Development Team

## matrixDimnamesDS *matrixDimnamesDS assign function called by ds.matrixDimnames*

### Description

Adds dimnames (row names, column names or both) to a matrix on the serverside.

### Usage

```
matrixDimnamesDS(M1.name = NULL, dimnames)
```

### Arguments

| | |
|---|---|
| M1.name | Specifies the name of the serverside matrix to which dimnames are to be added. Fully specified by <M1> argument of function ds.matrixDimnames. For more details see help for ds.matrixDimnames. |
| dimnames | A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. Fully specified by <dimnames> argument of function ds.matrixDimnames. For more details see help for ds.matrixDimnames. |

### Details

Adds dimnames (row names, column names or both) to a matrix on the serverside. Similar to the dimnames function in native R. For more details see help for function ds.matrixDimnames

### Value

Output is the serverside matrix specified by the <newobj> argument (or default name diag_<x1>) with specified dimnames (row and column names) which is written to the serverside.

### Author(s)

Paul Burton for DataSHIELD Development Team

## matrixDS *matrixDS assign function called by ds.matrix*

### Description

Creates a matrix A on the serverside

### Usage

```
matrixDS(mdata.transmit, from, nrows.transmit, ncols.transmit, byrow,
  dimnames)
```

### Arguments

| | |
|---|---|
| mdata.transmit | specifies the elements of the matrix to be created. Fully specified by <mdata> argument of ds.matrix |
| from | a character string specifying the source and nature of <mdata>. Fully specified by <from> argument of ds.matrix |
| nrows.transmit | specifies the number of rows in the matrix to be created. Fully specified by <nrows.scalar> argument of ds.matrix |
| ncols.transmit | specifies the number of columns in the matrix to be created. Fully specified by <ncols.scalar> argument of ds.matrix |
| byrow | a logical value specifying whether, when <mdata> is a vector, the matrix created should be filled row by row or column by column. Fully specified by <byrow> argument of ds.matrix |
| dimnames | A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names only. Fully specified by <dimnames> argument of ds.matrix |

### Details

Similar to the matrix() function in native R. Creates a matrix with dimensions specified by <nrows.scalar> and <ncols.scalar> arguments and assigns the values of all its elements based on the <mdata> argument

### Value

Output is the matrix A written to the serverside. For more details see help for ds.matrix

### Author(s)

Paul Burton for DataSHIELD Development Team

---

## matrixInvertDS *matrixInvertDS serverside assign function called by ds.matrixInvert*

---

### Description

Inverts a square matrix A and writes the output to the serverside

### Usage

```
matrixInvertDS(M1.name = NULL)
```

### Arguments

| | |
|---|---|
| M1.name | A character string specifying the name of the matrix to be inverted |

### Details

Undertakes standard matrix inversion. This operation is only possible if the number of columns and rows of A are the same and the matrix is non-singular - positive definite (eg there is no row or column that is all zeros)

### Value

Output is the matrix representing the inverse of A which is written to the serverside. For more details see help for ds.matrixInvert

### Author(s)

Paul Burton for DataSHIELD Development Team

---

## matrixMultDS *matrixMultDS serverside assign function called by ds.matrixMult*

---

### Description

Calculates the matrix product of two matrices and writes output to serverside

### Usage

```
matrixMultDS(M1.name = NULL, M2.name = NULL)
```

### Arguments

| | |
|---|---|
| M1.name | A character string specifying the name of the first matrix (M1) argument specified by the M1 argument in the original call to ds.matrixMult |
| M2.name | A character string specifying the name of the second matrix (M2) argument specified by the M1 argument in the original call to ds.matrixMult |

### Details

Undertakes standard matrix multiplication where with input matrices A and B with dimensions A: mxn and B: nxp the output C has dimensions mxp and each elemnt C[i,j] has value equal to the dot product of row i of A and column j of B where the dot product is obtained as sum(A[i,1]*B[1,j] + A[i,2]*B[2,j] + .... + A[i,n]*B[n,j]). This calculation is only valid if the number of columns of A is the same as the number of rows of B

### Value

Output is the matrix representing the product of M1 and M2 which is written to the serverside. For more details see help for ds.matrixMult

### Author(s)

Paul Burton for DataSHIELD Development Team

matrixTransposeDS *matrixTransposeDS serverside assign function called by ds.matrixTranspose*

## Description

Transposes a matrix A and writes the output to the serverside

## Usage

```
matrixTransposeDS(M1.name = NULL)
```

## Arguments

| | |
|---|---|
| M1.name | A character string specifying the name of the matrix to be transposed |

## Details

Undertakes standard matrix transposition. This operation converts matrix A to matrix C where element C[i,j] of matrix C equals element A[j,i] of matrix A. Matrix A therefore has the same number of rows as matrix C has columns and vice versa.

## Value

Output is the matrix representing the transpose of A which is written to the serverside. For more details see help for ds.matrixTranspose

## Author(s)

Paul Burton for DataSHIELD Development Team

meanDS *Computes statistical mean of a vectores*

## Description

Calculates the mean value.

## Usage

```
meanDS(xvect)
```

## Arguments

| | |
|---|---|
| xvect | a vector |

## Details

if the length of input vector is less than the set filter a missing value is returned.

## Value

a numeric, the statistical mean

## Author(s)

Gaye A, Burton PR

---

`meanSdGpDS` *MeanSdGpDS*

---

## Description

Serverside function called by ds.meanSdGp

## Usage

```
meanSdGpDS(X, INDEX)
```

## Arguments

| | |
|---|---|
| `X` | a clientside supplied character string identifying the variable for which means/SDs are to be calculated |
| `INDEX` | a clientside supplied character string identifying the factor across which means/SDs are to be calculated |

## Details

Computes the mean and standard deviation across groups defined by one factor

## Author(s)

Burton PR

mergeDS *mergeDS (assign function) called by ds.merge*

## Description

merges (links) two data.frames together based on common values in defined vectors in each data.frame

## Usage

```
mergeDS(x.name, y.name, by.x.names.transmit, by.y.names.transmit, all.x,
  all.y, sort, suffixes.transmit, no.dups, incomparables)
```

## Arguments

| | |
|---|---|
| x.name, | the name of the first data.frame to be merged specified in inverted commas. Specified via argument <x.name> of ds.merge function |
| y.name, | the name of the second data.frame to be merged specified in inverted commas. Specified via argument <y.name> of ds.merge function |
| by.x.names.transmit | the name of a single variable or a vector of names of multiple variables (in transmittable form) containing the IDs or other data on which data.frame x is to be merged/linked to data.frame y. Specified via argument <by.x.names> of ds.merge function |
| by.y.names.transmit | the name of a single variable or a vector of names of multiple variables (in transmittable form) containing the IDs or other data on which data.frame y is to be merged/linked to data.frame x. Specified via argument <by.y.names> of ds.merge function |
| all.x | logical, if TRUE, then extra rows will be added to the output, one for each row in x that has no matching row in y. Specified via argument <all.x> of ds.merge function. Default = FALSE. |
| all.y | logical, if TRUE, then extra rows will be added to the output, one for each row in y that has no matching row in x. Specified via argument <all.y> of ds.merge function. Default = FALSE. |
| sort | logical, if TRUE the merged result should be sorted on elements in the by.x.names and by.y.names columns. Specified via argument <sort> of ds.merge function. Default = TRUE. |
| suffixes.transmit | a character vector of length 2 (in transmittable form) specifying the suffixes to be used for making unique common column names in the two input data.frames when they both appear in the merged data.frame. Specified via argument <suffixes> of ds.merge function. Default '.x' and '.y'. |
| no.dups | logical, when TRUE suffixes are appended in more cases to rigorously avoid duplicated column names in the merged data.frame. Specified via argument <no.dups> of ds.merge function. Default TRUE but was apparently implicitly FALSE before R version 3.5.0. |

`incomparables`,
: values intended for merging on one column which cannot be matched. See 'match' in help for Native R merge function. Specified via argument <incomparables> of ds.merge

## Details

For further information see details of the native R function merge and the DataSHIELD clientside function ds.merge.

## Value

the merged data.frame specified by the <newobj> argument of ds.merge (or by default 'x.name_y.name' if the <newobj> argument is NULL) which is written to the serverside. In addition, two validity messages are returned to the clientside indicating whether <newobj> has been created in each data source and if so whether it is in a valid form. If its form is not valid in at least one study there may be a studysideMessage that can explain the error in creating the full output object. As well as appearing on the screen at run time,if you wish to see the relevant studysideMessages at a later date you can use the ds.message function. If you type ds.message(<newobj>) it will print out the relevant studysideMessage from any datasource in which there was an error in creating <newobj> and a studysideMessage was saved. If there was no error and <newobj> was created without problems no studysideMessage will have been saved and ds.message(<newobj>) will return the message: "ALL OK: there are no studysideMessage(s) on this datasource".

## Author(s)

Amadou Gaye, Paul Burton, for DataSHIELD Development Team

---

`messageDS` *messageDS*

---

## Description

This function allows for error messages arising from the running of a server-side assign function to be returned to the client-side

## Usage

```
messageDS(message.object.name)
```

## Arguments

`message.object.name`
: is a character string, containing the name of the list containing the message. See the header of the client-side function ds.message for more details.

### Details

Errors arising from aggregate server-side functions can be returned directly to the client-side. But this is not possible for server-side assign functions because they are designed specifically to write objects to the server-side and to return no meaningful information to the client-side. Otherwise, users may be able to use assign functions to return disclosive output to the client-side. ds.message calls messageDS which looks specifically for an object called $serversideMessage in a designated list on the server-side. Server-side functions from which error messages are to be made available, are designed to be able to write the designated error message to the $serversideMessage object into the list that is saved on the server-side as the primary output of that function. So only valid server-side functions of DataSHIELD can write a $studysideMessage, and as additional protection against unexpected ways that someone may try to get round this limitation, a $studysideMessage is a string that cannot exceed a length of nfilter.string a default of 80 characters.

### Value

a list object from each study, containing whatever message has been written by DataSHIELD into $studysideMessage.

### Author(s)

Burton PR

---

## namesDS *Returns the names of a list*

---

### Description

Returns the names of the object in the list.

### Usage

```
namesDS(xlist)
```

### Arguments

| | |
|---|---|
| xlist | a list |

### Details

This is similar to R base function `names` but restricted to list types only.

### Value

a character vector or NULL if the list does not have names

### Author(s)

Gaye, A.

numNaDS *Counts the number of missing values*

## Description

this function just counts the number of missing entries in a vector.

## Usage

```
numNaDS(xvect)
```

## Arguments

| | |
|---|---|
| xvect | a vector |

## Value

an integer, the number of missing values

## Author(s)

Gaye, A.

quantileMeanDS *Generates quantiles and mean information without maximum and minimum*

## Description

the probabilities 5 are used to compute the corresponding quantiles.

## Usage

```
quantileMeanDS(xvect)
```

## Arguments

| | |
|---|---|
| xvect | a numerical vector |

## Value

a numeric vector that represents the sample quantiles

## Author(s)

Burton, P.; Gaye, A.

## rangeDS *returns the minimum and maximum of a numeric vector*

### Description

this function is similar to R function range but instead to not return the real minimum and maximum, the computed values are multiplied by a very small random number.

### Usage

```
rangeDS(xvect)
```

### Arguments

| | |
|---|---|
| xvect | a numerical |

### Value

a numeric vector which contains the minimum and the maximum values of the vector

### Author(s)

Amadou Gaye, Demetris Avraam for DataSHIELD Development Team

## rbindDS *rbindDS called by ds.rbind*

### Description

serverside assign function that takes a sequence of vector, matrix or data-frame arguments and combines them by row to produce a matrix.

### Usage

```
rbindDS(x.names.transmit = NULL, colnames.transmit = NULL)
```

### Arguments

| | |
|---|---|
| x.names.transmit | This is a vector of character strings representing the names of the elemental components to be combined converted into a transmittable format. This argument is fully specified by the <x> argument of ds.rbind |
| colnames.transmit | This is NULL or a vector of character strings representing forced column names for the output object converted into a transmittable format. This argument is fully specified by the <force.colnames> argument of ds.cbind. |

## Details

A sequence of vector, matrix or data-frame arguments is combined row by row to produce a matrix which is written to the serverside. For more details see help for ds.rbind and the native R function rbind.

## Value

the object specified by the <newobj> argument of ds.rbind(or default name <rbind.out>) which is written to the serverside. As well as writing the output object as <newobj> on the serverside, two validity messages are returned indicating whether <newobj> has been created in each data source and if so whether it is in a valid form. If its form is not valid in at least one study - e.g. because a disclosure trap was tripped and creation of the full output object was blocked - ds.cbind() also returns any studysideMessages that can explain the error in creating the full output object. As well as appearing on the screen at run time,if you wish to see the relevant studysideMessages at a later date you can use the ds.message function. If you type ds.message("<newobj>") it will print out the relevant studysideMessage from any datasource in which there was an error in creating <newobj> and a studysideMessage was saved. If there was no error and <newobj> was created without problems no studysideMessage will have been saved and ds.message("<newobj>") will return the message: "ALL OK: there are no studysideMessage(s) on this datasource".

## Author(s)

Paul Burton for DataSHIELD Development Team

---

`rBinomDS` *rBinomDS serverside assign function*

---

## Description

primary serverside assign function called by ds.rBinom

## Usage

```
rBinomDS(n, size = 1, prob = 0.5)
```

## Arguments

| | |
|---|---|
| `n` | length of the pseudorandom number vector to be generated as specified by the argument <samp.size> in the function ds.rBinom |
| `size` | a scalar that must be a positive integer. Value set directly by <size> argument of ds.rBinom - for details see help for ds.rBinom. May be a scalar or a vector allowing the size to vary from observation to observation. |
| `prob` | a numeric scalar in range 0 > prob > 1 which specifies the probability of a positive response. Value set directly by <prob> argument of ds.rBinom - for details see help for ds.rBinom May be a scalar or a vector allowing the size to vary from observation to observation. |

### Details

Generates the vector of pseudorandom numbers from a binomial distribution in each data source as specified by the arguments of ds.rBinom. This serverside function is effectively the same as the function rbinom() in native R and its arguments are the same.

### Value

Writes the pseudorandom number vector with the characteristics specified in the function call as a new serverside vector on the data source on which it has been called. Also returns key information to the clientside: the random seed as specified by you in each source + (if requested) the full 626 length random seed vector this generated in each source (see info for the argument <return.full.seed.as.set>). It also returns a vector reporting the length of the pseudorandom vector created in each source.

### Author(s)

Paul Burton for DataSHIELD Development Team

---

## recodeLevelsDS *Recodes the levels of a categorical variables*

---

### Description

The functions uses the input factor and generates a new factor with new levels.

### Usage

```
recodeLevelsDS(x = NULL, classes = NULL)
```

### Arguments

| | |
|---|---|
| x | a factor vector |
| classes | a character vector the levels of the newt factor vector |

### Value

a factor vector with the new levels

### Author(s)

Gaye, A.

recodeValuesDS1 *recodeValuesDS1 an aggregate function called by ds.recodeValues*

## Description

First serverside function called by ds.recodeValues to convert specified values of elements in a vector into a matched set of alternative specified values.

## Usage

```
recodeValuesDS1(var.name.text = NULL, values2replace.text = NULL,
  new.values.text = NULL)
```

## Arguments

| | |
|---|---|
| `var.name.text` | a character string providing the name for the vector representing the variable to be recoded. The <var.name.text> argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues |
| `values2replace.text` | a character string specifying the values in the vector specified by the argument <var.name.text> that are to be replaced by new values as specified in the new.values.vector. The <values2replace.text> argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues. In effect, the <values2replace.vector> argument of the ds.recodeValues function is converted to a character string format that is acceptable to the DataSHIELD parser in Opal and so can be accepted by recodeValuesDS1 |
| `new.values.text` | a character string specifying the new values to which the specified values in the vector identified by the <var.name> argument are to be converted. The <new.values.text> argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues. In effect, the <new.values.vector> argument of the ds.recodeValues function is converted to a character string format that is acceptable to the DataSHIELD parser in Opal and so can be used in the call to recodeValuesDS1 |

## Details

For all details see the help header for ds.recodeValues

## Value

This first serverside function called by ds.recodeValues provides first level traps for a comprehensive series of disclosure risks which can be returned directly to the clientside because recodeValuesDS1 is an aggregate function. The second serverside function called by ds.dataFrameSubset (recodeValuesDS2) carries out some of the same disclosure tests, but it is an assign function because it writes the recoded vector to the serverside. In consequence, it records error messages as studysideMessages which can only be retrieved using ds.message

### Author(s)

DataSHIELD Development Team

---

`recodeValuesDS2` *recodeValuesDS2 an assign function called by ds.recodeValues*

---

### Description

Second serverside function called by ds.recodeValues to convert specified values of elements in a vector into a matched set of alternative specified values.

### Usage

```
recodeValuesDS2(var.name.text = NULL, values2replace.text = NULL,
  new.values.text = NULL, numeric.output.format.possible,
  force.output.format = "no", v2r.numeric = NULL)
```

### Arguments

`var.name.text` a character string providing the name for the vector representing the variable to be recoded. <var.name.text> argument generated and passed directly to recodeValuesDS2 by ds.recodeValues

`values2replace.text`
a character string specifying the values in the vector specified by the argument <var.name.text> that are to be replaced by new values as specified in the new.values.vector. The <values2replace.text> argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues. In effect, the <values2replace.vector> argument of the ds.recodeValues function is converted to a character string format that is acceptable to the DataSHIELD parser in Opal and so can be accepted by recodeValuesDS2

`new.values.text`
a character string specifying the new values to which the specified values in the vector <var.name> are to be converted. The <new.values.text> argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues. In effect, the <new.values.vector> argument of the ds.recodeValues function is converted to a character string format that is acceptable to the DataSHIELD parser in Opal and so can be used in the call to recodeValuesDS2

`numeric.output.format.possible`
logical, if TRUE the nature of <var.name>, <values2replace.vector> and <new.values.vector> are such that it is in principle possible for the output to be fully numeric. This argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues - its value determines how recodeValuesDS2 handles situations where a numeric output may be desirable.

`force.output.format`
character string. This argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues. For details see the equivalent parameter in the help header for ds.recodeValues

| | |
|---|---|
| v2r.numeric | logical. This argument is generated and passed directly to recodeValuesDS2 by ds.recodeValues. If TRUE it informs recodeValuesDS2 that the nature of <var.name>, <values2replace.vector>, <new.values.vector> and <force.output.format> are such that recodeValuesDS2 should convert the recoded (output) vector to numeric. If false, recodeValuesDS2 should write out the recoded (output) vector as character. |

## Details

For all details see the help header for ds.recodeValues

## Value

the object specified by the <newobj> argument (or default name '<var.name>_recoded') initially specified in calling ds.recodeValues. The output object (the required recoded variable called <newobj> is written to the serverside. In addition, two validity messages are returned via ds.recodeValues indicating whether <newobj> has been created in each data source and if so whether it is in a valid form. If its form is not valid in at least one study - e.g. because a disclosure trap was tripped and creation of the full output object was blocked - recodeValuesDS2 (via ds.recodeValues()) also returns any studysideMessages that can explain the error in creating the full output object. As well as appearing on the screen at run time,if you wish to see the relevant studysideMessages at a later date you can use the ds.message function. If you type ds.message("newobj") it will print out the relevant studysideMessage from any datasource in which there was an error in creating <newobj> and a studysideMessage was saved. If there was no error and <newobj> was created without problems no studysideMessage will have been saved and ds.message("newobj") will return the message: "ALL OK: there are no studysideMessage(s) on this datasource".

## Author(s)

DataSHIELD Development Team

---

# replaceNaDS *Replaces the missing values in a vector*

---

## Description

This function identifies missing values and replaces them by a value or values specified by the analyst.

## Usage

```
replaceNaDS(xvect, replacements)
```

## Arguments

| | |
|---|---|
| xvect | a character, the name of the vector to process. |
| replacements | a vector which contains the replacement value(s), a vector one or more values for each study. |

## Details

This function is used when the analyst prefer or requires complete vectors. It is then possible the specify one value for each missing value by first returning the number of missing values using the function numNaDS but in most cases it might be more sensible to replace all missing values by one specific value e.g. replace all missing values in a vector by the mean or median value. Once the missing values have been replaced a new vector is created.

## Value

a new vector without missing values

## Author(s)

Gaye, A.

---

# reShapeDS *reShapeDS (assign function) called by ds.reShape*

---

## Description

Reshapes a data frame containing longitudinal or otherwise grouped data from 'wide' to 'long' format or vice-versa

## Usage

```
reShapeDS(data.name, varying.transmit, v.names.transmit, timevar.name,
  idvar.name, drop.transmit, direction, sep)
```

## Arguments

| | |
|---|---|
| data.name, | the name of the data.frame to be reshaped. Specified via argument <data.name> of ds.reShape function |
| varying.transmit, | names of sets of variables in the wide format that correspond to single variables in long format (typically what may be called 'time-varying' or 'time-dependent' variables). Specified via argument <varying> of ds.reShape function. |
| v.names.transmit, | the names of variables in the long format that correspond to multiple variables in the wide format - for example, sbp7, sbp11, sbp15 (measured systolic blood pressure at ages 7, 11 and 15 years). Specified via argument <v.names> of ds.reShape function |
| timevar.name, | the variable in long format that differentiates multiple records from the same group or individual. Specified via argument <timevar.name> of ds.reShape function |
| idvar.name, | names of one or more variables in long format that identify multiple records from the same group/individual. This/these variable(s) may also be present in wide format. Specified via argument <idvar.name> of ds.reShape function |

`drop.transmit`,
: a vector of names of variables to drop before reshaping. Specified via argument <drop> of ds.reShape function

`direction`,
: a character string, partially matched to either "wide" to reshape from long to wide format, or "long" to reshape from wide to long format. Specified via argument <direction> of ds.reShape function

`sep`,
: a character vector of length 1, indicating a separating character in the variable names in the wide format. Specified via argument <sep> of ds.reShape function

## Details

This function is based on the native R function `reshape`. It reshapes a data frame containing longitudinal or otherwise grouped data between 'wide' format with repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records. The reshaping can be in either direction

## Value

a reshaped data.frame converted from long to wide format or from wide to long format which is written to the serverside and given the name provided as the <newobj> argument of ds.reShape or 'newObject' if no name is specified. In addition, two validity messages are returned to the clientside indicating whether <newobj> has been created in each data source and if so whether it is in a valid form (see header for ds.reShape.

## Author(s)

Demetris Avraam, Paul Burton for DataSHIELD Development Team

---

# rmDS *rmDS an aggregate function called by ds.rm*

---

## Description

deletes an R object on the serverside

## Usage

```
rmDS(x.name.transmit)
```

## Arguments

`x.name.transmit`,
: the name of the object to be deleted converted into transmissable form. The argument is specified via the <x.name> argument of ds.rm

### Details

this is a serverside function based on the rm() function in native R. It is an aggregate function which may be surprising because it modifies an object on the serverside, and would therefore be expected to be an assign function. However, as an assign function the last step in running it would be to write the modified object as newobj. But this would fail because the effect of the function is to delete the object and so it would be impossible to write it anywhere.

### Value

the specified object is deleted from the serverside. If this is successful the message "Object <x.name> successfully deleted" is returned to the clientside (where x.name is the name of the object to be deleted). If the object to be deleted is already absent on a given source, that source will return the message: "Object to be deleted, i.e. <x.name> , does not exist so does not need deleting". Finally, if the specified name of the object to be deleted is too long (>nfilter.stringShort) there is a potential disclosure risk (active code hidden in the name) and the rmDS returns a message such as: "Disclosure risk, number of characters in x.name must not exceed nfilter.stringShort which is currently set at: 25" where '25' is the current setting of the R_Option value of nfilter.stringShort.

### Author(s)

Paul Burton for DataSHIELD Development Team

---

## rNormDS *rNormDS serverside assign function*

---

### Description

primary serverside assign function called by ds.rNorm

### Usage

```
rNormDS(n, mean = 0, sd = 1, force.output.to.k.decimal.places = 9)
```

### Arguments

| | |
|---|---|
| n | length of the pseudorandom number vector to be generated as specified by the argument <samp.size> in the function ds.rNorm |
| mean | this specifies the mean of the pseudorandom number vector to be generated as specified by the argument <mean> in the function ds.rNorm. May be a scalar or a vector allowing the mean to vary from observation to observation. |
| sd | this specifies the standard deviation of the pseudorandom number vector to be generated as specified by the argument <sd> in the function ds.rNorm May be a scalar or a vector allowing the sd to vary from observation to observation. |

`force.output.to.k.decimal.places`
: scalar integer. Forces the output random number vector to have k decimal places. If 0 rounds it coerces decimal random number output to integer, a k in range 1-8 forces output to have k decimal places. If k = 9, no rounding occurs of native output. Default=9. Value specified by <force.output.to.k.decimal.places> argument in ds.rNorm

## Details

Generates the vector of pseudorandom numbers from a normal distribution in each data source as specified by the arguments of ds.rNorm. This serverside function is effectively the same as the function rnorm() in native R and its arguments are the same.

## Value

Writes the pseudorandom number vector with the characteristics specified in the function call as a new serverside vector on the data source on which it has been called. Also returns key information to the clientside: the random seed as specified by you in each source + (if requested) the full 626 length random seed vector this generated in each source (see info for the argument <return.full.seed.as.set>). It also returns a vector reporting the length of the pseudorandom vector created in each source.

## Author(s)

Paul Burton for DataSHIELD Development Team

---

# rowColCalcDS *Computes sums and means of rows or columns of numeric arrays*

---

## Description

The function is similar to R base functions 'rowSums', 'colSums', 'rowMeans' and 'colMeans'.

## Usage

```
rowColCalcDS(dataset, operation)
```

## Arguments

| | |
|---|---|
| `dataset` | an array of two or more dimensions. |
| `operation` | an integer that indicates the operation to carry out: 1 for 'rowSums', 2 for 'colSums', 3 for 'rowMeans' or 4 for 'colMeans' |

## Details

the output is returned to the user only the number of entries in the output vector is greater or equal to the allowed size.

### Value

a numeric vector

### Author(s)

Gaye, A.

---

`rPoisDS` *rPoisDS serverside assign function*

---

### Description

primary serverside assign function called by ds.rPois

### Usage

```
rPoisDS(n, lambda = 1)
```

### Arguments

| | |
|---|---|
| `n` | length of the pseudorandom number vector to be generated as specified by the argument <samp.size> in the function ds.rPois |
| `lambda` | a numeric scalar specifying the expected count of the Poisson distribution used to generate the random counts. Specified directly by the lambda argument in ds.rPois. May be a scalar or a vector allowing lambda to vary from observation to observation. |

### Details

Generates the vector of pseudorandom numbers (non-negative integers) from a Poisson distribution in each data source as specified by the arguments of ds.rPois. This serverside function is effectively the same as the function rpois() in native R and its arguments are the same.

### Value

Writes the pseudorandom number vector with the characteristics specified in the function call as a new serverside vector on the data source on which it has been called. Also returns key information to the clientside: the random seed as specified by you in each source + (if requested) the full 626 length random seed vector this generated in each source (see info for the argument <return.full.seed.as.set>). It also returns a vector reporting the length of the pseudorandom vector created in each source.

### Author(s)

Paul Burton for DataSHIELD Development Team

rUnifDS *rUnifDS serverside assign function*

## Description

primary serverside assign function called by ds.rUnif

## Usage

```
rUnifDS(n, min = 0, max = 1, force.output.to.k.decimal.places = 9)
```

## Arguments

| | |
|---|---|
| n | length of the pseudorandom number vector to be generated as specified by the argument <samp.size> in the function ds.rUnif |
| min | a numeric scalar specifying the minimum of the range across which the random numbers will be generated in each source. Specified directly by the min argument in ds.rUnif. May be a scalar or a vector allowing the min to vary from observation to observation. |
| max | a numeric scalar specifying the maximum of the range across which the random numbers will be generated in each source. Specified directly by the max argument in ds.rUnif. May be a scalar or a vector allowing the min to vary from observation to observation. |
| force.output.to.k.decimal.places | scalar integer. Forces the output random number vector to have k decimal places. If 0 rounds it coerces decimal random number output to integer, a k in range 1-8 forces output to have k decimal places. If k = 9, no rounding occurs of native output. Default=9. Value specified by <force.output.to.k.decimal.places> argument in ds.rUnif |

## Details

Generates the vector of pseudorandom numbers from a uniform distribution in each data source as specified by the arguments of ds.rUnif. This serverside function is effectively the same as the function runif() in native R and its arguments are the same.

## Value

Writes the pseudorandom number vector with the characteristics specified in the function call as a new serverside vector on the data source on which it has been called. Also returns key information to the clientside: the random seed as specified by you in each source + (if requested) the full 626 length random seed vector this generated in each source (see info for the argument <return.full.seed.as.set>). It also returns a vector reporting the length of the pseudorandom vector created in each source.

## Author(s)

Paul Burton for DataSHIELD Development Team

scatterPlotDS *Calculates the coordinates of the data to be plot*

## Description

This function uses two disclosure control methods to generate non-disclosive coordinates that are returned to the client that generates the non-disclosive scatter plots.

## Usage

```
scatterPlotDS(x, y, method.indicator, k, noise)
```

## Arguments

| | |
|---|---|
| x | the name of a numeric vector, the x-variable. |
| y | the name of a numeric vector, the y-variable. |
| method.indicator | an intiger either 1 or 2. If the user selects the deterministic method in the client side function the method.inticator is set to 1 while if the user selects the probabilistic method this argument is set to 2. |
| k | the number of the nearest neghbours for which their centroid is calculated if the deterministic method is selected. |
| noise | the percentage of the initial variance that is used as the variance of the embedded noise if the probabilistic method is selected. |

## Details

If the user chooses the deteministic approach, the function finds the k-1 nearest neighbours of each data point in a 2-dimensional space. The nearest neighbours are the data points with the minimum Euclidean distances from the point of interest. Each point of interest and its k-1 nearest neighbours are then used for the calculation of the coordinates of the centroid of those k points. Centroid here is referred to the centre of mass, i.e. the x-coordinate of the centroid is the average value of the x-coordinates of the k nearest neighbours and the y-coordinate of the centroid is the average of the y-coordinates of the k nearest neighbours. If the user chooses the probabilistic approach, the function adds random noise to $x$ and $y$ separately. Each random noise follows a normal distribution with zero mean and variance equal to 10 disclosure we fix the random number generator in a value that is specified by the input variables. Thus the function returns always the same noisy data for a given pair of variables.

## Value

a list with the x and y coordinates of the data to be plot

## Author(s)

Demetris Avraam for DataSHIELD Development Team

scoreVectDS *Generates the score vector and information matrix*

## Description

This function is called by the client function 'ds.gee' to produced the score vector and information matrix.

## Usage

```
scoreVectDS(data, formula, family, clusterID, corstr, alpha, phi,
  startBetas, zcor = NULL)
```

## Arguments

| | |
|---|---|
| data | the input dataframe which contains the variable specified in the formula. |
| formula | a regression formula. |
| family | the link function for the regression. |
| clusterID | the name of the column that holds the cluster IDs. |
| corstr | the correlation structure. |
| alpha | the parameter alpha |
| phi | the parameter alpha |
| startBetas | a character, the starting values concatenated by comma |
| zcor | the user defined matrix user if the correlation structure is 'fixed'. |

## Details

the score vector and information matrix are calculated according to the correlation structure.

## Value

a list

## Author(s)

Gaye, A.; Jones EM.

seqDS *seqDS called by ds.seq*

## Description

assign function seqDS called by ds.seq

## Usage

```
seqDS(FROM.value.char, BY.value.char, LENGTH.OUT.value.char,
  ALONG.WITH.name)
```

## Arguments

`FROM.value.char`
: the starting value for the sequence expressed as an integer in character form. e.g. FROM.value.char="1" will start at 1, FROM.value.char="-10" will start at -10. Default = "1". The value of this argument is usually specified by the value provided to the argument <FROM.value.char> in the function ds.seq

`BY.value.char`
: the value to increment each step in the sequence expressed as a numeric e.g. BY.value.char="10" will increment by 10, while BY.value.char="-3.37" will reduce the value of each sequence element by -3.37. Default = "1" but does not have to be integer. The value of this argument is usually specified by the value provided to the argument <BY.value.char> in the function ds.seq

`LENGTH.OUT.value.char`
: The length of the sequence at which point its extension should be stopped. e.g. LENGTH.OUT.value.char="1000" will generate a sequence of length 1000. Default = NULL (must be specified) but must be a positive integer. The value of this argument is usually specified by the value provided to the argument <LENGTH.OUT.value.char> in the function ds.seq

`ALONG.WITH.name`
: For convenience, rather than specifying a value for LENGTH.OUT it can often be better to specify a variable name as the <ALONG.WITH.name> argument. e.g. ALONG.WITH.name = "vector.name". This can be particularly useful in DataSHIELD where the length of the sequence you need to generate in each data set depends on the standard length of vectors in that data set and this will in general vary. The value of this argument is usually specified by the value provided to the argument <ALONG.WITH.name> in the function ds.seq

## Details

An assign function that uses the native R function seq() to create any one of a flexible range of sequence vectors that can then be used to help manage and analyse data. As it is an assign function the resultant vector is written as a new object onto all of the specified data source servers. For the purposes of creating the DataSHIELD equivalent to seq() in native R we have used all of the original arguments (see below) except the <to> argument. This simplifies the function and prevents some combinations of arguments that lead to an error in native R. The effect of the <to> argument

- see help(seq) in native R - is to specify the terminal value of the sequence. However, when using seq() one can usually specify other arguments (see below) to mimic the desire effect of <to>. These include: <from>, the starting value of the sequence; <by>, its increment (+ or -), and <length.out> the length of the final vector in each data source.

## Value

the object specified by the <newobj> argument of function ds.seq (or default name newObj) which is written to the serverside. As well as writing the output object as <newobj> on the serverside, two validity messages are returned indicating whether <newobj> has been created in each data source and if so whether it is in a valid form. If its form is not valid in at least one study - e.g. because a disclosure trap was tripped and creation of the full output object was blocked - ds.seq() also returns any studysideMessages that can explain the error in creating the full output object. As well as appearing on the screen at run time,if you wish to see the relevant studysideMessages at a later date you can use the ds.message function. If you type ds.message("<newobj>") it will print out the relevant studysideMessage from any datasource in which there was an error in creating <newobj> and a studysideMessage was saved. If there was no error and <newobj> was created without problems no studysideMessage will have been saved and ds.message("<newobj>") will return the message: "ALL OK: there are no studysideMessage(s) on this datasource".

## Author(s)

Paul Burton for DataSHIELD Development Team

---

# setSeedDS

*setSeedDs called by ds.setSeed, ds.rNorm, ds.rUnif, ds.rPois and ds.rBinom*

---

## Description

An aggregate serverside function that primes the pseudorandom number generator in a data source

## Usage

```
setSeedDS(seedtext = NULL, kind = NULL, normal.kind = NULL)
```

## Arguments

| | |
|---|---|
| seedtext | this is simply the value of the <seed.as.integer> argument of ds.setSeed, ds.rNorm, ds.rUnif, ds.rPois of ds.rBinom coerced into character format. This is done by the clientside functions themselves and does not require the DataSHIELD user to do anything. Please see the help for these clientside functions, and in particular, the information for the argument <seed.as.integer> for more details. |
| kind | see help for set.seed() function in native R |
| normal.kind | see help for set.seed() function in native R |

### Details

setSeedDS is effectively equivalent to the native R function set.seed() and so the help for that function can provide many additional details. The only very minor difference is that the first argument of setSeedDS, <seedtext> takes the integer priming seed in character format. However, for the user that integer is still specified directly as an integer as the <seed.as.integer> argument of one of the clientside functions ds.setSeed, ds.rNorm ..... Each of these clientside functions coerces the integer to character format calls setSeedDS and the first active line of code in setSeedDS converts the character string back to an integer and treats it as the first argument <seed> of the native R function set.seed(). The two other arguments of set.seed() in native R, <kind> and <normal.kind> are both defaulted by specifying them as NULL. This defaulting is hard wired into the setSeedDS function and as this cannot be changed by the analyst it means that setSeedDS is much less flexible than native R's set.seed() function. If any DataSHIELD user requires some aspect of this flexibility returned the development team can be approached, but unless you are actually doing theoretical work with random number generators it is likely that the

### Value

Sets the values of the vector of integers of length 626 known as .Random.seed on each data source that is the true current state of the random seed in each source.

### Author(s)

Paul Burton for DataSHIELD Development Team

---

## subsetByClassDS *Breaks down a dataframe or a factor into its sub-classes*

---

### Description

The function takes a categorical vector or dataframe as input and generates subset(s) vectors or dataframes for each category. Subsets are considered invalid if they hold between 1 and 4 observations.

### Usage

```
subsetByClassDS(data = NULL, variables = NULL)
```

### Arguments

| | |
|---|---|
| data | a string character, the name of the dataframe or the factor vector |
| variables | a vector of string characters, the names of the the variables to subset on. |

### Details

If the input data object is a dataframe it is possible to specify the variables to subset on. If a subset is not 'valid' all its the values are reported as missing (i.e. NA), the name of the subsets is labelled as '_INVALID'. If no variables are specified to subset on, the dataframe will be subset on each of its factor variables. And if none of the columns holds a factor variable a message is issued as output. A message is also issued as output if the input vector is not of type factor.

## Value

a list which contains the subsetted datasets

## Author(s)

Gaye, A.

---

`subsetDS` *Generates a valid subset of a table or a vector*

---

## Description

The function uses the R classical subsetting with squared brackets '[]' and allows also to subset using a logical oprator and a threshold. The object to subset from must be a vector (factor, numeric or charcater) or a table (data.frame or matrix).

## Usage

```
subsetDS(dt = NULL, complt = NULL, rs = NULL, cs = NULL,
  lg = NULL, th = NULL, varname = NULL)
```

## Arguments

| | |
|---|---|
| `dt` | a string character, the name of the dataframe or the factor vector and the range of the subset. |
| `complt` | a boolean that tells if the subset to subset should include only complete cases |
| `rs` | a vector of two integers that give the range of rows de extract. |
| `cs` | a vector of two integers or one or more characters; the indices of the columns to extract or the names of the columns (i.e. names of the variables to extract). |
| `lg` | a character, the logical parameter to use if the user wishes to subset a vector using a logical operator. This parameter is ignored if the input data is not a vector. |
| `th` | a numeric, the threshold to use in conjunction with the logical parameter. This parameter is ignored if the input data is not a vector. |
| `varname` | a character, if the input data is a table, if this parameter is provided along with the 'logical' and 'threshold' parameters, a subtable is based the threshold applied to the speicified variable. This parameter is however ignored if the parameter 'rows' and/or 'cols' are provided. |

### Details

If the input data is a table: The user specifies the rows and/or columns to include in the subset if the input object is a table; the columns can be refered to by their names. The name of a vector (i.e. a variable) can also be provided with a logical operator and a threshold (see example 3). If the input data is a vector: when the parameters 'rows', 'logical' and 'threshold' are all provided the last two are ignored ( 'rows' has precedence over the other two parameters then). If the requested subset is not valid (i.e. contains less than the allowed number of observations), the subset is not generated, rather a table or a vector of missing values is generated to allow for any subsequent process using the output of the function to proceed after informing the user via a message.

### Value

a subset of the vector, matric or dataframe as specified is stored on the server side

### Author(s)

Gaye, A.

---

## table1DDS *Creates 1-dimensional contingency tables*

---

### Description

This function generates a 1-dimensional table where potentially disclosive cells. (based on the set threshold) are replaced by a missing value ('NA').

### Usage

```
table1DDS(xvect)
```

### Arguments

| | |
|---|---|
| xvect | a numerical vector with discrete values - usually a factor. |

### Details

It generates a 1-dimensional tables where valid (non-disclosive) 1-dimensional tables are defined as data from sources where no table cells have counts between 1 and the set threshold. When the ouput table is invalid all cells but the total count are replaced by missing values. Only the total count is visible on the table returned to the client site. A message is also returned with the 1-dimensional; the message says "invalid table - invalid counts present" if the table is invalid and 'valid table' otherwise.

### Value

a list which contains two elements: 'table', the 1-dimensional table and 'message' a message which informs about the validity of the table.

### Author(s)

Gaye A.

---

`table2DDS` *table2DDS (aggregate function) called by ds.table2D*

---

### Description

This function generates a 2-dimensional contingency table where potentially disclosive cells (based on a set threshold) are replaced by a missing value ('NA').

### Usage

```
table2DDS(xvect, yvect)
```

### Arguments

| | |
|---|---|
| `xvect` | a numerical vector with discrete values - usually a factor. |
| `yvect` | a numerical vector with discrete values - usually a factor. |

### Details

It generates 2-dimensional contingency tables where valid (non-disclosive) tables are defined as those where none of their cells have counts between 1 and the set threshold "nfilter.tab". When the ouput table is invalid all cells except the total counts are replaced by missing values. Only the total counts are visible on the table returned to the client side. A message is also returned with the 2-dimensional table; the message says "invalid table - invalid counts present" if the table is invalid and 'valid table' otherwise.

### Value

a list which contains two elements: 'table', the 2-dimensional table and 'message' a message which informs about the validity of the table.

### Author(s)

Amadou Gaye, Paul Burton, Demetris Avraam for DataSHIELD Development Team

## tapplyDS *tapplyDS called by ds.tapply*

### Description

Apply one of a selected range of functions to summarize an outcome variable over one or more indexing factors and write the resultant summary to the clientside

### Usage

```
tapplyDS(X.name, INDEX.names.transmit, FUN.name)
```

### Arguments

| | |
|---|---|
| X.name, | the name of the variable to be summarized. Specified via argument <X.name> of ds.tapply function |
| INDEX.names.transmit, | the name of a single factor or a vector of names of factors to index the variable to be summarized. Specified via argument <INDEX.names> of ds.tapply function |
| FUN.name, | the name of one of the allowable summarizing functions to be applied. Specified via argument <FUN.name> of ds.tapply function. |

### Details

see details for ds.tapply function

### Value

an array of the summarized values created by the tapplyDS function. This array is returned to the clientside. It has the same number of dimensions as INDEX.

### Author(s)

Paul Burton, Demetris Avraam for DataSHIELD Development Team

## tapplyDS.assign *tapplyDS.assign called by ds.tapply.assign*

### Description

Apply one of a selected range of functions to summarize an outcome variable over one or more indexing factors and write the resultant summary as a newobj on the serverside

### Usage

```
tapplyDS.assign(X.name, INDEX.names.transmit, FUN.name)
```

### Arguments

| | |
|---|---|
| X.name, | the name of the variable to be summarized. Specified via argument <X.name> of ds.tapply.assign function |
| INDEX.names.transmit, | the name of a single factor or a vector of names of factors to index the variable to be summarized. Specified via argument <INDEX.names> of ds.tapply.assign function |
| FUN.name, | the name of one of the allowable summarizing functions to be applied. Specified via argument <FUN.name> of ds.tapply.assign function. |

### Details

see details for ds.tapply.assign function

### Value

an array of the summarized values created by the tapplyDS.assign function. This array is written as a newobj on the serverside. It has the same number of dimensions as INDEX.

### Author(s)

Paul Burton, Demetris Avraam for DataSHIELD Development Team

---

## testObjExistsDS *testObjExistsDS*

---

### Description

The serverside function called by ds.testObjExists

### Usage

```
testObjExistsDS(test.obj.name = NULL)
```

### Arguments

| | |
|---|---|
| test.obj.name | a clientside provided character string specifying the variable whose presence is to be tested in each data source |

### Details

Tests whether a given object exists in all sources. It is called at the end of all recently written assign functions to check the new (assigned) object has been created in all sources

### Author(s)

Burton PR

unListDS *unListDS a serverside aggregate function called by ds.unList*

## Description

Coerces an R object back from a list towards the class it was before being coerced to a list

## Usage

```
unListDS(x.name, recursive, newobj)
```

## Arguments

| | |
|---|---|
| x.name | the name of the input object to be coerced back from class list. It must be specified in inverted commas. But this argument is usually specified directly by <x.name> argument of the clientside function ds.unList |
| recursive | argument required for native R unlist function - see native R help for unlist function |
| newobj | is the object hard assigned '«-' to be the output of the function written to the serverside |

## Details

Unlike most other class coercing functions this is an aggregate function rather than an assign function. This is because the datashield.assign function in opal deals specially with a created object (newobj) if it is of class list. Reconfiguring the function as an aggregate function works around this problem. This aggregate function is based on the native R function unlist and so additional information can be found in the help for unlist. When an object is coerced to a list, depending on the class of the original object some information may be lost. Thus, for example, when a data.frame is coerced to a list information that underpins the structure of the data.frame is lost and when it is subject to the function ds.unlist it is returned to a simpler class than data.frame eg 'numeric' (basically a numeric vector containing all of the original data in all variables in the data.frame but with no structure). If you wish to reconstruct the original data.frame you therefore need to specify this structure again e.g. the column names etc

## Value

the object specified by the <newobj> argument (or its default name <x.name>.mat) which is written to the serverside. In addition, two validity messages are returned. The first confirms an output object has been created, the second states its class.

## Author(s)

Amadou Gaye, Paul Burton for DataSHIELD Development Team

varDS *Computes the variance of vector*

## Description

Calculates the variance.

## Usage

```
varDS(xvect)
```

## Arguments

| | |
|---|---|
| xvect | a vector |

## Details

if the length of input vector is less than the set filter a missing value is returned.

## Value

a list, with the sum of the input variable, the sum of squares of the input variable, the number of missing values, the number of valid values, the number of total lenght of the variable, and a study message indicating whether the number of valid is less than the disclosure threshold

## Author(s)

Amadou Gaye, Demetris Avraam, for DataSHIELD Development Team

# Index