Package ‘DSLite’

May 18, 2020

Type Package

Version 1.1.0

Title 'DataSHIELD' Implementation on Local Datasets

Depends R (>= 3.5.0), DSI (>= 1.1), methods, R6

Suggests resourcer, knitr, testthat, rmarkdown

Description 'DataSHIELD' is an infrastructure and series of R packages that enables the remote and 'non-disclosive' analysis of sensitive research data. This 'DataSHIELD Interface' implementation is for analyzing datasets living in the current R session. The purpose of this is primarily for lightweight 'DataSHIELD' analysis package development.

License LGPL (>= 2.1)

URL http://www.datashield.ac.uk https://doi.org/10.1093/ije/dyu188

BugReports https://github.com/datashield/DSLite

RoxygenNote 7.1.0

VignetteBuilder knitr

Encoding UTF-8

Collate 'DSLiteDriver.R' 'DSLiteConnection.R' 'DSLiteResult.R' 'DSLiteServer.R' 'data.cnsim.R' 'data.dasim.R' 'data.discordant.R' 'data.survival.R' 'data.testing.dataset.R' 'defaultDSConfiguration.R' 'getDSLiteData.R' 'setupCNSIMTest.R' 'setupDASIMTest.R' 'setupDATASETTest.R' 'setupDISCORDANTTest.R' 'setupDSLiteServer.R' 'setupSURVIVALTest.R'

NeedsCompilation no

Author Yannick Marcon [aut, cre] (<https://orcid.org/0000-0003-0138-2023>)

Maintainer Yannick Marcon <yannick.marcon@obiba.org>

Repository CRAN

Date/Publication 2020-05-18 13:40:02 UTC

R topics documented:

CNSIM1 *Simulated dataset CNSIM 1*

Description

Simulated dataset CNSIM 1, in a data.frame with 2163 observations of 11 harmonized variables. The CNSIM dataset contains synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. This dataset does contain some NA values.

Details

Variable	**Description**	**Type**	**N**
LAB_TSC	Total Serum Cholesterol	numeric	m
LAB_TRIG	Triglycerides	numeric	m
LAB_HDL	HDL Cholesterol	numeric	m
LAB_GLUC_ADJUSTED	Non-Fasting Glucose	numeric	m
PM_BMI_CONTINUOUS	Body Mass Index (continuous)	numeric	kg
DIS_CVA	History of Stroke	factor	0
MEDI_LPD	Current Use of Lipid Lowering Medication (from categorical assessment item)	factor	0
DIS_DIAB	History of Diabetes	factor	0
DIS_AMI	History of Myocardial Infarction	factor	0
GENDER	Gender	factor	0
PM_BMI_CATEGORICAL	Body Mass Index (categorical)	factor	1

CNSIM2 *Simulated dataset CNSIM 2*

Description

Simulated dataset CNSIM 1, in a data.frame with 3088 observations of 11 harmonized variables variables. The CNSIM dataset contains synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. This dataset does contain some NA values.

Details

Variable	**Description**	**Type**	**N**
LAB_TSC	Total Serum Cholesterol	numeric	m
LAB_TRIG	Triglycerides	numeric	m
LAB_HDL	HDL Cholesterol	numeric	m
LAB_GLUC_ADJUSTED	Non-Fasting Glucose	numeric	m
PM_BMI_CONTINUOUS	Body Mass Index (continuous)	numeric	kg
DIS_CVA	History of Stroke	factor	0
MEDI_LPD	Current Use of Lipid Lowering Medication (from categorical assessment item)	factor	0
DIS_DIAB	History of Diabetes	factor	0
DIS_AMI	History of Myocardial Infarction	factor	0
GENDER	Gender	factor	0
PM_BMI_CATEGORICAL	Body Mass Index (categorical)	factor	1

`CNSIM3` *Simulated dataset CNSIM 3*

Description

Simulated dataset CNSIM 1, in a data.frame with 4128 observations of 11 harmonized variables variables. The CNSIM dataset contains synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. This dataset does contain some NA values.

Details

Variable	**Description**	**Type**	**N**
LAB_TSC	Total Serum Cholesterol	numeric	m
LAB_TRIG	Triglycerides	numeric	m
LAB_HDL	HDL Cholesterol	numeric	m
LAB_GLUC_ADJUSTED	Non-Fasting Glucose	numeric	m
PM_BMI_CONTINUOUS	Body Mass Index (continuous)	numeric	kg
DIS_CVA	History of Stroke	factor	0
MEDI_LPD	Current Use of Lipid Lowering Medication (from categorical assessment item)	factor	0
DIS_DIAB	History of Diabetes	factor	0
DIS_AMI	History of Myocardial Infarction	factor	0
GENDER	Gender	factor	0
PM_BMI_CATEGORICAL	Body Mass Index (categorical)	factor	1

DASIM1 *Simulated dataset DASIM 1*

Description

Simulated dataset DASIM 1, in a data.frame with 10000 observations of 10 harmonized variables. The DASIM dataset contains synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. This dataset does not contain some NA values.

Details

Variable	**Description**	**Type**	**Note**
LAB_TSC	Total Serum Cholesterol	numeric	mmol/L
LAB_TRIG	Triglycerides	numeric	mmol/L
LAB_HDL	HDL Cholesterol	numeric	mmol/L
LAB_GLUC_FASTING	Fasting Glucose	numeric	mmol/L
PM_BMI_CONTINUOUS	Body Mass Index (continuous)	numeric	kg/m2
DIS_CVA	History of Stroke	factor	0 = Never had stroke, 1 = Has had stroke
DIS_DIAB	History of Diabetes	factor	0 = Never had diabetes, 1 = Has had diabetes
DIS_AMI	History of Myocardial Infarction	factor	0 = Never had myocardial infarction, 1 = Has ha
GENDER	Gender	factor	0 = Female, 1 = Male
PM_BMI_CATEGORICAL	Body Mass Index (categorical)	factor	1 = Less than 25 kg/m2, 2 = 25 to 30 kg/m2, 3 =

DASIM2 *Simulated dataset DASIM 2*

Description

Simulated dataset DASIM 2, in a data.frame with 10000 observations of 10 harmonized variables. The DASIM dataset contains synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. This dataset does not contain some NA values.

Details

Variable	**Description**	**Type**	**Note**
LAB_TSC	Total Serum Cholesterol	numeric	mmol/L
LAB_TRIG	Triglycerides	numeric	mmol/L
LAB_HDL	HDL Cholesterol	numeric	mmol/L
LAB_GLUC_FASTING	Fasting Glucose	numeric	mmol/L

PM_BMI_CONTINUOUS	Body Mass Index (continuous)	numeric	kg/m2
DIS_CVA	History of Stroke	factor	0 = Never had stroke, 1 = Has had stroke
DIS_DIAB	History of Diabetes	factor	0 = Never had diabetes, 1 = Has had diabetes
DIS_AMI	History of Myocardial Infarction	factor	0 = Never had myocardial infarction, 1 = Has ha
GENDER	Gender	factor	0 = Female, 1 = Male
PM_BMI_CATEGORICAL	Body Mass Index (categorical)	factor	1 = Less than 25 kg/m2, 2 = 25 to 30 kg/m2, 3 =

DASIM3 *Simulated dataset DASIM 3*

Description

Simulated dataset DASIM 3, in a data.frame with 10000 observations of 10 harmonized variables. The DASIM dataset contains synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. This dataset does not contain some NA values.

Details

Variable	**Description**	**Type**	**Note**
LAB_TSC	Total Serum Cholesterol	numeric	mmol/L
LAB_TRIG	Triglycerides	numeric	mmol/L
LAB_HDL	HDL Cholesterol	numeric	mmol/L
LAB_GLUC_FASTING	Fasting Glucose	numeric	mmol/L
PM_BMI_CONTINUOUS	Body Mass Index (continuous)	numeric	kg/m2
DIS_CVA	History of Stroke	factor	0 = Never had stroke, 1 = Has had stroke
DIS_DIAB	History of Diabetes	factor	0 = Never had diabetes, 1 = Has had diabetes
DIS_AMI	History of Myocardial Infarction	factor	0 = Never had myocardial infarction, 1 = Has ha
GENDER	Gender	factor	0 = Female, 1 = Male
PM_BMI_CATEGORICAL	Body Mass Index (categorical)	factor	1 = Less than 25 kg/m2, 2 = 25 to 30 kg/m2, 3 =

defaultDSConfiguration *Default DataSHIELD configuration*

Description

Find the R packages that have DataSHIELD server configuration information in them and extract this information in a data frame of aggregation/assignment methods and a named list of R options. The DataSHIELD packages can be filtered by specifying explicitly the package names to be included or excluded. The package exclusion prevails over the inclusion.

Usage

```
defaultDSConfiguration(include = NULL, exclude = NULL)
```

Arguments

`include`	Character vector of package names to be explicitly included. If NULL, do not filter packages.
`exclude`	Character vector of package names to be explicitly excluded. If NULL, do not filter packages.

Examples

```
{
# detect DS packages
defaultDSConfiguration()
# exclude a DS package
defaultDSConfiguration(exclude="dsBase")
# include explicitely some DS packages
defaultDSConfiguration(include=c("dsBase", "dsOmics"))
}
```

`DISCORDANT_STUDY1` *Simulated dataset DISCORDANT 1*

Description

Simulated dataset DISCORDANT 1, in a data.frame with 12 observations of 2 discordant variables.

Details

Variable	Description	Type
A	Dummy data	integer
B	Dummy data	integer

`DISCORDANT_STUDY2` *Simulated dataset DISCORDANT 2*

Description

Simulated dataset DISCORDANT 2, in a data.frame with 12 observations of 2 discordant variables.

Details

Variable	**Description**	**Type**
A	Dummy data	integer
C	Dummy data	integer

`DISCORDANT_STUDY3` *Simulated dataset DISCORDANT 3*

Description

Simulated dataset DISCORDANT 3, in a data.frame with 12 observations of 2 discordant variables.

Details

Variable	**Description**	**Type**
B	Dummy data	integer
C	Dummy data	integer

`dsAggregate,DSLiteConnection-method` *Aggregate data*

Description

Aggregate some data from the DataSHIELD R session using a valid R expression. The aggregation expression must satisfy the data repository's DataSHIELD configuration.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsAggregate(conn, expr, async = TRUE)
```

Arguments

`conn`	`DSLiteConnection-class` object.
`expr`	Expression to evaluate.
`async`	Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

`dsAssignExpr,DSLiteConnection-method`
Assign the result of an expression

Description

Assign a result of the execution of an expression in the DataSHIELD R session.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsAssignExpr(conn, symbol, expr, async = TRUE)
```

Arguments

`conn`	`DSLiteConnection-class` object.
`symbol`	Name of the R symbol.
`expr`	A R expression with allowed assign functions calls.
`async`	Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

Value

A `DSLiteResult-class` object.

dsAssignResource,DSLiteConnection-method

Assign a resource

Description

Assign a DSLite resource in the DataSHIELD R session.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsAssignResource(conn, symbol, resource, async = TRUE)
```

Arguments

conn	DSLiteConnection-class object.
symbol	Name of the R symbol.
resource	Fully qualified name of a resource object living in the DSLite server.
async	Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

Value

A DSLiteResult-class object.

dsAssignTable,DSLiteConnection-method

Assign a table

Description

Assign a DSLite dataset in the DataSHIELD R session.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsAssignTable(
  conn,
  symbol,
  table,
  variables = NULL,
  missings = FALSE,
  identifiers = NULL,
  id.name = NULL,
  async = TRUE
)
```

Arguments

`conn`	`DSLiteConnection-class` object.
`symbol`	Name of the R symbol.
`table`	Fully qualified name of a dataset living in the DSLite server.
`variables`	List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refere to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
`missings`	If TRUE, missing values will be pushed from Opal to R, default is FALSE. Ignored if value is an R expression.
`identifiers`	Name of the identifiers mapping to use when assigning entities to R (currently NOT supported by DSLite).
`id.name`	Name of the column that will contain the entity identifiers. If not specified, the identifiers will be the data frame row names. When specified this column can be used to perform joins between data frames.
`async`	Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

Value

A `DSLiteResult-class` object.

`dsConnect,DSLiteDriver-method`
Connect to a DSLite server

Description

Connect to a DSLite server, with provided datasets symbol names.

Usage

```
## S4 method for signature 'DSLiteDriver'
dsConnect(drv, name, url, restore = NULL, ...)
```

Arguments

`drv`	`DSLiteDriver-class` class object.
`name`	Name of the connection, which must be unique among all the DataSHIELD connections.
`url`	A R symbol that refers to a DSLiteServer object that holds the datasets of interest. The option "datashield.env" can be used to specify where to search for this symbol value. If not specified, the environment is the global one.
`restore`	Workspace name to be restored in the newly created DataSHIELD R session.
`...`	Unused, needed for compatibility with generic.

Value

A DSLiteConnection-class object.

---

dsDisconnect,DSLiteConnection-method

Disconnect from a DSLite server

Description

Save the session in a local file if requested.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsDisconnect(conn, save = NULL)
```

Arguments

conn	DSLiteConnection-class class object
save	Save the DataSHIELD R session with provided ID (must be a character string).

---

dsFetch,DSLiteResult-method

Fetch the result

Description

Fetch the DataSHIELD operation result.

Usage

```
## S4 method for signature 'DSLiteResult'
dsFetch(res)
```

Arguments

res	DSLiteResult-class object.

Value

TRUE if table exists.

`dsGetInfo,DSLiteResult-method` *Get result info*

Description

Get the information about a command (if still available).

Usage

```
## S4 method for signature 'DSLiteResult'
dsGetInfo(dsObj, ...)
```

Arguments

`dsObj`	`DSLiteResult-class` class object
`...`	Unused, needed for compatibility with generic.

Value

The result information, including its status.

`dsHasResource,DSLiteConnection-method` *Verify DSLite server resource*

Description

Verify resource exists and can be accessible for performing DataSHIELD operations.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsHasResource(conn, resource)
```

Arguments

`conn`	`DSLiteConnection-class` class object.
`resource`	The fully qualified name of the resource.

Value

TRUE if dataset exists.

dsHasTable,DSLiteConnection-method

Verify DSLite server dataset

Description

Verify dataset exists and can be accessible for performing DataSHIELD operations.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsHasTable(conn, table)
```

Arguments

`conn`	`DSLiteConnection-class` class object.
`table`	The fully qualified name of the dataset.

Value

TRUE if dataset exists.

dsIsAsync,DSLiteConnection-method

DSLite asynchronous support

Description

No asynchronicity on any DataSHIELD operations.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsIsAsync(conn)
```

Arguments

`conn`	`DSLiteConnection-class` class object

Value

The named list of logicals detailing the asynchronicity support.

dsListMethods,DSLiteConnection-method

List methods

Description

List methods defined in the DataSHIELD configuration.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsListMethods(conn, type = "aggregate")
```

Arguments

conn	DSLiteConnection-class class object
type	Type of the method: "aggregate" (default) or "assign".

Value

A data frame.

dsListPackages,DSLiteConnection-method

List packages

Description

List packages defined in the DataSHIELD configuration.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsListPackages(conn)
```

Arguments

conn	DSLiteConnection-class class object

Value

A data frame.

dsListResources,DSLiteConnection-method

List DSLite server resources

Description

List resource names living in the DSLite server for performing DataSHIELD operations.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsListResources(conn)
```

Arguments

conn	DSLiteConnection-class class object

Value

The fully qualified names of the resources.

dsListSymbols,DSLiteConnection-method

List R symbols

Description

List symbols living in the DataSHIELD R session.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsListSymbols(conn)
```

Arguments

conn	DSLiteConnection-class class object

Value

A character vector.

dsListTables,DSLiteConnection-method
List DSLite server datasets

Description

List dataset names living in the DSLite server for performing DataSHIELD operations.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsListTables(conn)
```

Arguments

conn	DSLiteConnection-class class object

Value

The fully qualified names of the tables.

dsListWorkspaces,DSLiteConnection-method
List workspaces

Description

List workspaces saved in the data repository.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsListWorkspaces(conn)
```

Arguments

conn	DSLiteConnection-class class object

Value

A data frame.

`DSLite` *Create a DSLite driver*

Description

Convenient function for creating a DSLiteDriver object.

Usage

```
DSLite()
```

`DSLiteServer` *Lightweight DataSHIELD server-side component*

Description

DSLiteServer mimics a DataSHIELD server by holding datasets and exposing DataSHIELD-like functions: aggregate and assign. A DataSHIELD session is a R environment where the assignment and the operations happen.

Methods

Public methods:

- `DSLiteServer$new()`
- `DSLiteServer$config()`
- `DSLiteServer$strict()`
- `DSLiteServer$home()`
- `DSLiteServer$workspaces()`
- `DSLiteServer$workspace_save()`
- `DSLiteServer$workspace_rm()`
- `DSLiteServer$aggregateMethods()`
- `DSLiteServer$aggregateMethod()`
- `DSLiteServer$assignMethods()`
- `DSLiteServer$assignMethod()`
- `DSLiteServer$options()`
- `DSLiteServer$option()`
- `DSLiteServer$newSession()`
- `DSLiteServer$hasSession()`
- `DSLiteServer$getSession()`
- `DSLiteServer$getSessionIds()`
- `DSLiteServer$getSessionData()`
- `DSLiteServer$closeSession()`

- `DSLiteServer$tableNames()`
- `DSLiteServer$hasTable()`
- `DSLiteServer$resourceNames()`
- `DSLiteServer$hasResource()`
- `DSLiteServer$symbols()`
- `DSLiteServer$symbol_rm()`
- `DSLiteServer$assignTable()`
- `DSLiteServer$assignResource()`
- `DSLiteServer$assignExpr()`
- `DSLiteServer$aggregate()`
- `DSLiteServer$clone()`

Method `new()`: Create new DSLiteServer instance. See defaultDSConfiguration function for including or excluding packages when discovering the DataSHIELD configuration from the DataSHIELD server-side packages (meta-data from the DESCRIPTION files).

Usage:

```
DSLiteServer$new(
  tables = list(),
  resources = list(),
  config = DSLite::defaultDSConfiguration(),
  strict = TRUE,
  home = file.path(tempdir(), ".dslite")
)
```

Arguments:

`tables` A named list of data.frames representing the harmonized tables.

`resources` A named list of `resourcer::Resource` objects representing accessible data or computation resources.

`config` The DataSHIELD configuration. Default is to discover it from the DataSHIELD server-side R packages.

`strict` Logical to specify whether the DataSHIELD configuration must be strictly applied. Default is TRUE.

`home` Folder location where are located the session work directory and where to read and dump workspace images. Default is in a hidden folder of the R session's temporary directory.

Returns: A DSLiteServer object

Method `config()`: Get or set the DataSHIELD configuration.

Usage:

```
DSLiteServer$config(value)
```

Arguments:

`value` The DataSHIELD configuration: aggregate/assign methods in data frames and a named list of options.

Returns: The DataSHIELD configuration, if no parameter is provided.

Method `strict()`: Get or set the level of strictness (stop when function call is not configured)

Usage:

```
DSLiteServer$strict(value)
```

Arguments:

`value` The `strict` logical field.

Returns: The strict field if no parameter is provided.

Method `home()`**:** Get or set the home folder location where are located the session work directories and where to read and dump workspace images.

Usage:

```
DSLiteServer$home(value)
```

Arguments:

`value` The path to the home folder.

Returns: The home folder path if no parameter is provided.

Method `workspaces()`**:** List the saved workspaces in the `home` folder.

Usage:

```
DSLiteServer$workspaces(prefix = NULL)
```

Arguments:

`prefix` Filter workspaces starting with provided prefix (optional).

Method `workspace_save()`**:** Save the session's workspace image identified by the `sid` identifier with the provided `name` in the `home` folder.

Usage:

```
DSLiteServer$workspace_save(sid, name)
```

Arguments:

`sid,` Session ID

`name` The name to be given to the workspace's image.

Method `workspace_rm()`**:** Remove the workspace image with the provided `name` from the `home` folder.

Usage:

```
DSLiteServer$workspace_rm(name)
```

Arguments:

`name` The name of the workspace.

Method `aggregateMethods()`**:** Get or set the aggregate methods.

Usage:

```
DSLiteServer$aggregateMethods(value)
```

Arguments:

`value` A `data.frame` with columns: `name` (the client function call), `value` (the translated server call), `package` (relevant when extracted from a DataSHIELD server-side package), `version` (relevant when extracted from a DataSHIELD server-side package), `type` ("aggregate"), `class` ("function" for package functions or "script" for custom scripts).

Returns: The aggregate methods when no parameter is provided.

Method `aggregateMethod()`**:** Get or set an aggregate method.

Usage:

```
DSLiteServer$aggregateMethod(name, value)
```

Arguments:

`name` The client function call.

`value` The translated server call: either a package function reference or function expression. Remove the method when `NULL`.

Returns: The aggregate method when no `value` parameter is provided.

Method `assignMethods()`**:** Get or set the assign methods.

Usage:

```
DSLiteServer$assignMethods(value)
```

Arguments:

`value` A `data.frame` with columns: `name` (the client function call), `value` (the translated server call), `package` (relevant when extracted from a DataSHIELD server-side package), `version` (relevant when extracted from a DataSHIELD server-side package), `type` ("assign"), `class` ("function" for package functions or "script" for custom scripts).

Returns: The assign methods when no parameter is provided.

Method `assignMethod()`**:** Get or set an assign method.

Usage:

```
DSLiteServer$assignMethod(name, value)
```

Arguments:

`name` The client function call

`value` The translated server call: either a package function reference or function expression. Remove the method when `NULL`.

Returns: The assign method when no `value` parameter is provided.

Method `options()`**:** Get or set the DataSHIELD R options that are applied when a new DataSHIELD session is started.

Usage:

```
DSLiteServer$options(value)
```

Arguments:

`value` A named list of options.

Returns: The R options when no parameter is provided.

Method `option()`**:** Get or set a R option.

Usage:

```
DSLiteServer$option(key, value)
```

Arguments:

`key` The R option's name.

`value` The R option's value. Remove the option when `NULL`.

Returns: The R option's value when only `key` parameter is provided.

Method `newSession()`**:** Create a new DataSHIELD session (contained execution environment), apply options that are defined in the DataSHIELD configuration and restore workspace image if `restore` workspace name argument is provided.

Usage:

```
DSLiteServer$newSession(restore = NULL)
```

Arguments:

`restore` The workspace image to be restored (optional).

Method `hasSession()`**:** Check a DataSHIELD session is alive.

Usage:

```
DSLiteServer$hasSession(sid)
```

Arguments:

`sid` The session ID.

Method `getSession()`**:** Get the DataSHIELD session's environment.

Usage:

```
DSLiteServer$getSession(sid)
```

Arguments:

`sid` The session ID.

Method `getSessionIds()`**:** Get the DataSHIELD session IDs.

Usage:

```
DSLiteServer$getSessionIds()
```

Method `getSessionData()`**:** Get the symbol value from the DataSHIELD session's environment.

Usage:

```
DSLiteServer$getSessionData(sid, symbol)
```

Arguments:

`sid` The session ID.

`symbol` The symbol name.

Method `closeSession()`**:** Destroy DataSHIELD session and save workspace image if `save` workspace name argument is provided.

Usage:

```
DSLiteServer$closeSession(sid, save = NULL)
```

Arguments:

`sid` The session ID.

`save` The name of the workspace image to be saved (optional).

Method `tableNames()`**:** List the names of the tables that can be assigned.

Usage:

```
DSLiteServer$tableNames()
```

Method `hasTable()`**:** Check a table exists.

Usage:

```
DSLiteServer$hasTable(name)
```

Arguments:

`name` The table name to be looked for.

Method `resourceNames()`**:** List the names of the resources (`resourcer::Resource` objects) that can be assigned.

Usage:

```
DSLiteServer$resourceNames()
```

Method `hasResource()`**:** Check a resource (`resourcer::Resource` object) exists.

Usage:

```
DSLiteServer$hasResource(name)
```

Arguments:

`name` The resource name to be looked for.

Method `symbols()`**:** List the symbols living in a DataSHIELD session.

Usage:

```
DSLiteServer$symbols(sid)
```

Arguments:

`sid` The session ID.

Method `symbol_rm()`**:** Remove a symbol from a DataSHIELD session.

Usage:

```
DSLiteServer$symbol_rm(sid, name)
```

Arguments:

`sid` The session ID.

`name` The symbol name.

Method `assignTable()`**:** Assign a table to a symbol in a DataSHIELD session. Filter table columns with the variables names provided.

Usage:

```
DSLiteServer$assignTable(sid, symbol, name, variables = NULL, id.name = NULL)
```

Arguments:

`sid` The session ID.

`symbol` The symbol to be assigned.

`name` The table's name.

`variables` The variable names to be filtered in (optional).

`id.name` The column name to be used for the entity's identifier (optional).

Method `assignResource()`**:** Assign a resource as a `resourcer::ResourceClient` object to a symbol in a DataSHIELD session.

Usage:

```
DSLiteServer$assignResource(sid, symbol, name)
```

Arguments:

`sid` The session ID.

`symbol` The symbol name.

`name` The name of the resource.

Method `assignExpr()`**:** Evaluate an assignment expression in a DataSHIELD session.

Usage:

```
DSLiteServer$assignExpr(sid, symbol, expr)
```

Arguments:

`sid` The session ID.

`symbol` The symbol name.

`expr` The R expression to evaluate.

Method `aggregate()`**:** Evaluate an aggregate expression in a DataSHIELD session.

Usage:

```
DSLiteServer$aggregate(sid, expr)
```

Arguments:

`sid` The session ID.

`expr` The R expression to evaluate.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DSLiteServer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other server-side items: `newDSLiteServer()`

---

dsRmSymbol,DSLiteConnection-method

Remove a R symbol

Description

Remoe a symbol living in the DataSHIELD R session.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsRmSymbol(conn, symbol)
```

Arguments

conn	DSLiteConnection-class class object
symbol	Name of the R symbol.

---

dsRmWorkspace,DSLiteConnection-method

Remove a workspace

Description

Remove a workspace on the data repository.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsRmWorkspace(conn, name)
```

Arguments

conn	DSLiteConnection-class class object
name	Name of the workspace.

dsSaveWorkspace,DSLiteConnection-method

Save workspace

Description

Save workspace on the data repository.

Usage

```
## S4 method for signature 'DSLiteConnection'
dsSaveWorkspace(conn, name)
```

Arguments

`conn`	`DSLiteConnection-class` class object
`name`	Name of the workspace.

getDSLiteData

Get data value from DSLite connection(s)

Description

Get the data value corresponding to the variable with the symbol name from the DSLiteServer associated to the `DSConnection-class` object(s). Can be useful when developping a DataSHIELD package.

Usage

```
getDSLiteData(conns, symbol)
```

Arguments

`conns`	`DSConnection-class` object or a list of `DSConnection-class`s.
`symbol`	Symbol name identifying the variable in the DSLiteServer's "server-side" environment(s).

Value

The data value or a list of values depending on the connections parameter. The value is NA when the connection object is not of class `DSLiteConnection-class`.

Examples

```
{
# DataSHIELD login
logindata <- setupCNSIMTest()
conns <- datashield.login(logindata, assign=TRUE)
# retrieve symbol D value from each DataSHIELD connections
getDSLiteData(conns, "D")
# retrieve symbol D value from a specific DataSHIELD connection
getDSLiteData(conns$sim1, "D")
}
```

`logindata.dslite.cnsim` *DataSHIELD login data for the CNSIM simulated datasets*

Description

DataSHIELD login data.frame for connecting with CNSIM datasets. The CNSIM datasets contain synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. These datasets do contain some NA values.

Details

Field	**Description**	**Type**	**Note**
server	Server/study name	char	
url	Server/study URL	char	DSLiteServer instance symbol name
user	User name	char	Always empty for DSLiteServer
password	User password	char	Always empty for DSLiteServer
table	Table unique name	char	As registered in the DSLiteServer
options	Connection options	char	Always empty for DSLiteServer
driver	Connection driver	char	DSLiteServer

`logindata.dslite.dasim` *DataSHIELD login data for the DASIM simulated datasets*

Description

DataSHIELD login data.frame for connecting with DASIM datasets. The DASIM datasets contain synthetic data based on a model derived from the participants of the 1958 Birth Cohort, as part of the obesity methodological development project. These datasets do not contain some NA values.

Details

Field	Description	Type	Note
server	Server/study name	char	
url	Server/study URL	char	DSLiteServer instance symbol name
user	User name	char	Always empty for DSLiteServer
password	User password	char	Always empty for DSLiteServer
table	Table unique name	char	As registered in the DSLiteServer
options	Connection options	char	Always empty for DSLiteServer
driver	Connection driver	char	DSLiteServer

`logindata.dslite.discordant`

DataSHIELD login data for the DISCORDANT simulated datasets

Description

DataSHIELD login data.frame for connecting with DISCORDANT datasets which purpose is to test datasets that are NOT harmonized.

Details

Field	Description	Type	Note
server	Server/study name	char	
url	Server/study URL	char	DSLiteServer instance symbol name
user	User name	char	Always empty for DSLiteServer
password	User password	char	Always empty for DSLiteServer
table	Table unique name	char	As registered in the DSLiteServer
options	Connection options	char	Always empty for DSLiteServer
driver	Connection driver	char	DSLiteServer

`logindata.dslite.survival.expand_with_missing`

DataSHIELD login data for the simulated survival expand-with-missing datasets

---

Description

DataSHIELD login data.frame for connecting with SURVIVAL datasets which purpose is to perform survival tests. The datasets contain synthetic data based on a simulated survival model, including a censoring indicator.

Details

Field	**Description**	**Type**	**Note**
server	Server/study name	char	
url	Server/study URL	char	DSLiteServer instance symbol name
user	User name	char	Always empty for DSLiteServer
password	User password	char	Always empty for DSLiteServer
table	Table unique name	char	As registered in the DSLiteServer
options	Connection options	char	Always empty for DSLiteServer
driver	Connection driver	char	DSLiteServer

`logindata.dslite.testing.dataset`

DataSHIELD login data for the TESTING.DATASET simulated datasets

---

Description

DataSHIELD login data.frame for connecting with TESTING.DATASET datasets which purpose is to evaluate each base data types.

Details

Field	**Description**	**Type**	**Note**
server	Server/study name	char	
url	Server/study URL	char	DSLiteServer instance symbol name
user	User name	char	Always empty for DSLiteServer
password	User password	char	Always empty for DSLiteServer
table	Table unique name	char	As registered in the DSLiteServer
options	Connection options	char	Always empty for DSLiteServer
driver	Connection driver	char	DSLiteServer

newDSLiteServer *Create a new DSLite server*

Description

Shortcut function to create a new `DSLiteServer` instance.

Usage

```
newDSLiteServer(
  tables = list(),
  resources = list(),
  config = DSLite::defaultDSConfiguration(),
  strict = TRUE,
  home = file.path(tempdir(), ".dslite")
)
```

Arguments

`tables`	A named list of data.frames representing the harmonized tables.
`resources`	A named list of `resourcer::Resource` objects representing accessible data or computation resources.
`config`	The DataSHIELD configuration. Default is to discover it from the DataSHIELD server-side R packages. See defaultDSConfiguration function for including or excluding packages when discovering the DataSHIELD configuration from the DataSHIELD server-side packages (meta-data from the DESCRIPTION files).
`strict`	Logical to specify whether the DataSHIELD configuration must be strictly applied. Default is TRUE.
`home`	Folder location where are located the session work directory and where to read and dump workspace images. Default is in a hidden folder of the R session's temporary directory.

See Also

Other server-side items: `DSLiteServer`

setupCNSIMTest *Setup a test environment based on the CNSIM simulated datasets*

Description

Load the CNSIM datasets, the corresponding login data object, instanciate a new DSLiteServer hosting these datasets and verify that the required DataSHIELD server-side packages are installed.

Usage

```
setupCNSIMTest(packages = c(), env = parent.frame())
```

Arguments

`packages`	DataSHIELD server-side packages which local installation must be verified so that the DSLiteServer can auto-configure itself and can execute the DataSHIELD operations. Default is none.
`env`	The environment where DataSHIELD objects should be looked for: the DSLite-Server and the DSIConnection objects. Default is the Global environment.

Value

The login data for the datashield.login function.

See Also

Other setup functions: `setupDASIMTest()`, `setupDATASETTest()`, `setupDISCORDANTTest()`, `setupDSLiteServer()`, `setupSURVIVALTest()`

Examples

```
{
logindata <- setupCNSIMTest()
conns <- datashield.login(logindata, assign=TRUE)
# do DataSHIELD analysis
datashield.logout(conns)
}
```

`setupDASIMTest` *Setup a test environment based on the DASIM simulated datasets*

Description

Load the DASIM datasets, the corresponding login data object, instanciate a new DSLiteServer hosting these datasets and verify that the required DataSHIELD server-side packages are installed.

Usage

```
setupDASIMTest(packages = c(), env = parent.frame())
```

Arguments

`packages`	DataSHIELD server-side packages which local installation must be verified so that the DSLiteServer can auto-configure itself and can execute the DataSHIELD operations. Default is none.
`env`	The environment where DataSHIELD objects should be looked for: the DSLite-Server and the DSIConnection objects. Default is the Global environment.

Value

The login data for the datashield.login function.

See Also

Other setup functions: `setupCNSIMTest()`, `setupDATASETTest()`, `setupDISCORDANTTest()`, `setupDSLiteServer()`, `setupSURVIVALTest()`

Examples

```
{
logindata <- setupDASIMTest()
conns <- datashield.login(logindata, assign=TRUE)
# do DataSHIELD analysis
datashield.logout(conns)
}
```

`setupDATASETTest` *Setup a test environment based on the TESTING.DATASET simulated datasets*

Description

Load the TESTING.DATASET datasets, the corresponding login data object, instanciate a new DSLiteServer hosting these datasets and verify that the required DataSHIELD server-side packages are installed.

Usage

```
setupDATASETTest(packages = c(), env = parent.frame())
```

Arguments

`packages`	DataSHIELD server-side packages which local installation must be verified so that the DSLiteServer can auto-configure itself and can execute the DataSHIELD operations. Default is none.
`env`	The environment where DataSHIELD objects should be looked for: the DSLite-Server and the DSIConnection objects. Default is the Global environment.

Value

The login data for the datashield.login function.

See Also

Other setup functions: `setupCNSIMTest()`, `setupDASIMTest()`, `setupDISCORDANTTest()`, `setupDSLiteServer()`, `setupSURVIVALTest()`

Examples

```
{
logindata <- setupDATASETTest()
conns <- datashield.login(logindata, assign=TRUE)
# do DataSHIELD analysis
datashield.logout(conns)
}
```

`setupDISCORDANTTest` *Setup a test environment based on the DISCORDANT simulated datasets*

Description

Load the DISCORDANT datasets, the corresponding login data object, instanciate a new DSLite-Server hosting these datasets and verify that the required DataSHIELD server-side packages are installed.

Usage

```
setupDISCORDANTTest(packages = c(), env = parent.frame())
```

Arguments

`packages`	DataSHIELD server-side packages which local installation must be verified so that the DSLiteServer can auto-configure itself and can execute the DataSHIELD operations. Default is none.
`env`	The environment where DataSHIELD objects should be looked for: the DSLite-Server and the DSIConnection objects. Default is the Global environment.

Value

The login data for the datashield.login function.

See Also

Other setup functions: `setupCNSIMTest()`, `setupDASIMTest()`, `setupDATASETTest()`, `setupDSLiteServer()`, `setupSURVIVALTest()`

Examples

```
{
logindata <- setupDISCORDANTTest()
conns <- datashield.login(logindata, assign=TRUE)
# do DataSHIELD analysis
datashield.logout(conns)
}
```

setupDSLiteServer *Setup an environment based on named datasets and logindata*

Description

Load the provided datasets and the corresponding logindata object, instanciate a new DSLiteServer hosting these datasets, verifies that the required DataSHIELD server-side packages are installed. All the data structures are loaded by data which supports various formats (see data() documentation).

Usage

```
setupDSLiteServer(
  packages = c(),
  datasets,
  logindata,
  pkgs = NULL,
  dslite.server = NULL,
  env = parent.frame()
)
```

Arguments

`packages`	DataSHIELD server-side packages which local installation must be verified so that the DSLiteServer can auto-configure itself and can execute the DataSHIELD operations. Default is none.
`datasets`	Names of the datasets to be loaded using data.
`logindata`	Name of the login data object to be loaded using data.
`pkgs`	The package(s) to look in for datasets, default is all, then the 'data' subdirectory (if present) of the current working directory (same behavior as 'package' argument in data).
`dslite.server`	Symbol name to which the DSLiteServer should be assigned to. If not provided, the symbol name will be the first not null one specified in the 'url' column of the loaded login data.
`env`	The environment where DataSHIELD objects should be looked for: the DSLiteServer and the DSIConnection objects. Default is the Global environment.

Value

The login data for the datashield.login function.

See Also

Other setup functions: `setupCNSIMTest()`, `setupDASIMTest()`, `setupDATASETTest()`, `setupDISCORDANTTest()`, `setupSURVIVALTest()`

Examples

```
{
logindata <- setupDSLiteServer(
                 datasets = c("CNSIM1", "CNSIM2", "CNSIM3"),
                 logindata = "logindata.dslite.cnsim", pkgs = "DSLite",
                 dslite.server = "dslite.server")
conns <- datashield.login(logindata, assign=TRUE)
# do DataSHIELD analysis
datashield.logout(conns)
}
```

`setupSURVIVALTest` *Setup a test environment based on the SURVIVAL (EXPAND_WITH_MISSING) simulated datasets*

Description

Load the SURVIVAL (EXPAND_WITH_MISSING) datasets, the corresponding login data object, instanciate a new DSLiteServer hosting these datasets and verify that the required DataSHIELD server-side packages are installed.

Usage

```
setupSURVIVALTest(packages = c(), env = parent.frame())
```

Arguments

`packages`	DataSHIELD server-side packages which local installation must be verified so that the DSLiteServer can auto-configure itself and can execute the DataSHIELD operations. Default is none.
`env`	The environment where DataSHIELD objects should be looked for: the DSLite-Server and the DSIConnection objects. Default is the Global environment.

Value

The login data for the datashield.login function.

See Also

Other setup functions: `setupCNSIMTest()`, `setupDASIMTest()`, `setupDATASETTest()`, `setupDISCORDANTTest()`, `setupDSLiteServer()`

Examples

```
{
logindata <- setupSURVIVALTest()
conns <- datashield.login(logindata, assign=TRUE)
# do DataSHIELD analysis
datashield.logout(conns)
}
```

SURVIVAL.EXPAND_WITH_MISSING1 *Simulated survival expand-with-missing dataset 1*

Description

Simulated dataset SURVIVAL.EXPAND_WITH_MISSING 1, in a data.frame with 2060 observations of 12 harmonized variables. The dataset contains synthetic data based on a simulated survival model, including a censoring indicator.

Details

Variable	**Description**	**Type**	**Note**
id	Unique individual ID	integer	
study.id	Study ID	integer	
time.id	Time ID	integer	
starttime	Start of follow up	numeric	years
endtime	End of follow up	numeric	years
survtime	Survtime	numeric	years
cens	Censoring status	factor	0 = not censored, 1 = censored
age.60	Age centred at 60	numeric	
female	Gender	factor	0 = Male, 1 = Female
noise.56	Noise pollution centred at 56	numeric	dB
pm10.16	Particulate matter centred at 16	numeric	µg/m3
bmi.26	Body mass index centred at 26	numeric	kg/m2

SURVIVAL.EXPAND_WITH_MISSING2 *Simulated survival expand-with-missing dataset 2*

Description

Simulated dataset SURVIVAL.EXPAND_WITH_MISSING 2, in a data.frame with 1640 observations of 12 harmonized variables. The dataset contains synthetic data based on a simulated survival model, including a censoring indicator.

Details

Variable	**Description**	**Type**	**Note**
id	Unique individual ID	integer	
study.id	Study ID	integer	
time.id	Time ID	integer	
starttime	Start of follow up	numeric	years
endtime	End of follow up	numeric	years
survtime	Survtime	numeric	years
cens	Censoring status	factor	0 = not censored, 1 = censored
age.60	Age centred at 60	numeric	
female	Gender	factor	0 = Male, 1 = Female
noise.56	Noise pollution centred at 56	numeric	dB
pm10.16	Particulate matter centred at 16	numeric	µg/m3
bmi.26	Body mass index centred at 26	numeric	kg/m2

`SURVIVAL.EXPAND_WITH_MISSING3`

Simulated survival expand-with-missing dataset 3

Description

Simulated dataset SURVIVAL.EXPAND_WITH_MISSING 3, in a data.frame with 2688 observations of 12 harmonized variables. The dataset contains synthetic data based on a simulated survival model, including a censoring indicator.

Details

Variable	**Description**	**Type**	**Note**
id	Unique individual ID	integer	
study.id	Study ID	integer	
time.id	Time ID	integer	
starttime	Start of follow up	numeric	years
endtime	End of follow up	numeric	years
survtime	Survtime	numeric	years
cens	Censoring status	factor	0 = not censored, 1 = censored
age.60	Age centred at 60	numeric	

female	Gender	factor	0 = Male, 1 = Female
noise.56	Noise pollution centred at 56	numeric	dB
pm10.16	Particulate matter centred at 16	numeric	µg/m3
bmi.26	Body mass index centred at 26	numeric	kg/m2

`TESTING.DATASET1` *Simulated dataset TESTING.DATASET 1*

Description

Simulated dataset TESTING.DATASET 1, in a data.frame with 71 observations of 17 harmonized variables.

Details

Variable	**Description**	**Type**
ID	Dummy data	integer
CHARACTER	Dummy data	char
LOGICAL	Dummy data	logical
NA_VALUES	Dummy data	logical
NULL_VALUES	Dummy data	logical
INTEGER	Dummy data	integer
NON_NEGATIVE_INTEGER	Dummy data	integer
POSITIVE_INTEGER	Dummy data	integer
NEGATIVE_INTEGER	Dummy data	integer
NUMERIC	Dummy data	numeric
NON_NEGATIVE_NUMERIC	Dummy data	numeric
POSITIVE_NUMERIC	Dummy data	numeric
NEGATIVE_NUMERIC	Dummy data	numeric
FACTOR_CHARACTER	Dummy data	factor
FACTOR_INTEGER	Dummy data	factor
IDENTIFIER	Dummy data	integer
CATEGORY	Dummy data	integer

`TESTING.DATASET2` *Simulated dataset TESTING.DATASET 2*

Description

Simulated dataset TESTING.DATASET 2, in a data.frame with 71 observations of 17 harmonized variables.

Details

Variable	**Description**	**Type**
ID	Dummy data	integer
CHARACTER	Dummy data	char
LOGICAL	Dummy data	logical
NA_VALUES	Dummy data	logical
NULL_VALUES	Dummy data	logical
INTEGER	Dummy data	integer
NON_NEGATIVE_INTEGER	Dummy data	integer
POSITIVE_INTEGER	Dummy data	integer
NEGATIVE_INTEGER	Dummy data	integer
NUMERIC	Dummy data	numeric
NON_NEGATIVE_NUMERIC	Dummy data	numeric
POSITIVE_NUMERIC	Dummy data	numeric
NEGATIVE_NUMERIC	Dummy data	numeric
FACTOR_CHARACTER	Dummy data	factor
FACTOR_INTEGER	Dummy data	factor
IDENTIFIER	Dummy data	integer
CATEGORY	Dummy data	integer

`TESTING.DATASET3` *Simulated dataset TESTING.DATASET 3*

Description

Simulated dataset TESTING.DATASET 3, in a data.frame with 71 observations of 17 harmonized variables.

Details

Variable	**Description**	**Type**
ID	Dummy data	integer
CHARACTER	Dummy data	char
LOGICAL	Dummy data	logical
NA_VALUES	Dummy data	logical
NULL_VALUES	Dummy data	logical
INTEGER	Dummy data	integer

NON_NEGATIVE_INTEGER	Dummy data	integer
POSITIVE_INTEGER	Dummy data	integer
NEGATIVE_INTEGER	Dummy data	integer
NUMERIC	Dummy data	numeric
NON_NEGATIVE_NUMERIC	Dummy data	numeric
POSITIVE_NUMERIC	Dummy data	numeric
NEGATIVE_NUMERIC	Dummy data	numeric
FACTOR_CHARACTER	Dummy data	factor
FACTOR_INTEGER	Dummy data	factor
IDENTIFIER	Dummy data	integer
CATEGORY	Dummy data	integer

Index