Package ‘DSLite’

March 16, 2021

Type Package
Version 1.2.0
Title 'DataSHIELD' Implementation on Local Datasets
Depends R (>= 3.5.0), DSI (>= 1.2), 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.cnsm.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 2021-03-16 09:10:05 UTC
R topics documented:

CNSIM1 ....................................................... 3
CNSIM2 ....................................................... 3
CNSIM3 ....................................................... 4
DASIM1 ....................................................... 5
DASIM2 ....................................................... 5
DASIM3 ....................................................... 6
defaultDSConfiguration .................................. 6
DISCORDANT_STUDY1 ..................................... 7
DISCORDANT_STUDY2 ..................................... 8
DISCORDANT_STUDY3 ..................................... 8
dsAggregate, DSLiteConnection-method ............... 8
dsAssignExpr, DSLiteConnection-method ............... 9
dsAssignResource, DSLiteConnection-method .......... 10
dsAssignTable, DSLiteConnection-method .............. 10
dsConnect, DLSiteDriver-method ......................... 11
dsDisconnect, DLSiteConnection-method ............... 12
dsFetch, DLSiteResult-method ............................ 12
dsGetInfo, DLSiteResult-method .......................... 13
dsHasResource, DLSiteConnection-method .............. 13
dsHasTable, DLSiteConnection-method ................. 14
dsIsAsync, DLSiteConnection-method .................... 14
dsListMethods, DLSiteConnection-method .............. 15
dsListPackages, DLSiteConnection-method .......... 15
dsListResources, DLSiteConnection-method .......... 16
dsListSymbols, DLSiteConnection-method .......... 16
dsListTables, DLSiteConnection-method ............... 17
dsListWorkspaces, DLSiteConnection-method .......... 17
DSLite .......................................................... 18
DSLiteServer .................................................. 18
dsRmSymbol, DLSiteConnection-method ............... 25
dsRmWorkspace, DLSiteConnection-method .......... 25
dsSaveWorkspace, DLSiteConnection-method .......... 26
getDSLiteData ................................................. 26
logindata.dslite.cnsim .................................... 27
logindata.dslite.dasim .................................... 27
logindata.dslite.dsicordant ............................. 28
logindata.dslite.survival.expand_with_missing ........ 28
logindata.dslite.testing.dataset ....................... 29
newDSLiteServer ........................................... 30
setupCNSIMTest ........................................... 30
setupDASIMTest ........................................... 31
setupDATASETTest ......................................... 32
setupDISCORDANTTest ..................................... 33
setupDSLiteServer ......................................... 34
setupSURVIVALTest ....................................... 35
SURVIVAL.EXPAND_WITH_MISSING1 ....................... 36
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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB_TSC</td>
<td>Total Serum Cholesterol</td>
<td>numeric mmol/L</td>
<td></td>
</tr>
<tr>
<td>LAB_TRIG</td>
<td>Triglycerides</td>
<td>numeric mmol/L</td>
<td></td>
</tr>
<tr>
<td>LAB_HDL</td>
<td>HDL Cholesterol</td>
<td>numeric mmol/L</td>
<td></td>
</tr>
<tr>
<td>LAB_GLUC_ADJUSTED</td>
<td>Non-Fasting Glucose</td>
<td>numeric mmol/L</td>
<td></td>
</tr>
<tr>
<td>PM_BMI_CONTINUOUS</td>
<td>Body Mass Index (continuous)</td>
<td>numeric kg/m²</td>
<td></td>
</tr>
<tr>
<td>DIS_CVA</td>
<td>History of Stroke</td>
<td>factor 0 : 1</td>
<td></td>
</tr>
<tr>
<td>MEDI_LPD</td>
<td>Current Use of Lipid Lowering Medication</td>
<td>factor 0 : 1</td>
<td></td>
</tr>
<tr>
<td>DIS_DIAB</td>
<td>History of Diabetes</td>
<td>factor 0 : 1</td>
<td></td>
</tr>
<tr>
<td>DIS_AMI</td>
<td>History of Myocardial Infarction</td>
<td>factor 0 : 1</td>
<td></td>
</tr>
<tr>
<td>GENDER</td>
<td>Gender</td>
<td>factor 0 : 1</td>
<td></td>
</tr>
<tr>
<td>PM_BMI_CATEGORICAL</td>
<td>Body Mass Index (categorical)</td>
<td>factor 1 : 3</td>
<td></td>
</tr>
</tbody>
</table>

CNSIM2

*Simulated dataset CNSIM 2*

### Description

Simulated dataset CNSIM 1, in a data.frame with 3088 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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB_TSC</td>
<td>Total Serum Cholesterol</td>
<td>numeric</td>
<td>mmol/L</td>
</tr>
<tr>
<td>LAB_TRIG</td>
<td>Triglycerides</td>
<td>numeric</td>
<td>mmol/L</td>
</tr>
<tr>
<td>LAB_HDL</td>
<td>HDL Cholesterol</td>
<td>numeric</td>
<td>mmol/L</td>
</tr>
<tr>
<td>LAB_GLUC_ADJUSTED</td>
<td>Non-Fasting Glucose</td>
<td>numeric</td>
<td>mmol/L</td>
</tr>
<tr>
<td>PM_BMI_CONTINUOUS</td>
<td>Body Mass Index (continuous)</td>
<td>numeric</td>
<td>kg/m²</td>
</tr>
<tr>
<td>DIS_CVA</td>
<td>History of Stroke</td>
<td>factor</td>
<td>0 = Never had stroke, 1 = Has had stroke</td>
</tr>
<tr>
<td>MEDI_LPD</td>
<td>Current Use of Lipid Lowering Medication (from categorical assessment item)</td>
<td>factor</td>
<td>0 = Not currently using lipid lowering medication, 1 = Currently using lipid lowering medication</td>
</tr>
<tr>
<td>DIS_DIAB</td>
<td>History of Diabetes</td>
<td>factor</td>
<td>0 = Never had diabetes, 1 = Has had diabetes</td>
</tr>
<tr>
<td>DIS_AMI</td>
<td>History of Myocardial Infarction</td>
<td>factor</td>
<td>0 = Never had myocardial infarction, 1 = Has had myocardial infarction</td>
</tr>
<tr>
<td>GENDER</td>
<td>Gender</td>
<td>factor</td>
<td>0 = Female, 1 = Male</td>
</tr>
<tr>
<td>PM_BMI_CATEGORICAL</td>
<td>Body Mass Index (categorical)</td>
<td>factor</td>
<td>1 = Less than 25 kg/m², 2 = 25 to 30 kg/m², 3 = Over 30 kg/m²</td>
</tr>
</tbody>
</table>

---

### Description

Simulated dataset CNSIM 1, in a data.frame with 4128 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.
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

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

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

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 explicitly some DS packages
  defaultDSConfiguration(include=c("dsBase", "dsOmics"))
}

Simulated dataset DISCORDANT 1

Description

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

Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>B</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
</tbody>
</table>
Simulated dataset DISCORDANT 2

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

Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>C</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
</tbody>
</table>

Simulated dataset DISCORDANT 3

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

Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>C</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
</tbody>
</table>

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

```r
## 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.

Description

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

Usage

```r
## 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

```r
## 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

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

Arguments

conn  
DSLiteConnection-class object.
symbol  
Name of the R symbol.
table  
Fully qualified name of a dataset living in the DSSLite 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 DSSLite).
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 DSSLiteResult-class object.

Description

Connect to a DSSLite 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 DSSLiteServer 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 DLSite server

Description

Save the session in a local file if requested.

Usage

```r
## 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

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

Arguments

- `res` : `DSLiteResult-class` object.

Value

TRUE if table exists.
dsGetInfo, DLSiteResult-method

*Get result info*

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

The result information, including its status.

---

dsHasResource, DLSiteConnection-method

*Verify DLSite server resource*

**Description**

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

**Usage**

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

**Arguments**

- `conn`: `DLSiteConnection-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: DMLElement-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: DMLElement-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**
```r
## 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**
```r
## 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, DLSiteConnection-method**

*List DLSite server datasets*

**Description**

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

**Usage**

```r
## S4 method for signature 'DLSiteConnection'
dsListTables(conn)
```

**Arguments**

- `conn` **DLSiteConnection-class** class object

**Value**

The fully qualified names of the tables.

---

**dsListWorkspaces, DLSiteConnection-method**

*List workspaces*

**Description**

List workspaces saved in the data repository.

**Usage**

```r
## S4 method for signature 'DLSiteConnection'
dsListWorkspaces(conn)
```

**Arguments**

- `conn` **DLSiteConnection-class** class object

**Value**

A data frame.
Create a DSLite driver

Description

Convenient function for creating a DSLiteDriver object.

Usage

DSLite()

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()
- DLSiteServer$workspaces()
- DLSiteServer$workspace_save()
- DLSiteServer$workspace_rm()
- DLSiteServer$aggregateMethods()
- DLSiteServer$aggregateMethod()
- DLSiteServer$assignMethods()
- DLSiteServer$assignMethod()
- DLSiteServer$options()
- DLSiteServer$option()
- DLSiteServer$newSession()
- DLSiteServer$hasSession()
- DLSiteServer$getSession()
- DLSiteServer$getSessionIds()
- DLSiteServer$getSessionData()
- DLSiteServer$closeSession()
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:
```r
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:
```r
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).
**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 key of the option.

- **value**  
  The value of the option.
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 resource: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

Remove a symbol living in the DataSHIELD R session.

Usage

## S4 method for signature 'DSLiteConnection'

\[\text{dsRmSymbol}(\text{conn}, \text{symbol})\]

Arguments

- \text{conn} \hspace{1cm} \text{DSLiteConnection-class} class object
- \text{symbol} \hspace{1cm} 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(\text{conn}, \text{name})

Arguments

- \text{conn} \hspace{1cm} \text{DSLiteConnection-class} class object
- \text{name} \hspace{1cm} 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

<table>
<thead>
<tr>
<th>conn</th>
<th>DSLiteConnection-class class object</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the workspace.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>conns</th>
<th>DSConnection-class object or a list of DSConnection-classes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbol</td>
<td>Symbol name identifying the variable in the DSLiteServer’s &quot;server-side&quot; environment(s).</td>
</tr>
</tbody>
</table>

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

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

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.
### 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

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

---

### logindata.dslite.survival.expand_with_missing

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

### Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>Server/study name</td>
<td>char</td>
<td>DSLiteServer instance symbol name</td>
</tr>
<tr>
<td>url</td>
<td>Server/study URL</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>user</td>
<td>User name</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>password</td>
<td>User password</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>table</td>
<td>Table unique name</td>
<td>char</td>
<td>As registered in the DSLiteServer</td>
</tr>
<tr>
<td>options</td>
<td>Connection options</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>driver</td>
<td>Connection driver</td>
<td>char</td>
<td>DSLiteServer</td>
</tr>
</tbody>
</table>
logindata.dslite.testing.dataset

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

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

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>Server/study name</td>
<td>char</td>
<td>DSLiteServer instance symbol name</td>
</tr>
<tr>
<td>url</td>
<td>Server/study URL</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>user</td>
<td>User name</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>password</td>
<td>User password</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>table</td>
<td>Table unique name</td>
<td>char</td>
<td>As registered in the DSLiteServer</td>
</tr>
<tr>
<td>options</td>
<td>Connection options</td>
<td>char</td>
<td>Always empty for DSLiteServer</td>
</tr>
<tr>
<td>driver</td>
<td>Connection driver</td>
<td>char</td>
<td>DSLiteServer</td>
</tr>
</tbody>
</table>
**newDSLiteServer**  
*Create a new DSLite server*

**Description**

Shortcut function to create a new DSLiteServer instance.

**Usage**

```r
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.
setupDASIMTest

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

```r
{ 
  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

```r
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 `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()`, `setupDISCORDANTTest()`, `setupDSLiteServer()`, `setupSURVIVALTest()`
setupDISCORDANTTest

Examples

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

Description

Load the DISCORDANT datasets, the corresponding login data object, instantiate a new DSLiteServer 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 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()`, `setupDSLiteServer()`, `setupSURVIVALTest()`

Examples

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

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

**Description**

Load the provided datasets and the corresponding logindata object, instantiate 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**

```r
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`. Default is none.

- **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

```r
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, instantiante 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 DSLiteServer and the DSIConection 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

```r
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**

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

**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

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

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

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>Dummy data</td>
<td>char</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>NA_VALUES</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>NULL_VALUES</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>NON_NEGATIVE_INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>POSITIVE_INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>NEGATIVE_INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>NON_NEGATIVE_NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>POSITIVE_NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>NEGATIVE_NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>FACTOR_CHARACTER</td>
<td>Dummy data</td>
<td>factor</td>
</tr>
<tr>
<td>FACTOR_INTEGER</td>
<td>Dummy data</td>
<td>factor</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
</tbody>
</table>

---

**TESTING.DATASET2**

*Simulated dataset TESTING.DATASET 2*
Testing.DATASET3

Description

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

Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>Dummy data</td>
<td>char</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>NA_VALUES</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>NULL_VALUES</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>NON_NEGATIVE_INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>POSITIVE_INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>NEGATIVE_INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>NON_NEGATIVE_NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>POSITIVE_NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>NEGATIVE_NUMERIC</td>
<td>Dummy data</td>
<td>numeric</td>
</tr>
<tr>
<td>FACTOR_CHARACTER</td>
<td>Dummy data</td>
<td>factor</td>
</tr>
<tr>
<td>FACTOR_INTEGER</td>
<td>Dummy data</td>
<td>factor</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
</tbody>
</table>

Simulated dataset TESTING.DATASET 3

Description

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

Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>Dummy data</td>
<td>char</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>NA_VALUES</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>NULL_VALUES</td>
<td>Dummy data</td>
<td>logical</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Dummy data</td>
<td>integer</td>
</tr>
<tr>
<td>Dummy data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Non-negative</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Positive</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Negative</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>Non-negative</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>Positive</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>Negative</td>
<td></td>
</tr>
<tr>
<td>Factor</td>
<td>Character</td>
<td></td>
</tr>
<tr>
<td>Factor</td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Identifier</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Category</td>
<td></td>
</tr>
</tbody>
</table>
Index

* data
  - CNSIM1, 3
  - CNSIM2, 3
  - CNSIM3, 4
  - DASIM1, 5
  - DASIM2, 5
  - DASIM3, 6
  - DISCORDANT_STUDY1, 7
  - DISCORDANT_STUDY2, 8
  - DISCORDANT_STUDY3, 8
  - logindata.dslite.cnsim, 27
  - logindata.dslite.dasim, 27
  - logindata.dslite.discordant, 28
  - logindata.dslite.survival.expand_with_missing, 28
  - logindata.dslite.testing.dataset, 29
  - SURVIVAL.EXPAND_WITH_MISSING1, 36
  - SURVIVAL.EXPAND_WITH_MISSING2, 36
  - SURVIVAL.EXPAND_WITH_MISSING3, 37
  - TESTING.DATASET1, 38
  - TESTING.DATASET2, 38
  - TESTING.DATASET3, 39

* server-side items
  - DSLiteServer, 18
  - newDSLiteServer, 30

* setup functions
  - setupCNSIMTest, 30
  - setupDASIMTest, 31
  - setupDATASETTest, 32
  - setupDISCORDANTTest, 33
  - setupDSLiteServer, 34
  - setupSURVIVALTest, 35

CNSIM1, 3
CNSIM2, 3
CNSIM3, 4
DASIM1, 5
DASIM2, 5
DASIM3, 6
data, 34
dataShield.login, 31–35
defaultDSConfiguration, 6, 19, 30
DISCORDANT_STUDY1, 7
DISCORDANT_STUDY2, 8
DISCORDANT_STUDY3, 8
dsAggregate, DSLiteConnection-method, 8
dsAssignExpr, DSLiteConnection-method, 9
dsAssignResource, DSLiteConnection-method, 10
dsAssignTable, DSLiteConnection-method, 10
dsConnect, DSLiteDriver-method, 11
dsDisconnect, DSLiteConnection-method, 12
dsFetch, DSLiteResult-method, 12
dsGetInfo, DSLiteResult-method, 13
dsHasResource, DSLiteConnection-method, 13
dsHasTable, DSLiteConnection-method, 14
dsIsAsync, DSLiteConnection-method, 14
dsListMethods, DSLiteConnection-method, 15
dsListPackages, DSLiteConnection-method, 15
dsListResources, DSLiteConnection-method, 16
dsListSymbols, DSLiteConnection-method, 16
dsListTables, DSLiteConnection-method, 17
dsListWorkspaces, DSLiteConnection-method, 17
DSLite, 18
DSLiteServer, 11, 18, 26, 30–35
dsRmSymbol, DSLiteConnection-method, 25
dsRmWorkspace, DSLiteConnection-method,
dsSaveWorkspace, DSQLiteConnection-method, 25
getDSQLiteData, 26
logindata.dslite.cnsim, 27
logindata.dslite.dasim, 27
logindata.dslite.discordant, 28
logindata.dslite.survival.expand_with_missing, 28
logindata.dslite.testing.dataset, 29
newDSQLiteServer, 24, 30
setupCNSIMTest, 30, 32–35
setupDASIMTest, 31, 31, 32–35
setupDATASETTest, 31, 32, 32, 33–35
setupDISCORDANTTest, 31, 32, 33, 34, 35
setupDSQLiteServer, 31–33, 34, 35
setupSURVIVALTest, 31–34, 35
SURVIVAL.EXPAND_WITH_MISSING1, 36
SURVIVAL.EXPAND_WITH_MISSING2, 36
SURVIVAL.EXPAND_WITH_MISSING3, 37
TESTING.DATASET1, 38
TESTING.DATASET2, 38
TESTING.DATASET3, 39