Package: dsDashboard (via r-universe)

May 19, 2026

Type Package

Title Dashboard Framework for DataSHIELD Applications

Version 0.4.1

Author Andreas M\u00e4ndle

Maintainer Andreas M\u00e4ndle <maendle@leibniz-bips.de>

Description Provides a framework for building interactive dashboards in R. The package is designed to support data analysis both on local data and on data on a DataSHIELD server.

License MIT + file LICENSE

Imports data.table, dplyr, dsBaseClient, DSI, jsonlite, magrittr, Matrix, methods, purrr, rlang, shiny, stats, tidyr, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests dsBase, DSLite, DSOpal, dsTidyverseClient

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libx11-dev zlib1g-dev

Repository https://bips-hb.r-universe.dev

Date/Publication 2026-03-20 17:00:56 UTC

RemoteUrl https://github.com/bips-hb/dsDashboard

RemoteRef HEAD

RemoteSha 44fbf77820a06387902810fed2a9dcc387b69385

Contents

`assignAllTables` *Assign all tables from all servers to a symbol in DataSHIELD*

Description

The function returns the table names in case of success.

Usage

```
assignAllTables(datasources)
```

Arguments

datasources	A list of DSConnection-class objects.

Value

A data.frame containing the table names.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conn1 <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

conns <- DSI::datashield.connections_find()
tabNamesTidy <- assignAllTables(conns)
tabNamesTidy
#        id     .x
# 1 server1 table1
# 2 server2 table2

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

checkDatasources *Check datasources*

Description

Checks if datasources are a list of DSConnection-class.

Usage

```
checkDatasources(datasources = DSI::datashield.connections_find())
```

Arguments

datasources	A list of DSConnection-class objects.

Value

An integer. If datasources check passes, number of connections are returned, otherwise -1 is returned.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

checkDatasources()
# 2

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

createFactorVars — *Create for each level of a (factor) variable a boolean factor on the server side in DataShield.*

Description

Create for each level of a (factor) variable a boolean factor on the server side in DataShield.

Usage

```
createFactorVars(x, datasources = DSI::datashield.connections_find())
```

Arguments

x	A character string referring to a table column in DataShield.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

createFactorVars("D$GENDER", datasources=conns)
# [[1]]
# [[1]]$is.object.created
# [1] "A data object <GENDER0> has been created in all specified data sources"
#
# [[1]]$validity.check
# [1] "<GENDER0> appears valid in all sources"
#
#
# [[2]]
# [[2]]$is.object.created
# [1] "A data object <GENDER1> has been created in all specified data sources"
#
# [[2]]$validity.check
# [1] "<GENDER1> appears valid in all sources"

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

ds.boxplot_data *Return boxplot data*

Description

Returns the first quantile, median, third quantile and a data privacy preserving safe range (ymin and ymax).

Usage

```
ds.boxplot_data(
  x,
  variables = NULL,
  group = NULL,
  group2 = NULL,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

x	A character string. Name of the DataShield table object.
variables	A character vector. Name of the requested columns of the DataShield table object.
group	A character string or NULL. If ! NUL, then the boxplot data are for grouped boxplots for each group level of the variable "group".
group2	A character string or NULL. If ! NUL, then the boxplot data are for grouped boxplots for each group level of the variable "group2".
datasources	A list of DSConnection-class objects.

Value

A data.frame containing the data type for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers
  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

  # Get the boxplot data
```

```
boxplot_data <- ds.boxplot_data("D", "LAB_HDL", datasources=DSI::datashield.connections_find())
print(boxplot_data$combined)
#          x  ymin lower middle   upper    ymax
#     <char> <num> <num>  <num>   <num>   <num>
# 1: LAB_HDL 0.876 1.302  1.585 1.85025 2.22755

# compute boxplot data, grouped according to GENDER and BMI CATEGORY
boxplot_data <- ds.boxplot_data("D", "LAB_HDL", "GENDER", "PM_BMI_CATEGORICAL", datasources=DSI::datashield.con
print(boxplot_data$combined)

# clear the Datashield R sessions and logout
DSI::datashield.logout(conns)

## End(Not run)
```

`ds.meta` *Gets metadata for multiple variables of a table*

Description

This function gets the metadata for multiple variables of a given object on the server. Basically the DataSHIELD function ds.metadata is applied to all variables given in x. By default the result is returned as a data.frame.

Usage

```
ds.meta(x = NULL, datasources = NULL, simplify = TRUE)
```

Arguments

`x`	a character string specifying the name of the table object.
`datasources`	a list of DSConnection-class objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().
`simplify`	a boolean. If TRUE (default) the result is simplified to a data.frame. Otherwise a list is returned.

Value

`ds.metadata` returns to the metadata of the associated table held at the server.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')
```

```
  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

  # Get the metadata associated with table 'D'
  metadata <- ds.meta(x = c('D$LAB_TSC','D$LAB_TRIG','D$LAB_HDL'), datasources = conns)
  print(metadata)
 #       label              opal.value_type opal.entity_type opal.repeatable opal.index opal.nature
 # LAB_TSC  "Total Serum Cholesterol" "decimal"     "Participant"   0          0       "CONTINUOUS"
 # LAB_TRIG "Triglycerides"        "decimal"     "Participant"   0          0       "CONTINUOUS"
 # LAB_HDL  "HDL Cholesterol"      "decimal"     "Participant"   0          0       "CONTINUOUS"

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds.meta2` *Gets metadata for multiple variables of a table*

Description

This function gets the metadata for multiple variables of a given object on the server. Data for each connection are returned – contrary to the function ds.meta.

Usage

```
ds.meta2(x = NULL, datasources = NULL, summarise_servers = T)
```

Arguments

`x`	a character string specifying the variables.
`datasources`	a list of DSConnection-class
`summarise_servers`	Logical. If FALSE, the result will be returned for each variable on each connection as a separate row in the resulting data.frame. Otherwise information wil be summarized over all connections. objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

ds.metadata returns to the metadata of the associated table held at the server.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

# Get the metadata associated with table 'D'
metadata <- ds.meta2(x = c('D$LAB_TSC','D$LAB_TRIG','D$LAB_HDL'), datasources = conns)
print(metadata)
#                   label opal.value_type opal.entity_type opal.repeatable opal.index opal.nature class  obj.name
# LAB_TSC  Total Serum Cholesterol      decimal    Participant          0       0 CONTINUOUS   NA  D$LAB_TSC  LAB_
# LAB_TRIG       Triglycerides      decimal    Participant          0       0 CONTINUOUS   NA D$LAB_TRIG LAB_TR
# LAB_HDL       HDL Cholesterol      decimal    Participant          0       0 CONTINUOUS   NA  D$LAB_HDL  LAB_HD

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

ds_get_boxplot_data *Get (ungrouped) boxplot data (combined) for all or some numeric variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ the boxplot data (for ungrouped boxplots only).

Usage

```
ds_get_boxplot_data(
  tab,
  vars = NULL,
  message_fun = message,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

tab	A character string. Name of the DataShield table object.
vars	A vector of character strings. Names of the variables in the table object 'tab'.
message_fun	default is message. The function used to output log information in case of errors.
datasources	a list of DSConnection-class. objects obtained after login. If the dsBaseClient::datasources argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the boxplot data for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_boxplot_data("D")
# A tibble: 1 × 3
#  variable feature  value
#  <chr>    <chr>    <chr>
#  1 bmi_T1   missings 2228#'
  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_get_cat_summary` *Get summary statistics for all or some categorical variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ summary statistics such as the number of observations (total, valid and missings), the levels (level labels, and number of observations per level) and the mode.

Usage

```
ds_get_cat_summary(
  tab,
  vars = NULL,
  check = T,
  shiny_notification = F,
  n_levels = NULL,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`tab`	A character string. Name of the DataShield table object.
`vars`	A vector of character strings. Names of the variables in the table object ‘tab‘.
`check`	Logical. If FALSE, then the check for nonnumerical variables is skipped. This will lead to errors, if numeric variables are included in the parameter vars.
`shiny_notification`	A logical or integer. If not ‘FALSE‘ the parameter specifies the number of seconds that a notification is shown in case of errors.
`n_levels`	Integer value. If known in advance, the number of possible level values can be given. This can help to get reliable results in more cases.
`datasources`	a list of DSConnection-class. objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the boxplot data for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
```

```
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_cat_summary("D")
# A tibble: 1 × 3
#  variable feature  value
#  <chr>    <chr>    <chr>
#  1 bmi_T1   missings 2228

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_get_hist` *Get histogram data (combined for all servers) for all or some numeric variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ the features histograms information including bin positions and sizes.

Usage

```
ds_get_hist(
  tab,
  vars = NULL,
  bins = 11,
  shiny_notification = F,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`tab`	A character string. Name of the DataShield table object.
`vars`	A vector of character strings. Names of the variables in the table object ‘tab‘.

`bins` A numeric. The number of bins for the histogram.

`shiny_notification` A logical or integer. If not 'FALSE' the parameter specifies the number of seconds that a notification is shown in case of errors.

`datasources` a list of DSConnection-class objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the histogram data for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_hist("D", datasources=DSI::datashield.connections_find())
# A tibble: 6 × 3
#   variable feature value
#   <chr>    <chr>   <chr>
# 1 bmi_T1  breaks [9.8445,11.2128,12.5812,13.9495,15.3179,16.6862,18.0546,19.4229,20.7912,22.1596,23. . .
# 2 bmi_T1  mids   [10.5287,11.897,13.2653,14.6337,16.002,17.3704,18.7387,20.1071,21.4754,22.8438,24.. . .
# 3 bmi_T1   counts  [0,7,113,410,434,267,166,88,69,54,23,22,10,9,4,3,0,2,0,0]
# 4 bmi_T1  density [0,0.0018,0.0289,0.1049,0.111,0.0683,0.0425,0.0225,0.0176,0.0138,0.0059,0.0056,0.0. . .
# 5 bmi_T1   min     11.0884345752723
# 6 bmi_T1   max     35.9674388820484

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

ds_get_NA *Get number of missing variables for all or some specified variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ either "numeric" if the variable is numeric or "categorical" if the variable is a factor or character string.

Usage

```
ds_get_NA(tab, vars = NULL, datasources = DSI::datashield.connections_find())
```

Arguments

tab	A character string. Name of the DataShield table object.
vars	A vector of character strings. Names of the variables in the table object ‘tab‘.
datasources	a list of DSConnection-class. objects obtained after login. If the dsBaseClient::datasources argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the number of missing values for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_NA("D", datasources=DSI::datashield.connections_find())
# A tibble: 11 × 3
#    variable           feature  value
```

```
#    <chr>              <chr>    <chr>
#  1 LAB_TSC            missings 1005
#  2 LAB_TRIG           missings 1017
#  3 LAB_HDL            missings 1015
#  4 LAB_GLUC_ADJUSTED  missings 950
#  5 PM_BMI_CONTINUOUS  missings 302
#  6 DIS_CVA            missings 0
#  7 MEDI_LPD           missings 0
#  8 DIS_DIAB           missings 0
#  9 DIS_AMI            missings 0
# 10 GENDER             missings 0
# 11 PM_BMI_CATEGORICAL missings 302

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_get_quantile_mean` *Get quantiles and mean (combined or seperate for all servers) for all or some numeric variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ the quantiles (5

Usage

```
ds_get_quantile_mean(
  tab,
  vars = NULL,
  combine = T,
  tidy = T,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`tab`	A character string. Name of the DataShield table object.
`vars`	A vector of character strings. Names of the variables in the table object ‘tab‘.
`combine`	Logical. If TRUE the results over all servers are combined. Otherwise results are returned for each server separately.
`tidy`	Logical. If tidy (and combine not FALSE), then the result is formatted as a data.frame, otherwise results for each variable are returned as elements of a list.
`datasources`	a list of DSConnection-class objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the histogram data for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_quantile_mean("D")
# A tibble: 1 × 3
#   variable feature  value
#   <chr>    <chr>    <chr>
# 1 bmi_T1   missings 2228

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_get_range` *Get range (combined over all servers) for all or some numeric variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function get_hist.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ the features minimum ‘min‘ and maximum ‘max‘.

Usage

```
ds_get_range(
  tab,
  vars = NULL,
```

```
  datasources = DSI::datashield.connections_find()
)
```

Arguments

tab	A character string. Name of the DataShield table object.
vars	A vector of character strings. Names of the variables in the table object 'tab'.
datasources	a list of DSConnection-class objects obtained after login. If the dsBaseClient::datasources argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the minimum and maximum for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers
  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_range("D")
# A tibble: 1 × 3
#   variable feature  value
#   <chr>    <chr>    <chr>
# 1 bmi_T1   missings 2228

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

ds_get_type *Get variable type as string ("numeric" or "categorical") for all variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries. It eturns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ either "numeric" if the variable is numeric or "categorical" if the variable is a factor or character string.

Usage

```
ds_get_type(tab, vars = NULL, datasources = DSI::datashield.connections_find())
```

Arguments

tab	A character string. Name of the DataShield table object.
vars	A vector of character strings. Names of the variables in the table object ‘tab‘.
datasources	a list of DSConnection-class. objects obtained after login. If the dsBaseClient::datasources argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the data type for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_type("D", datasources=DSI::datashield.connections_find())
# # A tibble: 11 × 3
#    variable           feature value
#    <chr>              <chr>   <chr>
#  1 LAB_TSC            type    numeric
#  2 LAB_TRIG           type    numeric
#  3 LAB_HDL            type    numeric
#  4 LAB_GLUC_ADJUSTED  type    numeric
#  5 PM_BMI_CONTINUOUS  type    numeric
#  6 DIS_CVA            type    categorical
#  7 MEDI_LPD           type    categorical
```

```
#  8 DIS_DIAB           type    categorical
#  9 DIS_AMI            type    categorical
# 10 GENDER             type    categorical
# 11 PM_BMI_CATEGORICAL type    categorical


  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_get_variance` *Get variance (combined for all servers) for all or some numeric variables of a table in unified form*

Description

The main purpose of this function is to be a helper function for the function all_summaries.

It returns for each of the requested variables (‘vars‘) in table ‘tab‘ from the ‘datasources‘ the variance.

Usage

```
ds_get_variance(
  tab,
  vars = NULL,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`tab`	A character string. Name of the DataShield table object.
`vars`	A vector of character strings. Names of the variables in the table object ‘tab‘.
`datasources`	a list of DSConnection-class objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the histogram data for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')
```

```
  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_get_variance("D")
# A tibble: 1 × 3
#   variable feature  value
#   <chr>    <chr>    <chr>
# 1 bmi_T1   missings 2228

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_quantileMean` *Get quantiles and mean (separate or combined for all servers) for one numeric variable*

Description

The main purpose of this function is to be a helper function for the function get_quantile_mean.

It returns for the requested variable (‘X‘) from the ‘datasources‘ the quantiles (5

Usage

```
ds_quantileMean(
  x,
  combine = T,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`x`	A character string. The name of the DataShield object.
`combine`	Logical. If TRUE the results over all servers are combined. Otherwise results are returned for each server separately.
`datasources`	a list of DSConnection-class objects obtained after login. If the `dsBaseClient::datasources` argument is not specified the default set of connections will be used: see datashield.connections_default().

Value

A data.frame containing the histogram data for each variable in table tab.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ds_quantileMean("D$LAB_TSC")
# A tibble: 1 × 3
#   variable feature  value
#   <chr>    <chr>    <chr>
# 1 bmi_T1   missings 2228

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`ds_safe_aggregate` *Safely aggregate an expression across multiple DataSHIELD servers*

Description

Performs a single `datashield.aggregate()` call across all provided datasources while safely handling individual server failures. Results from successful servers are returned; failures are silently ignored by default.

Usage

```
ds_safe_aggregate(expr, datasources, silent = TRUE)
```

Arguments

`expr`	An R expression to be evaluated on the server side.
`datasources`	A named list of DataSHIELD connections.
`silent`	Logical; if `TRUE` (default), server-level errors are suppressed. If `FALSE`, server errors are reported via `message()`.

Details

This function:

- Executes a single aggregate call across all datasources
- Captures per-server success results
- Prevents a single failing server from aborting the entire operation

Value

A named list containing results from servers that successfully evaluated the expression. Servers that failed are omitted.

See Also

`datashield.aggregate`

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

res <- ds_safe_aggregate(
  expr = quote(table(D$sex)),
  datasources = conns
)

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)
```

```
## End(Not run)
```

`dsBetter.metadata` *Returns the metadata for one (!) specified variable*

Description

This function returns metadata for a specified variable - without unnecessary checks and outputs (as in the slow ds.metadata() from dsBaseClient). Due to the restrictions in dsBase::metadataDS only the following attributes are returned: 'names', 'spec', 'class', 'label', 'opal.value_type', 'opal.entity_type', 'opal.repeatable', 'opal.index', 'opal.nature' The variable does not need to be specified in each server to obtain a result.

Usage

```
dsBetter.metadata(
  x,
  datasources = DSI::datashield.connections_find(),
  silent = FALSE
)
```

Arguments

`x`	a string character, specifying the variable
`datasources`	A list of DSConnection-class objects. Default is to use all findable connections.
`silent`	Logical. If TRUE, DataSHIELD errors are suppressed.

Details

The function returns the metadata, obtained from attributes function.

Value

a matrix containing the metadata.

Examples

```
## Not run:
  # connecting to the Opal servers
  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
```

```
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  # Then perform login in each server
  conns <- datashield.login(logins=logindata, assign = TRUE, symbol = "D")

  # Get the metadata associated with table 'D'
  dsBetter.metadata(x = 'D$LAB_TSC', datasources = conns)
 # server  label              opal.value_type opal.entity_type opal.repeatable opal.index opal.nature  class obj
 # LAB_TSC "study1" "Total Serum Cholesterol" "decimal"     "Participant"   0          0       "CONTINUOUS" NA   "
 # LAB_TSC "study3" "Total Serum Cholesterol" "decimal"     "Participant"   0          0       "CONTINUOUS" NA   "

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsCallGLM` *Perform generalized linear model regression in a non-disclosive (k-nearest-neighbor) way.*

Description

The function basically calls ds.glm, but, the data table name is added to the output. This function is basically a more comfortable wrapper for ‘dsBaseClient::ds.glm‘.

Usage

```
dsCallGLM(
  formula = NULL,
  data = NULL,
  family = c("gaussian", "binomial", "poisson"),
  offset = NULL,
  weights = NULL,
  checks = FALSE,
  maxit = 20,
  CI = 0.95,
  viewIter = FALSE,
  viewVarCov = FALSE,
  viewCor = FALSE,
  datasources = NULL
)
```

Arguments

`formula`	An object of class formula describing the model to be fitted.

data	A character string specifying the name of an (optional) data frame that contains all of the variables in the GLM formula.
family	A character string. Identifies the error distribution function to use in the model ("gaussian", "binomial" or "poisson").
offset	A character string specifying the name of a variable to be used as an offset.
weights	A character string specifying the name of a variable containing prior regression weights for the fitting process.
checks	A logical. If TRUE ds.glm checks the structural integrity of the model. Default FALSE. For more information see Details.
maxit	A numeric scalar denoting the maximum number of iterations that are permitted before non-convergence is declared.
CI	A numeric value specifying the confidence interval (default 0.95).
viewIter	A logical. If TRUE the results of the intermediate iterations are printed. If FALSE only final results are shown. Default FALSE.
viewVarCov	A logical. If TRUE the variance-covariance matrix of parameter estimates is returned. Default FALSE.
viewCor	A logical. If TRUE the correlation matrix of parameter estimates is returned. Default FALSE.
datasources	A list of DSConnection-class objects.

Value

A list with parameters similar to glm objects.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

mod_height_ds_lm <- dsCallGLM(formula = "PM_BMI_CONTINUOUS ~ GENDER + LAB_TRIG",
                           data = "D",
                           family = "gaussian",
                           datasources = conns)
```

```
mod_height_ds_lm$coefficients
#               Estimate Std. Error   z-value      p-value  low0.95CI high0.95CI
# (Intercept) 26.2806538  0.1574577 166.90614 0.000000e+00 25.9720424 26.5892651
# GENDER1     -0.7579016  0.1521374  -4.98169 6.303141e-07 -1.0560855 -0.4597176
# LAB_TRIG     0.7469029  0.0481524  15.51123 2.912536e-54  0.6525259  0.8412798

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

dsCatAgg *Summary for one (and only one!) categorical variable of a table across all servers*

Description

Summary for one (and only one!) categorical variable of a table across all servers

Usage

```
dsCatAgg(x = NULL, datasources = NULL)
```

Arguments

x	A character string specifying the name of a data.frame.
datasources	A list of DSConnection-class objects.

Value

A data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
```

```
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsCatAgg("D$GENDER")
#   GENDER number of observations
# 1      0                   2677
# 2      1                   2574

dsCatAgg("D$DIS_AMI")
#   DIS_AMI number of observations
# 1 invalid                    NA

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsCatSummary` *Summary for all non-numeric variables in a DataSHIELD table*

Description

Summary for all non-numeric variables in a DataSHIELD table

Usage

```
dsCatSummary(x = NULL, datasources = DSI::datashield.connections_find())
```

Arguments

`x`	A character string referring to a DataSHIELD table.
`datasources`	A list of DSConnection-class objects.

Value

A list of data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
```

```
                user="administrator", password="password",
                table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                user="administrator", password="password",
                table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsCatSummary("D")
# $DIS_CVA
# level number of observations
# 1     0                   5248
# 2     1                      3
#
# $MEDI_LPD
# level number of observations
# 1     0                   5144
# 2     1                    107
#
# $DIS_DIAB
# level number of observations
# 1     0                   5174
# 2     1                     77
#
# $DIS_AMI
# level number of observations
# 1 invalid                    NA
#
# $GENDER
# level number of observations
# 1     0                   2677
# 2     1                   2574
#
# $PM_BMI_CATEGORICAL
# level number of observations
# 1     1                   1540
# 2     2                   1989
# 3     3                   1475

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsCatTidy` *A tidy summary for all non-numeric variables in a DataSHIELD table*

Description

A tidy summary for all non-numeric variables in a DataSHIELD table

Usage

```
dsCatTidy(x = NULL, datasources = DSI::datashield.connections_find())
```

Arguments

x	A character string referring to a DataSHIELD table.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.

Value

A data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsCatTidy("D")
#              variable   level number of observations
# 1             DIS_CVA       0                   5248
# 2             DIS_CVA       1                      3
# 3            MEDI_LPD       0                   5144
# 4            MEDI_LPD       1                    107
# 5            DIS_DIAB       0                   5174
# 6            DIS_DIAB       1                     77
# 7             DIS_AMI invalid                     NA
# 8              GENDER       0                   2677
# 9              GENDER       1                   2574
# 10 PM_BMI_CATEGORICAL       1                   1540
# 11 PM_BMI_CATEGORICAL       2                   1989
# 12 PM_BMI_CATEGORICAL       3                   1475

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

dsCatVarSummary *Summary for categorical variable across all servers*

Description

Summary for categorical variable across all servers

Usage

```
dsCatVarSummary(x = NULL, datasources = NULL)
```

Arguments

x	Name of referring to a data.frame column as string.
datasources	A list of 'OpalConnection'.

Value

A data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsCatVarSummary("D$GENDER")
#    server length level value
# 1 server1   2163     0  1092
# 2 server2   3088     0  1585
# 3 server1   2163     1  1071
# 4 server2   3088     1  1503

dsCatVarSummary("D$DIS_AMI")
#        id length   level value
```

```
# 1 invalid     NA invalid    NA


  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

dsGapply *Apply a function to subsets of a table in DataSHIELD for each different outcome of a categorical grouping variable*

Description

This is useful if an analysis has to be performed for each factor level of a grouping variable.

Usage

```
dsGapply(
  x,
  G,
  FUN,
  lazy = FALSE,
  datasources = DSI::datashield.connections_find(),
  ...
)
```

Arguments

x	A character string specifying a table which is available as a DataShield object.
G	A character string specifying the group variable in X according to which the data are split before applying function FUN.
FUN	A function which creates a data.frame. It will be applied to groupwise subsets of the table X.
lazy	A boolean. If TRUE then objects for groupwise tables are only created if they don't exist.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.
...	A additional parameters to be passed to FUN.

Value

A data.frame containing the results of FUN, applied to subgroups G of table X.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "tab1")

# get summary statistics for all variables grouped by GENDER
genderwise_summary <- dsGapply("tab1","GENDER",dsSummary, datasources=conns[1])
head(genderwise_summary)
#   GENDER variable feature   value       tabname
# 1      0  LAB_TSC       N 1092.00 tab1.GENDER.0
# 2      0  LAB_TSC      5%    4.07 tab1.GENDER.0
# 3      0  LAB_TSC     10%    4.50 tab1.GENDER.0
# 4      0  LAB_TSC     25%    5.09 tab1.GENDER.0
# 5      0  LAB_TSC     50%    5.90 tab1.GENDER.0
# 6      0  LAB_TSC     75%    6.67 tab1.GENDER.0

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

dsGet_xyg — *Get two numeric columns and a factor column of a DataShield data.frame in a non-disclosive (k-nearest-neighbor) way.*

Description

The result can be used e.g. for a grouped scatterplot and for further analysis on the client side.

Usage

```
dsGet_xyg(
  x,
  y,
  g,
```

```
  tab,
  method = 1,
  k = 3,
  noise = 0.25,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

x	A character string. Name of the x-variable in the DataShield table object.
y	A character string. Name of the y-variable in the DataShield table object.
g	A character string. Name of the factor variable (group) in the DataShield table object.
tab	A character string. Name of the DataShield table object.
method	method A character string that specifies the method that is used to generated non-disclosive coordinates to be displayed in a scatter plot. This argument can be set as 'deterministic' (method=1) or 'probabilistic' (method=2). Default 'deteministic'.
k	Numeric. Only used if 'deterministic' method is used. The number of the nearest neighbors for which their centroid is calculated. Default 3.
noise	Numeric. Only used if 'probabilistic' method is used. The percentage of the initial variance that is used as the variance of the embedded noise if the argument method is set to 'probabilistic'.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.

Value

A data.frame with coloumns for x, y, group g and server (source).

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")
```

```
pair_data <- dsGet_xyg("LAB_TSC","LAB_TRIG","GENDER","D")

head(pair_data)
#    LAB_TSC  LAB_TRIG GENDER  server
# 1 6.379269 1.4462051      0 server1
# 2 5.427365 4.3632667      0 server1
# 3 7.877391 3.2380028      0 server1
# 4 5.521844 2.1337910      0 server1
# 5 5.139187 2.5045780      0 server1
# 6 6.204872 0.6240902      0 server1

# a grouped scatterplot
plot(pair_data$LAB_TSC, pair_data$LAB_TRIG, col=as.numeric(pair_data$GENDER)+1)

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsGetPairs` *Get two numeric columns and a factor column of a DataShield data.frame in a non-disclosive (k-nearest-neighbor) way.*

Description

This function is basically a wrapper for dsGet_xyg, with slightly different syntax and output.

Usage

```
dsGetPairs(
  tab,
  vars,
  group,
  method = 1,
  k = 3,
  noise = 0.25,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`tab`	A character string. Name of the DataShield table object.
`vars`	A character vector. Names of the variables in the DataShield table.
`group`	A character string. Name of the factor variable (group) in the DataShield table object.
`method`	method A character string that specifies the method that is used to generated non-disclosive coordinates to be displayed in a scatter plot. This argument can be set as 'deterministic' (method=1) or 'probabilistic' (method=2). Default 'deteministic'.

k	Numeric. Only used if 'deterministic' method is used. The number of the nearest neighbors for which their centroid is calculated. Default 3.
noise	Numeric. Only used if 'probabilistic' method is used. The percentage of the initial variance that is used as the variance of the embedded noise if the argument method is set to 'probabilistic'.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.

Value

A data.frame with coloumns for x, y, group g and server (source).

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

pair_data <- dsGetPairs("D",c("LAB_TSC","LAB_TRIG"),"GENDER")

head(pair_data)
#      xName    yName        x         y GENDER  server
#  1 LAB_TSC LAB_TRIG 6.379269 1.4462051      0 server1
#  2 LAB_TSC LAB_TRIG 5.427365 4.3632667      0 server1
#  3 LAB_TSC LAB_TRIG 7.877391 3.2380028      0 server1
#  4 LAB_TSC LAB_TRIG 5.521844 2.1337910      0 server1
#  5 LAB_TSC LAB_TRIG 5.139187 2.5045780      0 server1
#  6 LAB_TSC LAB_TRIG 6.204872 0.6240902      0 server1

# a grouped scatterplot
plot(pair_data$x, pair_data$y, col=as.numeric(pair_data$GENDER)+1)

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

dsGLM *Perform generalized linear model regression and return the model object*

Description

including non-disclosive k-nearest neighbor estimates for residuals.

Usage

```
dsGLM(
  formula = NULL,
  data = NULL,
  family = c("gaussian", "binomial", "poisson"),
  offset = NULL,
  weights = NULL,
  maxit = 20,
  CI = 0.95,
  viewIter = FALSE,
  viewCor = FALSE,
  datasources = NULL,
  add_tab_to_depvar = T
)
```

Arguments

formula	An object of class formula describing the model to be fitted.
data	A character string specifying the name of an (optional) data frame that contains all of the variables in the GLM formula.
family	A character string. Identifies the error distribution function to use in the model ("gaussian", "binomial" or "poisson").
offset	A character string specifying the name of a variable to be used as an offset.
weights	A character string specifying the name of a variable containing prior regression weights for the fitting process.
maxit	A numeric scalar denoting the maximum number of iterations that are permitted before non-convergence is declared.
CI	A numeric value specifying the confidence interval (default 0.95).
viewIter	A logical. If TRUE the results of the intermediate iterations are printed. If FALSE only final results are shown. Default FALSE.
viewCor	A logical. If TRUE the correlation matrix of parameter estimates is returned. Default FALSE.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.
add_tab_to_depvar	A boolean. If TRUE, the table name + "$" is added as prefix to the dependent variable. Default is TRUE.

Value

A list with parameters similar to glm objects.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

foo <- dsGLM(formula = "PM_BMI_CONTINUOUS ~ GENDER + LAB_TRIG",
             data = "D",
             family = "gaussian",
             datasources = conns)
## qq plot
qqnorm(foo$y)
qqline(foo$y)

foo <- dsGLM(formula = "PM_BMI_CONTINUOUS ~ LAB_HDL + LAB_TRIG",
             data = "D", family = "gaussian", datasources = conns)
# histogram of privacy preserving residuals
hist(x = foo$residual, type = "combine", breaks = 18, datasources = conns)
# compare that histogram with `dsBaseClient::ds.histogram`
dsBaseClient::ds.histogram(x = "mod_residuals",
            type = "combine",
            num.breaks = 19,
            datasources = conns)
# scatterplot
dsBaseClient::ds.scatterPlot(x="mod_predicted", y="mod_residuals", type="combine", datasources=conns)

# attention: it is still not fully compatible with the models from stats::lm:
# e.g. this does not work due to lack of the data slot:
#stats:::plot.lm(foo)

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsIsNumeric` *Check if columns of a DataSHIELD table are numeric*

Description

Check if columns of a DataSHIELD table are numeric

Usage

```
dsIsNumeric(
  x = NULL,
  x.vars = NULL,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

`x`	A character string referring to a data.frame.
`x.vars`	Optional character string vector; a subset of the variables in table x, on which the function shall be applied
`datasources`	A list of DSConnection-class objects. Default is to use all findable connections.

Value

A vector of boolean.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsIsNumeric("D")
# LAB_TSC           LAB_TRIG            LAB_HDL   LAB_GLUC_ADJUSTED   PM_BMI_CONTINUOUS
```

```
#    TRUE              TRUE              TRUE              TRUE              TRUE
# DIS_CVA          MEDI_LPD          DIS_DIAB          DIS_AMI            GENDER
#   FALSE             FALSE             FALSE             FALSE             FALSE
# PM_BMI_CATEGORICAL
#             FALSE

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

dsNumSummary *A summary for a numeric variable in a DataSHIELD table*

Description

The summary contains the number of observations, quantiles (5 75

Usage

```
dsNumSummary(x, datasources = DSI::datashield.connections_find())
```

Arguments

x	A character string referring to a numeric column in a DataSHIELD table.
datasources	A list of DSConnection-class objects. Default is to use all findable datasources.

Details

A similar function for all columns of a table is ‘dsNumVarSummary‘ which returns results for all numeric columns of a table – however it does not return the number of observations.

Value

A data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
```

```
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsNumSummary("D$LAB_HDL")
#    LAB_HDL        value
#  1       N 5251.0000000
#  2      5%    0.8606589
#  3     10%    1.0385205
#  4     25%    1.2964949
#  5     50%    1.5704848
#  6     75%    1.8418712
#  7     90%    2.0824057
#  8     95%    2.2191369
#  9    Mean    1.5619572

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsNumVarSummary` *Summary for all numeric variables in a DataSHIELD table- deprecated*

Description

Quantiles (5 numeric variables.

Usage

```
dsNumVarSummary(x = NULL, datasources = DSI::datashield.connections_find())
```

Arguments

`x`	A character string referring to a DataSHIELD table.
`datasources`	A list of DSConnection-class objects. Default is to use all findable connections as datasource.

Value

A data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsNumVarSummary("D")
#       LAB_TSC    LAB_TRIG   LAB_HDL LAB_GLUC_ADJUSTED PM_BMI_CONTINUOUS
# 5%   4.118277 -0.47120403 0.8606589          4.095639          19.45742
# 10%  4.503466  0.07673459 1.0385205          4.539658          21.20121
# 25%  5.132161  1.01642189 1.2964949          5.222477          24.16518
# 50%  5.834384  2.07638965 1.5704848          6.046338          27.36816
# 75%  6.543257  3.08549769 1.8418712          6.857015          30.66310
# 90%  7.233349  4.04673538 2.0824057          7.642129          33.76176
# 95%  7.648063  4.58631916 2.2191369          8.126197          35.57591
# Mean 5.856428  2.06854312 1.5619572          6.110854          27.44250

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsResiduals` *Compute residuals for models returned by the dsCallGLM function on the server side.*

Description

This function is basically a helper function for ‘dsBaseClient::ds.glm‘.

Usage

```
dsResiduals(mod, datasources)
```

Arguments

`mod`	A list object as returned by dsCallGLM.
`datasources`	A list of DSConnection-class objects.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

mod_height_ds_lm <- dsCallGLM(formula = "PM_BMI_CONTINUOUS ~ GENDER + LAB_TRIG",
                          data = "D",
                          family = "gaussian",
                          datasources = conns)

dsResiduals(mod_height_ds_lm, datasources=conns)
# results are on the server only and can be used by `dsGLM`

# test using DSLite:
getDSLiteData(conns, "mod_residuals")
getDSLiteData(conns, "mod_predicted")

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsSubsetLevels` *Get subsets of a DataShield object for each factor level of a factor variable*

Description

Get subsets of a DataShield object for each factor level of a factor variable

Usage

```
dsSubsetLevels(
  var,
  tab,
```

```
  lazy = FALSE,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

var	A character string. Name of the variable in the DataShield table object.
tab	A character string. Name of the DataShield table object.
lazy	A boolean. If TRUE then objects for groupwise tables are only created if they don't exist.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.

Value

A list of the names of the created DataShield objects.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsSubsetLevels("PM_BMI_CATEGORICAL","D")
# then these new tables canbe used, e.g. for a plot:
ggHistDS("D.PM_BMI_CATEGORICAL.1$LAB_TRIG")

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

dsSummary *A summary for all (numeric and non-numeric) variables in a DataSHIELD table*

Description

A summary for all (numeric and non-numeric) variables in a DataSHIELD table

Usage

```
dsSummary(x = NULL, datasources = DSI::datashield.connections_find())
```

Arguments

x	A character string referring to a numeric column in a DataSHIELD table.
datasources	A list of DSConnection-class objects.

Value

A data.frame.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

rs <- dsSummary("D")
head(rs, 20)
#              variable feature   value
# 1             LAB_TSC       N 5251.00
# 2             LAB_TSC      5%    4.12
# 3             LAB_TSC     10%    4.50
# 4             LAB_TSC     25%    5.13
# 5             LAB_TSC     50%    5.83
```

```
# 6             LAB_TSC     75%    6.54
# 7             LAB_TSC     90%    7.23
# 8             LAB_TSC     95%    7.65
# 9             LAB_TSC    Mean    5.86
# 10           LAB_TRIG       N 5251.00
# 11           LAB_TRIG      5%   -0.47
# 12           LAB_TRIG     10%    0.08
# 13           LAB_TRIG     25%    1.02
# 14           LAB_TRIG     50%    2.08
# 15           LAB_TRIG     75%    3.09
# 16           LAB_TRIG     90%    4.05
# 17           LAB_TRIG     95%    4.59
# 18           LAB_TRIG    Mean    2.07
# 19            LAB_HDL       N 5251.00
# 20            LAB_HDL      5%    0.86

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

`dsTableSummary` *Summary for a table in DataSHIELD*

Description

Output: number.of.rows, number.of.columns, variables.held for a table object per server

Usage

```
dsTableSummary(x = NULL, datasources = NULL)
```

Arguments

`x`	Table name as string.
`datasources`	A list of 'OpalConnection'.

Details

The output is more convenient than using the ds.summary function directly, as it turns the information in a data.frame.

Value

A data.frame

Examples

```
if (require(DSLite) && require('dsBase') ) {
  require('DSI')
  require('DSOpal')
  require('dsBaseClient')
  require('dsBase')

data('CNSIM1')
data('CNSIM2')

# build a DSLite server with the datasets inside
dslite.server1 <- DSLite::newDSLiteServer(tables=list(table1=CNSIM1))
dslite.server2 <- DSLite::newDSLiteServer(tables=list(table2=CNSIM2))

# build DS login information
builder <- DSI::newDSLoginBuilder()
builder$append(server = 'server1', driver = 'DSLiteDriver', url = 'dslite.server1')
builder$append(server = 'server2', driver = 'DSLiteDriver', url = 'dslite.server2')
logindata <- builder$build()

# do login and table assignment
conns <- DSI::datashield.login(logindata)
DSI::datashield.assign.table(conns, 'tab1', table = list(server1='table1', server2='table2'))

dsTableSummary("tab1")

# output:
#    server number.of.rows number.of.columns variables.held
# 1 server1           2163                11   LAB_TSC,....
# 2 server2           3088                11   LAB_TSC,....

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)
} else {
  message("This example requires DSLite and dsBase")
}
```

`dsUniqueLevels` *Get factor levels for a variable over all server connections*

Description

Get factor levels for a variable over all server connections

Usage

```
dsUniqueLevels(
  var,
  tab = NULL,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

var	A character string. Name of the variable in the DataShield table object.
tab	A character string. Name of the DataShield table object.
datasources	A list of DSConnection-class objects. Default is to use all findable datasources.

Value

A numeric vector with the names of the factor levels.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsUniqueLevels("PM_BMI_CATEGORICAL","D")
# [1] "1" "2" "3"
dsUniqueLevels("D$PM_BMI_CATEGORICAL")
# [1] "1" "2" "3"

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

dsUniqueVarnames *Get unique variable names of a table object across all servers*

Description

Get unique variable names of a table object across all servers

Usage

```
dsUniqueVarnames(
  x = NULL,
  datasources = DSI::datashield.connections_find(),
  combined = T
)
```

Arguments

x	Table name as string.
datasources	A list of 'OpalConnection'. Default is to use all findable connections.
combined	If FALSE, the available variables are returned per server, otherwise accross all servers.

Value

A vector.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

dsUniqueVarnames("D")
# output:
# [1] "LAB_TSC"        "LAB_TRIG"       "LAB_HDL"        "LAB_GLUC_ADJUSTED" "PM_BMI_CONTINUOUS" "DIS_CVA"
# [8] "DIS_DIAB"           "DIS_AMI"            "GENDER"             "PM_BMI_CATEGORICAL"

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

get_xy *Get two numeric columns of a DataShield data.frame in a non-disclosive (k-nearest-neighbor) way.*

Description

These data can than be used e.g. for a scatterplot or an analysis on the client side.

Usage

```
get_xy(
  x,
  y,
  method = 1,
  k = 3,
  noise = 0.25,
  datasources = DSI::datashield.connections_find()
)
```

Arguments

x	A character string. Name of the x-variable in the DataShield table object.
y	A character string. Name of the y-variable in the DataShield table object.
method	method A character string that specifies the method that is used to generated non-disclosive coordinates to be displayed in a scatter plot. This argument can be set as 'deterministic' (method=1) or 'probabilistic' (method=2). Default 'deteministic'.
k	Numeric. Only used if 'deterministic' method is used. The number of the nearest neighbors for which their centroid is calculated. Default 3.
noise	Numeric. Only used if 'probabilistic' method is used. The percentage of the initial variance that is used as the variance of the embedded noise if the argument method is set to 'probabilistic'.
datasources	A list of DSConnection-class objects. Default is to use all findable connections.

Value

A data.frame with coloumns for x, y and server (source).

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')
```

```
  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

pair_data <- get_xy("D$LAB_TSC","D$LAB_TRIG")

head(pair_data)
#          x         y  server
# 1 7.220168 2.4789300 server1
# 2 6.033267 1.3209085 server1
# 3 6.378069 1.4572257 server1
# 4 6.844664 4.4676767 server1
# 5 6.441833 0.7183122 server1
# 6 6.052160 2.9344455 server1

# a simple scatterplot
plot(pair_data$x, pair_data$y)

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

getRange *Get Range*

Description

Uses the histogramDS1 server function to get a safe range (spanning over minimum and maximum, were the exact range is increased with some privacy preserving noise).

Usage

```
getRange(
  x,
  type = "combined",
  k = 3,
  noise = 0.25,
  safemode = F,
  datasources = DSI::datashield.connections_find(),
  stdnames = names(datasources),
  method = 1
)
```

Arguments

x	A character string referring to a numeric column in a DataSHIELD table.
type	Boolean. If "combined" then the results are aggregated over all servers, otherwise a list of all server results is returned.
k	Numeric. Only used if 'method==2'. The number of the nearest neighbors for which their centroid is calculated. Default 3.
noise	Numeric. Only used if 'method==3'. The percentage of the initial variance that is used as the variance of the embedded noise if the argument method is set to 'probabilistic'.
safemode	Boolean, if TRUE, histogramDS1 is called for each datasource separately, to ensure that a result can be returned even if one datasource creates an error.
datasources	A list of DSConnection-class objects. Default is to use all findable datasources.
stdnames	Character vector. If there are more than one datasources and type is not "combined", then this defines how the rows of the resulting table are labelled. Default is to use the names of the datasources.
method	A numeric. Either 1, 2 or 3 indicating the method of disclosure control that is used for the generation of the histogram. If the value is equal to 1 then the 'smallCellsRule' is used. If the value is equal to 2 then the 'deterministic' method is used. If the value is set to 3 then the 'probabilistic' method is used.

Details

If safemode is FALSE the function will fail if histogramDS1 fails for any server.

Value

A numeric vector (if '"combined" or only 1 study) or a matrix (else).

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")
```

```
getRange("D$LAB_TRIG")
#       min       max
# -3.483958 12.128934
getRange("D$LAB_TRIG", safemode=T)
#       min       max
# -3.483958 12.128934
getRange("D$LAB_TRIG", type="separate", safemode=T)
#               min      max
# server1 -3.483958 12.12893
# server2 -3.331776 12.02544

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

ggHistDS *Plot data in a histogram with R graphics*

Description

Plot data in a histogram with R graphics

Usage

```
ggHistDS(
  x,
  bins = "Sturges",
  plot = "combined",
  k = 3,
  noise = 0.25,
  freq = T,
  safemode = F,
  range = NULL,
  datasources = DSI::datashield.connections_find(),
  stdnames = names(datasources),
  method = 2
)
```

Arguments

x	A character string referring to a numeric column in a DataSHIELD table.
bins	either a numeric defining the number of bins or "Sturges" to choose the bin number automatically. Default is "Sturges".
plot	If "combined" then a plot combining the results from all servers, if "separate" plots for each datasource, if "all" plots for each datasource and combined are plotted. Otherwise only data are returned as a list and no plot is generated.

k	Numeric. Only used if ‘method==2‘. The number of the nearest neighbors for which their centroid is calculated. Default 3.
noise	Numeric. Only used if ‘method==3‘. The percentage of the initial variance that is used as the variance of the embedded noise if the argument method is set to ’probabilistic’.
freq	Boolean.
safemode	Boolean. If FALSE, no plot is generated if any datasource fails, otherwise the failing datasources are ignored and the results shouw only data from non-failing datasources.
range	Numeric vector of length 2. A range for the histogram can be specified.
datasources	A list of DSConnection-class objects.
stdnames	Character vector. If there are more than one datasources and type is not "combined", then this defines how the rows of the resulting table are labelled. Default is to use the names of the datasources.
method	A numeric. Either 1, 2 or 3 indicating the method of disclosure control that is used for the generation of the histogram. If the value is equal to 1 then (only!) the ’smallCellsRule’ is used. If the value is equal to 2 then the ’deterministic’ method is used additionally to method 1 for bins with count zero according to method 1. If the value is set to 3 then the ’probabilistic’ method is used additionally to method 1 for bins with count zero according to method 1.

Value

A list containing the histogram object.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")

ggHistDS("D$LAB_TRIG")
ggHistDS("D$LAB_TRIG", safemode=T, plot="separate")
ggHistDS("D$LAB_TRIG", safemode=T, plot="all")
ggHistDS("D$LAB_TRIG", safemode=T, plot=F)
```

```
# [[1]]
# [[1]]$histobject
# $breaks
# [1] -3.4839583 -2.2829666 -1.0819749  0.1190168  1.3200085  2.5210003  3.7219920  4.9229837  6.1239754  7.3249671
# [13] 10.9279422 12.1289340
#
# $counts
# [1]   4  33 135 396 531 450 192  50   6   1   0   0   3
#
# $density
# [1] 0.001849295 0.015256686 0.062413717 0.183080236 0.245493952 0.208045722 0.088766175 0.023116191 0.002773943
# [12] 0.000000000 0.001386971
#
# $mids
# [1] -2.8834624 -1.6824707 -0.4814790  0.7195127  1.9205044  3.1214961  4.3224878  5.5234795  6.7244713  7.9254630
# [13] 11.5284381
#
# $xname
# [1] "xvect"
#
# $equidist
# [1] TRUE
#
# attr(,"class")
# [1] "histogram"
#
# [[1]]$invalidcells
# [1] 1
#
# [[1]]$simulatedObs
# [1] 3
#
# [[1]]$uncertainCells
# [1] 10 11 12
#
# ....

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

---

runDemo *Launch the Shiny Dashboard*

Description

Starts the Shiny dashboard demo.

Usage

```
runDemo()
```

Examples

```
## Not run:
runDemo()

## End(Not run)
```

---

tabSummary *Summarize the requested tables (DataSHIELD server list)*

Description

Gets row and column counts for all given tables.

Usage

```
tabSummary(tabList, datasources)
```

Arguments

tabList	A data.frame with the table names and server locations.
datasources	A list of DSConnection-class objects.

Value

A data.frame containing the table names, server location, symbols and dimensions.

Examples

```
## Not run:
  # connecting to the Opal servers

  require('DSI')
  require('DSOpal')
  require('dsBaseClient')

  builder <- DSI::newDSLoginBuilder()
  builder$append(server="study1", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM1")
  builder$append(server="study3", url="https://opal-demo.obiba.org",
                 user="administrator", password="password",
                 table = "CNSIM.CNSIM3")
  logindata <- builder$build()

  conns <- DSI::datashield.login(logins = logindata, assign = TRUE, symbol = "D")
```

```
conns <- DSI::datashield.connections_find()
tabSummary(tabNamesTidy, datasources=conns)
#     id  server   name rows cols
# 1 tab1 server1 table1 2163   11
# 2 tab2 server2 table2 3088   11

  # clear the Datashield R sessions and logout
  DSI::datashield.logout(conns)

## End(Not run)
```

Index