Package 'dsTidyverse'

February 27, 2025

Type Package

Title 'DataSHIELD' 'Tidyverse' Serverside Package

Version 1.0.4

Maintainer Tim Cadman <t.j.cadman@umcg.nl>

Description Implementation of selected 'Tidyverse' functions within 'DataSHIELD', an open-source federated analysis solution in R. Currently, DataSHIELD contains very limited tools for data manipulation, so the aim of this package is to improve the researcher experience by implementing essential functions for data manipulation, including subsetting, filtering, grouping, and renaming variables. This is the serverside package which should be installed on the server holding the data, and is used in conjuncture with the clientside package 'dsTidyverseClient' which is installed in the local R environment of the analyst. For more information, see <https://www.tidyverse.org/> and <https://datashield.org/>.

License LGPL (>= 2.1)

Encoding UTF-8

RoxygenNote 7.3.2

Imports rlang, cli

Suggests testthat (>= 3.0.0), tibble, DSLite, dsBaseClient, dsBase, DSI, dplyr, purrr, mockery

Config/testthat/edition 3

Additional_repositories https://cran.obiba.org/

NeedsCompilation no

Author Tim Cadman [aut, cre] (<https://orcid.org/0000-0002-7682-5645>), Mariska Slofstra [aut] (<https://orcid.org/0000-0002-0400-0468>), Stuart Wheater [aut], Demetris Avraam [aut]

Repository CRAN

Date/Publication 2025-02-27 09:40:06 UTC

Contents

`arrangeDS` *Order the rows of a data frame by the values of selected columns*

Description

DataSHIELD implentation of `dplyr::arrange`.

Usage

```
arrangeDS(tidy_expr, df.name, .by_group)
```

Arguments

`tidy_expr`	Variables, or functions of variables. Use `desc()` to sort a variable in descending order.
`df.name`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
`.by_group`	If TRUE, will sort first by grouping variable. Applies to grouped data frames only.

Value

An object of the same type as `df.name`, typically a data frame or tibble.

asTibbleDS *Coerce a data frame or matrix to a tibble*

Description

DataSHIELD implementation of `tibble::as_tibble`. Currently only implemented for data frames and matrices.

Usage

```
asTibbleDS(tidy_expr, x, .rows, .name_repair, rownames)
```

Arguments

`tidy_expr`	Unused in present function.
`x`	A data frame or matrix.
`.rows`	The number of rows, useful to create a 0-column tibble or just as an additional check.
`.name_repair`	Treatment of problematic column names: • "minimal": No name repair or checks, beyond basic existence. • "unique": Make sure names are unique and not empty. • "check_unique": (default value), no name repair, but check they are unique. • "universal": Make the names unique and syntactic.
`rownames`	How to treat existing row names of a data frame or matrix: • 'NULL': remove row names. This is the default. • 'NA': keep row names. • A string: the name of a new column. Existing rownames are transferred into this column and the `row.names` attribute is deleted. No name repair is applied to the new column name, even if `x` already contains a column of that name.

Value

A tibble.

bindColsDS *Bind multiple data frames by column*

Description

DataSHIELD implementation of `dplyr::bind_cols`.

Usage

```
bindColsDS(to_combine = NULL, .name_repair = NULL)
```

Arguments

`to_combine`	Data frames to combine. Each argument can either be a data frame, a list that could be a data frame, or a list of data frames. Columns are matched by name, and any missing columns will be filled with NA.
`.name_repair`	One of "unique", "universal", or "check_unique". See `vctrs::vec_as_names()` for the meaning of these options.

Value

A data frame the same type as the first element of `to_combine`

bindRowsDS *Bind multiple data frames by row.*

Description

DataSHIELD implementation of `dplyr::bind_rows`.

Usage

```
bindRowsDS(to_combine = NULL, .id = NULL)
```

Arguments

`to_combine`	Data frames to combine. Each argument can either be a data frame, a list that could be a data frame, or a list of data frames. Columns are matched by name, and any missing columns will be filled with NA.
`.id`	he name of an optional identifier column. Provide a string to create an output column that identifies each input. The column will use names if available, otherwise it will use positions.

Value

A data frame the same type as the first element of `to_combine`

caseWhenDS *Performs dplyr case_when*

Description

DataSHIELD implentation of `dplyr::case_when`.

Usage

```
caseWhenDS(tidy_expr = NULL, .default = NULL, .ptype = NULL, .size = NULL)
```

Arguments

`tidy_expr`	A sequence of two-sided formulas. The left hand side (LHS) determines which values match this case. The right hand side (RHS) provides the replacement value. The LHS inputs must evaluate to logical vectors. The RHS inputs will be coerced to their common type. All inputs will be recycled to their common size. That said, we encourage all LHS inputs to be the same size. Recycling is mainly useful for RHS inputs, where you might supply a size 1 input that will be recycled to the size of the LHS inputs. NULL inputs are ignored.
`.default`	The value used when all of the LHS inputs return either FALSE or NA.
`.ptype`	An optional prototype declaring the desired output type. If supplied, this overrides the common type of true, false, and missing.
`.size`	An optional size declaring the desired output size. If supplied, this overrides the size of condition.

Value

A vector with the same size as the common size computed from the inputs in `tidy_expr` and the same type as the common type of the RHS inputs in `tidy_expr`.

---

checkPermissivePrivacyControlLevel
checkPermissivePrivacyControlLevel

Description

This serverside function check that the server is running in "permissive" privacy control level.

Usage

```
checkPermissivePrivacyControlLevel(privacyControlLevels)
```

Arguments

`privacyControlLevels`
: is a vector of strings which contains the privacy control level names which are permitted by the calling method.

Details

Tests whether the R option "datashield.privacyControlLevel" is set to "permissive", if it isn't will cause a call to stop() with the message "BLOCKED: The server is running in 'non-permissive' mode which has caused this method to be blocked".

Value

Returns an error if the method is not permitted; otherwise, no value is returned.

Author(s)

Wheater, Dr SM., DataSHIELD Team.

`distinctDS` *Keep distinct/unique rows*

Description

DataSHIELD implentation of `dplyr::distinct`.

Usage

```
distinctDS(tidy_expr, df.name, .keep_all)
```

Arguments

`tidy_expr`	Optional variables to use when determining uniqueness. If there are multiple rows for a given combination of inputs, only the first row will be preserved. If omitted, will use all variables in the data frame.
`df.name`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
`.keep_all`	If TRUE, keep all variables in df.name If a combination of expr is not distinct, this keeps the first row of values.

Value

An object of the same type as `df.name`, typically a data frame or tibble.

filterDS *Performs dplyr filter*

Description

DataSHIELD implentation of `dplyr::filter`.

Usage

```
filterDS(tidy_expr, df.name, .by, .preserve)
```

Arguments

`tidy_expr`	Diffused expression that return a logical value, and are defined in terms of the variables in `df.name`.
`df.name`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
`.by`	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to `group_by`.
`.preserve`	Relevant when the df.name input is grouped. If .preserve = FALSE (the default), the grouping structure is recalculated based on the resulting data, otherwise the grouping is kept as is.

Value

An object of the same type as `df.name`, typically a data frame or tibble.

groupByDS *Group by one or more variables*

Description

DataSHIELD implentation of `dplyr::group_by`.

Usage

```
groupByDS(tidy_expr, df.name, .add, .drop)
```

Arguments

`tidy_expr`	Diffused grouping expression.
`df.name`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
`.add`	When FALSE, the default, `group_by()` will override existing groups. To add to the existing groups, use .add = TRUE.
`.drop`	Drop groups formed by factor levels that don't appear in the data? The default is TRUE except when df.name has been previously grouped with .drop = FALSE.

Value

A grouped data frame with class grouped_df, unless the combination of `tidy_expr` and `.add` yields a empty set of grouping columns, in which case a tibble will be returned.

`groupKeysDS` *Performs dplyr* `group_keys`.

Description

DataSHIELD implentation of `dplyr::group_keys`

Usage

```
groupKeysDS(tidy_select, x)
```

Arguments

`tidy_select`	Unused in this function.
`x`	a grouped tibble.

Value

A data frame describing the groups.

`ifElseDS` *Vectorised if-else*

Description

DataSHIELD implentation of `dply::if_else`.

Usage

```
ifElseDS(
  condition = NULL,
  true = NULL,
  false = NULL,
  missing = NULL,
  ptype = NULL,
  size = NULL
)
```

Arguments

condition	A list, specifying a logical vector in tidyverse syntax, ie data and column names unquoted.
true	Vector to use for TRUE value of condition.
false	Vector to use for FALSE value of condition.
missing	If not NULL, will be used as the value for NA values of condition. Follows the same size and type rules as true and false.
ptype	An optional prototype declaring the desired output type. If supplied, this overrides the common type of true, false, and missing.
size	An optional size declaring the desired output size. If supplied, this overrides the size of condition.

Value

A vector with the same size as `condition` and the same type as the common type of `true`, `false`, and `missing`.

`listPermittedTidyverseFunctionsDS`

List of Permitted Tidyverse Functions

Description

This function returns a vector of function names that are permitted to be passed within the dsTidyverse functions, e.g. within the ‘tidy_select‘ argument of ‘ds.mutate.‘

Usage

```
listPermittedTidyverseFunctionsDS()
```

Value

A character vector of function names, each representing a permitted function. Functions not included in this list will be blocked.

mutateDS *Create, modify, and delete columns*

Description

DataSHIELD implentation of mutate.

Usage

```
mutateDS(tidy_expr, df.name, .keep = NULL, .before = NULL, .after = NULL)
```

Arguments

tidy_expr	Name-value pairs. The name gives the name of the column in the output.
df.name	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
.keep	.keep Control which columns from df.name are retained in the output. Grouping columns and columns created by tidy_expr are always kept. • "all": Retains all columns from df.name. This is the default. • "used": Retains only the columns used in tidy_expr to create new columns. • "unused": Retains only the columns not used in tidy_expr to create new columns. This is useful if you generate new columns but no longer need the columns used to generate them. • "none": Doesn't retain any extra columns from df.name. Only the grouping variables and columns created by tidy_expr are kept. Grouping columns and columns created by tidy_expr are always kept.
.before	<tidy-select> Optionally, control where new columns should appear (the default is to add to the right hand side). See relocate for more details.
.after	<tidy-select> Optionally, control where new columns should appear (the default is to add to the right hand side). See relocate for more details.

Value

An object of the same type as df.name, typically a data frame or tibble.

renameDS *Rename columns*

Description

DataSHIELD implentation of`dplyr::rename`.

Usage

```
renameDS(tidy_expr, df.name)
```

Arguments

`tidy_expr`	list containing diffused expression.
`df.name`	A data frame or tibble.

Value

An object of the same type as `df.name`, typically a data frame or tibble.

selectDS *Keep or drop columns using their names and types*

Description

DataSHIELD implentation of `dplyr::select`.

Usage

```
selectDS(tidy_expr, df.name)
```

Arguments

`tidy_expr`	One or more unquoted expressions separated by commas.
`df.name`	A data frame or tibble.

Details

Performs dplyr select

Value

An object of the same type as `df.name`, typically a data frame or tibble.

sliceDS *Subset rows using their positions*

Description

DataSHIELD implentation of `dplyr::slice`.

Usage

```
sliceDS(tidy_expr, df.name, .by, .preserve)
```

Arguments

`tidy_expr`	Provide either positive values to keep, or negative values to drop. The values provided must be either all positive or all negative. Indices beyond the number of rows in the input are silently ignored.
`df.name`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
`.by`	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to `group_by`.
`.preserve`	Relevant when the df.name input is grouped. If .preserve = FALSE (the default), the grouping structure is recalculated based on the resulting data, otherwise the grouping is kept as is.

Value

An object of the same type as `df.name`, typically a data frame or tibble.

ungroupDS *Remove grouping from a tibble or data frame*

Description

DataSHIELD implentation of `dplyr::ungroup`.

Usage

```
ungroupDS(tidy_expr, x)
```

Arguments

`tidy_expr`	Unused in this function.
`x`	A tibble.

Value

An ungrouped data frame or tibble.

Index