Package 'madshapR'

April 23, 2024

Type Package

Title Support Technical Processes Following 'Maelstrom Research' Standards

Version 1.1.0

Description Functions to support rigorous processes in data cleaning, evaluation, and documentation across datasets from different studies based on Maelstrom Research guidelines. The package includes the core functions to evaluate and format the main inputs that define the process, diagnose errors, and summarize and evaluate datasets and their associated data dictionaries. The main outputs are clean datasets and associated metadata, and tabular and visual summary reports. As described in Maelstrom Research guidelines for rigorous retrospective data harmonization (Fortier I and al. (2017) <doi:10.1093/ije/dyw075>).

License GPL-3

LazyData true

Depends R (>= 3.4)

Imports dplyr (>= 1.1.0), rlang, stringr, crayon, ggplot2, tidytext, grDevices, graphics, lubridate, janitor, forcats, knitr, haven, bookdown, stats, DT, readr, tidyr, fs, utils, fabR (>= 2.0.0), lifecycle

URL `https://github.com/maelstrom-research/madshapR`

BugReports `https://github.com/maelstrom-research/madshapR/issues`

RoxygenNote 7.2.3

VignetteBuilder knitr

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Guillaume Fabre [aut, cre] (<`https://orcid.org/0000-0002-0124-9970`>), Maelstrom-research group [cph, fnd]

Maintainer Guillaume Fabre <`guijoseph.fabre@gmail.com`>

Repository CRAN

Date/Publication 2024-04-23 15:10:02 UTC

R topics documented:

`as_category` *Validate and coerce any object as a categorical variable.*

Description

[Experimental] Converts a vector object to a categorical object, typically a column in a data frame. The categories come from non-missing values present in the object and are added to an associated data dictionary (when present).

Usage

```
as_category(x)
```

Arguments

`x`	A vector object to be coerced to categorical.

Value

A vector with class haven_labelled.

See Also

`haven::labelled()`

Examples

```
{

library(dplyr)
mtcars <- tibble(mtcars)
as_category(mtcars[['cyl']])

head(mtcars %>% mutate(cyl = as_category(cyl)))


}
```

`as_dataset` *Validate and coerce any object as a dataset*

Description

Checks if an object is a valid dataset and returns it with the appropriate `madshapR::class` attribute. This function mainly helps validate inputs within other functions of the package but could be used separately to check if a dataset is valid.

Usage

```
as_dataset(object, col_id = NULL)
```

Arguments

`object`	A potential dataset object to be coerced.
`col_id`	An optional character string specifying the name(s) or position(s) of the column(s) used as identifiers.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame with `madshapR::class` 'dataset'.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

###### Example 1: A dataset can have an id column specified as an attribute.
dataset <- as_dataset(madshapR_DEMO$dataset_MELBOURNE, col_id = "id")
glimpse(dataset)

###### Example 2: Any data frame can be a dataset by definition.
glimpse(as_dataset(iris, col_id = "Species"))

}
```

`as_data_dict` *Validate and coerce any object as a data dictionary*

Description

Checks if an object is a valid data dictionary and returns it with the appropriate `madshapR::class` attribute. This function mainly helps validate inputs within other functions of the package but could be used to check if an object is valid for use in a function.

Usage

```
as_data_dict(object)
```

Arguments

`object`	A potential data dictionary object to be coerced.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) with `madshapR::class` 'data_dict'.

See Also

For a better assessment, please use `data_dict_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_PARIS
as_data_dict(data_dict)

}
```

`as_data_dict_mlstr` *Validate and coerce any object as an Opal data dictionary format*

Description

Validates the input object as a valid data dictionary compliant with formats used in Maelstrom Research ecosystem, including Opal, and returns it with the appropriate `madshapR::class` attribute. This function mainly helps validate input within other functions of the package but could be used to check if an object is valid for use in a function.

Usage

```
as_data_dict_mlstr(object, as_data_dict = FALSE, name_standard = FALSE)
```

Arguments

`object`	A potential valid data dictionary to be coerced.
`as_data_dict`	Whether the input data dictionary should not be coerced with specific format restrictions for compatibility with other Maelstrom Research software. FALSE by default.
`name_standard`	Whether the input data dictionary has variable names compatible with Maelstrom Research ecosystem, including Opal)or not. FALSE by default.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

The object may be specifically formatted to be compatible with additional Maelstrom Research software, in particular Opal environments.

Value

A list of data frame(s) with `madshapR::class` 'data_dict_mlstr'.

See Also

For a better assessment, please use `data_dict_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_MELBOURNE
as_data_dict_mlstr(madshapR_DEMO$data_dict_MELBOURNE)

}
```

`as_data_dict_shape` *Validate and coerce any object as a workable data dictionary structure*

Description

Validates the input object as a workable data dictionary structure and returns it with the appropriate `madshapR::class` attribute. This function mainly helps validate input within other functions of the package but could be used to check if a data dictionary is valid for use in a function.

Usage

```
as_data_dict_shape(object)
```

Arguments

`object`	A potential valid data dictionary to be coerced.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) with `madshapR::class` 'data_dict_shape'.

See Also

For a better assessment, please use `data_dict_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_PARIS
as_data_dict_shape(data_dict)

}
```

---

as_dossier *Validate and coerce any object as a dossier (list of dataset(s))*

Description

Checks if an object is a valid dossier (list of datasets) and returns it with the appropriate `madshapR::class` attribute. This function mainly helps validate inputs within other functions of the package but could be used to check if a dossier is valid.

Usage

```
as_dossier(object)
```

Arguments

`object`	A potential dossier object to be coerced.

Details

A dossier is a named list containing at least one data frame or more, each of them being datasets. The name of each tibble will be use as the reference name of the dataset.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A list of data frame(s) with `madshapR::class` 'dossier'.

See Also

For a better assessment, please use `dataset_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)
library(stringr)

###### Example 1: a dataset list is a dossier by definition.
dossier <-
  as_dossier(madshapR_DEMO[str_detect(names(madshapR_DEMO),"dataset_TOKYO")])

glimpse(dossier)

###### Example 2: any list of data frame can be a dossier by
# definition.
glimpse(as_dossier(list(dataset_1 = iris, dataset_2 = mtcars)))

}
```

`as_taxonomy` *Validate and coerce any object as a taxonomy*

Description

Confirms that the input object is a valid taxonomy and returns it as a taxonomy with the appropriate `madshapR::class` attribute. This function mainly helps validate input within other functions of the package but could be used to check if a taxonomy is valid.

Usage

```
as_taxonomy(object)
```

Arguments

`object`	A potential taxonomy to be coerced.

Details

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

Value

A list of data frame(s) with `madshapR::class` 'taxonomy'.

See Also

Opal documentation

Examples

```
{

# use madshapR_DEMO provided by the package

###### Example
as_taxonomy(madshapR_DEMO$taxonomy_PARIS)

}
```

`as_valueType` *Validate and coerce any object according to a given valueType*

Description

Attributes a valueType to an object, that can be a vector, or in a data frame using dplyr::mutate.

Usage

```
as_valueType(x, valueType = "text")
```

Arguments

`x`	Object to be coerced. Can be a vector.
`valueType`	A character string of the valueType used to coerce x.

Details

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

The object coerced accordingly to the input valueType.

See Also

Opal documentation

Examples

```
{

# use madshapR_DEMO provided by the package

dataset <- madshapR_DEMO$dataset_TOKYO
as_valueType(head(dataset$dob),'date')

# as_valueType is compatible with tidyverse philosophy
library(dplyr)
mtcars %>% mutate(cyl = as_valueType(cyl,'integer')) %>% head()

}
```

`bookdown_open` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

fabR `bookdown_open`

`bookdown_render` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

fabR `bookdown_render`

`bookdown_template` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

fabR `bookdown_template`

check_dataset_categories

Assess a data dictionary and associated dataset for category differences

Description

Generates a data frame report of any categorical value options (the combination of 'variable' and 'name' in 'Categories') in a data dictionary that are not in the associated dataset and any categorical variable values in a dataset that are not declared in the associated data dictionary. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_dataset_categories(
  dataset,
  data_dict = silently_run(data_dict_extract(dataset))
)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing categorical values which differ between dataset and their data dictionary.

Examples

```
{

# use madshapR_DEMO provided by the package
library(tidyr)

data_dict <-
  madshapR_DEMO$`data_dict_TOKYO - errors with data` %>%
  data_dict_filter('name == "prg_ever"')
dataset <- madshapR_DEMO$`dataset_TOKYO - errors with data`['prg_ever']

check_dataset_categories(dataset, data_dict)

}
```

`check_dataset_valueType` *Assess a data dictionary and associated dataset for valueType differences*

Description

Generates a data frame report of any incompatibility between variable values in a dataset and the declared valueType in the associated data dictionary. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_dataset_valueType(dataset, data_dict = NULL, valueType_guess = FALSE)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata to be evaluated.
`valueType_guess`	Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A data frame providing values which valueType differs between dataset and their data dictionary.

Examples

```
{

check_dataset_valueType(mtcars[2], valueType_guess = TRUE)

}
```

`check_dataset_variables` *Assess a data dictionary and associated dataset for undeclared variables*

Description

Generates a data frame report of any variable that is present in a dataset but not in the associated data dictionary or present in a data dictionary but not in the associated dataset. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_dataset_variables(dataset, data_dict = NULL)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing undeclared variables across a data dictionary.

Examples

```
{

# use madshapR_DEMO provided by the package

dataset <- madshapR_DEMO$`dataset_TOKYO - errors with data`
data_dict <- madshapR_DEMO$`data_dict_TOKYO - errors`
check_dataset_variables(dataset,data_dict)

}
```

---

check_data_dict_categories

Assess a data dictionary for potential issues in categories

Description

Generates a data frame report of any categorical variable name present in the 'Categories' element but not present in 'Variables'. The data frame also reports any non-unique combinations of 'variable' and 'name' in the 'Categories' element. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_data_dict_categories(data_dict)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A data frame providing categorical variables that has issues within a data dictionary.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_TOKYO - errors`
check_data_dict_categories(data_dict)

}
```

`check_data_dict_missing_categories`
Assess categorical variables for non-Boolean values in 'missing' column

---

Description

Generates a data frame report of any categorical variables with non-Boolean (or compatible with boolean) values in the 'missing' column of the 'Categories' element. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_data_dict_missing_categories(data_dict)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A data frame providing categorical values which 'missing' column is not a boolean.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_TOKYO - errors`
check_data_dict_missing_categories(data_dict)

}
```

`check_data_dict_valueType`
Assess a data dictionary for non-valid valueType values

Description

Generates a data frame report of any variable with a valueType that is not in the list of allowed valueType values. This function also assesses if the valueType is compatible with any associated categorical values declared. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_data_dict_valueType(data_dict)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A data frame providing non-standard valueType declared in a data dictionary.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_TOKYO - errors`
check_data_dict_valueType(data_dict)

}
```

`check_data_dict_variables` *Assess a data dictionary for potential issues in variables*

Description

Generates a data frame report of any non-unique variable names in the 'Variables' element. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_data_dict_variables(data_dict)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A data frame providing non unique variables across a data dictionary.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_TOKYO - errors`
check_data_dict_variables(data_dict)

}
```

`check_name_standards` *Assess variable names in a data dictionary for non-standard formats*

Description

Generates a data frame report of any variable names that are not compatible in Maelstrom Research ecosystem, including Opal. This report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
check_name_standards(var_names)
```

Arguments

`var_names`	A character vector of names.

Details

The object may be specifically formatted to be compatible with additional Maelstrom Research software, in particular Opal environments.

Value

A data frame providing non-standard names across a vector.

Examples

```
{

# use madshapR_DEMO provided by the package

check_name_standards(c("coucou", "cou cou", "$coucou",NA))
check_name_standards(
 madshapR_DEMO$`data_dict_TOKYO - errors`$Variables$name)

}
```

---

col_id *Return the id column names(s) of a dataset*

Description

Return the id column names(s) of a dataset if any. If not, the function returns a NULL object.

Usage

```
col_id(dataset)
```

Arguments

`dataset`	A data frame object.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

Name(s) of identifier column(s). NULL if not.

Examples

```
{

col_id(iris)

library(fabR)
iris <- add_index(iris)
iris <- as_dataset(iris, col_id = 'index')
col_id(iris)
}
```

`dataset_cat_as_labels` *Apply data dictionary category labels to the associated dataset variables*

Description

Applies category labels declared in a data dictionary to the associated columns (variables) in the dataset.

Usage

```
dataset_cat_as_labels(dataset, data_dict = NULL, col_names = names(dataset))
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`col_names`	A character string specifying the name(s) of the column(s) which refer to existing column(s) in the dataset. The column(s) can be named or indicated by position.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A data frame identifying a dataset.

Examples

```
{

dataset = madshapR_DEMO$dataset_PARIS
data_dict = as_data_dict_mlstr(madshapR_DEMO$data_dict_PARIS)
dataset_cat_as_labels(dataset, data_dict, col_names = 'SEX')

}
```

---

dataset_evaluate *Generate an assessment report for a dataset*

Description

Assesses the content and structure of a dataset object and generates reports of the results. This function can be used to evaluate data structure, presence of specific fields, coherence across elements, and data dictionary formats.

Usage

```
dataset_evaluate(
  dataset,
  data_dict = NULL,
  valueType_guess = FALSE,
  as_data_dict_mlstr = TRUE,
  taxonomy = NULL,
  dataset_name = .dataset_name,
  .dataset_name = NULL
)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`valueType_guess`	Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.
`as_data_dict_mlstr`	Whether the input data dictionary should be coerced with specific format restrictions for compatibility with other Maelstrom Research software. TRUE by default.
`taxonomy`	An optional data frame identifying a variable classification schema.
`dataset_name`	A character string specifying the name of the dataset (used internally in the function `dossier_evaluate()`).
`.dataset_name`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`. The function truncates each cell to a maximum of 10000 characters, to be readable and compatible with Excel.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

The object may be specifically formatted to be compatible with additional Maelstrom Research software, in particular Opal environments.

Value

A list of data frames containing assessment reports.

See Also

`dossier_evaluate()`

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

###### Example : Any data frame can be summarized
dataset <- as_dataset(
  madshapR_DEMO$`dataset_TOKYO - errors with data`,
  col_id = 'part_id') %>% slice(0)

glimpse(dataset_evaluate(dataset,as_data_dict_mlstr = FALSE))

}
```

`dataset_preprocess` *Generate an evaluation of all variable values in a dataset*

Description

Analyses the content of a dataset and its data dictionary (if any), identifies variable(s) data type and values accordingly and preprocess the variables. The elements of the data frame generated are evaluation of valid/non valid/missing values (based on the data dictionary information if provided). This function can be used to personalize report parameters and is internally used in the function `dataset_summarize()`.

Generates a data frame that evaluates and aggregates all columns in a dataset with (if any) its data dictionary. The data dictionary (if present) separates observations between open values, missing values, categorical values , and categorical missing values (which corresponds to the 'missing' column in the 'Categories' sheet). This internal function is mainly used inside summary functions.

Usage

```
dataset_preprocess(dataset, data_dict = NULL)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing summary elements of a dataset, including its values and data dictionary elements.

See Also

`summary_variables()`

Examples

```
{

###### Example : Any data frame can be a dataset by definition.
head(dataset_preprocess(dataset = iris))

}
```

`dataset_summarize` *Generate an assessment report and summary of a dataset*

Description

Assesses and summarizes the content and structure of a dataset and generates reports of the results. This function can be used to evaluate data structure, presence of specific fields, coherence across elements, and data dictionary formats, and to summarize additional information about variable distributions and descriptive statistics.

Usage

```
dataset_summarize(
  dataset,
  data_dict = data_dict_extract(dataset),
  group_by = NULL,
  taxonomy = NULL,
  dataset_name = .dataset_name,
  valueType_guess = FALSE,
  .dataset_name = NULL
)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`group_by`	A character string identifying the column in the dataset to use as a grouping variable. Elements will be grouped by this column.
`taxonomy`	An optional data frame identifying a variable classification schema.
`dataset_name`	A character string specifying the name of the dataset (internally used in the function `dossier_evaluate()`).
`valueType_guess`	Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.
`.dataset_name`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`. The function truncates each cell to a maximum of 10000 characters, to be readable and compatible with Excel.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A list of data frames containing assessment reports and summaries.

See Also

`dossier_evaluate()`

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

#' ###### Example : Any data frame can be summarized
dataset <- iris['Sepal.Width']
glimpse(dataset_summarize(dataset))

}
```

`dataset_visualize` *Generate a web-based visual report for a dataset*

Description

Generates a visual report of a dataset in an HTML bookdown document, with summary figures and statistics for each variable. The report outputs can be grouped by a categorical variable.

Usage

```
dataset_visualize(
  dataset = tibble(id = as.character()),
  bookdown_path,
  data_dict = data_dict_extract(dataset),
  group_by = NULL,
  valueType_guess = FALSE,
  taxonomy = NULL,
  dataset_name = .dataset_name,
  dataset_summary = .summary_var,
  .summary_var = NULL,
  .dataset_name = NULL
)
```

Arguments

`dataset`	A dataset object.
`bookdown_path`	A character string identifying the folder path where the bookdown report files will be saved.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`group_by`	A character string identifying the column in the dataset to use as a grouping variable. Elements will be grouped by this column.
`valueType_guess`	Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.
`taxonomy`	An optional data frame identifying a variable classification schema.
`dataset_name`	A character string specifying the name of the dataset (used internally in the function `dossier_evaluate()`).
`dataset_summary`	A list which identifies an existing summary produced by `dataset_summarize()` of the dataset. Using this parameter can save time in generating the visual report.
`.summary_var`	**[Deprecated]**
`.dataset_name`	**[Deprecated]**

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

Value

A folder containing files for the bookdown site. To open the bookdown site in a browser, open 'docs/index.html', or use `bookdown_open()` with the folder path.

See Also

`bookdown_open()` `as_category()`

Examples

```
{

# You can use our demonstration files to run examples

library(fs)
library(dplyr)

dataset <- madshapR_DEMO$dataset_TOKYO['height'] %>% slice(0)
dataset_summary <- madshapR_DEMO$`dataset_summary`

if(dir_exists(tempdir())) dir_delete(tempdir())
bookdown_path <- tempdir()

dataset_visualize(
  dataset,
  dataset_summary = dataset_summary,
  bookdown_path = bookdown_path)

# To open the file in browser, open 'bookdown_path/docs/index.html'.
# Or use bookdown_open(bookdown_path) function.

}
```

`dataset_zap_data_dict` *Remove labels (attributes) from a data frame, leaving its unlabelled columns*

Description

Removes any attributes attached to a data frame. Any value in columns will be preserved. Any 'Date' (typeof) column will be recast as character to preserve information.

Usage

```
dataset_zap_data_dict(dataset)
```

Arguments

`dataset`	A dataset object.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame identifying a dataset.

See Also

`haven::zap_labels()`.

Examples

```
{

# use madshapR_DEMO provided by the package

dataset <- madshapR_DEMO$dataset_TOKYO
data_dict <- as_data_dict_mlstr(madshapR_DEMO$data_dict_TOKYO)
dataset <- data_dict_apply(dataset,data_dict)
head(dataset_zap_data_dict(dataset))

}
```

`data_dict_apply` *Apply a data dictionary to a dataset*

Description

Applies a data dictionary to a dataset, creating a labelled dataset with variable attributes. Any previous attributes will be preserved. For variables that are factors, variables will be transformed into haven-labelled variables.

Usage

```
data_dict_apply(dataset, data_dict = NULL)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A labelled data frame with metadata as attributes, specified for each variable from the input data dictionary.

See Also

`attributes()`, `haven::labelled()`

Examples

```
{

# use madshapR_DEMO provided by the package

dataset <- madshapR_DEMO$dataset_MELBOURNE
data_dict <- as_data_dict_mlstr(madshapR_DEMO$data_dict_MELBOURNE)
head(data_dict_apply(dataset, data_dict))

}
```

`data_dict_collapse` *Transform multi-row category column(s) to single rows and join to "Variables"*

Description

Collapses a data dictionary element (the parameter 'from'), into column(s) in another element (the parameter 'to') If the element 'to' exists, and contains any column 'xx' or 'yy', these columns will be added to the element 'from' under the names 'to:xx' and 'to:yy'. (unique names will be generated if necessary). Each element of these column will gather all information to process the reverse operation. Separator of each element is the following structure : 'name = xx1 ; name = xx2'. This function is mainly used to collapse the 'Categories' element into columns in 'Variables'. This function is the reversed operation of `data_dict_expand()`

Usage

```
data_dict_collapse(
  data_dict,
  from = "Categories",
  to = "Variables",
  name_prefix = "Categories::"
)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.
`from`	A symbol identifying the name of the element (data frame) to take column(s) from. Default is 'Categories'.
`to`	A symbol identifying the name of the element (data frame) to create column(s) to. Default is 'Variables'.
`name_prefix`	A character string of the prefix of columns of interest. This prefix will be used to select columns, and to rename them in the 'to' element. Default is 'Categories::'.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a data dictionary.

See Also

`data_dict_expand()`

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_MELBOURNE
data_dict_collapse(data_dict)

}
```

`data_dict_evaluate` *Generate an assessment report for a data dictionary*

Description

Assesses the content and structure of a data dictionary and generates reports of the results. The report can be used to help assess data dictionary structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats.

Usage

```
data_dict_evaluate(data_dict, taxonomy = NULL, as_data_dict_mlstr = TRUE)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be evaluated.
`taxonomy`	An optional data frame identifying a variable classification schema.
`as_data_dict_mlstr`	Whether the input data dictionary should be coerced with specific format restrictions for compatibility with other Maelstrom Research software. TRUE by default.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`. The function truncates each cell to a maximum of 10000 characters, to be readable and compatible with Excel.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

The object may be specifically formatted to be compatible with additional Maelstrom Research software, in particular Opal environments.

Value

A list of data frames containing assessment reports.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

data_dict <- madshapR_DEMO$`data_dict_TOKYO - errors`
glimpse(data_dict_evaluate(data_dict))

}
```

`data_dict_expand` *Transform single-row category information to multiple rows as element*

Description

Expands data dictionary column(s) in a element (the parameter 'from'), into another element (the parameter 'to'). If the element `from` contains any column starting with 'prefix', (xx,yy), these columns will be added as 'xx' and 'yy' in the element identified by `to`. This data frame will be created if necessary, and columns will be added, from left to right. (unique names will be generated if necessary). Separator of each element is the following structure : 'name = xx1 ; name = xx2'. This function is mainly used to expand the column(s) 'Categories::xx' in "Variables" to "Categories" element with column(s) xx. This function is the reversed operation of `data_dict_collapse()`

Usage

```
data_dict_expand(
  data_dict,
  from = "Variables",
  name_prefix = "Categories::",
  to = "Categories"
)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.
`from`	A symbol identifying the name of the element (data frame) to take column(s) from. Default is 'Variables'.
`name_prefix`	Character string of the prefix of columns of interest. This prefix will be used to select columns, and to rename them in the 'to' element. Default is 'Categories::'.
`to`	A symbol identifying the name of the element (data frame) to create column(s) to. Default is 'Categories'.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a data dictionary.

See Also

`data_dict_collapse()`

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_PARIS - collapsed`
data_dict_expand(data_dict)

}
```

`data_dict_extract` *Generate a data dictionary from a dataset*

Description

Generates a data dictionary from a dataset. If the dataset variables have no associated metadata, a minimum data dictionary is created by using variable attributes.

Usage

```
data_dict_extract(dataset, as_data_dict_mlstr = TRUE)
```

Arguments

`dataset`	A dataset object.
`as_data_dict_mlstr`	Whether the input data dictionary should be coerced with specific format restrictions for compatibility with other Maelstrom Research software. TRUE by default.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame

'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

The object may be specifically formatted to be compatible with additional Maelstrom Research software, in particular Opal environments.

Value

A list of data frame(s) representing metadata of the dataset variables.

Examples

```
{

# use madshapR_DEMO provided by the package

###### Example 2: extract data dictionary from any dataset (the
# data dictionary will be created upon attributes of the dataset. Factors
# will be considered as categorical variables)
data_dict_extract(iris)

}
```

`data_dict_filter` *Subset data dictionary by row values*

Description

Subsets either or both the 'Variables' and 'Categories' elements of a data dictionary. Rows are conserved if their values satisfy the condition. This is a wrapper function analogous to `dplyr::filter()`.

Usage

```
data_dict_filter(
  data_dict,
  filter_var = NULL,
  filter_cat = NULL,
  filter_all = NULL
)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be filtered.
`filter_var`	Expressions that are defined in the element 'Variables' in the data dictionary.
`filter_cat`	Expressions that are defined in the element 'Categories' in the data dictionary.
`filter_all`	Expressions that are defined both in the 'Categories' and 'Variables' in the data dictionary.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a workable data dictionary structure.

See Also

`dplyr::filter()`

Examples

```
{

# use madshapR_DEMO provided by the package

# Create a list of data dictionaries where the column 'table' is added to
# refer to the associated dataset. The object created is not a
# data dictionary per say, but can be used as a structure which can be
# shaped into a data dictionary.
library(dplyr)

data_dict_list <- list(
  data_dict_1 <- madshapR_DEMO$data_dict_TOKYO ,
  data_dict_2 <- madshapR_DEMO$data_dict_MELBOURNE)
names(data_dict_list) = c("dataset_TOKYO","dataset_MELBOURNE")

data_dict_nest <- data_dict_list_nest(data_dict_list, name_group = 'table')

###### Example 1 search and filter through a column in 'Variables' element
data_dict_filter(data_dict_nest,filter_var = "valueType == 'text'")

###### Example 2 search and filter through a column in 'Categories' element
data_dict_filter(data_dict_nest,filter_cat = "missing == TRUE")

###### Example 3 search and filter through* a column in 'Variables' element.
# The column must exist in both 'Variables' and 'Categories' and have the
# same meaning
data_dict_filter(data_dict_nest,filter_all = "table == 'dataset_TOKYO'")

}
```

data_dict_group_by *Group listed data dictionaries by specified column names*

Description

Groups the data dictionary element(s) by the groups defined by the query. This function groups both the 'Variables' and 'Categories' elements (if the group exists under the same definition in in both). This function is analogous to running `dplyr::group_by()`. Each element is named using the group values. `data_dict_ungroup()` reverses the effect.

Usage

```
data_dict_group_by(data_dict, col)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.
`col`	variable to group by.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a workable data dictionary structure.

See Also

`dplyr::group_by()`, `data_dict_ungroup()`

Examples

```
{

# use madshapR_DEMO provided by the package
# Create a list of data dictionaries where the column 'table' is added to
# refer to the associated dataset. The object created is not a
# data dictionary per say, but can be used as a structure which can be
# shaped into a data dictionary.

data_dict_list <- list(
  data_dict_1 <- madshapR_DEMO$data_dict_TOKYO ,
  data_dict_2 <- madshapR_DEMO$data_dict_MELBOURNE)
names(data_dict_list) = c("dataset_TOKYO","dataset_MELBOURNE")

data_dict_nest <- data_dict_list_nest(data_dict_list, name_group = 'table')

data_dict_group_by(data_dict_nest, col = "table")

}
```

---

data_dict_group_split *Split grouped data dictionaries into a named list*

Description

Divides data dictionary element(s) into the groups defined by the query. This function divides both the 'Variables' and 'Categories' elements (if the group exists under the same definition in in both) into a list of data dictionaries, each with the rows of the associated group and all the original columns, including grouping variables. This function is analogous to running `dplyr::group_by()` and `dplyr::group_split()`. Each element is named using the group values. `data_dict_list_nest()` reverses the effect.

Usage

```
data_dict_group_split(data_dict, ...)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.
`...`	Column in the data dictionary to split it by. If not provided, the splitting will be done on the grouping element of a grouped data dictionary.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a list of workable data dictionary structure.

See Also

`dplyr::group_by()`, `dplyr::group_split()` , `data_dict_group_by()`, `data_dict_list_nest()`

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

# Create a list of data dictionaries where the column 'table' is added to
# refer to the associated dataset. The object created is not a
# data dictionary per say, but can be used as a structure which can be
# shaped into a data dictionary.

data_dict_list <- list(
  data_dict_1 <- madshapR_DEMO$data_dict_TOKYO ,
  data_dict_2 <- madshapR_DEMO$data_dict_MELBOURNE)
names(data_dict_list) = c("dataset_TOKYO","dataset_MELBOURNE")

data_dict_nest <-
  data_dict_list_nest(data_dict_list, name_group = 'table') %>%
  data_dict_group_by(col = "table")

glimpse(data_dict_group_split(data_dict_nest,col = "table"))

}
```

`data_dict_list_nest` *Bind listed data dictionaries*

Description

Binds a list of data dictionaries into one data dictionary. This is a wrapper function analogous to `dplyr::bind_rows()`.

Usage

```
data_dict_list_nest(data_dict_list, name_group = NULL)
```

Arguments

`data_dict_list`	A list of data frame(s) representing metadata to be transformed.
`name_group`	A character string of one column in the dataset that can be taken as a grouping column.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a workable data dictionary structure.

See Also

`dplyr::bind_rows()`

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

# Create a list of data dictionaries where the column 'table' is added to
# refer to the associated dataset. The object created is not a
# data dictionary per say, but can be used as a structure which can be
# shaped into a data dictionary.

data_dict_list <- list(
  data_dict_1 <- madshapR_DEMO$data_dict_TOKYO ,
  data_dict_2 <- madshapR_DEMO$data_dict_MELBOURNE)
names(data_dict_list) = c("dataset_TOKYO","dataset_MELBOURNE")

glimpse(data_dict_list_nest(data_dict_list, name_group = 'table'))

}
```

`data_dict_match_dataset`

Inner join between a dataset and its associated data dictionary

---

Description

Performs an inner join between a dataset and its associated data dictionary, keeping only variables present in both. This function returns the matched dataset rows, the matched data dictionary rows, or both, in a list.

Usage

```
data_dict_match_dataset(
  dataset,
  data_dict,
  data_dict_apply = FALSE,
  output = c("dataset", "data_dict")
)
```

Arguments

`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset.
`data_dict_apply`	Whether data dictionary(ies) should be applied to associated dataset(s), creating labelled dataset(s) with variable attributes. Any previous attributes will be preserved. FALSE by default.
`output`	A vector of character string which indicates if the function returns a dataset ('dataset'), data dictionary ('data_dict') of both. Default is c('dataset','data_dict').

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

Either a data frame, identifying the dataset, or a list of data frame(s) identifying a data dictionary. Returns both in a list by default.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

dataset <- madshapR_DEMO$dataset_MELBOURNE %>% select(-1)
data_dict <- madshapR_DEMO$data_dict_MELBOURNE
head(data_dict_match_dataset(dataset, data_dict, out = 'dataset'))
glimpse(data_dict_match_dataset(dataset, data_dict, out = 'data_dict'))

}
```

data_dict_pivot_longer *Transform column(s) of a data dictionary from wide format to long format*

Description

Transforms column(s) of a data dictionary from wide format to long format. If a taxonomy is provided, the corresponding columns in the data dictionary will be converted to a standardized format with fewer columns. This operation is equivalent to performing a `tidyr::pivot_longer()` on these columns following the taxonomy structure provided. Variable names in the data dictionary must be unique.

Usage

```
data_dict_pivot_longer(data_dict, taxonomy = NULL)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.
`taxonomy`	An optional data frame identifying a variable classification schema.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

Value

A list of data frame(s) identifying a data dictionary.

See Also

`tidyr::pivot_longer()`, `as_data_dict()`

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_PARIS - collapsed`
taxonomy <- madshapR_DEMO$taxonomy_PARIS
data_dict_pivot_longer(data_dict,taxonomy)

}
```

`data_dict_pivot_wider` *Transform column(s) of a data dictionary from long format to wide format*

Description

Transforms column(s) of a data dictionary from long format to wide format. If a taxonomy is provided, the corresponding columns in the data dictionary will be converted to a format with the taxonomy expanded. This operation is equivalent to performing a `tidyr::pivot_wider()` on these columns following the taxonomy structure provided. Variable names in the data dictionary must be unique.

Usage

```
data_dict_pivot_wider(data_dict, taxonomy = NULL)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.
`taxonomy`	An optional data frame identifying a variable classification schema.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

Value

A list of data frame(s) identifying a data dictionary.

See Also

`tidyr::pivot_wider()`, `as_data_dict()`

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$`data_dict_PARIS - collapsed`
taxonomy  <- madshapR_DEMO$taxonomy_PARIS
data_dict_pivot_wider(data_dict, taxonomy)

}
```

`data_dict_ungroup` *Ungroup data dictionary*

Description

Ungroups the data dictionary element(s). This function ungroups both the 'Variables' and 'Categories' elements (if both are grouped data frames). This function is analogous to running `dplyr::ungroup()`. `data_dict_group_by()` allows to group a data dictionary and this function reverses the effect.

Usage

```
data_dict_ungroup(data_dict)
```

Arguments

`data_dict`	A list of data frame(s) representing metadata to be transformed.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A list of data frame(s) identifying a workable data dictionary structure.

See Also

dplyr::ungroup() data_dict_group_by()

Examples

```
{

# use madshapR_DEMO provided by the package
# Create a list of data dictionaries where the column 'table' is added to
# refer to the associated dataset. The object created is not a
# data dictionary per say, but can be used as a structure which can be
# shaped into a data dictionary.

library(dplyr)

data_dict_list <- list(
  data_dict_1 <- madshapR_DEMO$data_dict_TOKYO ,
  data_dict_2 <- madshapR_DEMO$data_dict_MELBOURNE)
names(data_dict_list) = c("dataset_TOKYO","dataset_MELBOURNE")

data_dict_nest <-
  data_dict_list_nest(data_dict_list, name_group = 'table') %>%
  data_dict_group_by(col = "table")

 data_dict_ungroup(data_dict_nest)
}
```

---

data_extract   *Create an empty dataset from a data dictionary*

Description

Creates an empty dataset using information contained in a data dictionary. The column names are taken from 'name' in the 'Variables' element of the data dictionary. If a 'valueType' or alternatively 'typeof' column is provided, the class of each column is set accordingly (default is text).

Usage

```
data_extract(data_dict, data_dict_apply = FALSE)
```

Arguments

data_dict	A list of data frame(s) representing metadata.
data_dict_apply	Whether data dictionary(ies) should be applied to associated dataset(s), creating labelled dataset(s) with variable attributes. Any previous attributes will be preserved. FALSE by default.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A data frame identifying the dataset created from the variable names list in 'Variables' element of the data dictionary.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_MELBOURNE
data_extract(data_dict)

}
```

`dossier_create` *Generate a dossier from a list of one or more datasets*

Description

Generates a dossier object (list of one or more datasets).

Usage

```
dossier_create(dataset_list, data_dict_apply = FALSE)
```

Arguments

`dataset_list`	A list of data frame, each of them being dataset object.
`data_dict_apply`	Whether data dictionary(ies) should be applied to associated dataset(s), creating labelled dataset(s) with variable attributes. Any previous attributes will be preserved. FALSE by default.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A list of data frame(s), containing input dataset(s).

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

###### Example 1: datasets can be gathered into a dossier which is a list.
dossier <- dossier_create(
 dataset_list = list(
   dataset_MELBOURNE = madshapR_DEMO$dataset_MELBOURNE,
   dataset_PARIS = madshapR_DEMO$dataset_PARIS ))

glimpse(dossier)

###### Example 2: Any data frame can be gathered into a dossier
glimpse(dossier_create(list(iris, mtcars)))

}
```

`dossier_evaluate` *Generate an assessment report of a dossier*

Description

Assesses the content and structure of a dossier object (list of datasets) and generates reports of the results. This function can be used to evaluate data structure, presence of specific fields, coherence across elements, and data dictionary formats.

Usage

```
dossier_evaluate(dossier, taxonomy = NULL, as_data_dict_mlstr = TRUE)
```

Arguments

`dossier`	List of data frame, each of them being datasets.
`taxonomy`	An optional data frame identifying a variable classification schema.
`as_data_dict_mlstr`	Whether the input data dictionary should be coerced with specific format restrictions for compatibility with other Maelstrom Research software. TRUE by default.

Details

A dossier is a named list containing at least one data frame or more, each of them being datasets. The name of each data frame will be use as the reference name of the dataset.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

The object may be specifically formatted to be compatible with additional Maelstrom Research software, in particular Opal environments.

Value

A list of data frames containing assessment reports.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

###### Example : a dataset list is a dossier by definition.

dataset <- as_dataset(
   madshapR_DEMO$`dataset_TOKYO - errors with data`,
   col_id = 'part_id') %>% slice(0)

dossier <- as_dossier(list(dataset = dataset))

glimpse(dossier_evaluate(dossier,as_data_dict_mlstr = FALSE))

}
```

`dossier_summarize` *Generate an assessment report and summary of a dossier*

Description

Assesses and summarizes the content and structure of a dossier (list of datasets) and generates reports of the results. This function can be used to evaluate data structure, presence of specific fields, coherence across elements, and data dictionary formats, and to summarize additional information about variable distributions and descriptive statistics.

Usage

```
dossier_summarize(
  dossier,
  group_by = NULL,
  taxonomy = NULL,
  valueType_guess = FALSE
)
```

Arguments

`dossier`	List of data frame(s), each of them being datasets.
`group_by`	A character string identifying the column in the dataset to use as a grouping variable. Elements will be grouped by this column.
`taxonomy`	An optional data frame identifying a variable classification schema.
`valueType_guess`	Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.

Details

A dossier is a named list containing at least one data frame or more, each of them being datasets. The name of each data frame will be use as the reference name of the dataset.

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A list of data frames containing overall assessment reports and summaries grouped by dataset.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

###### Example 1: Combine functions and summarize datasets.
dossier <- list(iris = tibble())

dossier_summary <- dossier_summarize(dossier)
glimpse(dossier_summary)

}
```

`drop_category` *Validate and coerce any object as a non-categorical variable.*

Description

[Experimental] Converts a vector object to a non-categorical object, typically a column in a data frame. The categories come from non-missing values present in the object and are suppressed from an associated data dictionary (when present).

Usage

```
drop_category(x)
```

Arguments

`x`	object to be coerced.

Value

A R object.

Examples

```
{

head(iris[['Species']])
head(drop_category(iris[['Species']]))

}
```

`is_category` *Test if an object is a valid dataset*

Description

Tests if the input object is a valid dataset. This function mainly helps validate input within other functions of the package but could be used to check if a dataset is valid.

[Experimental] Test if vector object is a categorical variable, typically a column in a data frame. This function mainly helps validate input within other functions of the package.

Usage

```
is_category(x, threshold = NULL)
```

Arguments

`x`	object to be coerced.
`threshold`	Optional. The function returns TRUE if the number of unique values in the input vector is lower.

Value

A logical.

Examples

```
{

library(dplyr)
iris %>% summarise(across(everything(), is_category))
is_category(iris[['Species']])

}
```

`is_dataset` *Test if an object is a valid dataset*

Description

Tests if the input object is a valid dataset. This function mainly helps validate input within other functions of the package but could be used to check if a dataset is valid.

Usage

```
is_dataset(object)
```

Arguments

`object`	A potential dataset to be evaluated.

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A logical.

See Also

For a better assessment, please use `dataset_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package
# any data frame can be a dataset by definition.

is_dataset(madshapR_DEMO$dataset_MELBOURNE)
is_dataset(iris)
is_dataset(AirPassengers)

}
```

`is_data_dict` *Test if an object is a valid data dictionary*

Description

Tests if the input object is a valid data dictionary. This function mainly helps validate input within other functions of the package but could be used to check if an object is valid for use in a function.

Usage

```
is_data_dict(object)
```

Arguments

`object`	A potential data dictionary to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A logical.

See Also

For a better assessment, please use `data_dict_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_MELBOURNE
is_data_dict(data_dict)
is_data_dict(iris)

}
```

`is_data_dict_mlstr` *Test if an object is a valid Maelstrom data dictionary*

Description

Tests if the input object is a valid data dictionary compliant with formats used in Maelstrom Research ecosystem, including Opal. This function mainly helps validate input within other functions of the package but could be used to check if an object is valid for use in a function.

Usage

```
is_data_dict_mlstr(object)
```

Arguments

`object`	A potential Maelstrom formatted data dictionary to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A logical.

See Also

For a better assessment, please use `data_dict_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_MELBOURNE
is_data_dict_mlstr(data_dict)
is_data_dict_mlstr(iris)

}
```

`is_data_dict_shape` *Test if an object is a workable data dictionary structure*

Description

Tests if the input object has adequate structure to work with functions involving data dictionary shaping. This function mainly helps validate input within other functions of the package but could be used to check if an object is valid for use in a function.

Usage

```
is_data_dict_shape(object)
```

Arguments

`object`	A potential data dictionary structure to be evaluated.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

Value

A logical.

See Also

For a better assessment, please use `data_dict_evaluate()`.

Examples

```
{

# use madshapR_DEMO provided by the package

data_dict <- madshapR_DEMO$data_dict_MELBOURNE
is_data_dict_shape(data_dict)
is_data_dict_shape(iris)

}
```

`is_dossier` *Test if an object is a valid dossier (list of dataset(s))*

Description

Tests if the input object is a valid dossier. This function mainly helps validate input within other functions of the package but could be used to check if a dossier is valid.

Usage

```
is_dossier(object)
```

Arguments

`object`	A potential dossier to be evaluated.

Details

A dossier is a named list containing at least one data frame or more, each of them being datasets. The name of each tibble will be use as the reference name of the dataset.

Value

A logical.

Examples

```
{

# use madshapR_DEMO provided by the package
# Any list of data frame can be a dossier by definition.
library(stringr)

is_dossier(madshapR_DEMO[str_detect(names(madshapR_DEMO),"dataset")])
is_dossier(list(dataset_1 = iris, dataset_2 = mtcars))
is_dossier(iris)

}
```

`is_taxonomy` *Test if an object is a valid taxonomy*

Description

Confirms whether the input object is a valid taxonomy. This function mainly helps validate input within other functions of the package but could be used to check if a taxonomy is valid.

Usage

```
is_taxonomy(object)
```

Arguments

`object`	A potential taxonomy to be evaluated.

Details

A taxonomy is a classification schema that can be defined for variable attributes. A taxonomy is usually extracted from an Opal environment, and a taxonomy object is a data frame that must contain at least the columns `taxonomy`, `vocabulary`, and `terms`. Additional details about Opal taxonomies are available online.

Value

A logical.

Examples

```
{

# use madshapR_DEMO provided by the package

is_taxonomy(madshapR_DEMO$taxonomy_PARIS)

}
```

`is_valueType` *Test if a character object is one of the valid valueType values*

Description

Confirms whether the input object is a valid valueType. This function mainly helps validate input within other functions of the package but could be used to check if a valueType is valid.

Usage

```
is_valueType(object)
```

Arguments

`object`	A potential valueType name to be evaluated.

Details

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A logical.

See Also

Opal documentation

Examples

```
{

is_valueType('integer')
is_valueType('integre')

}
```

`madshapR_DEMO` *Built-in material allowing the user to test the package with demo data*

Description

Demo datasets and data dictionaries, and taxonomy, to provide illustrative examples of objects used by madshapR.

Usage

```
madshapR_DEMO
```

Format

`list`:

A list with 12 elements (data frames and lists) providing example objects for testing the package:

data_dict_MELBOURNE Example Data dictionary for Melbourne dataset

data_dict_PARIS Example Data dictionary for Paris dataset

data_dict_PARIS - collapsed Example Data dictionary for Paris with collapsed categories

data_dict_TOKYO Example Data dictionary for Tokyo dataset

data_dict_TOKYO - errors Data dictionary for Tokyo dataset with errors

data_dict_TOKYO - errors with data Example Data Dictionary for Tokyo dataset with errors with Tokyo dataset

dataset_MELBOURNE Example Dataset for MELBOURNE dataset

dataset_PARIS Example Dataset for PARIS dataset

dataset_TOKYO Example Dataset for TOKYO dataset

dataset_TOKYO - errors with data Example dataset of Tokyo with errors with Tokyo data dictionary

taxonomy_PARIS Example Taxonomy for Paris dataset

dataset_summary Example of dataset summary ...

Examples

```
{

 print(madshapR_DEMO$dataset_TOKYO)

}
```

`madshapR_website` *Call to online documentation*

Description

Direct call to the online documentation for the package, which includes a description of the latest version of the package, vignettes, user guides, and a reference list of functions and help pages.

Usage

```
madshapR_website()
```

Value

Nothing to be returned. The function opens a web page.

Examples

```
{

madshapR_website()

}
```

`summary_variables` *Provide descriptive statistics for variables in a dataset*

Description

Summarizes (in a data frame) the columns in a dataset and its data dictionary (if any). The summary provides information about quality, type, composition, and descriptive statistics of variables. Statistics are generated by valueType.

Usage

```
summary_variables(
  dataset_preprocess = .dataset_preprocess,
  dataset = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)
```

Arguments

`dataset_preprocess`	A data frame which provides summary of the variables (used for internal processes and programming).
`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`.dataset_preprocess`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing statistical description of variables present in a dataset.

Examples

```
{

library(dplyr)

###### Example : Any data frame can be a dataset by definition.
dataset_preprocess <- dataset_preprocess(dataset = iris)
glimpse(summary_variables(dataset_preprocess = dataset_preprocess))

}
```

`summary_variables_categorical` *Provide descriptive statistics for variables of categorical in a dataset*

Description

Summarizes (in a data frame) the columns of type 'categorical' in a dataset and its data dictionary (if any). The summary provides information about quality, type, composition, and descriptive statistics of variables. Statistics are generated by valueType.

Usage

```
summary_variables_categorical(
  dataset_preprocess = .dataset_preprocess,
  dataset = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)
```

Arguments

`dataset_preprocess`	A data frame which provides summary of the variables (for internal processes and programming).
`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`.dataset_preprocess`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A data frame providing statistical description of 'categorical' variables present in a dataset.

Examples

```
{

library(dplyr)

###### Example : Any data frame can be a dataset by definition.
dataset_preprocess <- dataset_preprocess(dataset = iris['Species'])
glimpse(summary_variables_categorical(dataset_preprocess = dataset_preprocess))

}
```

`summary_variables_date` *Provide descriptive statistics for variables of type 'date' in a dataset*

Description

Summarizes (in a data frame) the columns of type 'date' in a dataset and its data dictionary (if any). The summary provides information about quality, type, composition, and descriptive statistics of variables. Statistics are generated by valueType.

Usage

```
summary_variables_date(
  dataset_preprocess = .dataset_preprocess,
  dataset = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)
```

Arguments

`dataset_preprocess`	A data frame which provides summary of the variables (for internal processes and programming).
`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`.dataset_preprocess`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing statistical description of 'date' variables present in a dataset.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)
library(fabR)

dataset <-
  madshapR_DEMO$dataset_TOKYO %>%
    mutate(dob = as_any_date(dob)) %>%
    select(dob) %>%
    head()

dataset_preprocess <- dataset_preprocess(dataset = dataset)

summary_variables_date(dataset_preprocess = dataset_preprocess)

}
```

`summary_variables_datetime` *Provide descriptive statistics for variables of type 'datetime' in a dataset*

Description

Summarizes (in a data frame) the columns of type 'datetime' in a dataset and its data dictionary (if any). The summary provides information about quality, type, composition, and descriptive statistics of variables. Statistics are generated by valueType.

Usage

```
summary_variables_datetime(
  dataset_preprocess = .dataset_preprocess,
  dataset = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)
```

Arguments

`dataset_preprocess`	A data frame which provides summary of the variables (for internal processes and programming).
`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`.dataset_preprocess`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing statistical description of 'datetime' variables present in a dataset.

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)
library(lubridate)
library(fabR)

dataset_preprocess <-
  madshapR_DEMO$dataset_TOKYO %>%
  mutate(dob = as_datetime(as_any_date(dob))) %>%
  select(dob) %>%
  head() %>%
  dataset_preprocess

glimpse(summary_variables_datetime(dataset_preprocess = dataset_preprocess))

}
```

---

summary_variables_numeric

Provide descriptive statistics for variables of type 'numeric' in a dataset

---

Description

Summarizes (in a data frame) the columns of type 'numeric' in a dataset and its data dictionary (if any). The summary provides information about quality, type, composition, and descriptive statistics of variables. Statistics are generated by valueType.

Usage

```
summary_variables_numeric(
  dataset_preprocess = .dataset_preprocess,
  dataset = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)
```

Arguments

`dataset_preprocess`	A data frame which provides summary of the variables (for internal processes and programming).
`dataset`	A dataset object.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`.dataset_preprocess`	**[Deprecated]**

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing statistical description of 'numerical' variables present in a dataset.

Examples

```
{

library(dplyr)

###### Example : Any data frame can be a dataset by definition.
dataset_preprocess <- dataset_preprocess(dataset = iris)
glimpse(summary_variables_numeric(dataset_preprocess = dataset_preprocess))

}
```

`summary_variables_text` *Provide descriptive statistics for variables of type 'text' in a dataset*

Description

Summarizes (in a data frame) the columns of type 'text' in a dataset and its data dictionary (if any). The summary provides information about quality, type, composition, and descriptive statistics of variables. Statistics are generated by valueType.

Usage

```
summary_variables_text(
  dataset_preprocess = .dataset_preprocess,
  dataset = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)
```

Arguments

`dataset_preprocess`	A data frame which provides summary of the variables (for internal processes and programming).
`dataset`	A dataset object.

`data_dict` A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.

`.dataset_preprocess`
[Deprecated]

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

Value

A data frame providing statistical description of 'text' variables present in a dataset.

Examples

```
{

###### Example : Any data frame can be a dataset by definition.
library(dplyr)

dataset_preprocess <- dataset_preprocess(dataset = starwars['homeworld'])
glimpse(summary_variables_text(dataset_preprocess = dataset_preprocess))

}
```

`valueType_adjust` *Attribute the valueType from a data dictionary to a dataset, or vice versa*

Description

Takes the valueType of the input (from) and attributes it to the output (to). The parameters 'from' and 'to' can be either a dataset or a data dictionary. Depending on the input provided, the valueType replaced is either in the 'valueType' column of a data dictionary or cast to a column in a dataset. If 'to' is not provided, the function calls `valueType_self_adjust()` instead. The possible values returned are 'date', 'boolean', 'integer', 'decimal', and text'.

Usage

```
valueType_adjust(from, to = NULL)
```

Arguments

`from`	Object to be adjusted. Can be either a dataset or a data dictionary.
`to`	Object to be adjusted. Can be either a dataset or a data dictionary. NULL by default.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

Either a data frame, identifying the dataset, or a list of data frame(s) identifying a data dictionary, depending which is 'to'.

See Also

`valueType_self_adjust()`

Examples

```
{

# use madshapR_DEMO provided by the package
library(dplyr)

dataset <- madshapR_DEMO$dataset_TOKYO[c(1:4),'prg_ever']
data_dict <-
  madshapR_DEMO$data_dict_TOKYO %>%
  data_dict_filter(filter_var = 'name == "prg_ever"') %>%
  as_data_dict_mlstr()

head(valueType_adjust(from = data_dict,to = dataset))

}
```

`valueType_guess` *Guess the first possible valueType of an object (Can be a vector)*

Description

Provides the first possible valueType of a variable. The function tries to assign the valueType of the object first to 'boolean', then 'integer', then 'decimal', then 'date'. If all others fail, the default valueType is 'text'.

Usage

```
valueType_guess(x)
```

Arguments

`x`	Object. Can be a vector.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A character string which is the first possible valueType of the input object.

See Also

Opal documentation

Examples

```
{

# use madshapR_DEMO provided by the package

dataset <- madshapR_DEMO$dataset_TOKYO
valueType_guess(dataset$dob)

valueType_guess(mtcars$cyl)

}
```

`valueType_list` *Built-in data frame of allowed valueType values*

Description

Provides a built-in data frame showing the list of allowed Opal valueType values and their corresponding R data types. This data frame is mainly used for internal processes and programming.

Usage

```
valueType_list
```

Format

`data.frame`**:**
A data frame with 12 rows and 7 columns:

valueType data type as described in Opal
typeof data type provided by base::typeof
class data class provided by base::class
call function to transpose object according base::do.call function
toValueType ensemble data type as described in Opal
toTypeof ensemble data type provided by base::typeof
genericType ensemble data type which valueType belongs ...

Details

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

See Also

Opal documentation

Examples

```
{

print(valueType_list)

}
```

---

valueType_of *Return the valueType of an object*

Description

Determines the valueType of an object based on `typeof()` and `class()`. The possible values returned are 'date', 'boolean', 'integer', 'decimal', and 'text'.

Usage

```
valueType_of(x)
```

Arguments

`x`	Object. Can be a vector.

Details

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A character string which is the valueType of the input object.

See Also

`typeof()`, `class()` Opal documentation

Examples

```
{

# use madshapR_DEMO provided by the package

dataset <- madshapR_DEMO$dataset_MELBOURNE
valueType_of(dataset$Gender)
valueType_of(iris$Sepal.Length)

}
```

`valueType_self_adjust` *Guess and attribute the valueType of a data dictionary or dataset variable*

Description

Determines the valueType of an object based on `base::typeof()` and `base::class()`. The possible values returned are 'date', 'boolean', 'integer', 'decimal', and 'text'.

Usage

```
valueType_self_adjust(...)
```

Arguments

`...`	Object that can be either a dataset or a data dictionary.

Details

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

Either a data frame, identifying the dataset, or a list of data frame(s) identifying a data dictionary, depending which the input refers to.

See Also

`valueType_adjust()`

Examples

```
{

###### Example : The valueType of a dataset can be adjusted. each column is
# evaluated as whole, and the best valueType match found is applied. If
# there is no better match found, the column is left as it is.

head(valueType_self_adjust(mtcars['cyl']))

}
```

`variable_visualize` *Generate a list of charts, figures and summary tables of a variable*

Description

Analyses the content of a variable and its data dictionary (if any), identifies its data type and values accordingly and generates figures and summaries (datatable format). The figures and tables are representations of data distribution, statistics and valid/non valid/missing values (based on the data dictionary information if provided and the data type of the variable). This function can be used to personalize report parameters and is internally used in the function `dataset_visualize()`. Up to seven objects are generated which include : One datatable of the key elements of the data dictionary, one datatable summarizing statistics (such as mean, quartile, most seen value, most recent date, ... , depending on the data type of the variable), two graphs showing the distribution of the variable, One bar chart for categorical values (if any), One bar chart for missing values (if any), One pie chart for the proportion of valid and missing values (if any). The variable can be grouped using `group_by` parameter, which is a (categorical) column in the dataset. The user may need to use `as_category()`

in this context. To fasten the process (and allow recycling object in a workflow) the user can feed the function with a `variable_summary`, which is the output of the function `dataset_summarize()` of the column(s) `col` and `group_by`. The summary must have the same parameters to operate.

Usage

```
variable_visualize(
  dataset = tibble(id = as.character()),
  col,
  data_dict = NULL,
  group_by = NULL,
  valueType_guess = FALSE,
  variable_summary = .summary_var,
  .summary_var = NULL
)
```

Arguments

`dataset`	A dataset object.
`col`	A character string specifying the name of the column.
`data_dict`	A list of data frame(s) representing metadata of the input dataset. Automatically generated if not provided.
`group_by`	A character string identifying the column in the dataset to use as a grouping variable. Elements will be grouped by this column.
`valueType_guess`	Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.
`variable_summary`	A summary list which is the summary of the variables.
`.summary_var`	**[Deprecated]**

Details

A dataset is a data table containing variables. A dataset object is a data frame and can be associated with a data dictionary. If no data dictionary is provided with a dataset, a minimum workable data dictionary will be generated as needed within relevant functions. Identifier variable(s) for indexing can be specified by the user. The id values must be non-missing and will be used in functions that require it. If no identifier variable is specified, indexing is handled automatically by the function.

A data dictionary contains the list of variables in a dataset and metadata about the variables and can be associated with a dataset. A data dictionary object is a list of data frame(s) named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the data frame 'Variables' must contain at least the `name` column, with all unique and non-missing entries, and the data frame 'Categories' must contain at least the `variable` and `name` columns, with unique combination of `variable` and `name`.

The valueType is a declared property of a variable that is required in certain functions to determine handling of the variables. Specifically, valueType refers to the OBiBa data type of a variable. The valueType is specified in a data dictionary in a column 'valueType' and can be associated with variables as attributes. Acceptable valueTypes include 'text', 'integer', 'decimal', 'boolean', datetime', 'date'. The full list of OBiBa valueType possibilities and their correspondence with R data types are available using valueType_list. The valueType can be used to coerce the variable to the corresponding data type.

Value

A list of up to seven elements (charts and figures and datatables) which can be used to summarize visualize data.

See Also

`DT::datatable()`, `ggplot2::ggplot()` `dataset_summarize()`, `dataset_visualize()`

Examples

```
{

 library(dplyr)
 library(fs)

 dataset <- madshapR_DEMO$dataset_TOKYO

 variable_summary <- madshapR_DEMO$`dataset_summary`

 variable_visualize(
   dataset, col = 'height',
   variable_summary =  variable_summary,valueType_guess = TRUE)

 variable_visualize(
   dataset, col = 'height',
   variable_summary =  variable_summary,valueType_guess = TRUE)



}
```

Index