Package ‘madshapR’

June 20, 2023

Type Package

Title Support Technical Processes Following ‘Maelstrom Research’ Standards

Version 1.0.0

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, rlang, stringr, crayon, ggplot2, tidytext, grDevices, graphics, lubridate, janitor, forcats, knitr, bookdown, stats, DT, xfun, readr, tidyr, fs, utils, fabR

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], Samuel El Bouzaïdi Tiali [ctb]

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

Repository CRAN

Date/Publication 2023-06-20 17:50:02 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as_dataset</td>
<td>3</td>
</tr>
<tr>
<td>as_data_dict</td>
<td>4</td>
</tr>
<tr>
<td>as_data_dict_mlstr</td>
<td>5</td>
</tr>
<tr>
<td>as_data_dict_shape</td>
<td>6</td>
</tr>
<tr>
<td>as_dossier</td>
<td>7</td>
</tr>
<tr>
<td>as_taxonomy</td>
<td>8</td>
</tr>
<tr>
<td>as_valueType</td>
<td>9</td>
</tr>
<tr>
<td>check_dataset_categories</td>
<td>10</td>
</tr>
<tr>
<td>check_dataset_valueType</td>
<td>11</td>
</tr>
<tr>
<td>check_dataset_variables</td>
<td>12</td>
</tr>
<tr>
<td>check_data_dict_categories</td>
<td>14</td>
</tr>
<tr>
<td>check_data_dict_missing_categories</td>
<td>15</td>
</tr>
<tr>
<td>check_data_dict_valueType</td>
<td>16</td>
</tr>
<tr>
<td>check_data_dict_variables</td>
<td>17</td>
</tr>
<tr>
<td>check_name_standards</td>
<td>18</td>
</tr>
<tr>
<td>dataset_cat_as_labels</td>
<td>19</td>
</tr>
<tr>
<td>dataset_evaluate</td>
<td>20</td>
</tr>
<tr>
<td>dataset_preprocess</td>
<td>22</td>
</tr>
<tr>
<td>dataset_summarize</td>
<td>23</td>
</tr>
<tr>
<td>dataset_visualize</td>
<td>25</td>
</tr>
<tr>
<td>dataset_zap_data_dict</td>
<td>27</td>
</tr>
<tr>
<td>data_dict_apply</td>
<td>28</td>
</tr>
<tr>
<td>data_dict_collapse</td>
<td>29</td>
</tr>
<tr>
<td>data_dict_evaluate</td>
<td>30</td>
</tr>
<tr>
<td>data_dict_expand</td>
<td>32</td>
</tr>
<tr>
<td>data_dict_extract</td>
<td>33</td>
</tr>
<tr>
<td>data_dict_filter</td>
<td>34</td>
</tr>
<tr>
<td>data_dict_group_by</td>
<td>36</td>
</tr>
<tr>
<td>data_dict_group_split</td>
<td>37</td>
</tr>
<tr>
<td>data_dict_list_nest</td>
<td>38</td>
</tr>
<tr>
<td>data_dict_match_dataset</td>
<td>40</td>
</tr>
<tr>
<td>data_dict_pivot_longer</td>
<td>41</td>
</tr>
<tr>
<td>data_dict_pivot_wider</td>
<td>42</td>
</tr>
<tr>
<td>data_dict_ungroup</td>
<td>43</td>
</tr>
<tr>
<td>data_extract</td>
<td>45</td>
</tr>
<tr>
<td>DEMO_files</td>
<td>46</td>
</tr>
<tr>
<td>dossier_create</td>
<td>47</td>
</tr>
<tr>
<td>dossier_evaluate</td>
<td>48</td>
</tr>
<tr>
<td>dossier_summarize</td>
<td>49</td>
</tr>
<tr>
<td>is_dataset</td>
<td>50</td>
</tr>
<tr>
<td>is_data_dict</td>
<td>51</td>
</tr>
<tr>
<td>is_data_dict_mlstr</td>
<td>52</td>
</tr>
<tr>
<td>is_data_dict_shape</td>
<td>53</td>
</tr>
<tr>
<td>is_dossier</td>
<td>54</td>
</tr>
<tr>
<td>is_taxonomy</td>
<td>55</td>
</tr>
<tr>
<td>is_valueType</td>
<td>56</td>
</tr>
</tbody>
</table>
as_dataset

Validate and coerce an object to dataset format

Description

Confirms that the input object is a valid dataset and returns it as a dataset 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 dataset is valid.

Usage

as_dataset(object, col_id = NULL)

Arguments

object A potential dataset to be coerced.

col_id A character string specifying the name(s) of the column(s) which refer to key identifier of the dataset. The column(s) can be named or indicated by position.

Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

A tibble identifying a dataset.
as_data_dict

Validate and coerce any object as data dictionary

Description

Validates the input object as a valid data dictionary and coerces 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(object)

Arguments

object

A potential valid data dictionary to be coerced.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

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

See Also

For a better assessment, please use data_dict_evaluate().
Examples
{
  # use DEMO_files provided by the package
  data_dict <- DEMO_files$dd_PARIS_format_maelstrom
  as_data_dict(data_dict)
}

as_data_dict_mlstr

Validate and coerce an object to 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)

Arguments
object
  A potential valid data dictionary to be coerced.

as_data_dict
  Whether the output data dictionary has a simple data dictionary structure or not (meaning has a Maelstrom data dictionary structure, compatible with Maelstrom Research ecosystem, including Opal). FALSE by default.

Details
A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value
A list of tibble(s) identifying a data dictionary.
See Also

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

Examples

```r
{
  # use DEMO_files provided by the package

  data_dict <- DEMO_files$dd_MELBOURNE_1_format_maelstrom
  as_data_dict_mlstr(DEMO_files$dd_MELBOURNE_1_format_maelstrom)
}
```

as_data_dict_shape Validate and coerce an object to 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

```r
as_data_dict_shape(object)
```

Arguments

- **object**
  - A potential valid data dictionary to be coerced.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named ‘Variables’ (required) and ‘Categories’ (if any). To be usable in any function, the ‘Variables’ element must contain at least the ‘name’ column, and the ‘Categories’ element must contain at least the ‘variable’ and ‘name’ columns. To be considered as a minimum workable data dictionary, in ‘Variables’ the ‘name’ column must also have unique and non-null entries, and in ‘Categories’ the combination of ‘variable’ and ‘name’ columns must also be unique.

Value

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

See Also

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

{
  # use DEMO_files provided by the package

  data_dict <- DEMO_files$dd_PARIS_format_maelstrom
  as_data_dict_shape(data_dict)

}

as_dossier

Validate and coerce an object to dossier format

Description

Confirms that the input object is a valid dossier and returns it as a dossier 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 dossier is valid.

Usage

as_dossier(object)

Arguments

object       A potential dossier to be coerced.

Details

A dossier must be a named list containing at least one data frame or data frame extension (e.g. a
tibble), each of them being datasets. The name of each tibble will be use as the reference name of
the dataset.

Value

A list of tibble(s), each of them identifying datasets in a dossier.

See Also

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

{  
  # use DEMO_files provided by the package
  library(stringr)

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

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

---

as_taxonomy | Validate and coerce an object to taxonomy format

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 classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value

A tibble identifying a taxonomy (generally generated from Opal taxonomy).
as_valueType

See Also

Opal documentation

Examples

{

  # use DEMO_files provided by the package

  ###### Example
  as_taxonomy(DEMO_files$taxonomy_PARIS)

}

as_valueType

Validate and coerce an object according to a given valueType

Description

Attributes a valueType to an object, that can be a vector, or in a tibble 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 property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

The object coerced accordingly to the input valueType.

See Also

Opal documentation
check_dataset_categories

Assess a data dictionary and associated dataset for category differences

Description

Generates a tibble 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 = NULL)

Arguments

dataset A tibble identifying the input dataset observations.
data_dict A list of tibble(s) representing meta data to be evaluated.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Examples

```
{
  # use DEMO_files provided by the package
  dataset <- DEMO_files$dataset_TOKYO
  valueType_of(dataset$dob)
  valueType_guess(dataset$dob)
  as_valueType(dataset$dob,'date')

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

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

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

Examples

```r
{
# use DEMO_files provided by the package
data_dict <- DEMO_files$'dd_TOKYO_format_maelstrom_tagged' - ERROR WITH DATA`
dataset <- DEMO_files$'dataset_TOKYO' - ERROR WITH DATA`
check_dataset_categories(dataset, data_dict)
}
```

---

check_dataset_valueType

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

Description

Generates a tibble 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

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

Arguments

- `dataset` A tibble identifying the input dataset observations.
- `data_dict` A list of tibble(s) representing meta data to be evaluated.
- `valueType_guess` Whether the output should include a more accurate valueType that could be applied to the dataset. TRUE by default.
check_dataset_variables

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

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

Examples

{  
  # use DEMO_files provided by the package
  dataset <- DEMO_files$'dataset_TOKYO - ERROR WITH DATA'
  data_dict <- DEMO_files$'dd_TOKYO_format_maelstrom_tagged - ERROR WITH DATA'
  dataset <- data_dict_apply(dataset, data_dict)
  check_dataset_valueType(dataset, data_dict,valueType_guess = TRUE)
}

check_dataset_variables

Assess a data dictionary and associated dataset for undeclared variables
check_dataset_variables

Description
Generates a tibble 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 tibble identifying the input dataset observations.
data_dict A list of tibble(s) representing meta data to be evaluated.

Details
A data dictionary contains metadata about variables and can be associated with a dataset. It must
be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories'
(if any). To be usable in any function, the 'Variables' element must contain at least the 'name'
column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To
be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also
have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name'
columns must also be unique'.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no
data dictionary is provided, a minimum workable data dictionary will be generated as needed by
relevant functions. An identifier id column for sorting can be specified by the user. If specified, the
id values must be non-missing and will be used in functions that require it. If no identifier column
is specified, indexing is handled automatically by the function.

Value
A tibble providing undeclared variables across a data dictionary.

Examples
{
  # use DEMO_files provided by the package
  dataset <- DEMO_files$'dataset_TOKYO - ERROR WITH DATA`
data_dict <- DEMO_files$'dd_TOKYO_format_maelstrom_tagged - ERROR'
  check_dataset_variables(dataset, data_dict)
}
check_data_dict_categories

Assess a data dictionary for potential issues in categories

Description

Generates a tibble report of any categorical variable name present in the 'Categories' element but not present in 'Variables'. The tibble 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 tibble(s) representing meta data to be evaluated.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

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

Examples

{
    # use DEMO_files provided by the package

data_dict <- DEMO_files$'dd_TOKYO_format_maelstrom_tagged - ERROR'
check_data_dict_categories(data_dict)
}

check_data_dict_missing_categories

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

Description

Generates a tibble 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 tibble(s) representing meta data to be evaluated.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

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

Examples

{

# use DEMO_files provided by the package

data_dict <- DEMO_files$'dd_TOKYO_format_maelstrom_tagged - ERROR'
check_data_dict_missing_categories(data_dict)

}
check_data_dict_valueType

Assess a data dictionary for non-valid valueType values

Description
Generates a tibble 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 tibble(s) representing meta data to be evaluated.

Details
A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique'.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiaBa-internal type of a variable. It is specified in a data dictionary in a column valueType and can be associated with variables as attributes. Acceptable valueType possibilities include 'text', 'integer', 'decimal', 'boolean', 'datetime', 'date'). The full list of OBiaBa valueType possibilities and their correspondence with R data types are available using valueType_list.

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

Examples
{
  # use DEMO_files provided by the package

  data_dict <- DEMO_files$`dd_TOKYO_format_maelstrom_tagged - ERROR`
  check_data_dict_valueType(data_dict)
}
check_data_dict_variables

Assess a data dictionary for potential issues in variables

Description

Generates a tibble 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 tibble(s) representing meta data to be evaluated.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

A tibble providing non unique variables across a data dictionary.

Examples

{
    # use DEMO_files provided by the package

data_dict <- DEMO_files$`dd_TOKYO_format_maelstrom_tagged - ERROR`
check_data_dict_variables(data_dict)
}

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

Description

Generates a tibble 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 user must provide element which respect a certain structure to work with the functions of the package or its environment (Maelstrom and/or Obiba suite). In addition, any element may be compatible with Maelstrom Research ecosystem, including Opal, and its content can be evaluated accordingly, such as naming convention restriction, columns like 'valueType' and 'label(xx)' and/or any taxonomy provided.

Value

A tibble providing non-standard names across a vector.

Examples

{
  # use DEMO_files provided by the package
  check_name_standards(c("coucou", "cou cou", "$coucou", NA))
  check_name_standards(  
    DEMO_files$`dd_TOKYO_format_maelstrom_tagged - ERROR`$Variables$name)
}
dataset_cat_as_labels  

**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 tibble identifying the input dataset observations associated to its data dictionary.

data_dict  
A list of tibble(s) representing meta data of an associated dataset (to be generated).

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 must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

**Value**

A tibble identifying a dataset.
dataset_evaluate

Examples

```r
{
  dataset_cat_as_labels(iris['Sepal.Length'])
}
```

dataset_evaluate  Generate a quality assessment report of a dataset

Description

Assesses the content and structure of a dataset and reports possible issues in the dataset and data dictionary to facilitate assessment of input data. The report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats. This report is compatible with Excel and can be exported as an Excel spreadsheet.

Usage

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

Arguments

dataset  A tibble identifying the input dataset observations associated to its data dictionary.

data_dict  A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.

taxonomy  A tibble identifying the scheme used for variables classification.

.dataset_name  A character string specifying the name of the dataset (internally used in the function `dossier_evaluate()`).

as_data_dict_mlstr  Whether the output data dictionary has a simple data dictionary structure or not (meaning has a Maelstrom data dictionary structure, compatible with Maelstrom Research ecosystem, including Opal). TRUE by default.
Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value

A list of tibbles of report for one data dictionary.

See Also
dossier_evaluate()

Examples

```r
{
  # use DEMO_files provided by the package
dlplyr
deprecated('fabR') # add_index

  '#' ###### Example : any data frame (or tibble) can be summarized
dataset <- iris[,Sepal.Width]
dataset_evaluate(dataset)
}
```
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 tibble 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 tibble 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

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

Arguments

- **dataset**: A tibble identifying the input dataset observations associated to its data dictionary.
- **data_dict**: A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier `id` column for sorting can be specified by the user. If specified, the `id` values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

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

See Also

summary_variables()

Examples

{

    ###### Example : any data frame (or tibble) can be a dataset by definition.
    dataset_preprocess(iris)

}

dataset_summarize Generate a report and summary of a dataset

Description

Assesses and summarizes the content and structure of a dataset and data dictionary and reports potential issues to facilitate the assessment of input. The report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats. The summary provides additional information about variable distributions and descriptive statistics. This report is compatible with Excel and can be exported as an Excel spreadsheet.

Usage

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

Arguments

dataset A tibble identifying the input dataset observations associated to its data dictionary.

data_dict A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.

group_by A character string of one column in the dataset that can be taken as a grouping column. The visual element will be grouped and displayed by this column.

taxonomy A tibble identifying the scheme used for variables classification.

.dataset_name A character string specifying the name of the dataset (internally used in the function dossier_evaluate()).
dataset_summarize

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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

A list of tibbles of report for one data dictionary.

See Also

dossier_evaluate()

Examples

```r

# use DEMO_files provided by the package
library(dplyr)

#' ###### Example : any data frame (or tibble) can be summarized
```
dataset_visualize

```r
dataset <- iris['Sepal.Width']
dataset_summarize(dataset)
```

Description

Generates a visual report for a dataset in an HTML bookdown document. The report provides figures and descriptive statistics for each variable to facilitate the assessment of input data. Statistics and figures are generated according to variable data type. The report can be used to help assess data structure, coherence across elements, and taxonomy or data dictionary formats. The summaries and figures provide additional information about variable distributions and descriptive statistics. The charts and tables are produced based on their data type. The variable can be grouped using `group_by` parameter, which is a (categorical) column in the dataset. The user may need to use `as.factor()` in this context. To fasten the process (and allow recycling object in a workflow) the user can feed the function with a `.summary_var`, 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

```r
dataset_visualize(
  dataset = tibble(id = as.character()),
  data_dict = data_dict_extract(dataset),
  group_by = NULL,
  to,
  taxonomy = NULL,
  valueType_guess = FALSE,
  .summary_var = NULL,
  .dataset_name = NULL,
  .keep_files = TRUE
)
```

Arguments

- **dataset**: A tibble identifying the input dataset observations associated to its data dictionary.
- **data_dict**: A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.
- **group_by**: A character string of one column in the dataset that can be taken as a grouping column. The visual element will be grouped and displayed by this column.
- **to**: A character string identifying the folder path where the bookdown report will be saved.
dataset_visualize

taxonomy A tibble identifying the scheme used for variables classification.

valueType_guess Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.

.summary_var A list which is the summary of the variables.

dataset_name A character string specifying the name of the dataset (used for internal processes and programming).

.keep_files whether to keep the R-markdown files. TRUE by default. (used for internal processes and programming)

Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique'.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It is specified in a data dictionary in a column valueType and can be associated with variables as attributes. Acceptable valueType 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.

A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value

A bookdown folder containing files in the specified output folder. To open the file in browser, open 'docs/index.html'. Or use open_visual_report()

See Also

open_visual_report()
dataset_zap_data_dict

Examples

```r
{

dataset <- DEMO_files$dataset_TOKYO['height']
data_dict <-
  data_dict_filter(
    DEMO_files$dd_TOKYO_format_maelstrom_tagged,
    filter_var = "name == 'height'")

.summary_var <- DEMO_files$summary_var
to <- tempdir()
dataset_visualize(dataset, data_dict,.summary_var =.summary_var, to = to)

# To open the file in browser, open 'to/docs/index.html'.
# Or use open_visual_report function
}
```

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

Description

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

Usage

```r
dataset_zap_data_dict(dataset)
```

Arguments

dataset  A tibble identifying the input dataset observations associated to its data dictionary.

Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

A tibble identifying a dataset.
data_dict_apply

Apply a data dictionary to a dataset

Description

Applies a data dictionary to a data structure, creating a labelled dataset. All previous attributes will be preserved. For factors, the attribute 'levels' will be transformed into attribute 'labels' and values will be recast into appropriate datatypes.

Usage

data_dict_apply(dataset, data_dict = NULL)

Arguments

dataset A tibble identifying the input dataset observations associated to its data dictionary.
data_dict A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.

Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name'

See Also

haven::zap_labels().

Examples

{

  # use DEMO_files provided by the package

  dataset <- DEMO_files$dataset_TOKYO
  data_dict <- as_data_dict_mlstr(DEMO_files$dd_TOKYO_format_maelstrom_tagged)
  dataset <- data_dict_apply(dataset, data_dict)
  dataset_zap_data_dict(dataset)

}
column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value
A tibble identifying the dataset with the data dictionary applied to each variable as attributes.

See Also
base::attributes()

Examples
{
  # use DEMO_files provided by the package
  dataset <- DEMO_files$dataset_MELBOURNE_1
  data_dict <- as_data_dict_mlstr(DEMO_files$dd_MELBOURNE_1_format_maelstrom)
  data_dict_apply(dataset, data_dict)
}

---

**data_dictCollapse**

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_dictCollapse(
  data_dict,  
  from = "Categories",  
  to = "Variables",  
  name_prefix = "Categories::"
)
**Arguments**

- **data_dict**
  A list of tibble(s) representing metadata to be transformed. Automatically generated if not provided.

- **from**
  Symbol identifying the name of the element (tibble) to take column(s) from. Default is 'Categories'.

- **to**
  Symbol identifying the name of the element (tibble) to create column(s) to. 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::'.

**Details**

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

**Value**

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

**See Also**

data_dict_expand()

**Examples**

```r
{
  # use DEMO_files provided by the package
  data_dict <- DEMO_files$dd_MELBOURNE_1_format_maelstrom
  data_dictCollapse(data_dict)
}
```

---

**data_dict_evaluate**

*Generate a quality assessment report of a data dictionary*
data_dict_evaluate

Description

Assesses the content and structure of a data dictionary and reports potential issues to facilitate the assessment of input data. The report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats. This report is compatible with Excel and can be exported as an Excel spreadsheet.

Usage

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

Arguments

data_dict: A list of tibble(s) representing meta data to be evaluated.
taxonomy: A tibble identifying the scheme used for variables classification as a tibble.
as_data_dict_mlstr: Whether the output data dictionary has a simple data dictionary structure or not (meaning has a Maelstrom data dictionary structure, compatible with Maelstrom Research ecosystem, including Opal). TRUE by default.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique'.

A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value

A list of tibbles of report for one data dictionary.

Examples

{
  # use DEMO_files provided by the package
  data_dict <- DEMO_files$'dd_TOKYO_format_maelstrom_tagged - ERROR'
  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 tibble 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_dictCollapse()

Usage

data_dict.expand(
  data_dict,
  from = "Variables",
  name_prefix = "Categories::",
  to = "Categories"
)

Arguments

data_dict A list of tibble(s) representing meta data to be transformed. Automatically generated if not provided.
from Symbol identifying the name of the element (tibble) 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 Symbol identifying the name of the element (tibble) to create column(s) to. Default is 'Categories'.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
data_dict_extract

Create a data dictionary from a dataset

Description

Creates a data dictionary in a format compliant with formats used in Maelstrom Research ecosystem, including Opal (with ‘Variables’ and ‘Categories’ in separate tibbles and standard columns in each) from any dataset in tibble format. If the input dataset has no associated metadata, a data dictionary with minimal required information is created from the column (variable) names to create the data dictionary structure required for 'madshapR'. All columns except variable names will be blank.

Usage

data_dict_extract(dataset, as_data_dict_mlstr = TRUE)

Arguments

dataset A tibble identifying the input dataset observations which contains meta data as attributes.

as_data_dict_mlstr Whether the output data dictionary has a simple data dictionary structure or not (meaning has a Maelstrom data dictionary structure, compatible with Maelstrom Research ecosystem, including Opal). TRUE by default.
Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier `id` column for sorting can be specified by the user. If specified, the `id` values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named `Variables` (required) and `Categories` (if any). To be usable in any function, the `Variables` element must contain at least the `name` column, and the `Categories` element must contain at least the `variable` and `name` columns. To be considered as a minimum workable data dictionary, in `Variables` the `name` column must also have unique and non-null entries, and in `Categories` the combination of `variable` and `name` columns must also be unique.

Value

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

Examples

```r
# use DEMO_files provided by the package

dataset <- DEMO_files$dataset_MELBOURNE_1
data_dict <- as_data_dict_mlstr(DEMO_files$dd_MELBOURNE_1_format_maelstrom)
dataset <- data_dict_apply(dataset, data_dict)
data_dict_extract(dataset)

#### 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 tibble(s) representing meta data to be transformed.
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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

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

See Also
dplyr::filter()

Examples

{
  # use DEMO_files 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 <-
    DEMO_files$dd_MELBOURNE_1_format_maelstrom %>%

lapply(function(x){x %>% mutate(table = "MELBOURNE_1"))
data_dict_2 <- DEMO_files$dd_MELBOURNE_2_format_maelstrom %>%
  lapply(function(x){x %>% mutate(table = "MELBOURNE_2"}))
data_dict_list <-
  list(Variables = bind_rows(data_dict_1$Variables,data_dict_2$Variables),
       Categories = bind_rows(data_dict_1$Categories,data_dict_2$Categories))

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

# Example 2 search and filter through a column in 'Categories' element
data_dict_filter(data_dict_list,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_list,filter_all = "table == "MELBOURNE_1"")

---

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 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 tibble(s) representing meta data to be transformed.
- `col`: variable to group by.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
Value

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

See Also

dplyr::group_by(), data_dict_ungroup()

Examples

{

  # use DEMO_files 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 <-
    DEMO_files[c('dd_MELBOURNE_1_format_maelstrom',
                'dd_MELBOURNE_2_format_maelstrom')] %>%
    data_dict_list_nest(name_group = 'table')

  data_dict_group_by(data_dict_list,col = "table")
}

description

Split grouped data dictionaries into a named list

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 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 tibble(s) representing meta data 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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

A list of tibble(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 DEMO_files 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 <- DEMO_files[
    c('dd_MELBOURNE_1_format_maelstrom',
    'dd_MELBOURNE_2_format_maelstrom')]

  data_dict_list_nest(name_group = 'table')

  data_dict_group_split(data_dict_list, 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

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

Arguments

- `data_dict_list`: A list of tibble(s) representing meta data 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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

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

See Also

- `dplyr::bind_rows()`

Examples

```r
{
  # use DEMO_files 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 <- DEMO_files[
    c('dd_MELBOURNE_1_format_maelstrom',
      'dd_MELBOURNE_2_format_maelstrom')]

  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**

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

**Arguments**

- `dataset`: A tibble identifying the input dataset observations.
- `data_dict`: A list of tibble(s) representing meta data.
- `data_dict_apply`: whether to apply the data dictionary to its dataset. The resulting tibble will have for each column its associated meta data as attributes. The factors 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 must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
Value

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

Examples

{
  # use DEMO_files provided by the package
  library(dplyr)
  dataset <- DEMO_files$dataset_MELBOURNE_1 %>% select(-1)
  data_dict <- DEMO_files$dd_MELBOURNE_1_format_maelstrom
  data_dict_match_dataset(dataset, 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 tibble(s) representing meta data to be transformed.

taxonomy A tibble identifying the scheme used for variables classification.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value
A list of tibble(s) identifying a data dictionary.

See Also
tidyr::pivot_longer(), as_data_dict()

Examples
{
  # use DEMO_files provided by the package
  taxonomy <- DEMO_files$taxonomy_opal_mlstr
  data_dict <- DEMO_files$dd_TOKYO_format_opal_tagged
  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 tibble(s) representing meta data to be transformed.
taxonomy  A tibble identifying the scheme used for variables classification.
**Details**

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

**Value**

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

**See Also**

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

**Examples**

```r
{
  # use DEMO_files provided by the package

taxonomy <- DEMO_files$taxonomy_opal_mlstr
data_dict <- DEMO_files$dd_TOKYO_format_maelstrom_tagged
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 tibbles). 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 tibble(s) representing meta data to be transformed.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

Value

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

See Also

dplyr::ungroup() data_dict_group_by()

Examples

{
# use DEMO_files 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 <- DEMO_files[
  c('dd_MELBOURNE_1_format_maelstrom',
    'dd_MELBOURNE_2_format_maelstrom')] %>%
data_dict_list_nest(name_group = 'table') %>%
data_dict_group_by(col = "table")

data_dict_ungroup(data_dict_list)
}
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**

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

**Arguments**

- `data_dict`: A list of tibble(s) representing meta data of an associated dataset (to be generated).
- `data_dict_apply`: whether to apply the data dictionary to its dataset. The resulting tibble will have for each column its associated meta data as attributes. The factors will be preserved. FALSE by default.

**Details**

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique'.

**Value**

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

**Examples**

```r
{
# use DEMO_files provided by the package

data_dict <- DEMO_files$dd_MELBOURNE_1_format_maelstrom
data_extract(data_dict)
}
```
DEMO_files

**Description**

Built-in tibbles and lists allowing the user to test the package with demo material.

**Usage**

DEMO_files

**Format**

```
list:
A list with 19 elements used for testing the package (data frames and lists):
dd_MELBOURNE_1_format_maelstrom Data dictionary (1) of Melbourne dataset
dd_MELBOURNE_2_format_maelstrom Data dictionary (2) of Melbourne dataset
dd_PARIS_format_maelstrom Data dictionary of Paris dataset
dd_PARIS_format_preprocessed - ERROR Data dictionary of Paris dataset containing errors
dd_PARIS_format_preprocessed Data dictionary of Paris in preprocessed format
dd_TOKYO_format_maelstrom_tagged - ERROR WITH DATA Data dictionary of Tokyo dataset containing errors
dd_TOKYO_format_maelstrom_tagged - ERROR Data dictionary of Tokyo dataset containing errors
dd_TOKYO_format_maelstrom_tagged Data dictionary of Tokyo dataset
dd_TOKYO_format_opal_tagged - ERROR WITH TAXO Data dictionary of Tokyo dataset containing errors
dd_TOKYO_format_opal_tagged Data dictionary of Tokyo dataset opal format
dataset_MELBOURNE_1 Dataset of Melbourne (1)
dataset_MELBOURNE_2 Dataset of Melbourne (2)
dataset_PARIS Dataset of Paris
dataset_TOKYO - ERROR WITH DATA Dataset of Tokyo containing errors
dataset_TOKYO Dataset of Tokyo
PARIS_taxonomy Taxonomy specific to Paris dataset
taxonomy_mlstr Maelstrom Taxonomy
taxonomy_opal Opal Taxonomy
summary_var Variables summary for testing purpose ...
```

**Examples**

```
{
  print(DEMO_files$dataset_TOKYO)
}
```
dossier_create

Create a dossier object from a list of dataset(s)

Description

Assembles a dossier object from the listed datasets. A dossier is a list containing at least one valid dataset and is the input used by key functions of the package.

Usage

dossier_create(dataset_list, data_dict_apply = FALSE)

Arguments

dataset_list  A list of tibble(s), identifying the input data observations.
data_dict_apply whether to apply the data dictionary to its dataset. The resulting tibble will have for each column its associated meta data as attributes. The factors will be preserved. FALSE by default.

Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

A list of tibble(s), each of them identifying datasets in a dossier.

Examples

{

# use DEMO_files provided by the package

#### Example 1: datasets can be gathered into a dossier which is a list.

dossier <- dossier_create(
    dataset_list = list(
        dataset_MELBOURNE_1 = DEMO_files$dataset_MELBOURNE_1,
        dataset_MELBOURNE_2 = DEMO_files$dataset_MELBOURNE_2))

#### Example 2: any data frame (or tibble) can be gathered into a dossier

dossier_create(list(iris, mtcars))

}
dossier_evaluate  Generate a quality assessment report of a dossier (list of datasets)

Description
Assesses the content and structure of a dossier object (list of datasets) and reports possible issues in the datasets and data dictionaries to facilitate assessment of input data. The report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats. This report is compatible with Excel and can be exported as an Excel spreadsheet.

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

Arguments
dossier       List of tibble, each of them being datasets.
taxonomy      A tibble identifying the scheme used for variables classification.
as_data_dict_mlstr Whether the output data dictionary has a simple data dictionary structure or not (meaning has a Maelstrom data dictionary structure, compatible with Maelstrom Research ecosystem, including Opal). TRUE by default.

Details
A dossier must be a named list containing at least one data frame or data frame extension (e.g. a tibble), each of them being datasets. The name of each tibble will be use as the reference name of the dataset.

A taxonomy is classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value
A list of tibbles of report for each dataset.

Examples
{
  # use DEMO_files provided by the package
  library(dplyr)
dossier_summarize

library(stringr)

#### Example: a dataset list is a dossier by definition.
dossier_evaluation <- dossier_evaluate(
  DEMO_files[str_detect(names(DEMO_files), "dataset_MELBOURNE")]
)
glimpse(dossier_evaluation)

---

**dossier_summarize**  
*Generate a report and summary of a dossier (list of datasets)*

**Description**

Assesses and summarizes the content and structure of a dossier (list of datasets) and reports potential issues to facilitate the assessment of input data. The report can be used to help assess data structure, presence of fields, coherence across elements, and taxonomy or data dictionary formats. The summary provides additional information about variable distributions and descriptive statistics. This report is compatible with Excel and can be exported as an Excel spreadsheet.

**Usage**

`dossier_summarize(dossier, taxonomy = NULL, valueType_guess = TRUE)`

**Arguments**

- **dossier**: List of tibble, each of them being datasets.
- **taxonomy**: A tibble identifying the scheme used for variables classification.
- **valueType_guess**: Whether the output should include a more accurate valueType that could be applied to the dataset. TRUE by default.

**Details**

A dossier must be a named list containing at least one data frame or data frame extension (e.g. a tibble), each of them being datasets. The name of each tibble will be used as the reference name of the dataset. This report is compatible with Excel and can be exported as an Excel spreadsheet.

A taxonomy is a classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.
The `valueType` is a property of a variable and is required in certain functions to determine the handling of the variables. The `valueType` refers to the OBiBa-internal type of a variable. It is specified in a data dictionary in a column `valueType` and can be associated with variables as attributes. Acceptable `valueType` 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`.

**Value**

A list of tibbles of report for each listed dataset.

**Examples**

```r
{
    # use DEMO_files provided by the package
    library(dplyr)
    library(stringr) # glimpse

    # Example 1: Combine functions and summarise datasets.
    dossier <- list(iris = iris[1])

    dossier_summary <- dossier_summarize(dossier)
    glimpse(dossier_summary)
}
```

---

**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**

```r
is_dataset(object)
```

**Arguments**

- `object` A potential dataset to be evaluated.

**Details**

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier `1d` column for sorting can be specified by the user. If specified, the `1d` values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.
is_data_dict

Value
A logical.

See Also
For a better assessment, please use dataset_evaluate().

Examples
{
  # use DEMO_files provided by the package
  # any data frame (or tibble) can be a dataset by definition.
  is_dataset(DEMO_files$dataset_MELBOURNE_1)
  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 coerced.

Details
A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
Value

A logical.

See Also

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

Examples

```r
{
  # use DEMO_files provided by the package
  data_dict <- DEMO_files$dd_MELBOURNE_1_format_maelstrom
  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

```r
is_data_dict_shape(object)
```

Arguments

- **object**  
  A potential data dictionary structure to be evaluated.

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.
Value

A logical.

See Also

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

Examples

```r
{
  # use DEMO_files provided by the package
  data_dict <- DEMO_files$dd_MELBOURNE_1_format_maelstrom
  is_data_dict_shape(data_dict)
  is_data_dict_shape(iris)
}
```

---

### is_dossier

Test if an object is a valid dossier

#### 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

```r
is_dossier(object)
```

#### Arguments

- **object**
  
  A potential dossier to be evaluated.

#### Details

A dossier must be a named list containing at least one data frame or data frame extension (e.g. a tibble), each of them being datasets. The name of each tibble will be used as the reference name of the dataset.

#### Value

A logical.
is_taxonomy

Examples
{
  # use DEMO_files provided by the package
  # Any list of data frame (or tibble) can be a dossier by definition.
  library(stringr)

  is_dossier(DEMO_files[str_detect(names(DEMO_files),"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 classification scheme that can be defined for variable attributes. If defined, a taxonomy must be a data frame-like object. It must be compatible with (and is generally extracted from) an Opal environment. To work with certain functions, a valid taxonomy must contain at least the columns 'taxonomy', 'vocabulary', and 'terms'. In addition, the taxonomy may follow Maelstrom research taxonomy, and its content can be evaluated accordingly, such as naming convention restriction, tagging elements, or scales, which are specific to Maelstrom Research. In this particular case, the tibble must also contain 'vocabulary_short', 'taxonomy_scale', 'vocabulary_scale' and 'term_scale' to work with some specific functions.

Value
A logical.
Examples

{

# use DEMO_files provided by the package
is_taxonomy(DEMO_files$taxonomy_PARIS)
is_taxonomy(DEMO_files$taxonomy_opal_mlstr)
is_taxonomy(iris)

}

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 property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

A logical.

See Also

Opal documentation
madshapR_help

Examples
{
  is_valueType('integer')
  is_valueType('integre')
}

Call the help center for full documentation

Description
This function is a direct call to the documentation in the repository hosting the package. The user can access the description of the latest version of the package, the vignettes, and the list of functions.

Usage
madshapR_help()

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

Examples
{
  madshapR_help()
}

open_visual_report
Open a visual report in a browser

Description
Opens a previously generated HTML bookdown document. This is a shortcut function to access an existing visual report.

Usage
open_visual_report(report_path)
Arguments

report_path A character string specifying the path of the report to be opened.

Value

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

See Also

dataset_visualize()

Examples

{
  # use DEMO_files provided by the package
  dataset <- DEMO_files$dataset_TOKYO['height']
  data_dict <-
    data_dict_filter(
      DEMO_files$dd_TOKYO_format_maelstrom_tagged,
      filter_var = "name == 'height'"
    )
  .summary_var <- DEMO_files$summary_var

  to <- tempdir()
  dataset_visualize(dataset, data_dict,.summary_var =.summary_var, to = to)

  # To open the file in browser, you can also open 'to/docs/index.html'.
  open_visual_report(to)
}

summary_variables Provide descriptive statistics for variables in a dataset

Description

Summarises (in a tibble) 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 = NULL, data_dict = NULL, .dataset_preprocess = NULL)
summary_variables_categorical

Arguments

- **dataset**
  A tibble identifying the input dataset observations associated to its data dictionary.

- **data_dict**
  A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.

- **.dataset_preprocess**
  A tibble which provides summary of the variables (used for internal processes and programming).

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named `Variables` (required) and `Categories` (if any). To be usable in any function, the `Variables` element must contain at least the `name` column, and the `Categories` element must contain at least the `variable` and `name` columns. To be considered as a minimum workable data dictionary, in `Variables` the `name` column must also have unique and non-null entries, and in `Categories` the combination of `variable` and `name` columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier `id` column for sorting can be specified by the user. If specified, the `id` values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

A tibble providing statistical description of variables present in a dataset.

Examples

```r
#
# Example : any data frame (or tibble) can be a dataset by definition.
.dataset_preprocess <- dataset_preprocess(iris)
summary_variables(.dataset_preprocess = .dataset_preprocess)
```
Usage

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

Arguments

dataset  A tibble identifying the input dataset observations associated to its data dictionary.
data_dict  A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.
.dataset_preprocess  A tibble which provides summary of the variables (for internal processes and programming).

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

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

Examples

```r
{
  # use DEMO_files provided by the package
  library(dplyr)
}
Example: any data frame (or tibble) can be a dataset by definition.

```r
.dataset_preprocess <- dataset_preprocess(storms['status'])
summary_variables_categorical(.dataset_preprocess = .dataset_preprocess)
```

---

**summary_variables_date**

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

### Description

Summarises (in a tibble) 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

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

### Arguments

- **dataset**
  - A tibble identifying the input dataset observations associated to its data dictionary.

- **data_dict**
  - A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.

- **.dataset_preprocess**
  - A tibble which provides summary of the variables (for internal processes and programming).

### Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by
relevant functions. An identifier \texttt{id} column for sorting can be specified by the user. If specified, the \texttt{id} values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

**Value**

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

**Examples**

```r
### Example : any data frame (or tibble) can be a dataset by definition.
library(dplyr)
library(fabR)

.dataset_preprocess <-
storms %>%
sample_n(50) %>%
mutate(date_storm = as_any_date(paste(year, month, day,"-"),"ymd")) %>%
select(date_storm) %>%
dataset_preprocess

summary_variables_date(.dataset_preprocess = .dataset_preprocess)

}
```

---

**summary_variables_numeric**

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

**Description**

Summarises (in a tibble) 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 \texttt{valueType}.

**Usage**

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

- **dataset**: A tibble identifying the input dataset observations associated to its data dictionary.
- **data_dict**: A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.
- **.dataset_preprocess**: A tibble which provides summary of the variables (for internal processes and programming).

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier `id` column for sorting can be specified by the user. If specified, the `id` values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

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

Examples

```r
{

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

}
```

**summary_variables_text**

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

Description

Summarises (in a tibble) 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 = NULL,
  data_dict = NULL,
  .dataset_preprocess = NULL
)

Arguments

dataset A tibble identifying the input dataset observations associated to its data dictionary.
data_dict A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.
.dataset_preprocess A tibble which provides summary of the variables (for internal processes and programming).

Details

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

Value

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

Examples

{

    ###### Example : any data frame (or tibble) can be a dataset by definition.
    library(dplyr)
    .dataset_preprocess <- dataset_preprocess(starwars['homeworld'])
    summary_variables_text(.dataset_preprocess = .dataset_preprocess)

}
valueType_adjust

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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique'.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It is specified in a data dictionary in a column valueType and can be associated with variables as attributes. Acceptable valueType possibilities 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`.

Value

Either a tibble, identifying the dataset, or a list of tibble(s) identifying a data dictionary, depending which is 'to'.
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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by
relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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 \texttt{valueType\_list}.

\textbf{Value}

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

\textbf{See Also}

\texttt{Opal documentation}

\textbf{Examples}

\begin{verbatim}
{
    # use DEMO\_files provided by the package
    dataset <- DEMO\_files\$dataset\_TOKYO
    valueType\_of(dataset\$dob)
    valueType\_guess(dataset\$dob)

    valueType\_of(mtcars\$cyl)
    valueType\_guess(mtcars\$cyl)
}
\end{verbatim}

\begin{tabular}{ll}
\texttt{valueType\_list} & \textit{Built-in tibble of allowed valueType values} \\
\end{tabular}

\textbf{Description}

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

\textbf{Usage}

\texttt{valueType\_list}
**Format**

tibble:
A data frame with 12 rows and 6 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 property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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`.

**See Also**

- Opal documentation

**Examples**

```r
{
  valueType_list
}
```

---

**valueType_of**

*Return the valueType of an object*

**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_of(x)`

**Arguments**

- **x**: Object. Can be a vector.
Details

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

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

See Also

base::typeof(), base::class() valueType_list for insights about possible valueType and translation into type and class in R. Opal documentation

Examples

{
  # use DEMO_files provided by the package
  dataset <- DEMO_files$dataset_MELBOURNE_1
  valueType_of(dataset$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 metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

Either a tibble, identifying the dataset, or a list of tibble(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.

    valueType_self_adjust(mtcars['cyl'])

}
variable_visualize

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.factor() in this context. To fasten the process (and allow recycling object in a workflow) the user can feed the function with a .summary_var argument, 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,
  .summary_var = NULL
)

Arguments

dataset A tibble identifying the input dataset observations associated to its data dictionary.
col A character string specifying the name of the column.
data_dict A list of tibble(s) representing meta data of an associated dataset. Automatically generated if not provided.
group_by A character string of one column in the dataset that can be taken as a grouping column. The visual element will be grouped and displayed by this column.
valueType_guess Whether the output should include a more accurate valueType that could be applied to the dataset. FALSE by default.
.summary_var A summary list which is the summary of the variables.

Details

A dataset must be a data frame-like object and can be associated with a data dictionary. If no data dictionary is provided, a minimum workable data dictionary will be generated as needed by relevant functions. An identifier id column for sorting can be specified by the user. If specified, the
id values must be non-missing and will be used in functions that require it. If no identifier column is specified, indexing is handled automatically by the function.

A data dictionary contains metadata about variables and can be associated with a dataset. It must be a list of data frame-like objects with elements named 'Variables' (required) and 'Categories' (if any). To be usable in any function, the 'Variables' element must contain at least the 'name' column, and the 'Categories' element must contain at least the 'variable' and 'name' columns. To be considered as a minimum workable data dictionary, in 'Variables' the 'name' column must also have unique and non-null entries, and in 'Categories' the combination of 'variable' and 'name' columns must also be unique.

The valueType is a property of a variable and is required in certain functions to determine the handling of the variables. The valueType refers to the OBiBa-internal type of a variable. It 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.

Value

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

See Also

as.factor(), DT::datatable(), ggplot2::ggplot(), dataset_summarize(), dataset_visualize()

Examples

```
{

dataset <- iris[, 'Sepal.Width']
summary_variable <- dataset_summarize(dataset)

variable_viz <-
    variable_visualize(
        dataset, col = 'Sepal.Width',
        .summary_var = summary_variable)

variable_viz$summary_table
variable_viz$main_values_1
variable_viz$main_values_2
}
```
Index

datasets
  DEMO_files, 46
  valueType_list, 67

as.factor(), 25, 71, 72
as_data_dict, 4
as_data_dict(), 42, 43
as_data_dict_mlstr, 5
as_data_dict_shape, 6
as_dataset, 3
as_dossier, 7
as_taxonomy, 8
as_valueType, 9

base::attributes(), 29
base::class(), 68, 69
base::typeof(), 68, 69

check_data_dict_categories, 14
check_data_dict_missing_categories, 15
check_data_dict_valueType, 16
check_data_dict_variables, 17
check_dataset_categories, 10
check_dataset_valueType, 11
check_dataset_variables, 12
check_name_standards, 18

data_dict_apply, 28
data_dict_list_nest, 37, 38
data_dict_match_dataset, 40
data_dict_pivot_longer, 41
data_dict_pivot_wider, 42
data_dict_ungroup, 43
data_dict_ungroup(), 36, 37
data_extract, 45
dataset_cat_as_labels, 19
dataset_evaluate, 20
dataset_evaluate(), 7, 51
dataset_preprocess, 22
dataset_summarize, 23
dataset_summarize(), 22, 25, 71, 72
dataset_visualize, 25
dataset_visualize(), 58, 71, 72
dataset_zap_data_dict, 27
DEMO_files, 46
dossier_create, 47
dossier_evaluate, 48
dossier_evaluate(), 20, 21, 23, 24
dossier_summarize, 49
dplyr::bind_rows(), 38, 39
dplyr::filter(), 34, 35
dplyr::group_by(), 36–38
dplyr::group_split(), 37, 38
dplyr::ungroup(), 43, 44
DT::datatable(), 72

ggplot2::ggplot(), 72

haven::zap_labels(), 28

is_data_dict, 51
is_data_dict_mlstr, 52
is_data_dict_shape, 53
is_dataset, 50
is_dossier, 54
is_taxonomy, 55
is_valueType, 56
madshapR help, 57

open_visual_report, 57
open_visual_report(), 26

summary_variables, 58
summary_variables(), 23
summary_variables_categorical, 59
summary_variables_date, 61
summary_variables_numeric, 62
summary_variables_text, 63

tidyR::pivot_longer(), 41, 42
tidyR::pivot_wider(), 43

valueType_adjust, 65
valueType_adjust(), 70
valueType_guess, 66
valueType_list, 9, 12, 16, 24, 26, 50, 56, 60, 65, 67, 68–70, 72
valueType_of, 68
valueType_self_adjust, 69
valueType_self_adjust(), 65, 66
variable_visualize, 70