Package 'koboquest'

December 10, 2019

Title Reads and understands kobo questionnaires

Version 1.0.1

Description Use kobo questionnaires to identify data types, parse skip logic and apply labels.

Depends data.table,
dplyr,
magrittr,
stringi,
stringr,
utils

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat

R topics documented:

data *Example data set*

Description

Example data set

Author(s)

Impact Initiatives <martin.barner@impact-initiatives.org>

References

www.impact-initiatives.org

is_questionnaire_loaded *Determine if a questionnarie has been loaded*

Description

Checks if a questionnaire has been loaded into the global space with load_questionnaire

Usage

```
is_questionnaire_loaded()
```

Value

TRUE if load_questionnaire has been called successfully.

See Also

load_questionnaire

Examples

```
is_questionnaire_loaded()
```

koboquest *koboquest: Reading* R*hrefhttp://xlsform.org/en/XLSForm questionnaires*

Description

For data collected with kobotoolbox, ODK or similar.

Details

This package lets you load a questionnaire and match it with the data. It then provides three main functionalities:

- identifying question types
- converting xml style data headers and values to labels
- parsing questionnaire skiplogic ('relevant' column in XLSForms.)

A questionnaire (in .csv format) is loaded globally with `load_questionnaire`. All other functions then refer to that questionnaire automatically. (See below on using multiple questionnaires in parallel.)

Identifying question types

- `question_type` is the standard function to determine data types.

There are functions for groups of types:

- `question_is_categorical`,
- `question_is_numeric`,

And for testing individual types specifically:

- `question_is_select_multiple`,
- `question_is_select_one`,
- `question_is_sm_choice`,

Parsing skiplogic

Use `question_is_skipped` To find out which records of a certain question were skipped

Using Multiple Quesitonnaires

If multiple questionnaires need to be used in parallel, you can store the output of `link{load_questionniare}` in an object. All other functions in this package are then available relating to that specific questionnaire as a list element of that object. Example:

```
# load the first questionnaire:
q1<-load_questionnaire(...)

question_is_categorical(...) # global functions now refer to q1
q1$question_is_categorical(...) # always refers to q1
```

```
# load the second questionnaire:
q2<-load_quesitonniare(...)
question_is_categorical(...) # global functions now refer to q2
q1$question_is_categorical(...) # always refers to q1
q2$question_is_categorical(...) # always refers to q2
```

---

kobo_choices *Example kobo choices*

Description

Example kobo choices

Author(s)

Impact Initiatives <martin.barner@impact-initiatives.org>

References

www.impact-initiatives.org

---

kobo_questions *Example kobo questions*

Description

Example kobo questions

Author(s)

Impact Initiatives <martin.barner@impact-initiatives.org>

References

www.impact-initiatives.org

load_questionnaire *load_questionnaire*

Description

load_questionnaire

Usage

```
load_questionnaire(data, questions, choices,
  choices.label.column.to.use = NULL)
```

Arguments

data	data frame containing the data matching the questionnaire to be loaded.
questions	kobo form question sheet; either as a data frame, or a single character string with the name of a csv file
choices	questions kobo form choices sheet; either as a data frame, or a single character string with the name of a csv file
choices.label.column.to.use	The choices table has (sometimes multiple) columns with labels. They are often called "Label::English" or similar. Here you need to provide the _name of the column_ that you want to use for labels (see example!)

Value

A list containing the original questionnaire questions and choices, the choices matched 1:1 with the data columns, and all functions created by this function relating to the specific questionnaire

---

questionnaire *what is this? This package is using a specific kind of closures The factory is defined, but also the produced functions are *defined* and *documented* as part of the package. That allows us to make functions available once data is loaded, but still provide defaults for those functions and document them with vingnettes They consist of three parts: 1. The exported function; it calls an internal function that is not exported and therefore editable by the factory 2. A default for the internal function; usually throws and error or does some simple default behaviour 3. A factory; the factory fills the internal function with it's main code once the factory is called.*

Description

Here we define 1. and 2. All of the closures here are fabricated by load_questionnaire().

Usage

```
questionnaire
```

Format

An object of class NULL of length 0.

`question_get_choice_labels` *Convert kobo xml choice names to labels*

Description

Uses a loaded kobo questionnaire to get the labels of select_one and select_multiple type questions.

Usage

```
question_get_choice_labels(responses, variable.name)
```

Arguments

`responses`	A vector of responses in kobo xml name format
`variable.name`	The xml name of a kobo question. (as it appears in the kobo questionnaire and subsequently in the data column headers) Should be a question of type 'select_one' or 'select_multiple'.

Details

To use this you must first successfully run `load_questionnaire`. If conversion of a value in `responses` fails, the original input value is returned. This happens in the following cases:

- No questionnaire has been loaded with `load_questionnaire`
- `variable.name` could not be found in the questionnaire, or in the data that was supplied to `load_questionnaire`
- A value in `responses` is not listed in the loaded questionnaire
- `variable.name` is not listed as a question of type 'select_one' or 'select_multiple' in the questionnaire.

This does not work for concatenated responses of 'select_multiple' questions (e.g. strings of the form "choice_a choice_b"); The `responses` should have only a single response per element.

Value

A vector of strings with labels corresponding to the choices supplied in `responses`.

See Also

`load_questionnaire` must be run first. The equivalent of this but for question labels is `question_get_question_label`

Examples

```
question_get_choice_labels(mydata$location, "location")
```

question_get_question_label

Convert kobo xml question names to labels

Description

Uses a loaded kobo questionnaire to get the label corresponding to a question xml name / data column headers.

Usage

```
question_get_question_label(question.name)
```

Arguments

question.name	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run load_questionnaire. If conversion to label fails, the original input value is returned. This happens in the following cases:

- No questionnaire has been loaded with load_questionnaire
- question.name could not be found in the questionnaire, or in the data that was supplied to load_questionnaire

Value

A string with the kobo label of the question / data column header

See Also

load_questionnaire must be run first. The equivalent of this but for choice labels is question_get_choice_labels,

Examples

```
question_get_question_label("a_variable_name")
```

question_in_questionnaire

Determine if a question name can be found in the loaded questionnaire

Description

Determine if a question name can be found in the loaded questionnaire

Usage

```
question_in_questionnaire(question.name)
```

Arguments

question.name	The xml name of a (potential) kobo question as a string.

Value

TRUE if the question is listed in the loaded questionnaire. FALSE otherwise.

---

question_is_categorical

Determine if a kobo question is categorical

Description

Uses a loaded kobo questionnaire to look up the question type for a given question; returns true for select_one or select_multiple

Usage

```
question_is_categorical(question.name)
```

Arguments

question.name	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run load_questionnaire. This does not derive the data type from any actual data; it only looks up the type defined in the questionnaire. If type identification fails, the default return value is FALSE.This happens in the following cases:

- No questionnaire has been loaded with load_questionnaire
- question.name could not be found in the questionnaire, or in the data that was supplied to load_questionnaire

Value

TRUE if the question is listed as a select_one or select_multiple type in the questionnaire. FALSE if the question is listed as a different type. FALSE if the question type could not be determined from the questionnaire.

See Also

load_questionnaire must be run first. Use question_type for the most generalised way to guess the data type. Part of the question_is_* family of functions: testing for specific types: question_is_numeric, question_is_categorical, question_is_select_one, question_is_select_multiple, question_is_sm_choice

parsing kobo skip-logic: question_is_skipped

Examples

```
question_is_categorical("some_numeric_kobo_xml_question_name") # FALSE
question_is_categorical("a_select_one_kobo_xml_question_name") # TRUE
question_is_categorical("a_select_multiple_kobo_xml_question_name") # TRUE
question_is_categorical("some_unidentified_string") # FALSE
```

`question_is_numeric` *Determine if a kobo question is of type 'numeric'*

Description

Uses a loaded kobo questionnaire to look up the question type for a given question.

Usage

```
question_is_numeric(question.name)
```

Arguments

`question.name`	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run `load_questionnaire`. This does not derive the data type from any actual data; it only looks up the type defined in the questionnaire. If type identification fails, the default return value is `FALSE`.This happens in the following cases:

- No questionnaire has been loaded with `load_questionnaire`
- `question.name` could not be found in the questionnaire, or in the data that was supplied to `load_questionnaire`

Value

`TRUE` if the question is listed as a numeric type in the questionnaire. `FALSE` if the question is listed as a different type. `FALSE` if the question type could not be determined from the questionnaire.

See Also

`load_questionnaire` must be run first. Use `question_type` for the most generalised way to guess the data type. Part of the `question_is_*` family of functions: testing for specific types: `question_is_numeric`, `question_is_categorical`, `question_is_select_one`, `question_is_select_multiple`, `question_is_sm_choice`

parsing kobo skip-logic: `question_is_skipped`

Examples

```
question_is_numeric("some_numeric_kobo_xml_question_name") # TRUE
question_is_numeric("some_categorical_kobo_xml_question_name") # FALSE
question_is_numeric("some_unidentified_string") # FALSE
```

`question_is_select_multiple` *Determine if a kobo question is of type 'select_multiple'*

Description

Uses a loaded kobo questionnaire to look up the question type for a given question.

Usage

```
question_is_select_multiple(question.name)
```

Arguments

`question.name`	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run `load_questionnaire`. This does not derive the data type from any actual data; it only looks up the type defined in the questionnaire. If type identification fails, the default return value is `FALSE`.This happens in the following cases:

- No questionnaire has been loaded with `load_questionnaire`
- `question.name` could not be found in the questionnaire, or in the data that was supplied to `load_questionnaire`

Value

`TRUE` if the question is listed as a select_multiple type in the questionnaire. `FALSE` if the question is listed as a different type. `FALSE` if the question type could not be determined from the questionnaire.

See Also

`load_questionnaire` must be run first. Use `question_type` for the most generalised way to guess the data type. Part of the `question_is_*` family of functions: testing for specific types: `question_is_numeric`, `question_is_categorical`, `question_is_select_one`, `question_is_select_multiple`, `question_is_sm_choice`

parsing kobo skip-logic: `question_is_skipped`

Examples

```
question_is_select_multiple("some_numeric_kobo_xml_question_name") # FALSE
question_is_select_multiple("a_select_multiple_kobo_xml_question_name") # TRUE
question_is_select_multiple("a_select_one_kobo_xml_question_name") # FALSE
question_is_numeric("some_unidentified_string") # FALSE
```

question_is_select_one *Determine if a kobo question is of type 'select_one'*

Description

Uses a loaded kobo questionnaire to look up the question type for a given question.

Usage

```
question_is_select_one(question.name)
```

Arguments

question.name	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run load_questionnaire. This does not derive the data type from any actual data; it only looks up the type defined in the questionnaire. If type identification fails, the default return value is FALSE.This happens in the following cases:

- No questionnaire has been loaded with load_questionnaire
- question.name could not be found in the questionnaire, or in the data that was supplied to load_questionnaire

Value

TRUE if the question is listed as a select_one type in the questionnaire. FALSE if the question is listed as a different type. FALSE if the question type could not be determined from the questionnaire.

See Also

load_questionnaire must be run first. Use question_type for the most generalised way to guess the data type. Part of the question_is_* family of functions: testing for specific types: question_is_numeric, question_is_categorical, question_is_select_one, question_is_select_multiple, question_is_sm_choice

parsing kobo skip-logic: question_is_skipped

Examples

```
question_is_select_one("some_numeric_kobo_xml_question_name") # FALSE
question_is_select_one("a_select_one_kobo_xml_question_name") # TRUE
question_is_select_one("a_select_multiple_kobo_xml_question_name") # FALSE
question_is_numeric("some_unidentified_string") # FALSE
```

question_is_skipped *Determine if a variables records were skipped in the questionnaire*

Description

Uses a loaded kobo questionnaire and a dataset to determine which records have been skipped for a particular variable.

Usage

```
question_is_skipped(data, question.name)
```

Arguments

data	The dataset as a data.frame in standard kobo format, with the column headers matching the question names.
question.name	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run load_questionnaire. If for any reason skiplogic could not be determined, it returns FALSE for all records (with a warning).

Value

a logical vector with one value per row in data. TRUE if a record was skipped, FALSE otherwise.

See Also

load_questionnaire should be run first.

Examples

```
mydata<-lo
question_is_skipped("kobo_xml_question_name",mydata)
```

question_is_sm_choice *Determine if a data column header is a logical choice column of a select_multiple question*

Description

Determine if a data column header is a logical choice column of a select_multiple question

Usage

```
question_is_sm_choice(question.name)
```

Arguments

question.name	The xml name of a kobo question as a string. (as it appears in the kobo questionnaire and subsequently in the data column headers)

Details

To use this you must first successfully run load_questionnaire. This does not derive the data type from any actual data; it only looks up the type defined in the questionnaire. If type identification fails, the default return value is FALSE.This happens in the following cases:

- No questionnaire has been loaded with load_questionnaire
- question.name could not be found in the questionnaire, or in the data that was supplied to load_questionnaire

Value

TRUE if the question is listed as a logical column relating to a select_multiple type question in the questionnaire. FALSE if it isn't. FALSE if this could not be determined from the questionnaire.

See Also

load_questionnaire must be run first. Use question_type for the most generalised way to guess the data type. Part of the question_is_* family of functions: testing for specific types: question_is_numeric, question_is_categorical, question_is_select_one, question_is_select_multiple, question_is_sm_choice

parsing kobo skip-logic: question_is_skipped

Examples

```
question_is_sm_choice("a_select_multiple_question_name.a_choice_name") # TRUE
question_is_sm_choice("a_question_name") # FALSE
question_is_sm_choice("a_select_one_question_name") # FALSE
question_is_sm_choice("some_unidentified_string") # FALSE
```

---

question_type *question_type*

Description

Determines the kobo question type for a given variable name

Usage

```
question_type(variable.name, data = NULL, from.questionnaire = T,
  from.data = T)
```

Arguments

`variable.name`	the kobo question name for which the type should be determined. (works on a vector with multiple names)
`data`	the dataset matching the kobo questionnaire. This can be left empty if `from.data=F`. A provided dataset can be used as a fallback with `from.data=T` in case the data type can not be determined from the questionnaire.
`from.questionnaire`	if FALSE, prevent determinining data type from questionnaire. Can not be FALSE if from.data is also FALSE.
`from.data`	if TRUE, allows to determine data type from provided `data`. Can not be FALSE if from.data is also FALSE. If both from.questionnaire and from.data are TRUE, data types determined from the questionnaire have precedence.

Value

a string naming the question type. One of "select_one", "select_multiple" or "numeric".

See Also

Should be used after `load_questionnaire`, but can work without if data is provided as a fallback.

Examples

```
question_type("question_name_in_loaded_questionnaire")
```

Index