

Package ‘limonaid’

May 8, 2026

Title Working with 'LimeSurvey' Surveys and Responses

Version 25.5.5

Maintainer Gjalt-Jorn Peters <limonaid@opens.science>

License GPL (>= 3)

Description 'LimeSurvey' is Free/Libre Open Source Software for the development and administrations of online studies, using sophisticated tailoring capabilities to support multiple study designs (see <<https://www.limesurvey.org>>). This package supports programmatic creation of surveys that can then be imported into 'LimeSurvey', as well as user friendly import of responses from 'LimeSurvey' studies.

Encoding UTF-8

URL <https://limonaid.opens.science>

BugReports <https://codeberg.org/R-packages/limonaid/issues>

RoxygenNote 7.3.2

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports httr (>= 1.4), jsonlite (>= 1.7), R6 (>= 2.4), xml2

Suggests ggplot2, ggrepel, knitr, parallel, psyverse (>= 0.3),
rmarkdown, sticky, testthat

NeedsCompilation no

Author Gjalt-Jorn Peters [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-0336-9589>>),
Andrew Heiss [aut] (ORCID: <<https://orcid.org/0000-0002-3948-3914>>),
Urs Wilke [aut] (ORCID: <<https://orcid.org/0000-0001-7257-2524>>)

Repository CRAN

Date/Publication 2025-05-27 23:00:09 UTC

Contents

limonaid-package	3
add_answer_option_to_question	3
append_lsdf_rows	4
cat0	5
checkPkgs	5
convertToNumeric	6
emptyDf	7
export_with_languages	7
get_session_key	8
Group	9
limer_base64_to_df	13
limer_call_limer	13
limer_get_participants	14
limer_get_participant_property	15
limer_get_responses	15
limer_release_session_key	16
limer_upload_tsv_to_limesurvey	17
lsdf_for_language	18
ls_apply_script_bits	19
ls_eq_build	20
ls_eq_nestIfs	21
ls_import_data	22
ls_parse_data_import_script	24
ls_process_labels	25
ls_read_tsv	26
ls_recodeTable_to_equations	27
ls_tsv_get_group_rows	28
ls_tsv_get_rows	28
ls_tsv_rows	29
ls_write_lsg	29
ls_write_tsv	30
mail_registered_participant	31
massConvertToNumeric	31
opts	32
processLimeSurveyDropouts	33
Question	34
repeatStr	41
Survey	42
transpose_df	48
vecTxt	49

Index

51

limonaid-package *limonaid-package*

Description

Working With LimeSurvey Surveys and Responses

Details

LimeSurvey is Free/Libre Open Source Software for the development and administrations of online studies, using sophisticated tailoring capabilities to support multiple study designs. This package supports programmatic creation of surveys that can then be imported into LimeSurvey, as well as userfriendly import of responses from LimeSurvey studies.

Author(s)

Gjalt-Jorn Peters limonaid@opens.science

See Also

Useful links:

- <https://limonaid.opens.science>
- Report bugs at <https://codeberg.org/R-packages/limonaid/issues>

add_answer_option_to_question
Add an answer option to a question

Description

This is a convenience function that allows you to add an answer option to a question object.

Usage

```
add_answer_option_to_question(question, ...)
```

Arguments

question	The limonaid Question object
...	Options that are passed on to the Question's <code>add_answer_option()</code> method.

Value

The question object.

Examples

```
myQuestion <-
  limonaid::Question$new(
    code = 'myQuestion',
    type='radio'
  ) |>
  add_answer_option_to_question(
    code = 1,
    optionTexts = "First option"
  ) |>
  add_answer_option_to_question(
    code = 2,
    optionTexts = "Second option"
  );
```

append_lsdf_rows	<i>A home-rolled version of plyr::rbind.fill</i>
------------------	--

Description

This is used when creating dataframes for TSV exports.

Usage

```
append_lsdf_rows(data, row)
```

Arguments

data	The first dataframe.
row	The second dataframe.

Value

A merged dataframe.

Examples

```
limonaid::append_lsdf_rows(mtcars, iris);
```

cat0	<i>Concatenate to screen without spaces</i>
------	---

Description

The `cat0` function is to cat what `paste0` is to paste; it simply makes concatenating many strings without a separator easier.

Usage

```
cat0(..., sep = "")
```

Arguments

...	The character vector(s) to print; passed to <code>cat</code> .
sep	The separator to pass to <code>cat</code> , of course, "" by default.

Value

Nothing (invisible NULL, like `cat`).

Examples

```
cat0("The first variable is '", names(mtcars)[1], "'.");
```

checkPkgs	<i>Check for presence of a package</i>
-----------	--

Description

This function efficiently checks for the presence of a package without loading it (unlike `library()` or `require()`). This is useful to force yourself to use the `package::function` syntax for addressing functions; you can make sure required packages are installed, but their namespace won't attach to the search path.

Usage

```
checkPkgs(  
  ...,  
  install = FALSE,  
  load = FALSE,  
  repos = "https://cran.rstudio.com"  
)
```

Arguments

...	A series of packages. If the packages are named, the names are the package names, and the values are the minimum required package versions (see the second example).
install	Whether to install missing packages from repos.
load	Whether to load packages (which is exactly <i>not</i> the point of this function, but hey, YMMV).
repos	The repository to use if installing packages; default is the RStudio repository.

Value

Invisibly, a vector of the available packages.

Examples

```
limonaid::checkPkgs('base');

### Require a version
limonaid::checkPkgs(limonaid = "0.2.0");

### This will show the error message
tryCatch(
  limonaid::checkPkgs(
    base = "99",
    stats = "42.5",
    ufs = 20
  ),
  error = print
);
```

convertToNumeric	<i>Conveniently convert vectors to numeric</i>
------------------	--

Description

Tries to 'smartly' convert factor and character vectors to numeric.

Usage

```
convertToNumeric(vector, byFactorLabel = FALSE)
```

Arguments

vector	The vector to convert.
byFactorLabel	When converting factors, whether to do this by their label value (TRUE) or their level value (FALSE).

Value

The converted vector.

Examples

```
convertToNumeric(as.character(1:8));
```

emptyDf	<i>Create an empty dataframe</i>
---------	----------------------------------

Description

This function is used by [append_lsdf_rows\(\)](#), and you normally should not use it directly.

Usage

```
emptyDf(colnames, nrow, fillWith = "")
```

Arguments

colnames	The column names for the dataframe.
nrow	The number of rows.
fillWith	What to fill the dataframe with.

Value

The data.frame.

Examples

```
limonaid::emptyDf(c("x", "y"), 3);
```

export_with_languages	<i>Export a survey with a specific primary and additional languages</i>
-----------------------	---

Description

Sometimes it is useful to export a version of a survey with a different primary language, and/or less additional languages. This function allows that.

Usage

```
export_with_languages(
  x,
  language,
  path,
  additional_languages = NULL,
  new_sid = x$sid,
  backupLanguage = x$language,
  prefix = "limesurvey--",
  suffix = "",
  parallel = TRUE
)
```

Arguments

x	The Survey object.
language	The desired primary language.
path	The path where to save the .TSV file.
additional_languages	If specified, the selection of additional languages. If not specified, the survey's primary language will just be switched to language, and all original languages will be retained.
new_sid	If specified, a new sid to use.
backupLanguage	The language to use if an element is not specified in one of the languages.
prefix	The prefix to use in the filename.
suffix	The suffix to use in the filename.
parallel	Whether to use multiple cores when exporting the survey.

Value

Invisibly, the cloned and altered survey object.

Examples

```
### Add later
```

get_session_key	<i>Get a LimeSurvey API session key</i>
-----------------	---

Description

This function logs into the LimeSurvey API and provides an access session key. It was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```

get_session_key(
    username = getOption("lime_username"),
    password = getOption("lime_password"),
    hostname = getOption("lime_api")
)

```

Arguments

username	LimeSurvey username. Defaults to value set in options().
password	LimeSurvey password. Defaults to value set in options().
hostname	The host to use (if not using the one specified in the options). If no hostname is specified in the 'lime_api' option and no host name is passed as hostname, the subdomain stored in <code>limonaid::opts\$get("ls_subdomain")</code> will be combined with the domain stored in <code>limonaid::opts\$get("ls_domain")</code> to create the host name. You can change these using the <code>limonaid::opts\$set()</code> function.

Value

API token

Examples

```

## Not run:
get_session_key()

## End(Not run)

```

Group

R6 Class representing a LimeSurvey group

Description

R6 Class representing a LimeSurvey group
R6 Class representing a LimeSurvey group

Details

A group is mostly just a container for questions.

Public fields

`group_name` The group name / title / label
`description` The group description
`grelevance` The relevance equation for the group
`group_order` The group order (in the survey)
`randomization_group` The randomization group (that the group is a part of)
`language` The language of the group; or primary language, if there are multiple languages.
`additional_languages` Any additional languages for the title and description elements.
`id` The identifier of the group (a unique number in a survey)
`sid` The identifier of the survey that this group belongs to
`otherOptions` Any additional options, stored as a named list by assigning `as.list(...)`.
`questions` The questions in this group

Methods**Public methods:**

- `Group$new()`
- `Group$add_question()`
- `Group$export_to_lsg()`
- `Group$clone()`

Method `new()`: Create a new group object. Most of this text comes directly from the TSV manual page at https://www.limesurvey.org/manual/Tab_Separated_Value_survey_structure, so please see that page for more details.

Usage:

```

Group$new(
  group_name = "",
  description = "",
  grelevance = 1,
  group_order = 1,
  randomization_group = NULL,
  language = "en",
  additional_languages = "",
  id = NULL,
  sid = NULL,
  new_id_fun = NULL,
  uqid = NULL,
  repo_url = "https://operationalizations.com/questionnaires/json",
  ...
)
  
```

Arguments:

`group_name` The title of the group (if there are multiple languages, a named vector where every element is the title in another language and every element's name is the language code).

description The description of the group (if there are multiple languages, a named vector where every element is the title in another language and every element's name is the language code).
grelevance The group's relevance equation
group_order The group order (if the group is part of a survey)
randomization_group The group's randomization group
language The group's only or primary language
additional_languages Any additional languages
id Optionally, the id of the group.
sid Optionally, the identifier of the survey that this group belongs to.
new_id_fun A function to set identifiers (for XML exports, which mirrors MySQL tables and so needs identifiers). By default, new question objects receive this function from the group containing them; and groups receive it from the survey containing them. This ensures that identifiers are always unique in a survey (despite question objects not being able to 'see' anything in the group containing them, and group objects not being able to 'see' anything in the survey containing them; because they 'received' this function from the parent object, and it 'bubbles down' through groups to the questions, those functions still get and set a private identifier property in the 'top-most' object).
uqid A Unique Questionnaire Identifier (UQID) to import a questionnaire and populate the group with it.
repo_url The URL to a repo serving the questionnaire with the UQID in JSON.
 ... Any additional options, stored as a named list in the `otherOptions` property by assigning `as.list(...)`.

Returns: A new Group object.

Method `add_question()`: Add a question to a group object.

Usage:

```

Group$add_question(
  code,
  type = NULL,
  lsType = NULL,
  question_order = NULL,
  ...
)
  
```

Arguments:

code The question code.

type The question type.

lsType The question type, as LimeSurvey question type.

question_order The question order; automatically filled if left empty; starts counting at 0.

... Additional arguments are used to create the Question using `Question$new`.

Returns: Invisibly, the `thisQuestion` object that was just added. Note that you can further modify this, which will modify the question object "in" the survey group as well. This allows you to pipe the question creation on to, for example, add answer options.

Method `export_to_lsg()`: Export the group as an LSG (xml) file.

Usage:

```
Group$export_to_lsg(
  file = NULL,
  preventOverwriting = limonaid::opts$get("preventOverwriting"),
  encoding = limonaid::opts$get("encoding"),
  silent = limonaid::opts$get("silent"),
  backupLanguage = self$language
)
```

Arguments:

`file` The filename to which to save the file.
`preventOverwriting` Whether to prevent overwriting.
`encoding` The encoding to use
`silent` Whether to be silent or chatty.
`backupLanguage` The language to get content from if not from the primary language.
`parallel` Whether to work serially or in parallel.

Returns: Invisibly, the Survey object.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Group$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
myGroup <- limonaid::Group$new(
  group_name = "My Group"
);
myGroup$add_question(
  "testQuestion1",
  questionTexts = "First question",
  type="free text (short)"
);
myGroup$add_question(
  "testQuestion2",
  questionTexts = "Second question",
  type="radio"
);
myGroup$questions$testQuestion2$add_answer_option(
  "option1",
  "First option"
);
myGroup$questions$testQuestion2$add_answer_option(
  "option2",
  "Second option"
);

cat(as.character(myGroup$export_to_lsg()));
```

`limer_base64_to_df` *Convert base64 encoded data to a data frame*

Description

This function converts raw base64 results into a data frame. It was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```
limer_base64_to_df(  
  x,  
  encoding = NULL,  
  iconvArgs = list(from = "UTF-8", to = "UTF-8")  
)
```

Arguments

<code>x</code>	...
<code>encoding</code>	Either NULL or an encoding to pass to <code>textConnection()</code> .
<code>iconvArgs</code>	Arguments to pass to <code>[base::iconv()]</code> .

Examples

```
## Not run:  
limer_base64_to_df()  
  
## End(Not run)
```

`limer_call_limer` *Make a call to the LimeSurvey API*

Description

This function makes a generic call to the LimeSurvey API. See https://www.limesurvey.org/manual/RemoteControl_2_API for API documentation. It was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```
limer_call_limer(method, params = list(), ..., encoding = "utf-8")
```

Arguments

method	API function to call. Full list Defaults to value set in options().
params	Optional named list of parameters to pass to the function.
...	Other arguments passed to POST .
encoding	The encoding to use

Value

Results from the API (sometimes plain text, sometimes base64-encoded text).

Examples

```
## Not run:
limer_call_limer(method = "list_surveys")
limer_call_limer(method = "get_summary",
                 params = list(iSurveyID = 238481,
                              sStatname = "completed_responses"))

## End(Not run)
```

`limer_get_participants`

Export list of participants from a LimeSurvey survey

Description

This function exports and downloads the list of participants from a LimeSurvey survey.

Usage

```
limer_get_participants(iSurveyID, iStart, iLimit, bUnused, aAttributes)
```

Arguments

iSurveyID	...
iStart	...
iLimit	...
bUnused	...
aAttributes	...

Examples

```
## Not run:
limer_get_participants(12345, iStart=1, iLimit=10, bUnused=FALSE,
                      aAttributes=c('attribute_1','attribute_2'))
limer_get_participants(12345, iStart=1, iLimit=10, bUnused=FALSE, aAttributes=FALSE)

## End(Not run)
```

`limer_get_participant_property`*Get a participant property from a LimeSurvey survey*

Description

This function exports and downloads a participant property from a LimeSurvey survey. It was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```
limer_get_participant_property(  
  iSurveyID,  
  aTokenQueryProperties,  
  aTokenProperties  
)
```

Arguments

```
iSurveyID      ...  
aTokenQueryProperties  
              ...  
aTokenProperties  
              ...
```

Examples

```
## Not run:  
limer_get_participant_property(  
  iSurveyID = 12345,  
  aTokenQueryProperties = 1,  
  aTokenProperties = list("attribute_1")  
);  
  
## End(Not run)
```

`limer_get_responses` *Export data from a LimeSurvey survey*

Description

This function exports and downloads data from a LimeSurvey survey. It was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```

limer_get_responses(
  iSurveyID,
  sDocumentType = "csv",
  sLanguageCode = NULL,
  sCompletionStatus = "complete",
  sHeadingType = "code",
  sResponseType = "long",
  encoding_limerCall = NULL,
  encoding_txtCon = NULL,
  ...
)

```

Arguments

iSurveyID	The LimeSurvey survey identifier (the sid, usually 6 digits long).
sDocumentType	...
sLanguageCode	...
sCompletionStatus	...
sHeadingType	...
sResponseType	...
encoding_limerCall	The encoding to pass to the <code>limer_call_limer()</code> function.
encoding_txtCon	The encoding to pass to <code>limer_base64_to_df()</code> .
...	Further arguments to <code>limer_call_limer</code> .

Examples

```

## Not run:
limer_get_responses(12345)

## End(Not run)

```

```
limer_release_session_key
```

Release a LimeSurvey API session key

Description

This function clears the LimeSurvey API session key currently in use, effectively logging out. This function was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```
limer_release_session_key()
```

Examples

```
## Not run:  
limesurvey::limer_release_session_key()  
  
## End(Not run)
```

```
limer_upload_tsv_to_limesurvey  
Upload a tab separated limesurvey text file
```

Description

To use this function, you need to setup R for the LimeSuevey API, as described in vignette("limesurvey_api_setup").

Usage

```
limer_upload_tsv_to_limesurvey(  
  ls_txt_path,  
  open_url = "preview",  
  hostname = getOption("lime_api")  
)
```

Arguments

ls_txt_path	Path of the limesurvey text file
open_url	Character vector containing one or more of the strings in c("preview", "survey", "none"). If it contains "none", nothing is done. "preview" (the default) previews the survey on limesurvey. "survey" opens the survey summary.
hostname	The host to use (if not using the one specified in the options). If no hostname is specified in the 'lime_api' option and no host name is passed as hostname, the subdomain stored in limonaid::opts\$get("ls_subdomain") will be combined with the domain stored in limonaid::opts\$get("ls_domain") to create the host name. You can change these using the limonaid::opts\$set() function.

Value

The value of the id of your survey in the specified LimeSurvey installation,

Examples

```
## Not run:
### Log into the LimeSurvey API:
limonaid::get_session_key();

### Upload a tab separated values file:
limer_upload_tsv_to_limesurvey(
  "PATH/TO/YOUR/LIMESURVEY/TXT FILE",
  c("preview", "survey")
);

## End(Not run)
```

lsdf_for_language	<i>Produce the dataframe containing the survey for one language</i>
-------------------	---

Description

This is used when exporting surveys to LimeSurvey's TSV format.

Usage

```
lsdf_for_language(
  language,
  groups,
  exportGroupIdMapping,
  exportQuestionIdMapping,
  backupLanguage,
  silent = limonaid::opts$get("silent")
)
```

Arguments

language	The language for which to produce the data frame.
groups	The groups object in the Survey object.
exportGroupIdMapping, exportQuestionIdMapping	Used to map Survey object identifier onto the identifier model used in the LimeSurvey TSV.
backupLanguage	The language to get content from if not available in the primary language
silent	Whether to be silent or chatty.

Value

Invisibly, the Survey object.

ls_apply_script_bits *Apply specific code bits from LimeSurvey data import R script*

Description

This function applies specific code bits from the LimeSurvey data import R script, read by [ls_parse_data_import_script\(\)](#) for example to update variable names, set labels, etc.

Usage

```
ls_apply_script_bits(  
  data,  
  scriptBits,  
  setVarNames = TRUE,  
  setLabels = TRUE,  
  convertToCharacter = FALSE,  
  convertToFactor = FALSE,  
  categoricalQuestions = NULL,  
  massConvertToNumeric = TRUE,  
  silent = limonaid::opts$get("silent"),  
  sticky = limonaid::opts$get("sticky")  
)
```

Arguments

data	The dataframe.
scriptBits	The object returned by the call to ls_parse_data_import_script() .
setVarNames, setLabels, convertToCharacter, convertToFactor	Whether to set variable names or labels, or convert to character or factor, using the code isolated using the specified regular expression.
categoricalQuestions	Which variables (specified using LimeSurvey variable names) are considered categorical questions; for these, the script to convert the variables to factors, as extracted from the LimeSurvey import file, is applied.
massConvertToNumeric	Whether to convert all variables to numeric using massConvertToNumeric .
silent	Whether to be silent or verbose ('chatty').
sticky	Whether to make labels sticky (requires the sticky package).

Value

The dataframe.

 ls_eq_build

Building LimeSurvey Expression Manager equations

Description

These are a set of really basic functions that facilitate building LimeSurvey Expression Manager (LSEM) equations.

Usage

```
ls_eq_build(lhs, operator, rhs)
```

```
ls_eq_is(varCode, value, naok = TRUE)
```

```
ls_eq_isChecked(varCode, naok = TRUE)
```

```
ls_eq_isUnchecked(varCode, naok = TRUE)
```

```
ls_eq_if(cond, ifExpr, elseExpr)
```

```
ls_eq_ifRegex(regex, varCode, ifExpr, elseExpr, naok = TRUE)
```

```
ls_eq_brace(expr)
```

```
ls_eq_quote(expr)
```

Arguments

lhs	The left-hand side expression.
operator	The operator.
rhs	The right-hand side expression.
varCode	A LimeSurvey variable code.
value	A value.
naok	Whether to append ".NAOK" to the variable code.
cond	A condition, for example created by ls_eq_build() or ls_eq_is().
ifExpr, elseExpr, expr	An expression.
regex	A regular expression.

Details

ls_eq_build() just pastes together its three arguments in the same order using a space as separator. So it's mostly used for clarity when building LSEM equations.

ls_eq_is() uses ls_eq_build() to specify a logical expression that is true when varCode equals value.

ls_eq_if() builds an if/then/else expression; if cond evaluates to TRUE, the LSEM uses ifExpr; otherwise, it uses elseExpr.

ls_eq_ifRegex checks a question against a regular expression.

ls_eq_isChecked() and ls_eq_isUnchecked() return an expression evaluating whether a checkbox is checked (or not).

ls_eq_brace() simply embraces expr, an expression (i.e. it prepends { and appends }).

ls_eq_quote() simply embraces expr, an expression (i.e. it prepends ' and appends ').

Value

A character vector.

Examples

```
ls_eq_build("questionCode", "=", "Y");
```

ls_eq_nestIfs	<i>Create a series of nested LSEM if equations</i>
---------------	--

Description

This function takes a series of conditions and corresponding values, and builds an equation consisting of nested if statements.

Usage

```
ls_eq_nestIfs(conditions, values, elseExpr, quoteValues = FALSE)
```

Arguments

conditions	The conditions - in the right order, i.e. in the produced expression if nested if statements, the first condition in this list will be checked first, then the second, etc.
values	The values corresponding to each condition (in the same order!).
elseExpr	The value to return if there are no matches.
quoteValues	Whether to use double quotes to quote the values.

Value

A character value.

Examples

```
### Relatively simple example with four levels of nesting
ls_eq_nestIfs(c("age.NAOK > 80",
              "age.NAOK > 65",
              "age.NAOK > 40",
              "age.NAOK > 20"),
            c("Respectable",
              "Roughly retired",
              "Roughly middle-aged",
              "Quite young"),
            "Very young",
            quoteValue=TRUE);
```

 ls_import_data

Reading LimeSurvey data exported to R

Description

This function can be used to import files exported by LimeSurvey.

Usage

```
ls_import_data(
  sid = NULL,
  path = NULL,
  datafile = NULL,
  dataPath = NULL,
  datafileRegEx = NULL,
  scriptfile = NULL,
  setVarNames = TRUE,
  setLabels = TRUE,
  convertToCharacter = FALSE,
  convertToFactor = FALSE,
  categoricalQuestions = NULL,
  massConvertToNumeric = TRUE,
  dataHasVarNames = TRUE,
  dataEncoding = "UTF-8-BOM",
  scriptEncoding = NULL,
  sticky = limonaid::opts$get("sticky"),
  silent = limonaid::opts$get("silent")
)
```

Arguments

sid, path The easiest way to load data is to not rename the datafile and script file downloaded from LimeSurvey (so that both contain the Survey Identifier, the sid) and simply specify that sid and the path where both files are stored.

datafile	The path and filename of the file containing the data (comma separated values).
dataPath, datafileRegex	Path containing datafiles: this can be used to read multiple datafiles, if the data is split between those. This is useful when downloading the entire datafile isn't possible because of server restrictions, for example when the processing time for the script in LimeSurvey that generates the datafiles is limited. In that case, the data can be downloaded in portions, and specifying a path here enables reading all datafiles in one go. Use the regular expression to indicate which files in the path should be read.
scriptfile	The path and filename of the file containing the R script to import the data.
setVarNames, setLabels, convertToCharacter, convertToFactor	Whether to set variable names or labels, or convert to character or factor, using the code isolated using the specified regular expression.
categoricalQuestions	Which variables (specified using LimeSurvey variable names) are considered categorical questions; for these, the script to convert the variables to factors, as extracted from the LimeSurvey import file, is applied.
massConvertToNumeric	Whether to convert all variables to numeric using massConvertToNumeric .
dataHasVarNames	Whether the variable names are included as header (first line) in the comma separated values file (data file).
dataEncoding, scriptEncoding	The encoding of the files; can be used to override the setting in the limonaid options (i.e. in opts) in the encoding field (the default value is "UTF-8").
sticky	Whether to make labels sticky (requires the sticky package).
silent	Whether to be silent or verbose ('chatty').

Details

This function was intended to make importing data from LimeSurvey a bit easier. The default settings used by LimeSurvey are not always convenient, and this function provides a bit more control.

Value

The dataframe.

Examples

```
## Not run:
### Of course, you need valid LimeSurvey files. This is an example of
### what you'd do if you have them, assuming you specified that path
### containing the data in 'dataPath', the name of the datafile in
### 'dataFileName', the name of the script file in 'dataLoadScriptName',
### and that you only want variables 'informedConsent', 'gender', 'hasJob',
### 'currentEducation', 'prevEducation', and 'country' to be converted to
### factors.
dat <- limonaid::ls_import_data(
```

```
datafile = file.path(dataPath, dataFileName),
scriptfile = file.path(dataPath, dataLoadScriptName),
categoricalQuestions = c('informedConsent',
                        'gender',
                        'hasJob',
                        'currentEducation',
                        'prevEducation',
                        'country')
);

## End(Not run)
```

ls_parse_data_import_script

Extract specific code bits from LimeSurvey data import R script

Description

This function extracts specific code bits from the LimeSurvey data import R script, which can then be applied to imported data using `ls_apply_script_bits()`, for example to update variable names, set labels, etc.

Usage

```
ls_parse_data_import_script(
  scriptfile = NULL,
  scriptEncoding = limonaid::opts$get("encoding"),
  silent = limonaid::opts$get("silent")
)
```

Arguments

<code>scriptfile</code>	The path and filename of the script file.
<code>scriptEncoding</code>	The encoding of the script file; can be used to override the setting in the <code>limonaid</code> options (i.e. in <code>opts</code>) in the encoding field (the default value is "UTF-8").
<code>silent</code>	Whether to be silent or verbose ('chatty').

Value

A list with four components.

ls_process_labels *A function to conveniently process LimeSurvey labels*

Description

This function is meant to quickly parse the variable labels set by LimeSurvey. It works particularly well with dual anchor array questions, where the left and right anchors as well as the subquestions are extracted automatically.

Usage

```
ls_process_labels(  
  data,  
  varnameRegExPairs = NULL,  
  lengthToWrap = 50,  
  lengthToWrapAnchors = 20,  
  labelExtractionRegExPair = limonaid::opts$get("labelExtractionRegExPair"),  
  leftAnchorRegExPairs = limonaid::opts$get("leftAnchorRegExPairs"),  
  rightAnchorRegExPairs = limonaid::opts$get("rightAnchorRegExPairs")  
)
```

Arguments

data The dataframe as produced by `ls_import_data()`.

varnameRegExPairs Pairs of regular expressions to replace in the variable names. This is useful when some pattern can be applied to the variable names to, for example, add underscores to denote different parts of the variable name. This has to be a list of character vectors that each have length 2.

lengthToWrap At how many characters to wrap the subquestions.

lengthToWrapAnchors At how many characters to wrap the anchors.

labelExtractionRegExPair The regular expression pair used to extract the labels.

leftAnchorRegExPairs The regular expression pairs to use to extract the left anchors.

rightAnchorRegExPairs The regular expression pairs to use to extract the right anchors.

Details

This function processes LimeSurvey variable labels and applies regular expressions to automatically extract subquestions and left and right anchors.

Value

A dataframe.

Examples

```
### No examples provided yet; this would require data to be included,  
### and that's not available yet.
```

ls_read_tsv	<i>Read a LimeSurvey Tab-Separated Values file</i>
-------------	--

Description

Read a LimeSurvey Tab-Separated Values file

Usage

```
ls_read_tsv(file, encoding = limonaid::opts$get("encoding"))
```

Arguments

file	The filename to read.
encoding	The encoding to use when reading the file.

Value

A dataframe.

Examples

```
### Get location of one of the example files  
exampleFile <-  
  system.file(  
    "extdata",  
    "export-of-survey-with-one-question-as-tsv.txt",  
    package = "limonaid"  
  );  
  
### Import file  
lsrv <- limonaid::ls_read_tsv(exampleFile);
```

 ls_recodeTable_to_equations

Recode a set of LS variables codes and values into LSEM equations

Description

This function takes a dataframe with LimeSurvey (LS) variable codes and values, and builds a nested set of LimeSurvey Equation Manager (LSEM) if/then/else equations where the variable code in each row (in the varCodeCol) is compared to the corresponding value (i.e. the value in the same row in the valueCol column) using the operator specified in that row in the operatorCol column (or the == operator, if no operator is specified). In the case of a match, the value in the corresponding recodeToCol column is returned. If there is no match, the comparison on the next row is evaluated, all the way down. If nothing matches, the elseExpr is returned.

Usage

```
ls_recodeTable_to_equations(
  data,
  varCodeCol = limonaid::opts$get("recTab2Eq_varCodeCol"),
  valueCol = limonaid::opts$get("recTab2Eq_valueCol"),
  recodeToCol = limonaid::opts$get("recTab2Eq_recodeToCol"),
  operatorCol = limonaid::opts$get("recTab2Eq_operatorCol"),
  elseExpr = limonaid::opts$get("eq_elseExpr"),
  naok = TRUE
)
```

Arguments

data	The dataframe.
varCodeCol	The name or index of the column with the variable code.
valueCol	The name or index of the column with the values to compare the value of the variable code to.
recodeToCol	The name or index of the column with the value to return in the case of a match.
operatorCol	The name or index of the column with the operator used to build each logical expression.
elseExpr	The value to return if there are no matches.
naok	Whether to append ".NAOK" to variable codes by default.

Value

A character value.

Examples

```
### Provide later
```

ls_tsv_get_group_rows *Get all group rows from a LimeSurvey survey dataframe*

Description

Get all group rows from a LimeSurvey survey dataframe

Usage

```
ls_tsv_get_group_rows(data)
```

Arguments

data The LimeSurvey survey dataframe.

Value

A dataframe with the rows.

Examples

```
### Add
```

ls_tsv_get_rows *Display rows from a LimeSurvey dataframe that meet a criterion*

Description

Display rows from a LimeSurvey dataframe that meet a criterion

Usage

```
ls_tsv_get_rows(data, ...)
```

Arguments

data The datafram.
... For now, one column/value pair (the criterion).

Value

The rows, passed through `ls_tsv_rows()`.

Examples

```
### Add later
```

ls_tsv_rows	<i>Display one or more rows from a LimeSurvey dataframe, omitting empty columns</i>
-------------	---

Description

Display one or more rows from a LimeSurvey dataframe, omitting empty columns

Usage

```
ls_tsv_rows(dfRows)
```

Arguments

dfRows A dataframe with the selected rows.

Value

The rows, with empty columns omitted.

Examples

```
### Add later.
```

ls_write_lsg	<i>Write a data frame to a LimeSurvey Tab Separated Values file</i>
--------------	---

Description

Write a data frame to a LimeSurvey Tab Separated Values file

Usage

```
ls_write_lsg(  
  data,  
  file,  
  encoding = limonaid::opts$get("encoding"),  
  preventOverwriting = limonaid::opts$get("preventOverwriting"),  
  silent = limonaid::opts$get("silent")  
)
```

Arguments

data	The dataframe to write.
file	The file to write to.
encoding	The encoding to write to.
preventOverwriting	Whether to prevent overwriting, should the target file exist, already.
silent	Whether to be silent or chatty.

Value

The dataframe, adapted for writing, invisibly.

Examples

```
### Add example once something is available.
```

ls_write_tsv	<i>Write a data frame to a LimeSurvey Tab Separated Values file</i>
--------------	---

Description

Write a data frame to a LimeSurvey Tab Separated Values file

Usage

```
ls_write_tsv(
  data,
  file,
  encoding = limonaid::opts$get("encoding"),
  preventOverwriting = limonaid::opts$get("preventOverwriting"),
  silent = limonaid::opts$get("silent")
)
```

Arguments

data	The dataframe to write.
file	The file to write to.
encoding	The encoding to write to.
preventOverwriting	Whether to prevent overwriting, should the target file exist, already.
silent	Whether to be silent or chatty.

Value

The dataframe, adapted for writing, invisibly.

Examples

```
### Add example once something is available.
```

```
mail_registered_participant  
    Mail registered participant
```

Description

This function was adapted by Gjalt-Jorn Peters from a function originally written by Andrew Heiss.

Usage

```
mail_registered_participant(iSurveyID, tid)
```

Arguments

```
iSurveyID    ...  
tid          ...
```

Examples

```
## Not run:  
limonaid::mail_registered_participant(iSurveyID = 123456, tid = 2)  
  
## End(Not run)
```

```
massConvertToNumeric    Converting many dataframe columns to numeric
```

Description

This function makes it easy to convert many dataframe columns to numeric.

Usage

```
massConvertToNumeric(  
  dat,  
  byFactorLabel = FALSE,  
  ignoreCharacter = TRUE,  
  stringsAsFactors = FALSE  
)
```

Arguments

<code>dat</code>	The dataframe with the columns.
<code>byFactorLabel</code>	When converting factors, whether to do this by their label value (TRUE) or their level value (FALSE).
<code>ignoreCharacter</code>	Whether to convert (FALSE) or ignore (TRUE) character vectors.
<code>stringsAsFactors</code>	In the returned dataframe, whether to return string (character) vectors as factors or not.

Value

A data.frame.

Examples

```
### Create a dataset
a <- data.frame(var1 = factor(1:4),
                var2 = as.character(5:6),
                stringsAsFactors=FALSE);

### Ignores var2
b <- massConvertToNumeric(a);

### Converts var2
c <- massConvertToNumeric(a,
                          ignoreCharacter = FALSE);
```

opts

Options for the limonaid package

Description

The `limonaid::opts` object contains three functions to set, get, and reset options used by the `escalc` package. Use `limonaid::opts$set` to set options, `limonaid::opts$get` to get options, or `limonaid::opts$reset` to reset specific or all options to their default values.

Usage

```
opts
```

Format

An object of class `list` of length 4.

Details

It is normally not necessary to get or set limonaId options.

The following arguments can be passed:

... For limonaId::opts\$set, the dots can be used to specify the options to set, in the format option = value, for example, silent = FALSE. For limonaId::opts\$reset, a list of options to be reset can be passed.

option For limonaId::opts\$set, the name of the option to set.

default For limonaId::opts\$get, the default value to return if the option has not been manually specified.

The following options can be set:

silent Whether to be chatty or silent.

encoding The encoding to use when writing files.

preventOverwriting The name of the column with the missing values.

Examples

```
### Get the default silent setting
limonaId::opts$get('silent');

### Set it to FALSE
limonaId::opts$set(silent = FALSE);

### Check that it worked
limonaId::opts$get('silent');

### Reset this option to its default value
limonaId::opts$reset('silent');

### Check that the reset worked, too
limonaId::opts$get('silent');
```

```
processLimeSurveyDropouts
```

```
    Process LimeSurvey dropouts
```

Description

This function makes it easy to parse the dropouts from a LimeSurvey questionnaire.

Usage

```
processLimeSurveyDropouts(lastpage, pagenames = NULL, relevantPagenames = NULL)
```

Arguments

lastpage	A vector with the 'lastpage' variable as LimeSurvey stores it (an integer denoting the last page a participant visited, in other words, where they dropped out).
pagenames	Optional: names for each page.
relevantPagenames	Optional: the names of those pages that should be included.

Details

This will be described more in detail in a forthcoming publications.

Value

A list with information about the dropout, including plots.

Examples

```
limonaid::processLimeSurveyDropouts(c(1,2,1,1,2,3,2,2,3,2,1));
```

Question

R6 Class representing a LimeSurvey question

Description

R6 Class representing a LimeSurvey question

R6 Class representing a LimeSurvey question

Details

A question has at least a code and a primary language.

The human-readable question types are (with some additional variants also being valid, in any case the literal labels used at https://www.limesurvey.org/manual/Question_object_types#Current_question_types):

- "array dual scale"
- "5 point choice"
- "5 point array"
- "10 point array"
- "yes/no/uncertain array"
- "date"
- "increase/same/decrease array"
- "array" (this is the "array (flexible labels)" type)
- "gender"

- "array by column"
- "language switch"
- "multiple numerical input",
- "radio" (this is the "list" type)
- "checkboxes" (this is the "multiple choice" type)
- "numerical input",
- "list with comment"
- "multiple choice with comments"
- "multiple short text"
- "ranking"
- "short text"
- "long text"
- "huge text"
- "text display"
- "yes/no"
- "multiple texts array",
- "multiple dropdown array"
- "file"
- "dropdown"
- "equation".

Public fields

`code` The code of the question.

`id` The identifier of the question (a unique number in a survey).

`gid` The identifier of the group to which this question belongs.

`sid` The identifier of the survey to which this question belongs.

`type` The question type.

`lsType` The question type in LimeSurvey's format.

`questionTexts` The question text(s) in all languages.

`helpTexts` The question help text(s) in all languages.

`relevance` The relevance.

`validation` The question's validation.

`language` The primary language of the question.

`additional_languages` Any additional languages for the title and description elements.

`answerOptions` The answer options in the question.

`subquestions` The subquestions in the question.

`parent_qid` The question identifier of the parent question (or 0).

mandatory Whether the question is mandatory (Y or N).
other Whether the question has an 'other' option (Y or N).
otherReplaceTexts If the question has an 'other' option, its label if the default label should be overwritten (multilingual).
default The default value.
same_default Not entirely sure what this does.
array_filter The question code of the array filter question to apply.
question_order The question order (starts at 0)
cssclass The CSS class(es) to apply to this question.
hide_tip Whether to hide the tip (Y or N).
otherOptions Any additional options, stored as a named list by assigning as `.list(...)`.

Active bindings

has_subquestions Whether the question has subquestions.
has_answerOptions Whether the question has answer options

Methods

Public methods:

- `Question$new()`
- `Question$add_answer_option()`
- `Question$add_subquestion()`
- `Question$xmlExport_row_question()`
- `Question$xmlExport_row_subquestions()`
- `Question$xmlExport_row_question_l10ns()`
- `Question$xmlExport_row_answers()`
- `Question$xmlExport_row_answer_l10ns()`
- `Question$xmlExport_row_attributes()`
- `Question$clone()`

Method new(): Create a new question object. Most of this text comes directly from the TSV manual page at https://www.limesurvey.org/manual/Tab_Separated_Value_survey_structure, so please see that page for more details.

Usage:

```

Question$new(
  code,
  type = NULL,
  lsType = NULL,
  id = NULL,
  gid = NULL,
  sid = NULL,
  questionTexts = "",
  helpTexts = "",

```

```

    relevance = 1,
    validation = "",
    mandatory = "N",
    parent_qid = 0,
    other = "N",
    otherReplaceTexts = "",
    default = "",
    same_default = "0",
    array_filter = "",
    cssclass = "",
    hide_tip = "",
    language = "en",
    additional_languages = "",
    new_id_fun = NULL,
    question_order = 0,
    ...
)

```

Arguments:

`code` The question code.

`type` The human-readable question type (see details).

`lsType` The type as LimeSurvey type ("1"; "5"; "A" to "Y", except "J", "V" and "W"; "!"; ":"; ";"; "*"; or "|") –see https://www.limesurvey.org/manual/Question_object_types#Current_question_types.

`id` The identifier of the question (in a survey).

`gid` The identifier of the group to which this question belongs.

`sid` The identifier of the survey to which this question belongs.

`questionTexts` The question text(s).

`helpTexts` The help text(s).

`relevance` The question's relevance equation.

`validation` The question's validation.

`mandatory` Whether the question is mandatory (Y or N);.

`parent_qid` The question identifier of the parent question (or 0).

`other` Whether the question has an 'other' option (Y or N).

`otherReplaceTexts` If the question has an 'other' option, its label if the default label should be overwritten (multilingual).

`default` The default value.

`same_default` Y for true, in which case any default value set for the primary language applies to other languages.

`array_filter` The question code of the array filter question to apply.

`cssclass` The CSS class(es) to apply to this question.

`hide_tip` Whether to hide the tip (Y or N).

`language` The question's primary language.

`additional_languages` Any additional languages

`new_id_fun` A function to set identifiers (for XML exports, which mirrors MySQL tables and so needs identifiers). By default, new question objects receive this function from the group

containing them; and groups receive it from the survey containing them. This ensures that identifiers are always unique in a survey (despite question objects not being able to 'see' anything in the group containing them, and group objects not being able to 'see' anything in the survey containing them; because they 'received' this function from the parent object, and it 'bubbles down' through groups to the questions, those functions still get and set a private identifier property in the 'top-most' object).

`question_order` The question order (starts at 0)

... Any additional options, stored as a named list in the `otherOptions` property by assigning `as.list(...)`.

Returns: A new Question object.

Method `add_answer_option()`: Add an answer option to a question. Most of this text comes directly from the TSV manual page at https://www.limesurvey.org/manual/Tab_Separated_Value_survey_structure, so please see that page for more details.

Usage:

```
Question$add_answer_option(
  code,
  optionTexts,
  type.scale = 0,
  relevance = "",
  assessment.value = 0,
  sort.order = NULL
)
```

Arguments:

`code` The answer option code.

`optionTexts` The answer option text(s).

`type.scale` 0 or 1 (e.g. for dual-scale; 'scale_id').

`relevance` The answer option's relevance equation.

`assessment.value` If using assessment, this is the assessment value for the answer ('assessment_value').

`sort.order` The sort order (to manually specify); starts at 0. If left empty, new options are added at the bottom.

Returns: Invisibly, the question object.

Method `add_subquestion()`: Add a subquestion to a question. Most of this text comes directly from the TSV manual page at https://www.limesurvey.org/manual/Tab_Separated_Value_survey_structure, so please see that page for more details.

Usage:

```
Question$add_subquestion(
  code,
  subquestionTexts,
  relevance = "",
  helpTexts = NULL,
  type.scale = 0,
  validation = "",
  mandatory = "",
```

```

    default = "",
    same_default = "",
    subquestion.order = NULL
)

```

Arguments:

`code` The subquestions code.

`subquestionTexts` The subquestion text(s).

`relevance` When to show this subquestion.

`helpTexts` As far as I know not yet implemented in LimeSurvey; but the TSV help page says "(Future) to support subquestion-level help".

`type.scale` 0 or 1, depending upon question type (e.g. array text will have two scales)0 or 1, depending upon question type (e.g. array text will have two scales)."

`validation` As far as I know not yet implemented in LimeSurvey; but the TSV help page says "(Future) to support subquestion-level regular expression validation (e.g. for address parts)"

`mandatory` As far as I know not yet implemented in LimeSurvey; but the TSV help page says "(Future) to support subquestion-level mandatory (e.g. make only a few subquestions mandatory)"

`default` If set, then this is the default value for the subquestion (inserted into defaultvalues table).

`same_default` If set, then the default for the primary language is used for all other languages.

`subquestion.order` The subquestion order (to manually specify); starts at 0. If left empty, new options are added at the bottom.

Returns: Invisibly, the question object.

Method `xmlExport_row_question()`: Export the question in XML format (for lss, lsg, or lsq files).

Usage:

```
Question$xmlExport_row_question(silent = limonaid::opts$get("silent"))
```

Arguments:

`silent` Whether to be silent or chatty.

Returns: The produced XML

Method `xmlExport_row_subquestions()`: Export the question in XML format (for lss, lsg, or lsq files).

Usage:

```
Question$xmlExport_row_subquestions(
  returnRows = FALSE,
  silent = limonaid::opts$get("silent")
)
```

Arguments:

`returnRows` Whether to return a list with each row as element, or a rows node (as xml2 object) containing each row as nodes

`silent` Whether to be silent or chatty.

Returns: The produced XML

Method `xmlExport_row_question_l10ns()`: Export the question's `question_l10ns` info in a list of XML nodes.

Usage:

```
Question$xmlExport_row_question_l10ns(
  id_fun = private$new_id(),
  silent = limonaid::opts$get("silent")
)
```

Arguments:

`id_fun` The function to use to produce unique identifiers
`silent` Whether to be silent or chatty.

Returns: The produced list of XML nodes

Method `xmlExport_row_answers()`: Export the answer options in XML format (for `lss`, `lsg`, or `lsq` files).

Usage:

```
Question$xmlExport_row_answers(
  returnRows = FALSE,
  silent = limonaid::opts$get("silent")
)
```

Arguments:

`returnRows` Whether to return a list with each row as element, or a rows node (as `xml2` object) containing each row as nodes
`silent` Whether to be silent or chatty.

Returns: The produced XML

Method `xmlExport_row_answer_l10ns()`: Export the question's answer options info in a list of XML nodes.

Usage:

```
Question$xmlExport_row_answer_l10ns(
  id_fun = private$new_id(),
  silent = limonaid::opts$get("silent")
)
```

Arguments:

`id_fun` The function to use to produce unique identifiers
`silent` Whether to be silent or chatty.

Returns: The produced list of XML nodes

Method `xmlExport_row_attributes()`: Export the question's attributes in a list of XML nodes.

Usage:

```
Question$xmlExport_row_attributes(silent = limonaid::opts$get("silent"))
```

Arguments:

`silent` Whether to be silent or chatty.

Returns: The produced list of XML nodes

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
Question$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

repeatStr	<i>Repeat a string a number of times</i>
-----------	--

Description

Repeat a string a number of times

Usage

```
repeatStr(n = 1, str = " ")
```

Arguments

n, str Normally, respectively the frequency with which to repeat the string and the string to repeat; but the order of the inputs can be switched as well.

Value

A character vector of length 1.

Examples

```
### 10 spaces:  
repStr(10);
```

```
### Three euro symbols:  
repStr("\u20ac", 3);
```

 Survey

R6 Class representing a LimeSurvey survey

Description

R6 Class representing a LimeSurvey survey

R6 Class representing a LimeSurvey survey

Details

Create and work with a Survey to programmatically (or interactively) create a survey, export it to a tab separated values file, and import it to LimeSurvey.

Public fields

`titles` The title of the survey in the primary language and any additional languages

`descriptions` The descriptions of the survey in the primary language and any additional languages

`welcomeTexts` The welcome texts of the survey in the primary language and any additional languages

`endTexts` The end texts of the survey in the primary language and any additional languages

`endURLs` The end URLs of the survey in the primary language and any additional languages

`endURLdescriptions` The end URL descriptions of the survey in the primary language and any additional languages

`dateformats` The date format to use in the primary language and any additional languages; the index of the option from the dropdown in LimeSurvey (6 is the ISO standard, "YYYY-MM-DD").

`numberformats` The number format to use in the primary language and any additional languages (for periods as decimal marks, 0; for commas as decimal marks, 1).

`sid` The unique survey identifier; if this is free when importing the survey, this will be used.

`gsid` The Survey Group identifier.

`admin` The name of the survey administrator

`adminemail` The email address of the survey administrator

`anonymized` Whether the survey uses anonymized responses (Y or N).

`faxto` The contents of the "Fax to" field

`format` How to present the survey (Q for question by question; G for group by group; and A for all in one).

`savetimings` Whether to save timings of responses (Y or N).

`template` The name of the LimeSurvey theme to use.

`language` The primary language of the survey.

`additional_languages` Any additional languages the survey uses.

`datestamp` Whether to datestamp responses (Y or N).

usecookie Whether to use cookies to enable answer persistence.

allowregister Whether to allow public registration (Y or N).

allowsave Whether to allow users to save their responses and returning later (Y or N).

autonumber_start Where to start autonumbering

autoredirect Whether to automatically redirect users to a URL (Y or N).

allowprev Whether to allow users to return to previous pages (Y or N).

printanswers Whether to allow printing of answer (Y or N).

ipaddr Whether to store IP addresses (Y or N).

refurl Whether to store the referring URL (Y or N).

showsurvey policynotice Whether to show the data policy notice (Y or N).

publicstatistics Whether to have public statistics (Y or N).

publicgraphs Whether to show graphs in public statistics (Y or N).

listpublic Whether to list the survey publicly (Y or N).

htmlemail Whether to use HTML format for token emails (Y or N).

sendconfirmation Whether to send confirmation emails (Y or N).

tokenanswers persistence Whether to use token-based response persistence (Y or N).

assessments Whether to use assessments (Y or N).

usecaptcha Whether to use CAPTCHA's (Y or N).

usetokens Whether to use tokens (Y or N).

bounce_email Where bouncing emails should be sent.

emailresponseto Where detailed admin notifications emails should be sent.

emailnotificationto Where a notification should be sent for new responses.

tokenlength The token length.

showxquestions Whether to show "There are X questions in this survey" (Y or N).

showgroupinfo Whether to show group name and info (B for both, ?, or X to show nothing).

shownoanswer Whether to show the "No answer" option (Y or N).

showqnumcode Whether to show answer codes or numbers (Y, N, or X to show nothing).

bounceprocessing Whether to process bouncing emails? (Y or N).

showwelcome Whether to show the welcome page (Y or N).

showprogress Whether to show the progress bar (Y or N).

questionindex Whether to show the question index (\emptyset to disable; can also be set to incremental or full (1 and 2?)).

navigationdelay The navigation delay in seconds

nokeyboard Whether to show the on-screen keyboard (Y or N).

alloweditaftercompletion Whether to allow multiple reponses (N) or to allow updating responses with one token (Y)?

googleanalyticsstyle The google analytics settings; \emptyset for None, other values for other settings.

googleanalyticsapikey The google analytics API key.

groups The groups in the survey.

tsvData Used to store the dataframe saved to a file as tab separated values.

Active bindings

`get_group_ids` A list of all group ids.

`get_group_titles` A list of all group ids.

Methods**Public methods:**

- `Survey$new()`
- `Survey$add_group()`
- `Survey$add_question()`
- `Survey$export_to_tsv()`
- `Survey$find_group_id()`
- `Survey$clone()`

Method `new()`: Create a new survey object.

Usage:

```
Survey$new(  
  titles,  
  descriptions = "",  
  welcomeTexts = "",  
  endTexts = "",  
  endURLs = "",  
  endURLdescriptions = "",  
  dateformats = 6,  
  numberformats = 0,  
  sid = 1,  
  gsid = 1,  
  admin = "Admin Name",  
  adminemail = "email@add.ress",  
  anonymized = "Y",  
  faxto = "",  
  format = "G",  
  savetimings = "Y",  
  template = "vanilla",  
  language = "en",  
  additional_languages = "",  
  datestamp = "Y",  
  usecookie = "N",  
  allowregister = "N",  
  allowsave = "N",  
  autonumber_start = 0,  
  autoredirect = "Y",  
  allowprev = "N",  
  printanswers = "N",  
  ipaddr = "N",  
  refurl = "N",  
  showsurveypolicynotice = "0",
```

```

publicstatistics = "N",
publicgraphs = "N",
listpublic = "N",
htmlmail = "Y",
sendconfirmation = "N",
tokenanswerspersistence = "N",
assessments = "N",
usecaptcha = "N",
usetokens = "N",
bounce_email = "",
emailresponseto = "",
emailnotificationto = "",
tokenlength = 15,
showxquestions = "N",
showgroupinfo = "X",
shownoanswer = "N",
showqnumcode = "X",
bounceprocessing = "N",
showwelcome = "N",
showprogress = "N",
questionindex = "0",
navigationdelay = "0",
nokeyboard = "N",
alloweditaftercompletion = "N",
googleanalyticsstyle = 0,
googleanalyticsapikey = "",
new_id_fun = NULL
)

```

Arguments:

titles The titles of the survey in the primary language and optionally any additional languages.

descriptions The descriptions of the survey in the primary language and any additional languages

welcomeTexts The welcome texts of the survey in the primary language and any additional languages

endTexts The end texts of the survey in the primary language and any additional languages

endURLs The end URLs of the survey in the primary language and any additional languages

endURLdescriptions The end URL descriptions of the survey in the primary language and any additional languages

dateformats The date formats to use in the primary language and any additional languages; the index of the option from the dropdown in LimeSurvey (6 is the ISO standard, "YYYY-MM-DD").

numberformats The number formats to use in the primary language and any additional languages (for periods as decimal marks, 0; for commas as decimal marks, 1).

sid The unique survey identifier; if this is free when importing the survey, this will be used.

gsid The Survey Group identifier.

admin The name of the survey administrator

adminemail The email address of the survey administrator

anonymized Whether the survey uses anonymized responses (Y or N).

faxto The contents of the "Fax to" field

format How to present the survey (Q for question by question; G for group by group; and A for all in one).

savetimings Whether to save timings of responses (Y or N).

template The name of the LimeSurvey theme to use.

language The primary language of the survey.

additional_languages Any additional languages the survey uses.

datestamp Whether to datestamp responses (Y or N).

usecookie Whether to use cookies to enable answer persistence.

allowregister Whether to allow public registration (Y or N).

allowsave Whether to allow users to save their responses and returning later (Y or N).

autonumber_start Where to start autonumbering

autoredirect Whether to automatically redirect users to a URL (Y or N).

allowprev Whether to allow users to return to previous pages (Y or N).

printanswers Whether to allow printing of answer (Y or N).

ipaddr Whether to store IP addresses (Y or N).

refurl Whether to store the referring URL (Y or N).

showsurvey policynotice Whether to show the data policy notice (Y or N).

publicstatistics Whether to have public statistics (Y or N).

publicgraphs Whether to show graphs in public statistics (Y or N).

listpublic Whether to list the survey publicly (Y or N).

htmlmail Whether to use HTML format for token emails (Y or N).

sendconfirmation Whether to send confirmation emails (Y or N).

tokenanswerspersistence Whether to use token-based response persistence (Y or N).

assessments Whether to use assessments (Y or N).

usecaptcha Whether to use CAPTCHA's (Y or N).

usetokens Whether to use tokens (Y or N).

bounce_email Where bouncing emails should be sent.

emailresponseto Where detailed admin notifications emails should be sent.

emailnotificationto Where a notification should be sent for new responses.

tokenlength The token length.

showxquestions Whether to show "There are X questions in this survey" (Y or N).

showgroupinfo Whether to show group name and info (Y, N, or X to show nothing).

shownoanswer Whether to show the "No answer" option (Y or N).

showqnumcode Whether to show answer codes or numbers (Y, N, or X to show nothing).

bounceprocessing Whether to process bouncing emails? (Y or N).

showwelcome Whether to show the welcome page (Y or N).

showprogress Whether to show the progress bar (Y or N).

questionindex Whether to show the question index (0 to disable; can also be set to incremental or full (1 and 2?)).

navigationdelay The navigation delay in seconds
 nokeyboard Whether to show the on-screen keyboard (Y or N).
 alloweditaftercompletion Whether to allow multiple responses (N) or to allow updating responses with one token (Y)?
 googleanalyticsstyle The google analytics settings; 0 for None, other values for other settings.
 googleanalyticsapikey The google analytics API key.
 new_id_fun A function to set identifiers (for XML exports, which mirrors MySQL tables and so needs identifiers). By default, new question objects receive this function from the group containing them; and groups receive it from the survey containing them. This ensures that identifiers are always unique in a survey (despite question objects not being able to 'see' anything in the group containing them, and group objects not being able to 'see' anything in the survey containing them; because they 'received' this function from the parent object, and it 'bubbles down' through groups to the questions, those functions still get and set a private identifier property in the 'top-most' object).

Returns: A new Survey object.

Method `add_group()`: Add a group to a survey object.

Usage:

```
Survey$add_group(titles, descriptions = "", relevance = 1, random_group = "")
```

Arguments:

titles The group's title, either as a named character vector where each element is the group title in a different language, and every element's name is the language code; or as a single character value, in which case the survey's primary language is used.

descriptions The group description, either as a named character vector where each element is the group description in a different language, and every element's name is the language code; or as a single character value, in which case the survey's primary language is used.

relevance The group's relevance equation.

random_group The group's randomization group.

Returns: Invisibly, the Survey object.

Method `add_question()`: Add a question to a survey object.

Usage:

```
Survey$add_question(groupId, code, type = NULL, lsType = NULL, ...)
```

Arguments:

groupId The id of the group to add the question to.

code The question code.

type The question type.

lsType The question type, as LimeSurvey question type.

... Additional arguments are used to create the Question using `Question$new`.

Returns: Invisibly, the Survey object.

Method `export_to_tsv()`: Export the survey as a tab separated values file (see https://manual.limesurvey.org/Tab_Separated)

Usage:

```
Survey$export_to_tsv(
  file,
  preventOverwriting = limonaid::opts$get("preventOverwriting"),
  parallel = TRUE,
  encoding = limonaid::opts$get("encoding"),
  silent = limonaid::opts$get("silent"),
  backupLanguage = self$language
)
```

Arguments:

`file` The filename to which to save the file.
`preventOverwriting` Whether to prevent overwriting.
`parallel` Whether to work serially or in parallel.
`encoding` The encoding to use
`silent` Whether to be silent or chatty.
`backupLanguage` The language to get content from if not from the primary language.

Returns: Invisibly, the Survey object.

Method `find_group_id()`: Find the numeric group identifier by group title.

Usage:

```
Survey$find_group_id(title, titleLanguage = NULL)
```

Arguments:

`title` The survey title.
`titleLanguage` The language in which to search.

Returns: Invisibly, the Survey object.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Survey$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

 transpose_df

Transpose a data frame

Description

Returns a list of lists, where each list contains a row.

Usage

```
transpose_df(x)
```

Arguments

x The data frame.

Value

A list.

Examples

```
limonaid::transpose_df(
  mtcars[1:3, 1:3]
);
```

 vecTxt

Easily parse a vector into a character value

Description

Easily parse a vector into a character value

Usage

```
vecTxt(
  vector,
  delimiter = ", ",
  useQuote = "",
  firstDelimiter = NULL,
  lastDelimiter = " & ",
  firstElements = 0,
  lastElements = 1,
  lastHasPrecedence = TRUE
)

vecTxtQ(vector, useQuote = "'", ...)
```

Arguments

vector The vector to process.

delimiter, firstDelimiter, lastDelimiter
 The delimiters to use for respectively the middle, first firstElements, and last lastElements elements.

useQuote This character string is pre- and appended to all elements; so use this to quote all elements (useQuote="'"), doublequote all elements (useQuote='"), or anything else (e.g. useQuote='|'). The only difference between vecTxt and vecTxtQ is that the latter by default quotes the elements.

firstElements, lastElements
 The number of elements for which to use the first respective last delimiters

lastHasPrecedence

If the vector is very short, it's possible that the sum of firstElements and lastElements is larger than the vector length. In that case, downwardly adjust the number of elements to separate with the first delimiter (TRUE) or the number of elements to separate with the last delimiter (FALSE)?

... Any addition arguments to vecTxtQ are passed on to vecTxt.

Value

A character vector of length 1.

Examples

```
vecTxtQ(names(mtcars));
```

Index

- * **datasets**
 - opts, 32
- * **package**
 - limonaid-package, 3
- add_answer_option_to_question, 3
- append_lsdf_rows, 4
- append_lsdf_rows(), 7

- cat, 5
- cat0, 5
- checkPkgs, 5
- convertToNumeric, 6

- emptyDf, 7
- export_with_languages, 7

- get (opts), 32
- get_session_key, 8
- Group, 9

- library(), 5
- limer_base64_to_df, 13
- limer_base64_to_df(), 16
- limer_call_limer, 13, 16
- limer_call_limer(), 16
- limer_get_participant_property, 15
- limer_get_participants, 14
- limer_get_responses, 15
- limer_release_session_key, 16
- limer_upload_tsv_to_limesurvey, 17
- limonaid (limonaid-package), 3
- limonaid-package, 3
- ls_apply_script_bits, 19
- ls_apply_script_bits(), 24
- ls_eq_brace (ls_eq_build), 20
- ls_eq_build, 20
- ls_eq_if (ls_eq_build), 20
- ls_eq_ifRegex (ls_eq_build), 20
- ls_eq_is (ls_eq_build), 20
- ls_eq_isChecked (ls_eq_build), 20
- ls_eq_isUnchecked (ls_eq_build), 20
- ls_eq_nestIfs, 21
- ls_eq_quote (ls_eq_build), 20
- ls_import_data, 22
- ls_import_data(), 25
- ls_parse_data_import_script, 24
- ls_parse_data_import_script(), 19
- ls_process_labels, 25
- ls_read_tsv, 26
- ls_recodeTable_to_equations, 27
- ls_tsv_get_group_rows, 28
- ls_tsv_get_rows, 28
- ls_tsv_rows, 29
- ls_tsv_rows(), 28
- ls_write_lsg, 29
- ls_write_tsv, 30
- lsdf_for_language, 18

- mail_registered_participant, 31
- massConvertToNumeric, 19, 23, 31

- opts, 32

- POST, 14
- processLimeSurveyDropouts, 33

- Question, 34

- repeatStr, 41
- repStr (repeatStr), 41
- require(), 5
- reset (opts), 32

- set (opts), 32
- Survey, 42

- textConnection(), 13
- transpose_df, 48

- vecTxt, 49
- vecTxtQ (vecTxt), 49