Package 'preregr'

August 22, 2022

Title Specify (Pre)Registrations and Export Them Human- And Machine-Readably

Version 0.2.5

Description Preregistrations, or more generally, registrations, enable explicit timestamped and (often but not necessarily publicly) frozen documentation of plans and expectations as well as decisions and justifications. In research, preregistrations are commonly used to clearly document plans and facilitate justifications of deviations from those plans, as well as decreasing the effects of publication bias by enabling identification of research that was conducted but not published. Like reporting guidelines, (pre)registration forms often have specific structures that facilitate systematic reporting of important items. The 'preregr' package facilitates specifying (pre)registrations in R and exporting them to a human-readable format (using R Markdown partials or exporting to an 'HTML' file) as well as human-readable embedded data (using 'JSON'), as well as importing such exported (pre)registration specifications from such embedded 'JSON'.

URL https://r-packages.gitlab.io/preregr

BugReports https://gitlab.com/r-packages/preregr/-/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Imports cli (>= 3.0), jsonlite (>= 1.7), rmdpartials (>= 0.5.8), yaml (>= 2.2)

Suggests googlesheets4 (>= 1.0), haven (>= 2.4.3), justifier (>= 0.2.2), knitr (>= 1.34), openxlsx (>= 4.2), markdown, readxl (>= 1.3), rvest (>= 1.0), testthat (>= 3.0), writexl (>= 1.4), XLConnect (>= 1.0), rmarkdown

Config/testthat/edition 3

Depends R (>= 4.1)

VignetteBuilder knitr

NeedsCompilation no

Author Gjalt-Jorn Peters [aut, cre] (<https://orcid.org/0000-0002-0336-9589>),
Szilvia Zörgő [ctb] (<https://orcid.org/0000-0002-6916-2097>),
Olmo den Akker [ctb] (<https://orcid.org/0000-0002-0712-3746>),
Aleksandra Lazić [ctb] (<https://orcid.org/0000-0002-0433-0483>)

Maintainer Gjalt-Jorn Peters <gjalt-jorn@behaviorchange.eu>

Repository CRAN

Date/Publication 2022-08-22 07:50:10 UTC

R topics documented:

`bipiped_value_to_vector` *Convert a "bipiped" value (or vector of values) to a vector*

Description

"Bipiped" means that different values are separated by a pair of pipes (`||`), like the logical OR operator in R. Use `bipiped_value_to_vector()` for single values, and `bipiped_values_to_vector()` for a vector of values, in which case a list is returned.

Usage

```
bipiped_value_to_vector(x)

bipiped_values_to_vector(x)
```

Arguments

`x`	The value or vector of values.

Value

A vector or list of vectors.

Examples

```
exampleValue <-
  paste0('"Purposefully select" || "Aselect" || ',
         '"Likely self-selected" || "Ameliorated self-selection"');
bipiped_value_to_vector(exampleValue);
```

cat0 *Concatenate to screen without spaces*

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 cat.
sep	The separator to pass to cat, of course, "" by default.

Value

Nothing (invisible NULL, like cat).

Examples

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

examplePrereg_1 *An example (pre)registration specification using the Inclusive General-Purpose Registration Form*

Description

This is a simple and relatively short partially completed (pre)registration specification.

Usage

```
examplePrereg_1
```

Format

An example of a (pre)registration specification

first_valid_value *Select the first valid value*

Description

From a vector or list of values, select the first valid value, valid being defined as a value that is not NULL or NA and has, after being trimmed of whitespace, a nonzero length. Optionally, only look at the element with a given name.

Usage

```
first_valid_value(x, selectName = NULL)
```

Arguments

x	The vector or list.
selectName	Optionally, the name to look at.

Value

The first valid value (or NULL).

forms *Included (pre)registration forms*

Description

Show an overview of all included (pre)registration forms.

Usage

```
forms()
```

Value

Invisibly, a list of form identifiers of the available forms.

Examples

```
forms();
```

`form_add_instruction` *Add an instruction, section, or item to a (pre)registration form*

Description

Add an instruction, section, or item to a (pre)registration form

Usage

```
form_add_instruction(x, heading, description, overwrite = TRUE)

form_add_section(x, id, label, description, overwrite = TRUE)

form_add_item(
  x,
  id,
  label,
  description,
  section_id,
  valueTemplate = "string",
  validValues = NA,
  validation = NA,
  overwrite = TRUE
)
```

Arguments

`x`	The (pre)registration form as created by `form_create()`.
`heading`	The instruction's heading
`description`	The description of the instruction, section, or item
`overwrite`	Whether to overwrite existing content or append the new content
`id`	The identifier of the section or item
`label`	The label (i.e. title) of the section or item
`section_id`	The section identifier of the section the item should be placed in
`valueTemplate`	The name of the value template of the item
`validValues`	The valid values (for categorical items)
`validation`	The validation statement (an R expression)

Value

The modified (pre)registration form

Examples

```
### Create an empty example form
exampleForm <-
  preregr::form_create(
    title = "Example form",
    version = "0.1.0"
  ) |>
  preregr::form_show();

### Add some stuff;
exampleForm <-
  exampleForm |>
  preregr::form_add_instruction(
    heading = "First Real Instruction",
    description = "Which normally also contains real instructions here"
  ) |>
  preregr::form_add_section(
    id = "first_section",
    label = "First Real Section",
    description = "This section is very, very important."
  ) |>
  preregr::form_add_section(
    id = "second_section",
    label = "Second Real Section",
    description = "This section is even more important then the first one."
  ) |>
  preregr::form_add_item(
    id = "study_title",
    label = "Study Title",
    section_id = "first_section",
    description = paste0(
      "Think of a catching title, preferably with a colon in ",
      "the middle. Bonus points for pop culture references."
    )
  ) |>
  preregr::form_add_item(
    id = "study_authors",
    label = "Authors",
    section_id = "first_section",
    description = "Maybe list the authors, too."
  ) |>
  preregr::form_add_item(
    id = "registration_type",
    label = "Registration type",
    section_id = "second_section",
    description = paste0(
      "Describe briefly why you are (pre)registering this ",
      "study. For example, this might be a preregistration ",
      "to allow others to know you're doing this study; or to ",
      "make it clear you value transparency in science; or to ",
      "remember your original plans later on. Or this might be ",
      "a registration to update your plans after the data came ",
```

```
      "in; or to document pragmatic changes in plans."
    )
  );

### Show the result of our hard labour
preregr::form_show(exampleForm);
```

`form_almostEmptyForm` *A mostly empty example form specification*

Description

This form specification is mostly empty, so it can be a useful start if you want to create your own form. The accompanying Google Sheet, which you can also copy, is https://docs.google.com/spreadsheets/d/14Qbak7JbBhTqmJaMgJ4tU9ZROaBbUfq37_UzkoHnM60

Usage

```
form_almostEmptyForm
```

Format

A (pre)registration specification

`form_create` *Create a new (pre)registration form*

Description

You can use this function to create a new (pre)registration form. The "Creating a (pre)registration form" vignette explains how this works. That is available at https://r-packages.gitlab.io/preregr/articles/creating_prereg_form.html or can be shown by running `vignette("creating_prereg_form",pac = "preregr")`.

Usage

```
form_create(
  title,
  version,
  author = NA,
  date = format(Sys.Date(), "%Y-%m-%d"),
  ...
)
```

Arguments

`title`	The form's title
`version`	The form's version. If there is only one version and the creators do not plan to release future version, the recommendation is to set the version to `1`. If the creators have no clear idea about how to version the form (i.e. there may be improvements in the future), the recommendation is to set the first version to `0.0.1`, and roughly adopt the system common for software: increment the first number when the form is substantially updated (e.g. such that preregistrations that used the previous version of the form may no longer be valid given the new version, for example becaue sections were added or removed, or value templates changed, etc), the third number for very small changes (e.g. typos, spelling corrections, clarification or extra explanations, bug fixes in regular expressions in value templates, etc), and the second number for changes in between (e.g. changing the order of items or moving an item to another section, or changing value templates to be more permissive (and so, retaining compatibility with (pre)registrations that used the previous version of the form)). In that case, use version `1.0.0` to signal that the form has reached maturity.
`author`	The authors of the form
`date`	The date the form was created
`...`	Additional field-content pairs to specify arbitrary metadata.

Value

The `preregr` form object prefilled with some examples.

Examples

```
exampleForm <-
  preregr::form_create(
    title = "Example form",
    version = "0.1.0"
  );

### Show the form summary
exampleForm;
```

`form_fromSpreadsheet` *Import a (pre)registration form specification from a spreadsheet*

Description

With this function, you can import a (pre) registration from a spreadsheet. See the "Creating a form from a spreadsheet" vignette for more information. That is available at https://r-packages.gitlab.io/preregr/articles/creating_form_from_spreadsheet.html or can be shown by running `vignette("creating_form_from_spreadsheet",package = "preregr")`

Usage

```
form_fromSpreadsheet(
  x,
  localBackup = NULL,
  exportGoogleSheet = TRUE,
  xlsxPkg = c("rw_xl", "openxlsx", "XLConnect"),
  silent = preregr::opts$get("silent")
)
```

Arguments

x	The URL or path to a file.
localBackup	If not NULL, a valid filename to write a local backup to.
exportGoogleSheet	If x is a URL to a Google Sheet, instead of using thr googlesheets4 package to download the data, by passing exportGoogleSheet=TRUE, an export link will be produced and the data will be downloaded as Excel spreadsheet.
xlsxPkg	Which package to use to work with Excel spreadsheets.
silent	Whether to be silent or chatty.

Details

An empty simple example spreadsheet is available at https://docs.google.com/spreadsheets/d/14Qbak7JbBhTqmJaMgJ4tU9ZROaBbUfq37_UzkoHnM60 and can be initialized as the almostEmptyForm form with prereg_initialize()

Value

The preregr form specification

---

form_generalPurpose_v1_1

Inclusive General-Purpose Registration Form

Description

This Inclusive General-Purpose Registration Form is designed to be applicable across disciplines (i.e., psychology, economics, law, physics, or any other field) and across study types (i.e., qualitative studies, quantitative studies, experiments, systematic reviews, case studies, archive studies, comparative legal studies, or any other type of study). This form, therefore, is a fall-back for more specialized forms and can be used if no specialized form or registration platform is available. If at all possible, it is recommended to use a specialized form, since this inclusive general-purpose registration form achieves that inclusiveness and general-purposeness at the cost of specificity and comprehensiveness. Still, if specialized forms don't fit for your study, this form may be a good backup.

Usage

```
form_generalPurpose_v1_1
```

Format

A (pre)registration form specification

---

form_inclSysRev_v0_92 *Inclusive Systematic Review Registration Form*

Description

This Systematic Review Registration Form is intended as a general-purpose registration form. The form is designed to be applicable to reviews across disciplines (i.e., psychology, economics, law, physics, or any other field) and across review types (i.e., scoping review, review of qualitative studies, meta-analysis, or any other type of review). That means that the reviewed records may include research reports as well as archive documents, case law, books, poems, etc. This form, therefore, is a fall-back for more specialized forms and can be used if no specialized form or registration platform is available.

Usage

```
form_inclSysRev_v0_92
```

Format

A (pre)registration form specification

Source

doi:10.31222/osf.io/3nbea

---

form_knit *Knit a (pre)registration form into an Rmd file*

Description

This function inserts a (pre)registration form, or one or more sections, into an R Markdown file.

Usage

```
form_knit(x, section = NULL, headingLevel = 2)
```

Arguments

x	The (pre)registration form (as produced by a call to form_create()) or initialized preregr object (as produced by a call to prereg_initialize()).
section	The section(s) to show; pass NULL (the default) to show everything.
headingLevel	The level to use for the top-most heading.

Value

x, invisibly

Examples

```
preregr::form_create(
  title = "Example form",
  version = "0.1.0"
) |>
  preregr::form_knit();
```

---

form_OSFprereg_v1 *OSF Prereg form*

Description

Preregistration is the act of submitting a study plan, ideally also with analytical plan, to a registry prior to conducting the work. Preregistration increases the discoverability of research even if it does not get published further. Adding specific analysis plans can clarify the distinction between planned, confirmatory tests and unplanned, exploratory research.

Usage

```
form_OSFprereg_v1
```

Format

A (pre)registration form specification

Details

This preprint contains a template for the "OSF Prereg" form available from the OSF Registry. An earlier version was originally developed for the Preregistration Challenge, an education campaign designed to initiate preregistration as a habit prior to data collection in basic research, funded by the Laura and John Arnold Foundation (now Arnold Ventures) and conducted by the Center for Open Science. More information is available at https://www.cos.io/initiatives/prereg/, and other templates are available at: https://osf.io/zab38/"

Source

doi:10.31222/osf.io/epgjd

form_OSFqual1_v1 *Qualitative Preregistration Template*

Description

The Qualitative Preregistration Template was created by researchers from the qualitative community for registration of primarily qualitative work. It is available as a registration option on the Open Science Framework (OSF).

Usage

```
form_OSFqual1_v1
```

Format

A (pre)registration specification

Details

This form was added to `preregr` by Aleksandra Lazic.

form_prereg2D_v1 *Preregistration Template for Secondary Data Analysis*

Description

Please cite the associated paper when using this preregistration template (see https://doi.org/10.15626/MP.2020.2625).

Usage

```
form_prereg2D_v1
```

Format

A (pre)registration specification

`form_preregQE_v0_94` *Preregistration Template for Qualitative and Quantitative Ethnographic Studies*

Description

A preregistration is a way to design your research project before you begin and to document your decisions, rationale. A template such as this one can be employed to think about what you want to do and how, and subsequently, if you wish, you can submit the finished preregistration to a registry, such as OSF's (https://osf.io/registries/). This template was developed to aid the preregistration of quantitative ethnographic studies, but due to its modular nature, it can be employed for qualitative studies as well.

Usage

```
form_preregQE_v0_94
```

Format

A (pre)registration form specification

Source

doi:10.23668/psycharchives.4584

`form_prpQuant_v1` *Psychological Research Preregistration-Quantitative (aka PRP-QUANT) Template*

Description

As an international effort toward increasing psychology's commitment to creating a stronger culture and practice of preregistration, a multi-society Preregistration Task Force* was formed, following the 2018 meeting of the German Psychological Society in Frankfurt, Germany. The Task Force created a detailed preregistration template that benefited from the APA JARS Quantitative Research guidelines, as well as a comprehensive review of many other preregistration templates.

Usage

```
form_prpQuant_v1
```

Format

A (pre)registration form specification

Source

doi:10.23668/psycharchives.4584

form_show *Show a (pre)registration form*

Description

This function shows (parts of) a (pre)registration form.

Usage

```
form_show(x, section = NULL)
```

Arguments

x	The (pre)registration form (as produced by a call to form_create()).
section	The section(s) to show; pass NULL (the default) to show everything.

Value

x, invisibly

Examples

```
### An empty form
preregr::form_create(
  "Example form",
  version = "1"
) |>
  preregr::form_show();

### A complete form
preregr::prereg_initialize("inclSysRev_v0_92") |>
  preregr::form_show();
```

form_to_html *Convert a (pre)registration form to html*

Description

Convert a (pre)registration form to html

Usage

```
form_to_html(
  x,
  file = NULL,
  section = NULL,
  headingLevel = 1,
  silent = preregr::opts$get("silent")
)
```

Arguments

x	The (pre)registration form (as produced by a call to form_create()) or initialized preregr object (as produced by a call to prereg_initialize()).
file	Optionally, a file to save the html to.
section	Optionally, one or multiple sections to include (if NULL, all sections are included).
headingLevel	The level of the top-most headings.
silent	Whether to be silent or chatty.

Value

x, invisibly

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Extract the form and show it as HTML
preregr::form_to_html(
  examplePrereg_1
);
```

---

form_to_json *Convert a (pre)registration specification to YAML or JSON*

Description

Convert a (pre)registration specification to YAML or JSON

Usage

```
form_to_json(x, file = NULL)

prereg_spec_to_json(x, includeFormSpec = TRUE, file = NULL)

## S3 method for class 'preregr_json'
print(x, ...)

prereg_spec_to_yaml(x, includeFormSpec = TRUE, file = NULL)

## S3 method for class 'preregr_yaml'
print(x, ...)
```

Arguments

x	The (pre)registration object (as produced by a call to prereg_initialize()), or, for the print method, the produced YAML or JSON.
file	Optionally, a file to save the YAML or JSON to.
includeFormSpec	Whether to include the (pre)registration form specification. Note that this includes metadata about the form fields such as their labels and descriptions - without the form specification, only the item identifiers are stored.
...	Any additional arguments are ignored.

Value

If a file is specified to write, to, x will be returned invisibly to allow building a pipe chain; if file=NULL, the resulting YAML/JSON will be returned as a character vector.

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Export to YAML
preregr::prereg_spec_to_json(
  examplePrereg_1
);
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Export to YAML
preregr::prereg_spec_to_yaml(
  examplePrereg_1
);
```

---

form_to_rmd_template *Convert a (pre)registration form to an R Markdown template*

Description

This function creates an R Markdown template from a preregr (pre)registrations form specification. Pass it the URL to a Google Sheet holding the (pre)registration form specification (in preregr format), see the "Creating a form from a spreadsheet" vignette), the path to a file with a spreadsheet holding such a specification, or a loaded or imported preregr (pre)registration form.

Usage

```
form_to_rmd_template(
  x,
  file = NULL,
  title = NULL,
  author = NULL,
  date = "`r format(Sys.time(), '%Y-%m-%d at %H:%M:%S %Z (UTC%z)')`",
  output = "html_document",
  yaml = list(title = title, author = author, date = date, output = output),
  includeYAML = TRUE,
  chunkOpts = "echo=FALSE, results='hide'",
  justify = FALSE,
  headingLevel = 2,
  showSpecification = FALSE,
  preventOverwriting = preregr::opts$get("preventOverwriting"),
  silent = preregr::opts$get("silent")
)

## S3 method for class 'preregr_rmd_template'
print(x, ...)
```

Arguments

x	The (pre)registration form (as produced by a call to form_create() or import_from_html()) or initialized preregr object (as produced by a call to prereg_initialize() or [preregr::import_from_html()); or, for the printing method, the R Markdown template produced by a call to form_to_rmd_template().
file	Optionally, a file to save the html to.
title	The title to specify in the template's YAML front matter.
author	The author to specify in the template's YAML front matter.
date	The date to specify in the template's YAML front matter.
output	The output format to specify in the template's YAML front matter.
yaml	It is also possible to specify the YAML front matter directly using this argument. If used, it overrides anything specified in title, author, date and output.
includeYAML	Whether to include the YAML front matter or omit it.
chunkOpts	The chunk options to set for the chunks in the template.
justify	Whether to use prereg_specify() as function for specifying the (pre)registration content (if FALSE), or prereg_justify() (if TRUE).
headingLevel	The level of the top-most heading to use (the title of the (pre)registration form).
showSpecification	Whether to show the specification in the Rmd output. When FALSE, the preregr option silent is set to TRUE at the start of the Rmd template; otherwise, it is set to FALSE.
preventOverwriting	Set to FALSE to override overwrite prevention.

silent	Whether to be silent or chatty.
...	Additional argument are ignored.

Value

x, invisibly

Examples

```
preregr::form_create(
  title = "Example form",
  version = "0.1.0"
) |>
  preregr::form_to_rmd_template();
```

---

form_to_xlsx *Export a (pre)registration form to an Excel spreadsheet*

Description

Export a (pre)registration form to an Excel spreadsheet

Usage

```
form_to_xlsx(x, file)
```

Arguments

x	The (pre)registration form (as produced by a call to form_create()) or initialized preregr object (as produced by a call to prereg_initialize()).
file	The file to write the spreadsheet to.

Value

x, invisibly

heading *Print a heading*

Description

This is just a convenience function to print a markdown or HTML heading at a given 'depth'.

Usage

```
heading(
  ...,
  headingLevel = ufs::opts$get("defaultHeadingLevel"),
  output = "markdown",
  cat = TRUE
)
```

Arguments

...	The heading text: pasted together with no separator.
headingLevel	The level of the heading; the default can be set with e.g. `ufs::opts$set(defaultHeadingLevel=1)`.
output	Whether to output to HTML ("html") or markdown (anything else).
cat	Whether to cat (print) the heading or just invisibly return it.

Value

The heading, invisibly.

Examples

```
heading("Hello ", "World", headingLevel=5);
### This produces: "\n\n##### Hello World\n\n"
```

import_from_html *Import a (pre)registration specification from JSON embedded in HTML*

Description

Import a (pre)registration specification from JSON embedded in HTML

Usage

```
import_from_html(x, select = 1)
```

Arguments

x	The HTML as URL, path to a file, or HTML that has already been imported to a character vector.
select	If multiple preregr specifications are present in x, the select argument can be used to select which one to import. select is a number corresponding to the order of the encountered preregr specifications (e.g., pass 2 to import the second specification, etc).

Value

The (pre)registration specification.

Examples

```
### Note that this example writes to a local file!

### Temporary file to save to
tmpFile <- tempfile(fileext = ".html");

### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Save it to an HTML file
preregr::prereg_spec_to_html(
  examplePrereg_1,
  file = tmpFile
);

### Import the example again
importedPreregr <-
  preregr::import_from_html(
    tmpFile
  );

### Show the result
preregr::prereg_show_item_completion(
  importedPreregr,
  section="metadata"
);
```

---

number_as_xl_date *Convert a number to a date using Excel's system*

Description

Convert a number to a date using Excel's system

Usage

```
number_as_xl_date(x)
```

Arguments

x	The number(s)

Value

The date(s)

Examples

```
preregr::number_as_xl_date(44113);
```

---

opts *Options for the preregr package*

Description

The `preregr::opts` object contains three functions to set, get, and reset options used by the preregr package. Use `preregr::opts$set` to set options, `preregr::opts$get` to get options, or `preregr::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 `preregr` options.

The following arguments can be passed:

... For `preregr::opts$set`, the dots can be used to specify the options to set, in the format `option = value`, for example, `utteranceMarker = "\n"`. For `preregr::opts$reset`, a list of options to be reset can be passed.

option For `preregr::opts$set`, the name of the option to set.

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

The following options can be set:

quridPrefix The prefix for quasi-unique record identifiers (QURIDs).

Two Second item

Examples

```
### Get the default "silence versus chattiness" setting
preregr::opts$get(silent);

### Set it to show all messages
preregr::opts$set(silent = FALSE);

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

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

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

---

prereg_initialize *Initialize a (pre)registration*

Description

To initialize a (pre)registration, pass the URL to a Google Sheet holding the (pre)registration form specification (in preregr format), see the "Creating a form from a spreadsheet" vignette), the path to a file with a spreadsheet holding such a specification, or a loaded or imported preregr (pre)registration form.

Usage

```
prereg_initialize(x, initialText = "Unspecified")
```

Arguments

x	The (pre)registration form specification, as a URL to a Google Sheet or online file or as the path to a locally stored file.
initialText	The text to initialize every field with.

Details

For an introduction to working with preregr (pre)registrations, see the "Specifying preregistration content" vignette.

Value

The empty (pre)registration specification.

Examples

```
preregr::prereg_initialize(
  "inclSysRev_v0_92"
);
```

`prereg_justify` *Justify (and optionally specify) the content for one or more (pre)registration items*

Description

Justify (and optionally specify) the content for one or more (pre)registration items

Usage

```
prereg_justify(
  x,
  item,
  decision = NULL,
  justification = NULL,
  assertion = NULL,
  source = NULL,
  content = NULL,
  append = TRUE,
  validate = TRUE,
  requireValidContent = TRUE,
  silent = preregr::opts$get("silent")
)
```

Arguments

`x`	The (pre)registration object (as produced by a call to `prereg_initialize()`).
`item`	The identifier of the item for which to specify the justification of the (pre)registration content.
`decision, justification, assertion, source`	The decision(s) (with optionally nested within it, one or more justifications), justification(s) (with optionally nested within it, one or more assertions), assertion(s) (with optionally nested within it, one or more sources), or source(s).
`content`	Optionally, content to specify or append for this item.
`append`	Whether to replace (`append=FALSE`) or append (`append=TRUE`) the content if an item already contains some content.
`validate`	Whether to validate the specified content for each item using the validation rules in the (pre)registration form.
`requireValidContent`	Whether to only store new content if it passes validation. Note that this is ignored if `validate=FALSE`.
`silent`	Whether to be silent or chatty.

Value

x, invisibly

Examples

```
### Start with an empty form, then specify and justify
### content for an item.
preregExampl <-
  preregr::prereg_initialize(
    "inclSysRev_v0_92"
  ) |>
    preregr::prereg_justify(
      item = "title",
      content = "Example title",
      decision = "We decide to call this study 'Example title'.",
      justification = "It seems a fitting title for an example."
    ) |>
    preregr::prereg_show_item_completion(
      section="metadata"
    );
```

---

prereg_knit_item_content

Knit the specified content for the items in a (pre)registration into an Rmd file

---

Description

This function inserts the specified content for the items in a (pre)registration, or in one or more sections, into an R Markdown file.

Usage

```
prereg_knit_item_content(x, section = NULL, headingLevel = 2)
```

Arguments

x	The (pre)registration object (as produced by a call to prereg_initialize()).
section	The section(s) to show; pass NULL (the default) to show everything.
headingLevel	The level to use for the top-most heading.

Value

x, invisibly

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Knit the contents of the "metadata" section
### as an R Markdown partial
examplePrereg_1 |>
  preregr::prereg_knit_item_content(
    section="metadata"
  );
```

`prereg_next_item` *Show the next item to specify content for*

Description

This function shows the next item (or items) in a (pre)registration for which to specify content (searching through all sections or through a selection of sections).

Usage

```
prereg_next_item(x, nrOfItems = 1, section = NULL)
```

Arguments

`x`	The (pre)registration object (as produced by a call to `prereg_initialize()`).
`nrOfItems`	The number of items to complete to show.
`section`	The section(s) to search; pass `NULL` (the default) to show everything.

Value

x, invisibly

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Check next item
examplePrereg_1 |>
  preregr::prereg_next_item();

### Specify content for this item
examplePrereg_1 <-
  preregr::prereg_specify(
    examplePrereg_1,
    funding = paste0(
      "No funding. There's never any ",
```

```
      "funding for this kind of stuff."
    )
  );

### Get the next three items
preregr::prereg_next_item(
  examplePrereg_1,
  nrOfItems = 3
);
```

`prereg_show_item_completion` *Show which items in a (pre)registration have been completed*

Description

This function shows which items in a (pre)registration, or in one or more sections, have been completed - or, more accurately, contain at least some content that is different from the default content.

Usage

```
prereg_show_item_completion(x, section = NULL)
```

Arguments

`x`	The (pre)registration object (as produced by a call to `prereg_initialize()`).
`section`	The section(s) to show; pass `NULL` (the default) to show everything.

Value

x, invisibly

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Show which items were completed
examplePrereg_1 |>
  preregr::prereg_show_item_completion(
    section="metadata"
  );
```

prereg_show_item_content *Show the specified content for the items in a (pre)registration*

Description

This function shows the specified content for the items in a (pre)registration, or in one or more sections.

Usage

```
prereg_show_item_content(x, section = NULL)
```

Arguments

x	The (pre)registration object (as produced by a call to prereg_initialize()).
section	The section(s) to show; pass NULL (the default) to show everything.

Value

x, invisibly

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Show the item content for the "metadata" section
examplePrereg_1 |>
  preregr::prereg_show_item_content(
    section="metadata"
  );
```

prereg_specify *Specify the content for one or more (pre)registration items*

Description

Specify the content for one or more (pre)registration items

Usage

```
prereg_specify(
  x,
  ...,
  append = TRUE,
  validate = TRUE,
  requireValidContent = TRUE,
  silent = preregr::opts$get("silent")
)
```

Arguments

x	The (pre)registration object (as produced by a call to prereg_initialize()).
...	Item-content pairings.
append	Whether to replace (append=FALSE) or append (append=TRUE) the content if an item already contains some content.
validate	Whether to validate the specified content for each item using the validation rules in the (pre)registration form.
requireValidContent	Whether to only store new content if it passes validation. Note that this is ignored if validate=FALSE.
silent	Whether to be silent or chatty.

Value

x, invisibly

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Specify some fields and show the results
examplePrereg_1 |>
  preregr::prereg_specify(
    tasks_and_roles = "All authors contributed equally",
    nonExistent_item = "This can't be stored anywhere",
    start_date = "2021-9-01"
  ) |>
  preregr::prereg_show_item_completion(
    section="metadata"
  );
```

prereg_spec_to_html *Convert a (pre)registration specification to html*

Description

Use this function to export your (pre)registration specification to an HTML file. To instead embed it in an R Markdown file, use prereg_knit_item_content().

Usage

```
prereg_spec_to_html(
  x,
  file = NULL,
  section = NULL,
  headingLevel = 1,
  silent = preregr::opts$get("silent")
)
```

Arguments

x	The (pre)registration object (as produced by a call to prereg_initialize()).
file	Optionally, a file to save the html to.
section	Optionally, one or multiple sections to include (if NULL, all sections are included).
headingLevel	The level of the top-most headings.
silent	Whether to be silent or chatty.

Value

The produced HTML, which will print in the viewer in RStudio.

Examples

```
### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

### Convert it to HTML and show the result
preregr::prereg_spec_to_html(
  examplePrereg_1
);
```

prereg_spec_to_pdf *Convert a (pre)registration specification to PDF*

Description

Use this function to export your (pre)registration specification to a PDF file. To embed it in an R Markdown file, use `prereg_knit_item_content()` instead.

Usage

```
prereg_spec_to_pdf(
  x,
  file,
  author = NULL,
  section = NULL,
  headingLevel = 1,
  silent = preregr::opts$get("silent")
)
```

Arguments

x	The (pre)registration object (as produced by a call to `prereg_initialize()`).
file	The filename to save the (pre)registration to.
author	The author to specify in the PDF.
section	Optionally, one or multiple sections to include (if `NULL`, all sections are included).
headingLevel	The level of the top-most headings.
silent	Whether to be silent or chatty.

Value

x, invisibly

Examples

```
### Use a temporary file to write to
tmpFile <- tempfile(fileext = ".pdf");

### Load an example (pre)registration specification
data("examplePrereg_1", package = "preregr");

preregr::prereg_spec_to_pdf(
  examplePrereg_1,
  file = tmpFile
);
```

randomSlug *Generate a random slug*

Description

idSlug is a convenience function with swapped argument order.

Usage

```
randomSlug(x = 10, id = NULL, chars = c(letters, LETTERS, 0:9))

idSlug(id = NULL, x = 10, chars = c(letters, LETTERS, 0:9))
```

Arguments

x	Length of slug
id	If not NULL, prepended to slug (separated with a dash) as id; in that case, it's also braces and a hash is added.
chars	Characters to sample from

Value

A character value.

Examples

```
randomSlug();
idSlug("identifier");
```

rbind_dfs *Simple alternative for rbind.fill or bind_rows*

Description

Simple alternative for rbind.fill or bind_rows

Usage

```
rbind_dfs(x, y, clearRowNames = TRUE)
```

Arguments

x	One dataframe
y	Another dataframe
clearRowNames	Whether to clear row names (to avoid duplication)

Value

The merged dataframe

Examples

```
rbind_dfs(Orange, mtcars);
```

---

rbind_df_list *Bind lots of dataframes together rowwise*

Description

Bind lots of dataframes together rowwise

Usage

```
rbind_df_list(x)
```

Arguments

x	A list of dataframes

Value

A dataframe

Examples

```
rbind_df_list(list(Orange, mtcars, ChickWeight));
```

---

read_spreadsheet *Convenience function to read spreadsheet-like files*

Description

Currently reads spreadsheets from Google Sheets or from `xlsx`, `csv`, or `sav` files. Normally, you don't use this, but instead you use `form_fromSpreadsheet()`.

Usage

```
read_spreadsheet(
  x,
  sheet = NULL,
  columnDictionary = NULL,
  localBackup = NULL,
  exportGoogleSheet = FALSE,
  flattenSingleDf = FALSE,
  xlsxPkg = c("rw_xl", "openxlsx", "XLConnect"),
  failQuietly = FALSE,
  silent = preregr::opts$get("silent")
)
```

Arguments

x	The URL or path to a file.
sheet	Optionally, the name(s) of the worksheet(s) to select.
columnDictionary	Optionally, a dictionary with column names to check for presence. A named list of vectors.
localBackup	If not NULL, a valid filename to write a local backup to.
exportGoogleSheet	If x is a URL to a Google Sheet, instead of using thr googlesheets4 package to download the data, by passing exportGoogleSheet=TRUE, an export link will be produced and the data will be downloaded as Excel spreadsheet.
flattenSingleDf	Whether to return the result as a data frame if only one data frame is returned as a result.
xlsxPkg	Which package to use to work with Excel spreadsheets.
failQuietly	Whether to give an error when x is not a valid URL or existing file, or just return NULL invisibly.
silent	Whether to be silent or chatty.

Value

A list of dataframes, or, if only one data frame was loaded and flattenSingleDf is TRUE, a data frame.

Examples

```
### Note that this example requires an internet connection!
read_spreadsheet(
  paste0(
    "https://docs.google.com/",
    "spreadsheets/d/",
    "1bHDzpCu4CwEa5_3_q_9vH2691XPhCS3e4Aj_HLhw_U8"
```

```
  )
);
```

`repeatStr` *Repeat a string a number of times*

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);
```

`serialized_data_to_dfs` *In an object imported from YAML or JSON, convert some elements to dataframes*

Description

In an object imported from YAML or JSON, convert some elements to dataframes

Usage

```
serialized_data_to_dfs(x)
```

Arguments

x	The just imported (pre)registration specification

Value

The restructured object

`serialize_df` *"Serialize" a data frame or (pre)registration specification*

Description

When exporting a (pre)registration specification to YAML or JSON, the most human-readable format differs from the way data frames are comprised of lists. Data frames are lists that are bound together as columns; and so, when saving a data frame to YAML or JSON, the data in each column is combined (e.g. first all item identifiers, then all item labels, then all item descriptions, etc). However, for humans, it makes more sense to have all data belonging to the same item close together. These functions do that processing.

Usage

```
serialize_df(x, idCol = NULL)

structure_for_serialization(x)
```

Arguments

x	For `serialize_df`, a data frame; for `structure_for_serialization`, the (pre)registration specification
idCol	If not `NULL`, the name of a column in the data frame to use as names for the lists.

Value

The restructured list

source *Programmatically constructing justifier elements*

Description

These functions can be used to programmatically construct decision, justifications, assertions, and sources using the `justifier` package.

Usage

```
source(label, description = NULL, type = NULL, id = NULL, xdoi = NULL, ...)

assert(label, description = "", type = NULL, id = NULL, source = NULL, ...)

justify(label, description = "", type = NULL, id = NULL, assertion = NULL, ...)

decide(
  label,
  description = NULL,
  type = NULL,
  id = NULL,
  alternatives = NULL,
  justification = NULL,
  ...
)
```

Arguments

`label`	A human-readable label for the `decision`, `justification`, `assertion`, or `source`. Labels are brief summaries of the core of the decision, justification, assertion, or source. More details, background information, context, and other comments can be placed in the description.
`description`	A human-readable description. This can be used to elaborate on the label. Note that the label should be reader-friendly and self-contained; but because they also have to be as short as possible, descriptions can be used to provide definitions, context, background information, or add any other metadata or comments.
`type`	Types are used when working with a framework. Frameworks define type identifiers, consisting of letters, digits, and underscores. By specifying these identifiers the type of a decision, justification, assertion, or source. Source types can be, for example, types of documents or other data providers, such as "empirical evidence', 'expert consensus', 'personal opinion', or 'that one meeting that we had in May'. Assertion types can be, for example, data types or types of facts, such as 'number', 'prevalence', 'causal relationship', or 'contact information'. Justification types can be, for example, types of reasoning or logical expressions, such as 'deduction', 'induction', or 'intersection'. Decision types are the most framework-specific, heavily depend on the specific context of the decision,

	and are used by frameworks to organise the decisions in a project. Examples of decision types are the decision to recruit a certain number of participants in a scientific study; the decision to target a certain belief in a behavior change intervention; the decision to merge two codes in a qualitative study; the decision to hire a staff member; or the decision to make a certain purchase.
`id`	The identifier (randomly generated if omitted).
`xdoi`	For `sources`, XDOI identifier (a DOI, or, if that does not exist, ISBN or other unique identifier of the source).
`...`	Additional fields and values to store in the element.
`source`	In assertions, the source (or sources) that the assertion is based on can be specified using `srce()`.
`assertion`	In justifications, the assertion (or assertions) that the justification is based on can be specified using `asrt()`.
`alternatives`	The alternatives that were considered in a decision.
`justification`	In decisions, the justification (or justifications) that the decision is based on can be specified using `jstf()`.

Value

The generated object.

Examples

```
### Programmatically create a partial justification object
exampleAssertion <-
  preregr::assert(
    "This is an assertion",
    source = c(
      preregr::source('This is a first source'),
      preregr::source('This is a second source')));

### Programmatically create a justification with two assertions
### but without sources
exampleJustification <-
  preregr::justify(
    "Icecream will make me feel less fit",
    assertion = c(
      preregr::assert('Icecream is rich in energy'),
      preregr::assert('Consuming high-energy foods makes me feel less fit')
    ),
    weight = -.5
  );

### Show it
exampleJustification;

### Programmatically create a simple decision
simpleDecision <-
  preregr::decide(
```

```
    "decision",
    justification = preregr::jstf(
      "justification",
      assertion = exampleAssertion
    )
  );

### Programmatically create a justification object for a full decision
fullJustifierObject <-
  preregr::decide(
    "I decide to go get an icecream",
    justification = c(
      preregr::justify(
        "Having an icecream now would make me happy",
        assertion = c(
          preregr::assert(
            "Decreasing hunger increases happiness",
            source = preregr::source(
              "My past experiences"
            )
          ),
          preregr::assert(
            "I feel hungry",
            source = preregr::source(
              "Bodily sensations"
            )
          )
        ),
        weight = 1
      ),
      exampleJustification,
      preregr::justify(
        "I can afford to buy an icecream.",
        assertion = c(
          preregr::assert(
            "My bank account balance is over 300 euro.",
            source = preregr::source(
              "My bank app"
            )
          ),
          preregr::assert(
            "I need to keep at least 100 euro in my bank account.",
            source = preregr::source(
              "Parental advice"
            )
          )
        ),
        weight = .3
      )
    )
  );

### Show the full object
```

```
fullJustifierObject;

### Combine both into a list of decisions
twoDecisions <-
  c(simpleDecision,
    fullJustifierObject);

### Show the combination
twoDecisions;
```

`validate_value` *Validate a value*

Description

This function validates a value. Before validation, it checks the validation expression and optionally performs replacements, where the replacement strings (delimited by the validation replacement delimiters, by default, {{ and }}) are replaced by the first valid corresponding value from the `replacementSources` (working through those sources consecutively, i.e. only looking in the second one of the first one doesn't contain a valid value for the relevant replacement string, valid being defined as a value that is not NULL or NA and has, after being trimmed of whitespace, a nonzero length).

Usage

```
validate_value(
  VALUE,
  validations,
  replacementSources,
  errorMessages,
  convert_bipiped = TRUE
)
```

Arguments

`VALUE`	The value to validate.
`validations`	The validations, a vector or list that will be consecutively searched for the first valid value (valid being defined as a value that is not NULL or NA and has, after being trimmed of whitespace, a nonzero length). That value has to be either as an R expression, or a character value (i.e. a length 1 character vector) that is a valid R expression, optionally after having performed the specified replacements.
`replacementSources`	A list of named lists (or 1-row data frames) that will be searched, consecutively, for values to replace the replacement strings with.
`errorMessages`	The errormessage to return if validation fails, of which the first valid one will be returned (valid being defined as a value that is not NULL or NA and has, after being trimmed of whitespace, a nonzero length).

convert_bipiped
: Whether to first run bipiped_value_to_vector() on the replacment values.

Details

To change the validation replacement delimiters, use preregr::opts$set(validation_replacementDelimiters = c("{{", "}}"));.

Value

The message resulting from the validation (i.e. an error or "").

Examples

```
### Set some validation variables
validationStatement <-
  paste(
    "is.na(VALUE) ||",
    "(VALUE %in% {{validValues}}) ||",
    "(VALUE %in% {{testField}})"
  );
replacementSources <-
  list(
    list(validValues = '"testValue" || "anotherValue"'),
    list(testField = "Yet another testvalue")
  );
errorMessages <-
  "No valid test value passed!";

### Run a passing validation
preregr::validate_value(
  "testValue",
  validations = validationStatement,
  replacementSources = replacementSources,
  errorMessages = errorMessages
);


### Run a failing validation
preregr::validate_value(
  "A testvalue that won't pass",
  validations = validationStatement,
  replacementSources = replacementSources,
  errorMessages = errorMessages
);
```

---

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,
  colFun = NULL
)

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`)?
`colFun`	A function to use for coloring.
`...`	Any addition arguments to `vecTxtQ` are passed on to `vecTxt`.

Value

A character vector of length 1.

Examples

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

wrapVector *Wrap all elements in a vector*

Description

Wrap all elements in a vector

Usage

```
wrapVector(x, width = 0.9 * getOption("width"), sep = "\n", ...)
```

Arguments

x	The character vector
width	The number of
sep	The glue with which to combine the new lines
...	Other arguments are passed to strwrap().

Value

A character vector

Examples

```
res <- wrapVector(
  c(
    "This is a sentence ready for wrapping",
    "So is this one, although it's a bit longer"
  ),
  width = 10
);

print(res);
cat(res, sep="\n");
```

yaml_to_prereg_spec *Convert a (pre)registration specification from YAML or JSON*

Description

Convert a (pre)registration specification from YAML or JSON

Usage

```
yaml_to_prereg_spec(x)
```

Arguments

x	The YAML or JSON as character vector, or a path to a file containing the YAML or JSON.

Value

The imported object.

Examples

```
### Get path to example file
examplePreregFile <-
  system.file(
    "extdata",
    "preregr-spec-example1.yml",
    package = "preregr"
  );

### Load it and show which items are completed
preregr::yaml_to_prereg_spec(
  examplePreregFile
) |>
  preregr::prereg_show_item_completion();
```

Index