Package 'psyverse'

May 9, 2026

Type Package

Title Decentralized Unequivocality in Psychological Science

Version 0.2.6

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

License GPL (>= 3)

Description The constructs used to study the human psychology have many definitions and corresponding instructions for eliciting and coding qualitative data pertaining to constructs' content and for measuring the constructs. This plethora of definitions and instructions necessitates unequivocal reference to specific definitions and instructions in empirical and secondary research. This package implements a human- and machine-readable standard for specifying construct definitions and instructions for measurement and qualitative research based on 'YAML'. This standard facilitates systematic unequivocal reference to specific construct definitions and corresponding instructions in a decentralized manner (i.e. without requiring central curation; Peters (2020) <doi:10.31234/osf.io/xebhn>).

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

URL https://psyverse.one

Encoding UTF-8

RoxygenNote 7.2.2

Depends R (>= 3.0.0)

Imports stats (>= 3.0.0), yaml (>= 2.1.19), yum (>= 0.0.1)

Suggests covr, DiagrammeR (>= 1.0.0), haven, googlesheets4, knitr, openxlsx, readxl, rmarkdown, rstudioapi, testthat, writexl, XLConnect

VignetteBuilder knitr

NeedsCompilation no

Author Gjalt-Jorn Peters [aut, cre, ctb] (ORCID: <https://orcid.org/0000-0002-0336-9589>), Rik Crutzen [ctb] (ORCID: <https://orcid.org/0000-0002-3731-6610>)

Repository CRAN

Date/Publication 2023-03-05 22:00:07 UTC

Contents

`apply_graph_theme` *Apply multiple DiagrammeR global graph attributes*

Description

Apply multiple DiagrammeR global graph attributes

Usage

```
apply_graph_theme(dctGraph, ...)
```

Arguments

`dctGraph`	The DiagrammeR::DiagrammeR graph to apply the attributes to.
`...`	One or more character vectors of length three, where the first element is the attribute, the second the value, and the third, the attribute type (`graph`, `node`, or `edge`).

Value

The DiagrammeR::DiagrammeR graph.

Examples

```
exampleSpec <-
  system.file("inst",
              "extdata",
              "example_dct_spec_1.dct",
              package="psyverse");
parsedSpecs <- load_dct_specs(exampleSpec);
dctGraph <- parsedSpecs$output$basic_graph;
dctGraph <- apply_graph_theme(dctGraph,
                              c("color", "#0000AA", "node"),
                              c("fillcolor", "#00FFFF", "node"));
```

---

base30toNumeric *Conversion between base10 and base30 & base36*

Description

The conversion functions from base10 to base30 are used by the `generate_id()` functions; the base36 functions are just left here for convenience.

Usage

```
base30toNumeric(x)

base36toNumeric(x)

numericToBase30(x)

numericToBase36(x)
```

Arguments

`x`	The vector to convert (numeric for the `numericTo` functions, character for the `base30to` and `base36to` funtions).

Details

The symbols to represent the 'base 30' system are the 0-9 followed by the alphabet without vowels but including the y. This vector is available as `base30`.

Value

The converted vector (numeric for the `base30to` and `base36to` funtions, character for the `numericTo` functions).

Examples

```
numericToBase30(654321);
base30toNumeric(numericToBase30(654321));
```

`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], "'.");
```

`dct_from_spreadsheet` *Import a DCT specification from a spreadsheet*

Description

This function reads a spreadsheet (from a Google sheet URL or a local file in `.xlsx`, `.csv`, or `.sav` format) and imports the DCT specifications in it.

Usage

```
dct_from_spreadsheet(
  x,
  path = NULL,
  sheet = NULL,
  localBackup = NULL,
  exportGoogleSheet = TRUE,
  xlsxPkg = c("rw_xl", "openxlsx", "XLConnect"),
  preventOverwriting = psyverse::opts$get("preventOverwriting"),
  encoding = psyverse::opts$get("encoding"),
  silent = psyverse::opts$get("silent")
)
```

Arguments

x	The URL or path to a file.
path	The path to save the DCT specifications.
sheet	Optionally, the name(s) of the worksheet(s) to select.
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 the 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.
preventOverwriting	Whether to prevent overwriting.
encoding	The encoding to use.
silent	Whether to be silent or chatty.

Value

Invisibly, an object with the worksheets and the DCT objects.

---

dct_object *Create a DCT object*

Description

Create a DCT object

Usage

```
dct_object(
  version = as.character(utils::packageVersion("psyverse")),
  id = NULL,
  prefix = paste(sample(letters, 4), collapse = ""),
  label = "",
  date = as.character(Sys.Date()),
  dct_version = "1",
  ancestry = "",
  retires = "",
  definition = "",
  measure_dev = "",
  measure_code = "",
  aspect_dev = "",
  aspect_code = "",
  comments = "",
  rel = NULL
)
```

Arguments

`version`	The version of the DCT specification format (normally the version of the `psyverse` package).
`id`	The Unique Construct Identifier (UCID); if not provided, this is created using the `prefix`.
`prefix`	The prefix to use to construct the Unique Construct Identifier (UCID); ignored i `id` is provided.
`label`	The human-readable label for the construct.
`date`	The date at which the construct was created.
`dct_version`	The version of the DCT specification. This can optionally be used to manage consecutive DCT versions.
`ancestry`	The DCT specification or specifications that this DCT was based on.
`retires`	The DCT specification or specifications that this DCT renders obsolete (note that this doesn't mean anything in itself; `psyverse` does not enforce this automatically, nor does PsyCoRe, without configuration).
`definition`	The definition of the construct. This has to be comprehensive, detailed, accurate, and clearly delineate the relevant aspects of the human psychology.
`measure_dev`	Instructions for developing measurement instruments that measure this construct.
`measure_code`	Instructions for coding measurement instruments (e.g. in systematic reviews) as measurement instruments that measure this construct. Note that explicitly defining boundary conditions often helps, for example by explaining the features that coders should look for to distinguish this construct from closely related constructs (ideally linking to those other constructs using the `dct:UCID` notations).

aspect_dev	Instructions for eliciting construct content. Note that this is not sensible for all constructs; some may be defined at a very general level, rendering their content insufficiently specific to discuss or describe.
aspect_code	Instructions for coding construct content (i.e. aspects). Note that explicitly defining boundary conditions often helps, for example by explaining the features that coders should look for to distinguish this construct from closely related constructs (ideally linking to those other constructs using the dct:UCID notations).
comments	Any additional comments.
rel	Relationships with other constructs.

Value

The DCT object.

Examples

```
exampleDCT <-
  psyverse::dct_object(
    prefix = "exampleConstruct",
    label = "An example construct",
    definition = "The definition goes here",
    measure_dev = "Here you can explain how to measure the construct"
  );
```

---

dct_object_to_html *Create an HTML fragment showing a DCT object*

Description

Create an HTML fragment showing a DCT object

Usage

```
dct_object_to_html(
  dctObject,
  headingLevel = 3,
  hyperlink_UCIDs = TRUE,
  collapseButtons = TRUE,
  urlPrefix = "#",
  sortDecreasing = FALSE
)
```

Arguments

dctObject	The DCT object
headingLevel	The level of the heading in the Markdown output that is produces.
hyperlink_UCIDs	Whether to create hyperlinks to UCIDs.
collapseButtons	Whether to include buttons to show/hide the definition and instructions.
urlPrefix	The prefix to insert before the URL in the produced hyperlink. The default, "#", results in a link to an anchor (an HTML a element) on the current page.
sortDecreasing	Whether to sort the constructs in decreasing order (TRUE), in increasing order (FALSE), or not at all (NULL).

Value

A character vector.

Examples

```
exampleDCT <-
  psyverse::dct_object(
    prefix = "exampleConstruct",
    label = "An example construct",
    definition = "The definition goes here",
    measure_dev = "Here you can explain how to measure the construct"
  );
### Only run this in an interactive R session,
### as it shows the HTML in the viewer.
if (interactive()) {
  dct_object_to_html(exampleDCT);
}
```

---

dct_object_to_yaml *Convert a DCT object to YAML*

Description

Convert a DCT object to YAML

Usage

```
dct_object_to_yaml(dctObject)
```

Arguments

dctObject	The DCT object

Value

A character vector.

dct_sheet_to_dct *Create a DCT object from a DCT sheet*

Description

Create a DCT object from a DCT sheet

Usage

```
dct_sheet_to_dct(dct_sheet)
```

Arguments

dct_sheet	A dataframe containing a DCT specification.

Value

A DCT created by `dct_object()`.

generate_construct_overview *Generate construct overviews and instruction overviews*

Description

These functions use a DCT specification to generate a construct overview or an instruction overview.

Usage

```
generate_construct_overview(
  dctSpec,
  include = c("definition", "measure_dev", "measure_code", "manipulate_dev",
    "manipulate_code", "aspect_dev", "aspect_code", "rel"),
  hideByDefault = NULL,
  divClass = "btn btn-secondary",
  headingLevel = 3,
  collapseButtons = TRUE,
  hyperlink_UCIDs = TRUE,
  HTMLoutput = FALSE,
  urlPrefix = "#",
  sortDecreasing = FALSE
)

generate_definitions_overview(
  dctSpecDf,
  headingLevel = 3,
```

```
  hyperlink_UCIDs = "Markdown",
  urlPrefix = "#",
  sortDecreasing = FALSE
)

generate_instruction_overview(
  dctSpecDf,
  type,
  headingLevel = 3,
  hyperlink_UCIDs = "Markdown",
  urlPrefix = "#",
  sortDecreasing = FALSE
)
```

Arguments

dctSpec	The DCT specification, as resulting from a call to load_dct_specs() or load_dct_dir().
include	Which elements to include in the construct overview.
hideByDefault	Which elements to hide by default.
divClass	The class of the button to collapse/expand sections.
headingLevel	The level of the heading in the Markdown output that is produces.
collapseButtons	Whether to include buttons to show/hide the definition and instructions.
hyperlink_UCIDs	Whether to create hyperlinks to UCIDs.
HTMLoutput	Whether to output to Markdown (FALSE) or HTML (TRUE).
urlPrefix	The prefix to insert before the URL in the produced hyperlink. The default, "#", results in a link to an anchor (an HTML a element) on the current page.
sortDecreasing	Whether to sort the constructs in decreasing order (TRUE), in increasing order (FALSE), or not at all (NULL).
dctSpecDf	The DCT specification dataframer, as produced by a call to load_dct_specs() or load_dct_dir(), and stored within the resulting object.
type	For instruction overviews, the type of instruction to generate can be specified: must be one of "measure_dev", "measure_code", "manipulate_dev", "manipulate_code", "aspect_dev", or "aspect_code".

Value

A character string with the overview.

Examples

```
exampleDCT <-
  psyverse::dct_object(
    prefix = "exampleConstruct",
    label = "An example construct",
```

```
    definition = "The definition goes here",
    measure_dev = "Here you can explain how to measure the construct"
  );
generate_construct_overview(exampleDCT);
```

`generate_dct_template` *DCT templates*

Description

These functions can generate one or more empty DCT templates.

Usage

```
generate_dct_template(
  prefix = paste(sample(letters, 4), collapse = ""),
  output = NULL,
  overwrite = FALSE,
  createDirs = FALSE,
  addComments = TRUE,
  stopOnIllegalChars = FALSE
)

generate_dct_templates(
  x,
  outputDir = NULL,
  createDirs = FALSE,
  addComments = FALSE,
  stopOnIllegalChars = FALSE
)
```

Arguments

`prefix`, `x`	The prefix (`prefix`) or vector of prefixes (`x`) to use.
`output`, `outputDir`	The filename or directory to which to write the templates.
`overwrite`	Whether to overwrite any existing files.
`createDirs`	Whether to recursively create the directories if the path specified in `output` or `outputPath` does not yet exist.
`addComments`	Whether to add comments to the DCT specification as extra explanation.
`stopOnIllegalChars`	DCT identifier prefixes can only contain upper- and lowercase letters and underscores. This argument specifies whether to remove illegal characters with a warning, or whether to throw an error (and stop) if illegal characters are found,

Value

The DCT template(s), either invisibly (if `output` or `outputDir` is specified) or visibly.

generate_id *Generate unique identifier(s)*

Description

To allow unique reference to constructs, they require unique identifiers. These functions generate such identifiers by combining one or more identifier prefixes (usually a human-readable construct name such as 'attitude') with a unique identifier based on the second the identifier was generated. The identifier prefix may only contain lowercase and uppercase letters and underscores.

Usage

```
generate_id(
  prefix = paste(sample(letters, 4), collapse = ""),
  stopOnIllegalChars = FALSE
)

generate_ids(x, stopOnIllegalChars = FALSE)
```

Arguments

prefix	An identifier prefix.
stopOnIllegalChars	Whether to base::stop() or produce a base::warning() when encountering illegal characters (i.e. anything other than a letter or underscore).
x	A vector of identifier prefixes.

Value

a character vector containing the identifier(s).

Examples

```
generate_id('attitude');
```

invert_id *Invert identifier*

Description

Invert the identifier (generated by generate_id() for one or more constructs. This means that the identifier prefix is stripped and the last part is converted back from base 30 to base 10.

Usage

```
invert_id(x)
```

Arguments

x	The identifier(s) as a character vector.

Value

The identifier(s) as a numeric vector.

Examples

```
invert_id(generate_id('example'));
```

---

load_dct_dir *Load DCT specifications from a file or multiple files*

Description

These function load DCT specifications from the YAML fragments in one (`load_dct_specs`) or multiple files (`load_dct_dir`).

Usage

```
load_dct_dir(
  path,
  recursive = TRUE,
  extension = "\\.rock|\\.dct\\.yaml|\\.yaml|\\.yml",
  regex,
  dctContainer = "dct",
  headingLevel = 2,
  delimiterRegEx = "^---$",
  ignoreOddDelimiters = FALSE,
  encoding = "UTF-8",
  sortDecreasing = FALSE,
  silent = TRUE
)

load_dct_specs(
  text,
  file,
  delimiterRegEx = "^---$",
  dctContainer = "dct",
  headingLevel = 2,
  ignoreOddDelimiters = FALSE,
  encoding = "UTF-8",
  silent = TRUE
)

## S3 method for class 'dct_specs'
```

```
print(x, ...)

## S3 method for class 'dct_specs'
plot(x, ...)
```

Arguments

path	The path containing the files to read.
recursive	Whether to also process subdirectories (TRUE) or not (FALSE).
extension	The extension of the files to read; files with other extensions will be ignored. Multiple extensions can be separated by a pipe (|).
regex	Instead of specifing an extension, it's also possible to specify a regular expression; only files matching this regular expression are read. If specified, regex takes precedece over extension,
dctContainer	The container of the DCT specifications in the YAML fragments. Because only DCT specifications are read that are stored in this container, the files can contain YAML fragments with other data, too, without interfering with the parsing of the DCT specifications.
headingLevel	The level of the Markdown headings that are produced.
delimiterRegEx	The regular expression used to locate YAML fragments
ignoreOddDelimiters	Whether to throw an error (FALSE) or delete the last delimiter (TRUE) if an odd number of delimiters is encountered.
encoding	The encoding to use when calling readLines(). Set to NULL to let readLines() guess.
sortDecreasing	Whether to sort the constructs in decreasing order (TRUE), in increasing order (FALSE), or not at all (NULL).
silent	Whether to be silent (TRUE) or informative (FALSE).
text, file	As text or file, you can specify a file to read with encoding encoding, which will then be read using base::readLines(). If the argument is named text, whether it is the path to an existing file is checked first, and if it is, that file is read. If the argument is named file, and it does not point to an existing file, an error is produced (useful if calling from other functions). A text should be a character vector where every element is a line of the original source (like provided by base::readLines()); although if a character vector of one element *and* including at least one newline character (\\n) is provided as text, it is split at the newline characters using base::strsplit(). Basically, this behavior means that the first argument can be either a character vector or the path to a file; and if you're specifying a file and you want to be certain that an error is thrown if it doesn't exist, make sure to name it file.
x	The parsed parsed_dct object.
...	Any other arguments are passed to the print command.

Details

load_dct_dir simply identifies all files and then calls load_dct_specs for each of them. load_dct_specs loads the YAML fragments containing the DCT specifications using yum::load_yaml_fragments() and then parses the DCT specifications into a visual representation as a DiagrammeR::DiagrammeR graph and Markdown documents with the instructions for creating measurement instruments or manipulations, and for coding measurement instruments, manipulations, or aspects of a construct.

Value

An object with the DiagrammeR::DiagrammeR graph stored in output$basic_graph, a DiagrammeR::DiagrammeR graph with a summary of which specifications are provided for each construct in output$completeness_graph and the instructions in output$instr.

Examples

```
exampleSpec <-
  system.file("extdata",
              "example.dct.yaml",
              package="psyverse");
dctObject <- load_dct_specs(exampleSpec);

## Not run:
psyverse::load_dct_dir(path="A:/some/path");

## End(Not run)
```

---

opts *Options for the psyverse package*

Description

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

The following arguments can be passed:

... For `psyverse::opts$set`, the dots can be used to specify the options to set, in the format `option = value`, for example, `encoding = "UTF-8"`. For `psyverse::opts$reset`, a list of options to be reset can be passed.

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

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

The following options can be set:

encoding The default encoding used to read or write files.

Examples

```
### Get the default encoding
psyverse::opts$get(encoding);

### Set it to UTF-8-BOM
psyverse::opts$set(encoding = "UTF-8-BOM");

### Check that it worked
psyverse::opts$get(encoding);

### Reset this option to its default value
psyverse::opts$reset(encoding);

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

---

parse_dct_specs *Parse DCT specifications*

Description

This function parses DCT specifications; it's normally called by `load_dct_dir()` or `load_dct_specs()`, so you won't have to use it directly.

Usage

```
parse_dct_specs(
  dctSpecs,
  headingLevel = 2,
  hyperlink_UCIDs = TRUE,
```

```
  urlPrefix = "#",
  HTMLoutput = FALSE,
  sortDecreasing = FALSE
)
```

Arguments

dctSpecs	The DCT specifications (a list).
headingLevel	The heading level for Markdown output.
hyperlink_UCIDs, urlPrefix, HTMLoutput	Passed on to the generate_instruction_overview() and generate_construct_overview() functions.
sortDecreasing	Whether to sort the constructs in decreasing order (TRUE), in increasing order (FALSE), or not at all (NULL).

Value

The object of parsed DCT specifications.

---

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 dct_from_spreadsheet().

Usage

```
read_spreadsheet(
  x,
  sheet = NULL,
  columnDictionary = NULL,
  localBackup = NULL,
  exportGoogleSheet = TRUE,
  flattenSingleDf = FALSE,
  xlsxPkg = c("rw_xl", "openxlsx", "XLConnect"),
  failQuietly = FALSE,
  silent = psyverse::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 the 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);
```

`save_to_yaml` *Save a psyverse object or YAML character vector to a file*

Description

Pretty much what it says on the box. But check the bit about encoding.

Usage

```
save_to_yaml(
  x,
  file,
  preventOverwriting = psyverse::opts$get("preventOverwriting"),
  encoding = psyverse::opts$get("encoding")
)
```

Arguments

`x`	The object to save.
`file`	The file to save to.
`preventOverwriting`	Whether to prevent overwriting.
`encoding`	The encoding to use. Note that in general, encoding seems to have been invented primarily as a source of frustration, and it rarely disappoints. If unsure, use UTF-8. If using UTF-8, the approach from https://kevinushey.github.io/blog/2018/02/21/string-encoding-and-r/ will be used.

Value

The character vector that was written to the file.

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));
```

`viewHTML` *Display HTML*

Description

This function displays HTML in the viewer, adding <body> and <head> tags (which should therefore not be included in the fragment).

Usage

```
viewHTML(x, title = "Psyverse", css = "body {font-size: 16px;}")
```

Arguments

`x`	The HTML fragment
`title`	The title
`css`	CSS

Value

Invisibly, `x`, with the extra HTML bits added.

Examples

```
### Only run this example in an interactive R session,
### as it shows the HTML in the viewer.
if (interactive()) {
  psyverse::viewHTML("<strong>Hello world!</strong>");
}
```

Index