Package: formr (via r-universe)

October 18, 2024

Title formr survey framework

Description The formr R package provides a few convenience functions that may be useful to the users of formr (formr.org), an online survey framework which heavily relies on R via openCPU. Some of the functions are for conveniently generating individual feedback graphics, some are just shorthands to make certain common operations in formr more palatable to R novices.

Version 0.10.2

Depends R (>= 3.0.2)

Imports stats, methods, dplyr, ggplot2, scales, haven, stringr, tidyr, knitr, httr, curl, jsonlite, lubridate, commonmark, rmarkdown, keyring

License MIT + file LICENSE

LazyData true

Date 2014-12-09 12:06:54

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Encoding UTF-8

Repository https://rforms.r-universe.dev

RemoteUrl https://github.com/rubenarslan/formr

RemoteRef HEAD

RemoteSha c9e5dcc6ff7905ae837a3015e62c456825518eaa

Contents

`aggregate_and_document_scale`

Aggregate variables and remember which variables this were

Description

Copied from codebook. The resulting variables will have the attribute `scale_item_names` containing the basis for aggregation. Its `label` attribute will refer to the common stem of the aggregated variable names (if any), the number of variables, and the aggregation function.

Usage

```
aggregate_and_document_scale(items, fun = rowMeans, stem = NULL)
```

Arguments

`items`	data.frame of the items that should be aggregated
`fun`	aggregation function, defaults to rowMeans with na.rm = FALSE
`stem`	common stem for the variables, specify if it should not be auto-detected as the longest common stem of the variable names

Examples

```
testdf <- data.frame(bfi_neuro_1 = rnorm(20), bfi_neuro_2 = rnorm(20),
                     bfi_neuro_3R = rnorm(20), age = rpois(20, 30))
item_names <- c('bfi_neuro_1', 'bfi_neuro_2', 'bfi_neuro_3R')
testdf$bfi_neuro <- aggregate_and_document_scale(testdf[, item_names])
testdf$bfi_neuro
```

as.data.frame.formr_item_list
Transform formr_item_list into a data.frame for ease of use

Description

This function just turns a formr_item_list into a data.frame. The reason, these lists don't come as data.frames as default is because the 'choices' are a list themselves. When transforming, the choice column contains a collapsed choice list, which may be less useful for some purposes.

Usage

```
## S3 method for class 'formr_item_list'
as.data.frame(x, row.names, ...)
```

Arguments

`x`	a formr_item_list
`row.names`	not used
`...`	not used

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
as.data.frame(formr_items(survey_name = 'training_diary' ))

## End(Not run)
items = formr_items(path =
system.file('extdata/gods_example_items.json', package = 'formr', mustWork = TRUE))
items_df = as.data.frame(items)
items_df[1,]
```

asis_knit_child
knit_child as is

Description

This slightly modifies the `knitr::knit_child()` function to have different defaults.

- the environment defaults to the calling environment.
- the output receives the class knit_asis, so that the output will be rendered "as is" by knitr when calling inside a chunk (no need to set results='asis' as a chunk option).
- defaults to quiet = TRUE

Usage

```
asis_knit_child(
  input = NULL,
  text = NULL,
  ...,
  quiet = TRUE,
  options = NULL,
  envir = parent.frame()
)
```

Arguments

`input`	if you specify a file path here, it will be read in before being passed to knitr (to avoid a working directory mess)
`text`	passed to `knitr::knit_child()`
`...`	passed to `knitr::knit_child()`
`quiet`	passed to `knitr::knit_child()`
`options`	defaults to NULL.
`envir`	passed to `knitr::knit_child()`

Details

Why default to the calling environment? Typically this function defaults to the global environment. This makes sense if you want to use knit_children in the same context as the rest of the document. However, you may also want to use knit_children inside functions to e.g. summarise a regression using a set of commands (e.g. plot some diagnostic graphs and a summary for a regression nicely formatted).

Some caveats:

- the function has to return to the top-level. There's no way to `cat()` this from loops or an if-condition without without setting results='asis'. You can however concatenate these objects with `paste.knit_asis()`

Examples

```
## Not run:
# an example of a wrapper function that calls asis_knit_child with an argument
# ensures distinct paths for cache and figures, so that these calls can be looped in parallel
regression_summary = function(model) {
   child_hash = digest::digest(model)
   options = list(
       fig.path = paste0(knitr::opts_chunk$get("fig.path"), child_hash, "-"),
       cache.path = paste0(knitr::opts_chunk$get("cache.path"), child_hash, "-"))
   asis_knit_child("_regression_summary.Rmd", options = options)
}

## End(Not run)
```

choice_labels_for_values

switch choice values with labels

Description

formr display labels for multiple choice items, but stores their values. We assume you prefer to analyse the values (e.g. numeric values for Likert-type items, or English values for international surveys), but sometimes you may wish to switch this around.

Usage

```
choice_labels_for_values(survey, item_name)
```

Arguments

`survey`	survey with item_list attribute
`item_name`	item name

Examples

```
example(formr_post_process_results)
table(processed_results$BFIK_extra_4)
table(choice_labels_for_values(processed_results, "BFIK_extra_4"))
```

crosstabs

xtabs with sensible defaults

Description

xtabs requires two arguments (na.action and exclude) to show missing values along with other values. This function defaults to including missings and has only one argument

Usage

```
crosstabs(x, ..., exclude = NULL)
```

Arguments

`x`	passed to xtabs if it is a formula, transformed into a formula if it's a single object
`...`	passed to xtabs
`exclude`	defaults to NULL (i.e. includes NA)

Examples

```
x = NA
crosstabs(~ x)
```

`current` *Gives the last element, doesn't omit missings*

Description

Just a simple shorthand to get the current element (in a formr df, where the last element is always the one from the current session).

Usage

```
current(x)
```

Arguments

`x`	vector of which you want the current element

Examples

```
current( c(1:10,NA) )
current( 1:10 )
```

`email_image` *generates valid email cids*

Description

can be used as an argument to knitr::opts_knit. If you attach the images properly, you can then send knit emails including plots. See the formr OpenCPU module on Github for a sample implementation.

Usage

```
email_image(x, ext = ".png")
```

Arguments

`x`	image ID
`ext`	extension, defaults to .png

Examples

```
## Not run:
library(knitr); library(formr)
opts_knit$set(upload.fun=formr::email_image)

## End(Not run)
```

`expired` *How many surveys were expired?*

Description

Just a simple to check how many times a survey (e.g. diary) has expired (i.e. user missed it). It defaults to checking the "expired" variable for this.

Usage

```
expired(survey, variable = "expired")
```

Arguments

`survey`	which survey are you asking about?
`variable`	which variable should be filled out, defaults to "ended"

Examples

```
survey = data.frame(expired = c(NA, "2016-05-29 10:11:00", NA))
expired(survey = survey)
```

`feedback_chunk` *Text feedback based on groups*

Description

If you pass in a z-standardised value (x - Mean)/SD, and a vector of feedback text chunks, that has either three or five elements, the text chunks will be used in this order [very low], low, average, high, [very high] corresponding to these intervals [low, -2], [-2, -1], [-1, 1], [1, 2], [2, high]

Usage

```
feedback_chunk(normed_value, chunks)
```

Arguments

`normed_value`	a z-standardised value
`chunks`	a three or five element long character vector containing the text chunks for feedback

Examples

```
feedback_chunk(normed_value = 0.7, chunks = c("You are rather introverted.",
"You're approximately as extraverted as most people.","You are rather extraverted."))
```

finished	*How many surveys were finished?*

Description

Just a simple to check how many times a survey (e.g. diary) was finished. It defaults to checking the "ended" variable for this.

Usage

```
finished(survey, variable = "ended")
```

Arguments

survey	which survey are you asking about?
variable	which variable should be filled out, defaults to "ended"

Examples

```
survey = data.frame(ended = c("2016-05-28 10:11:00", NA, "2016-05-30 11:18:28"))
finished(survey = survey)
```

first	*Gives the first non-missing element*

Description

Just a simple shorthand to get the first, non-missing argument per default. Can give more than one element and can include missing elements. The inverse of `last()`.

Usage

```
first(x, n = 1, na.rm = TRUE)
```

Arguments

x	vector of which you want the first element
n	number of elements to take from the beginning
na.rm	whether to remove missings first, defaults to TRUE

Examples

```
first( c(NA,1:10) )
first( c(NA, 1:10), 2, TRUE )
```

formr_aggregate *Aggregate data based on item table*

Description

If you've retrieved an item table using formr_items() you can use this function to aggregate your multiple choice items into mean scores. If you do not have a item table (e.g. your data was not collected using formr, you don't want another HTTP request in a time-sensitive process). Example: If your data contains Extraversion_1, Extraversion_2R and Extraversion_3, there will be two new variables in the result: Extraversion_2 (reversed to align with _1 and _2) and Extraversion, the mean score of the three.

Usage

```
formr_aggregate(
  survey_name,
  item_list = formr_items(survey_name, host = host),
  results = formr_raw_results(survey_name, host = host),
  host = formr_last_host(),
  compute_alphas = FALSE,
  fallback_max = 5,
  plot_likert = FALSE,
  quiet = FALSE,
  aggregation_function = rowMeans,
  ...
)
```

Arguments

survey_name	case-sensitive name of a survey your account owns
item_list	an item_list, will be auto-retrieved based on survey_name if omitted
results	survey results, will be auto-retrieved based on survey_name if omitted
host	defaults to formr_last_host(), which defaults to https://formr.org
compute_alphas	deprecated, functionality migrated to codebook package
fallback_max	defaults to 5 - if the item_list is set to null, we will use this to reverse
plot_likert	deprecated, functionality migrated to codebook package
quiet	defaults to FALSE - If set to true, likert plots and reliability computations are not echoed.
aggregation_function	defaults to rowMeans with na.rm = FALSE
...	passed to psych::alpha()

Examples

```
results = jsonlite::fromJSON(txt =
 system.file('extdata/gods_example_results.json', package = 'formr', mustWork = TRUE))
items = formr_items(path =
 system.file('extdata/gods_example_items.json', package = 'formr', mustWork = TRUE))
results = formr_recognise(item_list = items, results = results)
agg = formr_aggregate(item_list = items, results = results,
 compute_alphas = FALSE, plot_likert = FALSE)
agg[, c('religiousness', 'prefer')]
```

`formr_api_access_token` *Connect to formr API*

Description

Connects to formr using your client_id and client_secret (OAuth 2.0 grant type: client_credentials).

Usage

```
formr_api_access_token(
  client_id,
  client_secret,
  host = "https://api.formr.org/"
)
```

Arguments

`client_id`	your client_id
`client_secret`	your client_secret
`host`	defaults to https://formr.org

Examples

```
## Not run:
formr_api_access_token(client_id = 'your_id', client_secret = 'your_secret' )

## End(Not run)
```

`formr_api_results` *Get result from formr*

Description

After obtaining a token from formr, use this request

Usage

```
formr_api_results(request = NULL, token = NULL)
```

Arguments

`request`	parameter (see example, API docs)
`token`	defaults to last used token

Examples

```
## Not run:
request <-
 list(
  "run[name]" = 'widgets',
  "run[sessions]" =
    'PJ_nACjFQDEBhx7pMUfZQz3mV-OtetnpEdqT88aiY8eXE4-HegFI7Sri4yifxPXO',
  "surveys[all_widgets]" = "abode, yourstory, mc_god"
)
formr_api_results(request)

## End(Not run)
```

`formr_api_session` *Get current API session Return or set URL in list form for formr API (if available)*

Description

Get current API session Return or set URL in list form for formr API (if available)

Usage

```
formr_api_session()
```

`formr_connect` *Connect to formr*

Description

Connects to formr using your normal login and the httr library which supports persistent session cookies. Calling this function will persist the specified host (by default https://formr.org) in further formr_ function calls. You can change this by calling `formr_last_host()`

Usage

```
formr_connect(
  email = NULL,
  password = NULL,
  host = formr_last_host(),
  keyring = NULL
)
```

Arguments

`email`	your registered email address
`password`	your password
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org
`keyring`	a shorthand for the account you're using

Examples

```
## Not run:
formr_connect(keyring = "formr_diary_study_account" )

## End(Not run)
```

`formr_disconnect` *Disconnect from formr*

Description

Disconnects from formr if connected.

Usage

```
formr_disconnect(host = formr_last_host())
```

Arguments

`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_disconnect()

## End(Not run)
```

formr_inline_render *render inline text for formr*

Description

Render text

Usage

```
formr_inline_render(text, self_contained = TRUE, ...)
```

Arguments

`text`	that will be written to a tmp file and used as the input argument
`self_contained`	passed to markdown_custom_options
`...`	all other arguments passed to `rmarkdown::render()`

formr_items *Download items from formr*

Description

After connecting to formr using `formr_connect()` you can download items using this command. One of survey_name or path has to be specified, if both are specified, survey_name is preferred.

Usage

```
formr_items(survey_name = NULL, host = formr_last_host(), path = NULL)
```

Arguments

`survey_name`	case-sensitive name of a survey your account owns
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org
`path`	path to local JSON copy of the item table

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
formr_items(survey_name = 'training_diary' )

## End(Not run)
formr_items(path =
 system.file('extdata/gods_example_items.json', package = 'formr', mustWork = TRUE))[1:2]
```

`formr_item_displays` *Download detailed result timings and display counts from formr*

Description

After connecting to formr using `formr_connect()` you can download detailed times and display counts for each item using this command.

Usage

```
formr_item_displays(survey_name, host = formr_last_host())
```

Arguments

`survey_name`	case-sensitive name of a survey your account owns
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
formr_item_displays(survey_name = 'training_diary' )

## End(Not run)
```

`formr_knit` *knit rmarkdown to markdown for formr*

Description

Render text

Usage

```
formr_knit(text)
```

Arguments

`text`	rmarkdown that will be knit

`formr_last_host` *Get the last specified host*

Description

This function returns the default or the last specified host if called without an argument. It changes the host when called with an argument.

Usage

```
formr_last_host(host = NULL)
```

Arguments

`host`	defaults to https://formr.org

Value

the last specified host

Examples

```
formr_last_host("https://formr.org")
formr_last_host()
```

`formr_post_process_results` *Processed, aggregated results*

Description

This function chains `formr_recognise()` and `formr_aggregate()` in sequence. Useful if you want to post-process raw results before aggregating etc.

Usage

```
formr_post_process_results(
  item_list = NULL,
  results,
  compute_alphas = FALSE,
  fallback_max = 5,
  plot_likert = FALSE,
  quiet = FALSE,
  item_displays = NULL,
  tag_missings = !is.null(item_displays),
  remove_test_sessions = TRUE
)
```

Arguments

`item_list`	an item_list, defaults to NULL
`results`	survey results
`compute_alphas`	passed to formr_aggregate, defaults to TRUE
`fallback_max`	passed to formr_reverse, defaults to 5
`plot_likert`	passed to formr_aggregate, defaults to TRUE
`quiet`	passed to formr_aggregate, defaults to FALSE
`item_displays`	an item display table, necessary to tag missings
`tag_missings`	should missings that result from an item not being shown be distinguished from missings due to skipped questions?
`remove_test_sessions`	by default, formr removes results resulting from test session (animal names and null session codes)

Examples

```
results = jsonlite::fromJSON(txt =
 system.file('extdata/BFI_post.json', package = 'formr', mustWork = TRUE))
items = formr_items(path =
 system.file('extdata/BFI_post_items.json', package = 'formr', mustWork = TRUE))
item_displays = jsonlite::fromJSON(
 system.file('extdata/BFI_post_itemdisplay.json', package = 'formr', mustWork = TRUE))
processed_results = formr_post_process_results(items, results, item_displays = item_displays,
compute_alphas = FALSE, plot_likert = FALSE)
```

`formr_raw_results` *Download data from formr*

Description

After connecting to formr using `formr_connect()` you can download data using this command.

Usage

```
formr_raw_results(survey_name, host = formr_last_host())
```

Arguments

`survey_name`	case-sensitive name of a survey your account owns
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_raw_results(survey_name = 'training_diary' )

## End(Not run)
```

formr_recognise *Recognise data types based on item table*

Description

Once you've retrieved an item table using `formr_items()` you can use this function to correctly type your variables based on the item table (e.g. formr free text types will be character, but select_add_one will be factor, dates are also typed as Date, datetimes as POSIXct).

Usage

```
formr_recognise(
  survey_name = NULL,
  item_list = formr_items(survey_name, host = host),
  results = formr_raw_results(survey_name, host = host),
  host = formr_last_host()
)
```

Arguments

`survey_name`	case-sensitive name of a survey your account owns
`item_list`	an item_list, will be auto-retrieved based on survey_name if omitted
`results`	survey results, will be auto-retrieved based on survey_name if omitted
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
results = jsonlite::fromJSON(txt =
system.file('extdata/gods_example_results.json', package = 'formr', mustWork = TRUE))
class(results$created)
items = formr_items(path =
system.file('extdata/gods_example_items.json', package = 'formr', mustWork = TRUE))
results = formr_recognise(item_list = items, results = results)
class(results$created)
```

formr_render *render text for formr*

Description

Render text

Usage

```
formr_render(text, self_contained = FALSE, ...)
```

Arguments

`text`	that will be written to a tmp file and used as the input argument
`self_contained`	passed to markdown_custom_options
`...`	all other arguments passed to `rmarkdown::render()`

`formr_render_commonmark` *render inline text for formr*

Description

Render text

Usage

```
formr_render_commonmark(text)
```

Arguments

`text`	that will be passed to knitr

Examples

```
formr_render_commonmark("There are only `r sample(2:3, 1)` types of people.")
```

`formr_results` *Download processed, aggregated results from formr*

Description

After connecting to formr using `formr_connect()` you can download data and process it. This approach calls the following functions in the right sequence: `formr_raw_results()` `formr_items()`, `formr_item_displays()` and `formr_post_process_results()`. So, results are downloaded, metadata on items (labels etc.) is added, normal and missing values are labelled. In the end, items like bfi_extra_3R are reversed in place (maintaining labels but changing underlying numbers), and scales are aggregated (bfi_extra_1, bfi_extra_2, bfi_extra_3R become bfi_extra)

Usage

```
formr_results(survey_name, host = formr_last_host(), ...)
```

Arguments

`survey_name`	case-sensitive name of a survey your account owns
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org
`...`	passed to `formr_post_process_results()`

Examples

```
## Not run:
formr_results(survey_name = 'training_diary' )

## End(Not run)
```

formr_reverse *Reverse items based on item table or a fallback_max*

Description

Example: If your data contains Extraversion_1, Extraversion_2R and Extraversion_3, there will be two new variables in the result: Extraversion_2 (reversed to align with _1 and _2) and Extraversion, the mean score of the three. If you supply an item table, the maximum possible answer to the item will be used to reverse it. If you don't, the maximum actually given answer or the fallback_max argument will be used to reverse it. It's faster to do this without an item table, but this can lead to problems, if you mis-specify the fallback max or the highest possible value does not occur in the data.

Usage

```
formr_reverse(results, item_list = NULL, fallback_max = 5)
```

Arguments

`results`	survey results
`item_list`	an item_list, defaults to NULL
`fallback_max`	defaults to 5 - if the item_list is set to null, we will use this to reverse

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
icar_items = formr_items(survey_name='ICAR',host = 'http://localhost:8888/formr/')
# get some simulated data and aggregate it
sim_results = formr_simulate_from_items(icar_items)
reversed_items = formr_reverse(item_list = icar_items, results = sim_results)

## End(Not run)
results = jsonlite::fromJSON(txt =
 system.file('extdata/gods_example_results.json', package = 'formr', mustWork = TRUE))
items = formr_items(path =
 system.file('extdata/gods_example_items.json', package = 'formr', mustWork = TRUE))
formr_reverse(results, items)
```

`formr_shuffled` *Download random groups*

Description

formr has a specific module for randomisation. After connecting using `formr_connect()` you can download the assigned random groups and merge them with your data.

Usage

```
formr_shuffled(run_name, host = formr_last_host())
```

Arguments

`run_name`	case-sensitive name of the run in which you randomised participants
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
formr_shuffled(run_name = 'different_drills' )

## End(Not run)
```

`formr_simulate_from_items` *Simulate data based on item table*

Description

Once you've retrieved an item table using `formr_items()` you can use this function to sample data from the possible choices. At the moment random data is only generated for choice-type items and numeric ones, as these are most likely to enter data analysis. Does not yet handle dates, times, text, locations, colors

Usage

```
formr_simulate_from_items(item_list, n = 300)
```

Arguments

`item_list`	the result of a call to `formr_connect()`
`n`	defaults to 300

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
sim = formr_simulate_from_items(item_list = formr_items('training_diary'), n = 100)
summary(lm(pushups ~ pullups, data = sim))

## End(Not run)
items = formr_items(path =
system.file('extdata/gods_example_items.json', package = 'formr', mustWork = TRUE))
fakedata = formr_simulate_from_items(items, n = 20)
fakedata[1:2,]
```

`formr_store_keys` *Store keys in keyring*

Description

Store keys in the system keyring/keychain instead of plaintext.

Usage

```
formr_store_keys(account_name)
```

Arguments

`account_name`	a shorthand for the account you're using

Examples

```
## Not run:
formr_store_keys("formr_diary_study_account")

## End(Not run)
```

`formr_upload_items` *Upload new item table*

Description

To automatically create surveys using formr, you can upload survey item tables from R. Only file uploads are available. The file name determines the survey name. Updating existing surveys is not implemented and not recommended (because of the sanity checks we require to prevent data deletion).

Usage

```
formr_upload_items(survey_file_path, host = formr_last_host())
```

Arguments

`survey_file_path`	the path to an item table in csv/json/xlsx etc.
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
items <- system.file('extdata/gods_example_items.json', package = 'formr',
mustWork = TRUE)
formr_upload_items(items)

## End(Not run)
```

`formr_user_detail` *Download random groups*

Description

formr collects information about users' progression through the run After connecting using `formr_connect()` you can download a table showing their progression through the run.

Usage

```
formr_user_detail(run_name, host = formr_last_host())
```

Arguments

`run_name`	case-sensitive name of the run in which you randomised participants
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
formr_user_detail(run_name = 'different_drills' )

## End(Not run)
```

formr_user_overview *Download random groups*

Description

formr collects information about users' progression through the run After connecting using `formr_connect()` you can download a table showing where they are in the run.

Usage

```
formr_user_overview(run_name, host = formr_last_host())
```

Arguments

`run_name`	case-sensitive name of the run in which you randomised participants
`host`	defaults to `formr_last_host()`, which defaults to https://formr.org

Examples

```
## Not run:
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
formr_user_overview(run_name = 'different_drills' )

## End(Not run)
```

get_opencpu_rds *pass in the url to the RDS representation of a openCPU session object, get the object*

Description

useful to programmatically access openCPU session object stored in character variables etc.

Usage

```
get_opencpu_rds(session_url, local = TRUE)
```

Arguments

`session_url`	the session url, e.g. https://public.opencpu.org/ocpu/tmp/x02a93ec/R/.val/rds
`local`	defaults to FALSE, if true, will assume that the session is not on another server, and do some not-very-smart substitution to load it via the file system instead of HTTP/HTTPS

Examples

```
## Not run:
get_opencpu_rds('https://public.opencpu.org/ocpu/tmp/x02a93ec/R/.val/rds')

## End(Not run)
```

`ifelsena` *Like* `ifelse()`*, but allows you to assign a third value to missings.*

Description

Deprecated. Please use `dplyr::if_else()` in the future. Defaults to assigning the "no" value to missing values as well. Often missings encapsulate some sort of meaning for the variable you're trying to define.

Usage

```
ifelsena(test, yes, no, missing = no)
```

Arguments

`test`	passed to ifelse
`yes`	passed to ifelse
`no`	passed to ifelse
`missing`	defaults to the value for no

Examples

```
## Not run:
data(beavers)
beaver1$activ[1:10] = NA
beaver1$hyperactive = ifelse(beaver1$activ > 1, 1, 0)
table(beaver1$hyperactive)
beaver1$hyperactive = ifelsena(beaver1$activ > 1, 1, 0)
table(beaver1$hyperactive)

## End(Not run)
```

`if_na` *Replace NA values with something else*

Description

Often, you want to substitute missing values with some implicit known value (e.g. if the question on number of sexual partners was skipped for sexually inactive people, you know the missing should turn into zero)

Usage

```
if_na(x, missing)
```

Arguments

`x`	the variable
`missing`	What to replace missing values with

Examples

```
number_of_sex_partners <- c(1, 3, 5, 10, NA, 29)
if_na(number_of_sex_partners, 0)
```

`if_na_null` *This function makes sure you know what to expect when evaluating uncertain results in an if-clause. In most cases, you should not use this function, because it can lump a lot of very different cases together, but it may have some use for fool-proofing certain if-clauses on formr.org, where a field in a survey may either not exist, be missing or have a value to check.*

Description

This function makes sure you know what to expect when evaluating uncertain results in an if-clause. In most cases, you should not use this function, because it can lump a lot of very different cases together, but it may have some use for fool-proofing certain if-clauses on formr.org, where a field in a survey may either not exist, be missing or have a value to check.

Usage

```
if_na_null(test, na = FALSE, null = FALSE)
```

Arguments

`test`	condition. can only have length 0 or length 1
`na`	returned if the condition has a missing value
`null`	passed to ifelse

Examples

```
testdf = data.frame(test1 = 1, test2 = NA)
if ( if_na_null(testdf$test1 == 1) ) { print("go on") }
if ( if_na_null(testdf$test2 == 1) ) { print("not shown") }
if ( if_na_null(testdf$test3 == 1) ) { print("not shown") }
tryCatch({ if ( if_na_null(testdf2$test1 == 1) ) { print("causes error") } },
   error = function(e) { warning(e) })
```

`in_time_window` *checks whether the current time is in a certain time window*

Description

supply min,max as POSIXct

Usage

```
in_time_window(min, max)
```

Arguments

`min`	POSIXct < max
`max`	POSIXct > min

Examples

```
in_time_window(Sys.time() - 1, Sys.time() + 1)
```

`item` *get item from survey attribute*

Description

Shortcut for attributes(survey$item_name)$item. Fails with a warning.

Usage

```
item(survey, item_name)
```

Arguments

`survey`	survey with item_list attribute
`item_name`	item name

Examples

```
example(formr_post_process_results)
item(processed_results, "BFIK_extra_4")
```

`items` *get item list from survey attributes*

Description

get item list from survey attributes

Usage

```
items(survey)
```

Arguments

`survey`	survey with item_list attribute

Examples

```
example(formr_post_process_results)
items(processed_results)[[1]]
```

`knit_prefixed` *knit prefixed*

Description

Knit using knitr, but prefix file name to figure and cache folder (to knit in parallel on e.g. a cluster)

Usage

```
knit_prefixed(input, ...)
```

Arguments

`input`	input document
`...`	all arguments passed to `knitr::knit()`

`last` *Gives the last non-missing element*

Description

Just a simple shorthand to get the last, non-missing argument per default. Can give more than one element and can include missing elements. The inverse of `first()`.

Usage

```
last(x, n = 1, na.rm = TRUE)
```

Arguments

`x`	vector of which you want the last element
`n`	number of elements to take from the end
`na.rm`	whether to remove missings first, defaults to TRUE

Examples

```
last( c(1:10,NA) )
last( c(1:10,NA), 2, TRUE )
```

`loadRDS` *loads an RDS object, assigns it to an object of the base-filename*

Description

`saveRDS()` saves an object to a file, so unlike `save()` and `load()` you can assign the loaded object to a new variable using `readRDS()`. However, sometimes it may be more convenient to assign the object in the RDS file to an object of the same name as the file. This is what `loadRDS()` does. It extracts the filename using `basename()` and `tools::file_path_sans_ext()`

Usage

```
loadRDS(file, refhook = NULL, overwrite = FALSE)
```

Arguments

`file`	path to file
`refhook`	passed to readRDS
`overwrite`	whether to overwrite an existing object of the same name. defaults to false.

Examples

```
## Not run:
loadRDS(file = '~/Models/Spouses.rds') # assigns object contained in file to variable 'Spouses'

## End(Not run)
```

ls_by_class *get functions in the environment by their class. Useful to find e.g. all regression models you've stored in interactive programming.*

Description

get functions in the environment by their class. Useful to find e.g. all regression models you've stored in interactive programming.

Usage

```
ls_by_class(classes, envir = parent.frame(), top_class_only = FALSE, ...)
```

Arguments

`classes`	objects should have one of these classes
`envir`	defaults to looking in the calling environment of this function, passed to ls
`top_class_only`	defaults to FALSE. If false, also returns objects inheriting from one of the specified classes.
`...`	passed to ls

Examples

```
data(ChickWeight)
chickweight.m1 <- glm(weight ~ Time + Diet, family = gaussian, data = ChickWeight)
ls_by_class('lm')
c('chickweight.m1') %in% ls_by_class('lm')
c('chickweight.m1') %in% ls_by_class('lm', top_class_only = TRUE)
```

markdown_custom_options *custom markdown options for rmarkdown's pandoc*

Description

custom markdown options for rmarkdown's pandoc

Usage

```
markdown_custom_options(
  add_to_format = c("+autolink_bare_uris", "+ascii_identifiers",
    "+tex_math_single_backslash", "-implicit_figures"),
  fragment.only = FALSE,
  section_divs = TRUE,
  break_on_error = FALSE,
  ...
)
```

Arguments

`add_to_format`	add these arguments to the default specification
`fragment.only`	whether to get only a html fragment
`section_divs`	whether to disable –section-divs (headings generate section including everything up to the next same-or-higher-level heading)
`break_on_error`	should an error in the R code execution interrupt the rendering or should rendering continue, defaults to FALSE
`...`	all other arguments passed to `rmarkdown::html_document()` Custom rmarkdown input format options based on the standard `rmarkdown::html_document()`, but with options that you can specify. Find the format options here in the pandoc documentation: http://johnmacfarlane.net/pandoc/demo/example9/pandocs-markdown.html Pandoc's enhanced version of markdown includes syntax for footnotes, tables, flexible ordered lists, definition lists, fenced code blocks, superscript, subscript, strikeout, title blocks, automatic tables of contents, embedded LaTeX math, citations, and markdown inside HTML block elements or spoken in options: +escaped_line_breaks, +header_attributes, +yaml_metadata_block, +auto_identifiers, +implicit_header_references, +blank_before_blockquote, +fenced_code_blocks, +fenced_code_attributes, +line_blocks, +definition_lists, +startnum, +fancy_lists, +pipe_tables, +pandoc_title_block, +intraword_underscores, +strikeout, +superscript, +subscript, +tex_math_dollars, +raw_html, +markdown_in_html_blocks, +implicit_figures, +footnotes, +inline_notes, +citations. The current default rmarkdown additions to Pandoc's enhanced markdown are: +autolink_bare_uris, +ascii_identifiers, +tex_math_single_backslash, -implicit_figures.

markdown_github *github_markdown for rmarkdown*

Description

Custom template with github-flavoured markdown based on the standard `rmarkdown::html_document()`. Adds +pipe_tables, +raw_html, +tex_math_single_backslash, +fenced_code_blocks, +auto_identifiers, +ascii_identifiers, +backtick_code_blocks, +autolink_bare_uris, +intraword_underscores, +strikeout, +hard_line_breaks over markdown_strict. A number of pandoc features are disabled (see `markdown_custom_options()`), but +yaml_metadata_block is re-enabled, so that it is possible to specify this output function using YAML.

Usage

```
markdown_github(fragment.only = FALSE, break_on_error = FALSE, ...)
```

Arguments

`fragment.only`	whether to get only a html fragment
`break_on_error`	should an error in the R code execution interrupt the rendering or should rendering continue, defaults to FALSE
`...`	all other arguments passed to `rmarkdown::html_document()`

markdown_hard_line_breaks *hard line breaks*

Description

Custom rmarkdown template based on the standard `rmarkdown::html_document()`, but with hard line breaks. Will add the pandoc '+hard_line_breaks' argument if the origin format is markdown.

Usage

```
markdown_hard_line_breaks(...)
```

Arguments

`...`	all other arguments passed to `rmarkdown::html_document()`

`miss_frac` *percentage of missings for each variable in a data.frame*

Description

This functions simply reports the number of missings as the percentage of the maximum number of rows. It also works on single variables.

Usage

```
miss_frac(df, vars = 1:NCOL(df))
```

Arguments

`df`	data.frame or variable
`vars`	subset of variables, defaults to all

Examples

```
fruits = c('apple', 'banana', NA, 'pear', 'pinapple', NA)
pets = c('cat', 'dog', 'anteater', NA, NA, NA)
favorites = data.frame(fruits, pets)
miss_frac(favorites)
miss_frac(favorites$fruits)
miss_frac(favorites, 2)
```

`next_day` *checks whether a new day has broken (date has increased by at least one day)*

Description

a simple utility functions to avoid that looped Skip Backwards/Skip Forwards in formr are true repeatedly.

Usage

```
next_day(date = NULL)
```

Arguments

`date`	defaults to .formr$last_action_date, a hidden variable that is automatically set by formr.org. Will be coerced to POSIXct.

Examples

```
next_day(Sys.time())
```

`n_missing` *Returns the number of missings in a variable or dataset. If missings are an explicit level in a factor variable, this function defaults to reporting them anyway.*

Description

Returns the number of missings in a variable or dataset. If missings are an explicit level in a factor variable, this function defaults to reporting them anyway.

Usage

```
n_missing(x, exclude = NA)
```

Arguments

`x`	variable
`exclude`	only needed for factors. defaults to NA (count level=missing as missing), setting to 0 allows you to count level=missing as nonmissing

Examples

```
data(beavers)
beaver1$activ[1:10] = NA
n_missing(beaver1$activ)
beaver1$activ = factor(beaver1$activ, exclude = NULL)
sum(is.na(beaver1$activ))
n_missing(beaver1$activ)
n_missing(beaver1$activ, exclude = NULL)
```

`n_nonmissing` *Returns the number of nonmissings in a variable or dataset. If missings are an explicit level in a factor variable, this function defaults to excluding them anyway.*

Description

Returns the number of nonmissings in a variable or dataset. If missings are an explicit level in a factor variable, this function defaults to excluding them anyway.

Usage

```
n_nonmissing(x, exclude = NA)
```

Arguments

`x`	variable
`exclude`	only needed for factors. defaults to NA (count level=missing as missing), setting to 0 allows you to count level=missing as nonmissing

Examples

```
data(beavers)
beaver1$activ[1:10] = NA
n_nonmissing(beaver1$activ)
beaver1$activ = factor(beaver1$activ, exclude = NULL)
sum(!is.na(beaver1$activ))
n_nonmissing(beaver1$activ)
n_nonmissing(beaver1$activ, exclude = NULL)
```

`paste.knit_asis` *paste.knit_asis*

Description

Helper function for knit_asis objects, useful when e.g. `asis_knit_child()` was used in a loop.

Usage

```
paste.knit_asis(..., sep = "\n\n\n", collapse = "\n\n\n")
```

Arguments

`...`	passed to `paste()`
`sep`	defaults to two empty lines, passed to `paste()`
`collapse`	defaults to two empty lines, passed to `paste()`

Details

Works like `paste()` with both the sep and the collapse argument set to two empty lines

Examples

```
paste.knit_asis("# Headline 1", "## Headline 2")
```

print.knit_asis *Print new lines in knit_asis outputs*

Description

Print new lines in knit_asis outputs

Usage

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

Arguments

x	the knit_asis object
...	ignored

props *proportions table*

Description

quick and easy function to show proportions of values of a variable, defaults to including missings

Usage

```
props(..., exclude = NULL)
```

Arguments

...	passed to crosstabs
exclude	defaults to NULL (i.e. includes NA)

Examples

```
x = NA
props(~ x)
```

qplot_on_bar *Plot normed values as a barchart*

Description

Pass in a data.frame with z-standardised values (x - Mean)/SD, and variable names, get a bar chart. Getting your data.frame into this shape probably will mean using tidyr and dplyr If the data.frame has an se column or ymax/ymin columns, these will be displayed on top of the bars and the bars will become transparent.

Usage

```
qplot_on_bar(
  normed_data,
  ylab = "Your value",
  xlab = "Trait",
  title = "",
  y_ticks = c("--", "-", "0", "+", "++")
)
```

Arguments

normed_data	a dataset with a value column containing z-standardised value and a variable column containing labels for those values
ylab	Y-axis label, defaults to "Percentage of other people with this value"
xlab	X-axis label, empty by default, useful for labeling the plotted trait
title	Plot title
y_ticks	the ticks labels for -2,1,0,1 and 2 SDs around the mean, default to minuses, pluses and the average sign

Examples

```
normed_data = data.frame(variable = c("Extraversion","Openness",
"Agreeableness","Neuroticism","Conscientiousness"),
value = c(-3,1,-1,0.5,2)) # standardise value
qplot_on_bar(normed_data, title = "Your personality")
normed_data = data.frame(variable = c("Extraversion","Openness",
"Agreeableness","Neuroticism","Conscientiousness"),
value = c(-3,1,-1,0.5,2), se = c(0.2,0.3,0.2,0.25,0.4)) # standardise value
qplot_on_bar(normed_data, title = "Your personality")
```

qplot_on_normal *Plot a normed value on the standard normal*

Description

Pass in a z-standardised value (x - Mean)/SD, get a standard normal distribution.

Usage

```
qplot_on_normal(
  normed_value,
  ylab = "Percentage of other people with this value",
  xlab = "",
  colour = "blue",
  x_ticks = c("--", "-", "0", "+", "++")
)
```

Arguments

normed_value	a z-standardised value
ylab	Y-axis label, defaults to "Percentage of other people with this value"
xlab	X-axis label, empty by default, useful for labeling the plotted trait
colour	defaults to blue
x_ticks	the ticks labels for -2,1,0,1 and 2 SDs around the mean, default to minuses, pluses and the average sign

Examples

```
normed_value = scale(x = 20, center = 14, scale = 5) # standardise value
qplot_on_normal(normed_value, xlab = "Extraversion")
```

qplot_on_polar *Time-polar plot*

Description

Pass in a data.frame with z-standardised values (x - Mean)/SD, and variable names, get a bar chart. Getting your data.frame into this shape probably will mean using tidyr + dplyr. If the data.frame has an se column or ymax/ymin columns, these will be displayed on top of the bars and the bars will become transparent.

Usage

```
qplot_on_polar(normed_data, ylab = "Your value", title = "")
```

Arguments

`normed_data`	a dataset with a value column containing z-standardised value and a variable column containing labels for those values
`ylab`	Y-axis label, defaults to "Percentage of other people with this value"
`title`	Plot title

Examples

```
weekdays = c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday")
normed_data = data.frame(variable = factor(weekdays, weekdays),
 value = c(0,1,0.2,0.5,1.5,2,1)) # standardise value
qplot_on_polar(normed_data, title = "Your alcohol consumption across the week")
normed_data = data.frame(variable = factor(1:24,1:24),
 value = 3+rnorm(24), se = rep(0.2,24)) # standardise value
qplot_on_polar(normed_data, title = "Your mood around the clock")
```

`random_date_in_range` *Random date in range*

Description

taken from Dirk Eddelbuettel's answer here http://stackoverflow.com/a/14721124/263054

Usage

```
random_date_in_range(N, lower = "2012/01/01", upper = "2012/12/31")
```

Arguments

`N`	desired number of random dates
`lower`	lower limit
`upper`	upper limit

`render_text` *render text*

Description

Render text

Usage

```
render_text(text, ...)
```

Arguments

`text`	that will be written to a tmp file and used as the input argument
`...`	all other arguments passed to `rmarkdown::render()`

`rescue_attributes` *Rescue lost attributes*

Description

Taken from codebook You can use this function if some of your items have lost their attributes during wrangling Variables have to have the same name (Duh) and no attributes should be overwritten. But use with care. Similar to `labelled::copy_labels()`.

Usage

```
rescue_attributes(df_no_attributes, df_with_attributes)
```

Arguments

`df_no_attributes`	the data frame with missing attributes
`df_with_attributes`	the data frame from which you want to restore attributes

`reverse_labelled_values` *Reverse labelled values*

Description

Taken from codebook package reverse the underlying values for a numeric `haven::labelled()` vector while keeping the labels correct

Usage

```
reverse_labelled_values(x)
```

Arguments

`x`	a labelled vector

Value

return the labelled vector with the underlying values having been reversed

Examples

```
x <- haven::labelled(rep(1:3, each = 3), c(Bad = 1, Good = 5))
x
reverse_labelled_values(x)
```

text_message_clickatell

Send text message via Clickatell

Description

Connects to Clickatell using your token and sends a text message.

Usage

```
text_message_clickatell(To, Body, Token, return_result = F)
```

Arguments

`To`	the number you're texting to (usually without zeroes at the beginning)
`Body`	the text message body/text
`Token`	your Clickatell token
`return_result`	whether to return simply TRUE/FALSE on success/failure or the whole result

Examples

```
## Not run:
text_message_clickatell(
 To = '492222',
 Body = 'Hello friend',
 Token = 'Tokentokentoken')

## End(Not run)
```

text_message_massenversand

Send text message via Massenversand.de

Description

Connects to Massenversand.de using your token and sends a text message.

Usage

```
text_message_massenversand(
  To,
  From,
  Body,
  id,
  pw,
  time = "0",
  msgtype = "t",
  tarif = "OA",
  test = "0",
  return_result = F
)
```

Arguments

`To`	the number you're texting to (usually without zeroes at the beginning)
`From`	the number you're texting from
`Body`	the text message body/text
`id`	your Massenversand ID
`pw`	your Massenversand password
`time`	see provider API (defaults to immediate sending)
`msgtype`	see provider API
`tarif`	see provider API
`test`	see provider API
`return_result`	whether to return simply TRUE/FALSE on success/failure or the whole result

Examples

```
## Not run:
text_message_massenversand(
 To = '492222',
 From = '15005000',
 Body = 'Hello friend',
 id = 'ID',
 pw = 'Tokentokentoken')

## End(Not run)
```

`text_message_twilio` *Send text message via Twilio*

Description

Connects to Twilio using your token and sends a text message.

Usage

```
text_message_twilio(To, From, Body, Account, Token, return_result = F)
```

Arguments

`To`	the number you're texting to (usually without zeroes at the beginning)
`From`	the number you're texting from
`Body`	the text message body/text
`Account`	your Twilio account ID
`Token`	your Twili token
`return_result`	whether to return simply TRUE/FALSE on success/failure or the whole result

Examples

```
text_message_twilio(
 To = '492222',
 From = '15005000',
 Body = 'Hello friend',
 Account = 'ID', Token = 'Tokentokentoken')
```

`time_passed` *checks how much time has passed relative to the user's last action*

Description

checks how much time has passed. You can choose the unit. Implemented via `lubridate::dseconds()`, not periods, i.e. a minute has 60 seconds, an hour 60 minutes, a day 24 hours. Months and years are not well-defined durations, but we offer them anyway for convenience. Returns true or false.

Usage

```
time_passed(
  years = 0,
  months = 0,
  weeks = 0,
  days = 0,
  hours = 0,
  minutes = 0,
  seconds = 0,
  time = NULL
)
```

Arguments

`years`	365 days
`months`	30 days
`weeks`	7 days
`days`	24 hours

`hours`	60 minutes
`minutes`	60 seconds
`seconds`	argument to `lubridate::dseconds()`
`time`	defaults to .formr$last_action_time, a hidden variable that is automatically set by formr.org

Examples

```
time_passed(hours = 7, time = Sys.time())
```

`word_document` *word_document from rmarkdown, but has an added option not to break on error*

Description

Exactly like `rmarkdown::word_document()`, but with one added argument

Usage

```
word_document(..., break_on_error = FALSE)
```

Arguments

`...`	all other arguments passed to `rmarkdown::word_document()`
`break_on_error`	should an error in the R code execution interrupt the rendering or should rendering continue, defaults to FALSE

`%begins_with%` *check whether a character string begins with a string*

Description

Escapes any special RegExp characters in the search term. A way to check whether the search term (e.g. a variable name) is the beginning. Just a simple shorthand so that inexperienced R users won't have to use somewhat complex functions such as `grepl()` and `stringr::str_detect()`. You can also use \%starts_with\%.

Usage

```
haystack %begins_with% needle
```

Arguments

`haystack`	string in which you search
`needle`	string to search for

Examples

```
"1, 3, 4" %begins_with% "1" # TRUE
"1, 3, 4" %begins_with% 1 # unlike str_detect casts all needles as characters
"1, 3, 4" %begins_with% "." # FALSE
```

`%contains%` *check whether a character string contains another*

Description

Just a simple shorthand so that inexperienced R users don't have to use somewhat complex functions such as `grepl()` and `stringr::str_detect()` with non-default arguments (e.g. fixed params).

Usage

```
haystack %contains% needle
```

Arguments

`haystack`	string in which you search
`needle`	string to search for

Examples

```
"1, 2, 3, 4, you" %contains% "you"
"1, 2, 3, 4, you" %contains% 1 # unlike str_detect casts all needles as characters
"1, 2, 3, 4, you" %contains% 343
```

`%contains_word%` *check whether a character string contains another as a word*

Description

Looks for a string appearing on its own. This is needed e.g. when checking whether the replies to a mmc item, stored as a comma-separated list from 1 to 12 contain option 1 - you wouldn't want to get a hit for 11 and 12. Only works for search terms containing alphanumeric characters. Just a simple shorthand so that inexperienced R users don't have to use somewhat complex functions such as `grepl()` and `stringr::str_detect()`.

Usage

```
haystack %contains_word% needle
```

Arguments

`haystack`	string in which you search
`needle`	string to search for

Examples

```
"1, 3, 4" %contains_word% "1" # TRUE
"1, 3, 4" %contains_word% 1 # TRUE unlike str_detect casts all needles as characters
"12, 14, 17" %contains_word% "1" # FALSE even though 12 contains 1
```

`%ends_with%` *check whether a character string ends with a string*

Description

Escapes any special RegExp characters in the search term. A way to check whether the search term (e.g. a variable name) is the ending. Just a simple shorthand so that inexperienced R users don't have to use somewhat complex functions such as `grepl()` and `stringr::str_detect()`.

Usage

```
haystack %ends_with% needle
```

Arguments

`haystack`	string in which you search
`needle`	string to search for

Examples

```
"1, 3, 4" %ends_with% "4" # TRUE
"1, 3, 4" %ends_with% 4 # unlike str_detect casts all needles as characters
"1, 3, 4" %ends_with% "." # FALSE
```

Index