

# Package ‘formr’

June 16, 2026

**Title** Companion R Package for the 'formr' Survey Framework

**Description** Serves as a companion toolkit for the 'formr' survey framework (<<https://rforms.org>>). The package acts as a bridge between a 'formr' server and a local R environment. Key features include an API client for fetching, type-casting, and automatically scoring data; a project management workflow for syncing study assets (surveys, CSS) for local editing; and functions for use within 'formr' runs to generate dynamic, personalized feedback plots and to simplify survey logic.

**Version** 1.1.0

**Depends** R (>= 4.1.0)

**Imports** stats, methods, tools, utils, dplyr, ggplot2, scales, haven, stringr, tidyr, knitr, httr, curl, jsonlite, lubridate, rmarkdown, otp, keyring, purrr, readr, rlang

**Suggests** testthat (>= 3.0.0), vcr, mockery, withr, rstudioapi, plotly, commonmark

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**URL** <https://rubenarslan.github.io/formr/>,  
<https://github.com/rubenarslan/formr>

**BugReports** <https://github.com/rubenarslan/formr/issues>

**Config/roxygen2/version** 8.0.0

**RoxygenNote** 8.0.0

**NeedsCompilation** no

**Author** Ruben Arslan [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-6670-5658>>),  
Tim B Seidel [aut, ctb]

**Maintainer** Ruben Arslan <rubenarslan@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-06-16 20:10:13 UTC

## Contents

.formr . . . . .	4
aggregate_and_document_scale . . . . .	5
as.data.frame.formr_api_run_structure . . . . .	6
as.data.frame.formr_item_list . . . . .	6
asis_knit_child . . . . .	7
choice_labels_for_values . . . . .	8
current . . . . .	9
email_image . . . . .	10
expired . . . . .	10
feedback_chunk . . . . .	11
finished . . . . .	12
first . . . . .	12
formr_aggregate . . . . .	13
formr_api_aggregate . . . . .	14
formr_api_authenticate . . . . .	15
formr_api_backup_run . . . . .	16
formr_api_create_run . . . . .	16
formr_api_create_session . . . . .	17
formr_api_delete_all_files . . . . .	18
formr_api_delete_file . . . . .	18
formr_api_delete_run . . . . .	19
formr_api_delete_survey . . . . .	19
formr_api_fetch_results . . . . .	20
formr_api_files . . . . .	21
formr_api_is_authenticated . . . . .	21
formr_api_logout . . . . .	22
formr_api_pull_project . . . . .	22
formr_api_push_project . . . . .	23
formr_api_recognise . . . . .	24
formr_api_results . . . . .	24
formr_api_reverse . . . . .	25
formr_api_runs . . . . .	26
formr_api_run_settings . . . . .	26
formr_api_run_structure . . . . .	27
formr_api_session . . . . .	27
formr_api_sessions . . . . .	28
formr_api_session_action . . . . .	29
formr_api_surveys . . . . .	29
formr_api_survey_structure . . . . .	30
formr_api_token_expiry . . . . .	30
formr_api_unit_sessions . . . . .	31

formr_api_upload_file . . . . .	32
formr_api_upload_survey . . . . .	33
formr_backup_files . . . . .	33
formr_backup_study . . . . .	34
formr_backup_surveys . . . . .	35
formr_connect . . . . .	36
formr_default_dir . . . . .	37
formr_disconnect . . . . .	37
formr_inline_render . . . . .	38
formr_items . . . . .	38
formr_item_displays . . . . .	39
formr_knit . . . . .	40
formr_last_host . . . . .	40
formr_overview_sankey . . . . .	41
formr_post_process_results . . . . .	42
formr_raw_results . . . . .	43
formr_recognise . . . . .	44
formr_render . . . . .	45
formr_render_commonmark . . . . .	45
formr_results . . . . .	46
formr_reverse . . . . .	46
formr_run_structure . . . . .	47
formr_shuffled . . . . .	48
formr_simulate_from_items . . . . .	49
formr_store_keys . . . . .	50
formr_uploaded_files . . . . .	51
formr_upload_items . . . . .	52
formr_user_detail . . . . .	52
formr_user_overview . . . . .	53
get_opencpu_rds . . . . .	54
ifelse_na . . . . .	54
if_na . . . . .	55
if_na_null . . . . .	56
in_time_window . . . . .	57
item . . . . .	57
items . . . . .	58
knit_prefixed . . . . .	58
last . . . . .	59
markdown_custom_options . . . . .	59
markdown_github . . . . .	60
markdown_hard_line_breaks . . . . .	61
next_day . . . . .	61
paste.knit_asis . . . . .	62
print.formr_api_run_structure . . . . .	63
print.knit_asis . . . . .	63
qplot_on_bar . . . . .	64
qplot_on_normal . . . . .	65
qplot_on_polar . . . . .	65

random_date_in_range . . . . .	66
render_text . . . . .	67
rescue_attributes . . . . .	67
reverse_labelled_values . . . . .	68
summary.formr_results . . . . .	68
text_message_clickatell . . . . .	69
text_message_massenversand . . . . .	69
text_message_twilio . . . . .	71
time_passed . . . . .	72
word_document . . . . .	73
%begins_with% . . . . .	73
%contains% . . . . .	74
%contains_word% . . . . .	74
%ends_with% . . . . .	75

<b>Index</b>	<b>76</b>
--------------	-----------

---

.formr	<i>Per-request environment populated by rforms.org</i>
--------	--

---

## Description

An environment that the rforms.org server fills with per-request state when R code runs inside an OpenCPU session on a formr study (for example, on a CalculateUnit page or an OverviewScriptPage). Useful fields the server may set:

## Usage

.formr

## Format

An environment.

## Details

- `.formr$run_name` – the name of the current run.
- `.formr$host` – the API host (e.g. `https://api.rforms.org`).
- `.formr$access_token` – a short-lived OAuth token for the request.
- `.formr$last_action_time` / `.formr$last_action_date` – timestamps used by `time_passed()` and the other shorthands.

Several user-facing functions (e.g. `formr_api_authenticate()`, `formr_api_results()`, `formr_overview_sankey()`) default their `host` / `run_name` / `access_token` arguments to these fields, so the same code runs unchanged locally and on a formr study. Outside a formr session the environment is empty.

**Value**

An [environment](#) holding per-request state in its bindings (the `run_name`, `host`, `access_token`, `last_action_time` and `last_action_date` fields listed above). `.formr` is a data object, not a function, so it does not itself return a value; the `rforms.org` server populates these bindings before user code runs and the environment is empty outside a `formr/OpenCPU` session.

---

`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**

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

**Value**

A numeric vector (the aggregated scale score) carrying `scale_item_names` and `label` attributes.

**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_api_run_structure
```

*Convert formr run structure to data.frame*

---

### Description

Convert formr run structure to data.frame

### Usage

```
## S3 method for class 'formr_api_run_structure'  
as.data.frame(x, ...)
```

### Arguments

x	The object.
...	Additional arguments.

### Value

A data.frame with one row per unit and columns position, type, description and details.

---

```
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

**Value**

A data.frame of item metadata, one row per item, with choices collapsed to a comma-separated string.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
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	<i>knit_child as is</i>
-----------------	-------------------------

---

**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 <code>knitr::knit_child()</code>
...	passed to <code>knitr::knit_child()</code>

quiet	passed to <code>knitr::knit_child()</code>
options	defaults to NULL.
envir	passed to <code>knitr::knit_child()</code>

### 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()`

### Value

A length-1 character string of class `knit_asis` (the knitted child document).

### Examples

```
## Not run:
# Not run: requires a knitr session and an external child .Rmd document.
# 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

**Value**

A vector of choice labels mapped onto the supplied item values.

**Examples**

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

---

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
---	--

**Value**

A length-1 vector (the last element of x), keeping x's type.

**Examples**

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

---

email_image	<i>generates valid email cids</i>
-------------	-----------------------------------

---

### 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

### Value

A length-1 character string holding a cid: reference for inline email images, carrying the source path in its link attribute.

### Examples

```
## Not run:
# Not run: meant to run inside a knitr session as the figure upload hook.
library(knitr); library(formr)
opts_knit$set(upload.fun=formr::email_image)

## End(Not run)
```

---

expired	<i>How many surveys were expired?</i>
---------	---------------------------------------

---

### 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"

**Value**

An integer: the count of expired sessions (non-missing values in `survey[[variable]]`).

**Examples**

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

---

feedback_chunk	<i>Text feedback based on groups</i>
----------------	--------------------------------------

---

**Description**

If you pass in a z-standardised value  $(x - \text{Mean})/\text{SD}$ , 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

**Value**

A length-1 character string: the chunk selected for the interval containing `normed_value`.

**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	<i>How many surveys were finished?</i>
----------	--

---

**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"

**Value**

An integer: the count of non-missing values in `survey[[variable]]` (0 when the survey or column is empty).

**Examples**

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

---

first	<i>Gives the first non-missing element</i>
-------	--

---

**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

**Value**

A vector of the same type as `x` holding its first `n` elements (after dropping NAs when `na.rm = TRUE`); an empty vector if none remain.

**Examples**

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

---

formr_aggregate	<i>Aggregate data based on item table</i>
-----------------	---

---

**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**

<code>survey_name</code>	case-sensitive name of a survey your account owns
<code>item_list</code>	an <code>item_list</code> , will be auto-retrieved based on <code>survey_name</code> if omitted
<code>results</code>	survey results, will be auto-retrieved based on <code>survey_name</code> if omitted
<code>host</code>	defaults to <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>
<code>compute_alphas</code>	deprecated, functionality migrated to codebook package
<code>fallback_max</code>	defaults to 5 - if the <code>item_list</code> is set to null, we will use this to reverse
<code>plot_likert</code>	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

... formerly passed to psych::alpha(); ignored now that the reliability/Likert code has moved to the codebook package

**Value**

The results data.frame with one added numeric column per scale (the row mean of its items).

**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\_aggregate    *Aggregate Scales*

---

**Description**

Aggregate Scales

**Usage**

```
formr_api_aggregate(results, item_list, min_items = 2)
```

**Arguments**

results            A data frame/tibble containing the run results.

item\_list          A data frame containing item metadata (names, types, choices).

min\_items         Minimum number of valid items required to calculate a mean (default 2).

**Value**

The results data.frame with one added numeric column per scale (the row mean of its items); scale reliability stored in attributes.

---

`formr_api_authenticate`*Authenticate with formr*

---

## Description

Connects to the API. If no credentials are provided, the auto-pickup chain is: the package's hidden `.formr` env (set automatically when the code runs inside an OpenCPU session on `rforms.org`), then the calling-frame chain (for legacy injectors that wrote bare locals into the wrapper scope), then the keyring.

## Usage

```
formr_api_authenticate(  
  host = "https://rforms.org",  
  client_id = NULL,  
  client_secret = NULL,  
  access_token = NULL,  
  account = NULL,  
  verbose = TRUE  
)
```

## Arguments

<code>host</code>	API Base URL. Defaults to <code>.formr\$host</code> when running on <code>rforms.org</code> , otherwise <code>"https://rforms.org"</code> .
<code>client_id</code>	OAuth Client ID.
<code>client_secret</code>	OAuth Client Secret.
<code>access_token</code>	Direct Access Token.
<code>account</code>	Optional string identifier for multiple accounts on the same host.
<code>verbose</code>	Logical. If TRUE (default), reports success via <code>message()</code> .

## Value

Invisibly NULL; called for its side effect of obtaining and caching an OAuth access token (errors on failure).

---

formr\_api\_backup\_run *Backup a study*

---

### Description

Downloads the full run structure, all survey items, attached files, and results. Saves everything into a structured folder.

### Usage

```
formr_api_backup_run(run_name, dir = NULL, prompt = TRUE, verbose = TRUE)
```

### Arguments

run_name	Name of the run/study.
dir	Directory to write the backup into. Defaults to a sub-folder named after the run inside <code>formr_default_dir()</code> ; set that (or pass dir) since formr never writes to the working directory by default.
prompt	Logical. If TRUE (default), asks for confirmation before overwriting when run interactively; in a non-interactive session it errors instead of proceeding unattended. Pass <code>prompt = FALSE</code> to overwrite without confirmation (e.g. in scripts).
verbose	Logical. If TRUE (default), reports progress via <code>message()</code> .

### Value

Invisibly NULL; called for its side effect of writing the run structure (JSON), surveys, files and results (`results.rds`) into `dir`.

### Examples

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_api_backup_run("my_run", dir = tempdir())

## End(Not run)
```

---

formr\_api\_create\_run *Create a new run*

---

### Description

Creates one or more new runs on the server. Prints a confirmation message with the public link for each.

**Usage**

```
formr_api_create_run(name, verbose = TRUE)
```

**Arguments**

name            A character vector of names for the new runs (must be unique).  
verbose        Logical. If TRUE (default), reports progress via `message()`.

**Value**

Invisibly returns a data frame containing the name and link of the created runs.

---

```
formr_api_create_session  
                          Create Session(s)
```

---

**Description**

Creates one or more sessions. If codes is NULL, one random session is created. If codes is provided, tries to create sessions with those specific codes.

**Usage**

```
formr_api_create_session(  
  run_name,  
  codes = NULL,  
  testing = FALSE,  
  verbose = TRUE  
)
```

**Arguments**

run\_name        Name of the run.  
codes          Character vector of codes. If NULL, creates one random code.  
testing        Logical. Mark these sessions as testing?  
verbose        Logical. If TRUE (default), reports progress via `message()`.

**Value**

Invisibly the API response: a list with the created sessions and, for any that failed, an errors data.frame to inspect.

---

```
formr_api_delete_all_files
```

*Delete ALL files attached to a run*

---

### Description

CAUTION: This will permanently remove every file attached to the specified run. It first fetches the list of existing files, then iterates through them to delete.

### Usage

```
formr_api_delete_all_files(run_name, prompt = TRUE, verbose = TRUE)
```

### Arguments

run_name	Name of the run.
prompt	Logical. If TRUE (default), asks for interactive confirmation before deleting; in a non-interactive session it errors instead of proceeding unattended. Set to FALSE for automated scripts (use with care).
verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

### Value

Invisibly TRUE on success; called to delete all files from the run.

---

```
formr_api_delete_file Delete file(s) from a run
```

---

### Description

Removes file attachment(s) from the run. Accepts a single filename, a vector of filenames, or a local directory path (which will delete files on the server that match the names of the files in the local directory).

### Usage

```
formr_api_delete_file(run_name, file_name, verbose = TRUE)
```

### Arguments

run_name	Name of the run.
file_name	The name of the file(s) to delete (e.g. "image.png"), or a local directory path.
verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

### Value

Invisibly TRUE; called to delete the named file(s) from the run.

---

formr\_api\_delete\_run *Delete a Run*

---

### Description

Permanently deletes a run and all associated data (sessions, results).

### Usage

```
formr_api_delete_run(run_name, prompt = TRUE, verbose = TRUE)
```

### Arguments

run_name	Name of the run to delete.
prompt	Logical. If TRUE (default), asks for interactive confirmation; in a non-interactive session it errors instead of proceeding unattended. Pass prompt = FALSE to delete without confirmation (e.g. in scripts).
verbose	Logical. If TRUE (default), reports progress via <code>message()</code> .

### Value

Invisibly TRUE (single run) or a named logical vector (multiple runs) indicating per-run success; FALSE if the user declines the prompt.

---

formr\_api\_delete\_survey  
*Delete a Survey*

---

### Description

Permanently deletes a survey study. Note: The API may prevent deletion if this survey is currently used in an active run.

### Usage

```
formr_api_delete_survey(survey_name, prompt = TRUE, verbose = TRUE)
```

### Arguments

survey_name	Name of the survey to delete.
prompt	Logical. If TRUE (default), asks for interactive confirmation; in a non-interactive session it errors instead of proceeding unattended. Pass prompt = FALSE to delete without confirmation (e.g. in scripts).
verbose	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

Invisibly TRUE on success; FALSE if the user declines the prompt.

---

`formr_api_fetch_results`

*Lower-level API Result Fetcher*

---

**Description**

Fetches raw results. Advanced users can use this if they want completely raw data without any type coercion or processing.

**Usage**

```
formr_api_fetch_results(  
  run_name = .formr$run_name,  
  surveys = NULL,  
  session_ids = NULL,  
  item_names = NULL,  
  join = FALSE  
)
```

**Arguments**

<code>run_name</code>	Name of the run. Defaults to <code>.formr\$run_name</code> , which is set automatically when the code runs inside an OpenCPU session on <code>rforms.org</code> .
<code>surveys</code>	Optional character vector of survey names to filter by.
<code>session_ids</code>	Optional character vector of session IDs to filter by.
<code>item_names</code>	Optional character vector of item names to filter by.
<code>join</code>	Logical. If TRUE, joins the results into a single data frame.

**Value**

A tibble of survey results, or (when `join = FALSE`) a named list of tibbles, one per survey.

---

formr_api_files	<i>List files attached to a run</i>
-----------------	-------------------------------------

---

**Description**

Returns a data frame of all files uploaded to a specific run, including their public URLs and timestamps.

**Usage**

```
formr_api_files(run_name, verbose = TRUE)
```

**Arguments**

run_name	Name of the run.
verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

**Value**

A data.frame containing: id, name, path, url, created, modified.

---

formr_api_is_authenticated	<i>Check if currently authenticated</i>
----------------------------	---

---

**Description**

Checks if there is a valid, non-expired session. Does NOT verify token validity with the server (use `formr_api_session()` for that).

**Usage**

```
formr_api_is_authenticated()
```

**Value**

TRUE if authenticated and token not expired, FALSE otherwise.

---

formr_api_logout	<i>Revoke Access Token (Logout)</i>
------------------	-------------------------------------

---

**Description**

Invalidates the current access token on the server and clears the local session state.

**Usage**

```
formr_api_logout(verbose = TRUE)
```

**Arguments**

verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .
---------	--

**Value**

Invisibly TRUE on success (or FALSE if there was no active session); called to revoke the access token on the server and clear the local session.

---

formr_api_pull_project	
------------------------	--

*Pull Project from Server Scaffolds folder structure if missing, then overwrites local files with Server state.*

---

**Description**

Pull Project from Server Scaffolds folder structure if missing, then overwrites local files with Server state.

**Usage**

```
formr_api_pull_project(run_name, dir = NULL, prompt = TRUE, verbose = TRUE)
```

**Arguments**

run_name	Name of the run.
dir	Local directory to scaffold and write into. Defaults to <a href="#">formr_default_dir()</a> ; set that (or pass dir) since formr never writes to the working directory by default.
prompt	Logical. If TRUE (default), asks for confirmation before overwriting when run interactively (unless dir is empty); in a non-interactive session it errors instead of overwriting unattended. Pass prompt = FALSE to overwrite without confirmation (e.g. in scripts).
verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

**Value**

Invisibly NULL on success (or FALSE if the user declines the overwrite prompt); called for its side effect of scaffolding `dir` and writing the run's structure, settings, surveys and files from the server.

---

```
formr_api_push_project
```

*Push Project to Server*

---

**Description**

Uploads local project files (surveys, assets, settings) to the formr server. Optionally monitors the directory for subsequent changes (Watcher mode).

**Usage**

```
formr_api_push_project(  
  run_name,  
  dir = NULL,  
  watch = FALSE,  
  background = TRUE,  
  interval = 2,  
  verbose = TRUE  
)
```

**Arguments**

<code>run_name</code>	Name of the run.
<code>dir</code>	Local directory to push from. Defaults to <code>formr_default_dir()</code> ; set that (or pass <code>dir</code> ) since formr never writes to the working directory by default.
<code>watch</code>	Logical. If TRUE, keeps the connection open and uploads changes immediately when files are saved.
<code>background</code>	Logical. If TRUE (default), launches watcher as an RStudio Job.
<code>interval</code>	Seconds between checks (default 2).
<code>verbose</code>	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

Invisibly TRUE when the watcher is launched as a background RStudio job; otherwise invisibly NULL. Called for its side effect of uploading the local project in `dir` to the server (optionally starting a file-watcher).

---

formr\_api\_recognise     *Apply Type Definitions and Labels*

---

**Description**

Apply Type Definitions and Labels

**Usage**

```
formr_api_recognise(item_list, results)
```

**Arguments**

item\_list     A data frame containing item metadata.  
results     A data frame containing the raw results.

**Value**

The results data.frame with item types applied: POSIXct timestamps and choice items as haven::labelled vectors.

---

formr\_api\_results     *Get and Process Run Results*

---

**Description**

This is the main function for scientists. It fetches data from the API, automatically cleans types (dates/numbers), reverses items, computes scales, and joins everything into one dataframe.

**Usage**

```
formr_api_results(  
  run_name = .formr$run_name,  
  ...,  
  compute_scales = TRUE,  
  join = TRUE,  
  remove_test_sessions = TRUE,  
  verbose = TRUE  
)
```

**Arguments**

run_name	Name of the run. Defaults to <code>.formr\$run_name</code> , which is set automatically when the code runs inside an OpenCPU session on <code>rforms.org</code> – so portable run code can omit this argument.
...	Filters passed to API (e.g. <code>surveys = c("Daily", "Intake")</code> , <code>session_ids = "..."</code> ).
compute_scales	Logical. Should scales (e.g. <code>extraversion</code> ) be computed from items (e.g. <code>extra_1</code> , <code>extra_2</code> )?
join	Logical. If TRUE (default), joins all surveys into one wide dataframe.
remove_test_sessions	Logical. Filter out sessions marked as testing?
verbose	Logical. Print progress messages?

**Value**

A processed tibble with class `formr_results`.

---

`formr_api_reverse`      *Reverse Items and Update Labels*

---

**Description**

Reverses numeric items ending in 'R' based on metadata bounds. Critically, it also updates `haven::labelled` attributes so that the text labels point to the new, reversed values.

**Usage**

```
formr_api_reverse(results, item_list)
```

**Arguments**

results	A data frame containing the results.
item_list	A data frame containing item metadata.

**Value**

The results `data.frame` with reverse-keyed items (those ending in R) flipped and their value labels remapped.

---

formr_api_runs	<i>List all runs</i>
----------------	----------------------

---

**Description**

Returns a data frame of all runs accessible to the user, including status flags and timestamps.

**Usage**

```
formr_api_runs()
```

**Value**

A data.frame containing run details: id, name, title, public (bool), cron\_active (bool), locked (bool), created (POSIXct), modified (POSIXct).

---

formr_api_run_settings	<i>Get or Update Run Settings</i>
------------------------	-----------------------------------

---

**Description**

Retrieve the settings for one or more runs as a tidy data frame, or update them by providing a named list of new values.

**Usage**

```
formr_api_run_settings(run_name, settings = NULL, verbose = TRUE)
```

**Arguments**

run_name	Name of the run (or a vector of names).
settings	A list of settings to update (e.g., <code>list(public = 1, locked = TRUE)</code> ). If NULL, returns the current settings.
verbose	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

- If settings is NULL: A data.frame/tibble with details for all requested runs.
- If settings is provided: Invisibly returns TRUE on success.

---

`formr_api_run_structure`*Get or Update Run Structure (Run Units)*

---

**Description**

Export the current run structure as a list (GET) or replace it by importing a JSON file (PUT).

**Usage**

```
formr_api_run_structure(  
    run_name,  
    structure_json_path = NULL,  
    file = NULL,  
    verbose = TRUE  
)
```

**Arguments**

<code>run_name</code>	Name of the run.
<code>structure_json_path</code>	Optional path to a JSON file to IMPORT (PUT) structure. If provided, the function uploads this file to the server.
<code>file</code>	Optional path to save the DOWNLOADED (GET) structure as a .json file. This ensures a perfect 1:1 backup of the server configuration.
<code>verbose</code>	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

- GET (default): A `formr_run_structure` object (list) for inspection.
- GET (file provided): Invisibly returns the file path.
- PUT: Invisibly returns TRUE on success.

---

`formr_api_session`*Get Current API session*

---

**Description**

Returns the current session object or NULL if not authenticated.

**Usage**

```
formr_api_session()
```

**Value**

A list, or NULL if not authenticated:

- `base_url`: parsed URL (http style).
- `token`: the bearer access token.
- `scope`: space-delimited string of scopes granted to this token. `NA_character_` when the auth path couldn't introspect (direct access-token authentication, or older server). `""` means the credential was issued with no scopes — every API call will 403 until the user picks scopes at `admin/account#api`.
- `expires_at`: POSIXct of token expiry (or NULL).

---

formr\_api\_sessions      *List Sessions in a Run*

---

**Description**

Returns a tidy data frame of sessions. Can either list all sessions (with filtering) or fetch specific sessions by their codes.

**Usage**

```
formr_api_sessions(
  run_name,
  session_codes = NULL,
  active = NULL,
  testing = NULL,
  limit = 1000,
  offset = 0,
  verbose = TRUE
)
```

**Arguments**

<code>run_name</code>	Name of the run.
<code>session_codes</code>	Optional. A character vector of session codes to fetch specific details for. If provided, <code>active</code> , <code>limit</code> , and <code>offset</code> are ignored.
<code>active</code>	Filter: TRUE for ongoing, FALSE for finished, NULL for all.
<code>testing</code>	Filter: TRUE for test sessions, FALSE for real users, NULL for all.
<code>limit</code>	Pagination limit (default 1000).
<code>offset</code>	Pagination offset (default 0).
<code>verbose</code>	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

A combined tibble of session states and details.

---

formr\_api\_session\_action  
*Perform Action on Session(s)*

---

**Description**

Controls the flow of one or more sessions.

**Usage**

```
formr_api_session_action(  
  run_name,  
  session_codes,  
  action,  
  position = NULL,  
  verbose = TRUE  
)
```

**Arguments**

run_name	Name of the run.
session_codes	A single code or vector of session codes.
action	One of: "end_external", "toggle_testing", "move_to_position", "execute", "advance".
position	Required only if action is "move_to_position".
verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

**Value**

A logical vector indicating success for each session.

---

formr\_api\_surveys      *List Surveys*

---

**Description**

Returns a list of all surveys owned by the user.

**Usage**

```
formr_api_surveys(name_pattern = NULL, verbose = TRUE)
```

**Arguments**

name_pattern	Optional. Filter surveys by name (partial match).
verbose	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

A tibble of surveys (id, name, created, modified, results\_table).

---

formr\_api\_survey\_structure  
*Get Survey Structure (Items)*

---

**Description**

Retrieves the item table for a survey. Can return a tibble (JSON) or download the original Excel file (XLSX).

**Usage**

```
formr_api_survey_structure(survey_name, format = "json", file_path = NULL)
```

**Arguments**

survey_name	The name of the survey.
format	The format to retrieve: "json" (default) or "xlsx".
file_path	Optional. Required if format is "xlsx".

**Value**

In JSON mode (the default) a tibble of the survey's items; in xlsx mode invisibly the file\_path written.

---

formr\_api\_token\_expiry  
*Get token expiry information*

---

**Description**

Returns information about when the current token expires.

**Usage**

```
formr_api_token_expiry()
```

**Value**

A list with:

- `expires_at`: POSIXct of expiry time (or NULL if unknown)
- `seconds_left`: Seconds until expiry (or NA if unknown)
- `is_expired`: TRUE if token has expired

---

`formr_api_unit_sessions`

*List Per-Unit Sessions in a Run*

---

**Description**

Returns one row per (participant × unit × iteration) for the run — the history view that complements [formr\\_api\\_sessions\(\)](#) (which gives one row per participant with their *current* unit only).

**Usage**

```
formr_api_unit_sessions(
  run_name,
  session_codes = NULL,
  testing = NULL,
  since = NULL,
  limit = 1000,
  offset = 0,
  verbose = TRUE
)
```

**Arguments**

<code>run_name</code>	Name of the run.
<code>session_codes</code>	Optional character vector — restrict to one or more participants' histories.
<code>testing</code>	Filter: TRUE for test sessions only, FALSE for real participants only, NULL for both.
<code>since</code>	Optional ISO 8601 datetime string. Returns only unit sessions whose created is at-or-after this — handy for incremental polling.
<code>limit</code>	Pagination limit (default 1000, max 10000).
<code>offset</code>	Pagination offset (default 0).
<code>verbose</code>	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

**Details**

Use this for trajectory plots (Sankey, alluvial), drop-off analytics, and debugging stuck participants. The rows arrive ordered by (session, created, unit\_session\_id), so `dplyr::group_by(session) |> dplyr::mutate(next_unit = dplyr::lead(unit_description))` gives the edges of a trajectory plot directly.

Special units (OverviewScriptPage, ServiceMessagePage, ReminderEmail) surface with position = NA because they live outside the ordered run flow.

**Value**

A tidy tibble with columns: unit\_session\_id, session, testing, unit\_id, unit\_type, unit\_description, position, iteration, created, expires, ended, expired, result, state.

---

formr\_api\_upload\_file *Upload File(s) to Run*

---

**Description**

Uploads local file(s) to the run. Accepts a single file path, a vector of file paths, or a directory path (which will upload all files within that directory).

**Usage**

```
formr_api_upload_file(run_name, path, verbose = TRUE)
```

**Arguments**

run_name	Name of the run.
path	Local path to the file, a vector of paths, or a directory path.
verbose	Logical. If TRUE (default), reports progress via <code>message()</code> .

**Value**

Invisibly returns a list of server responses.

---

formr\_api\_upload\_survey      *Upload/Update Survey*

---

**Description**

Uploads a survey structure.

**Usage**

```
formr_api_upload_survey(  
  file_path = NULL,  
  google_sheet_url = NULL,  
  verbose = TRUE  
)
```

**Arguments**

file\_path      Path to a local file.  
google\_sheet\_url      Google Sheet URL.  
verbose      Logical. If TRUE (default), reports progress via [message\(\)](#).

**Value**

Invisibly the server response; called to upload or update a survey from a local file or Google Sheet.

---

formr\_backup\_files      *Backup uploaded files from formr*

---

**Description**

After connecting to formr using [formr\\_connect\(\)](#) you can backup uploaded files using this command.

**Usage**

```
formr_backup_files(  
  survey_name,  
  overwrite = FALSE,  
  save_path = NULL,  
  host = formr_last_host()  
)
```

**Arguments**

survey_name	case-sensitive name of a survey your account owns
overwrite	should existing files be overwritten? defaults to FALSE
save_path	directory to write the files into. Defaults to a sub-folder named after the survey inside <code>formr_default_dir()</code> ; set that (or pass <code>save_path</code> ) since formr never writes to the working directory by default.
host	defaults to <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>

**Value**

Invisibly the file list with an updated downloaded field; called to download a survey's user-uploaded files into `save_path`.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_backup_files(survey_name = 'training_diary', save_path = tempdir() )

## End(Not run)
```

---

formr\_backup\_study      *Backup a study*

---

**Description**

Backup a study by downloading all surveys, results, item displays, run shuffle, user overview and user details. This function will save the data in a folder named after the study.

**Usage**

```
formr_backup_study(
  study_name,
  save_path = NULL,
  host = formr_last_host(),
  overwrite = FALSE
)
```

**Arguments**

study_name	case-sensitive name of a study your account owns
save_path	directory to write the backup into. Defaults to a sub-folder named after the study inside <code>formr_default_dir()</code> ; set that (or pass <code>save_path</code> ) since formr never writes to the working directory by default.
host	defaults to <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>
overwrite	should existing files be overwritten?

**Value**

Invisibly NULL; called for its side effect of downloading a whole study (run structure, surveys, files and results) into `save_path`.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_default_dir(tempdir())
formr_backup_study(study_name = 'training_diary' )

## End(Not run)
```

---

formr\_backup\_surveys *Backup surveys*

---

**Description**

Backup surveys by downloading item lists, results, item displays and file lists.

**Usage**

```
formr_backup_surveys(
  survey_names,
  surveys = list(),
  save_path = NULL,
  overwrite = FALSE,
  host = formr_last_host()
)
```

**Arguments**

<code>survey_names</code>	case-sensitive names of surveys your account owns
<code>surveys</code>	a list of survey data (from a run structure), optional
<code>save_path</code>	directory to write the surveys into. Defaults to <code>formr_default_dir()</code> ; set that (or pass <code>save_path</code> ) since formr never writes to the working directory by default.
<code>overwrite</code>	should existing files be overwritten?
<code>host</code>	defaults to <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>

**Value**

Invisibly NULL; called for its side effect of downloading surveys (items, results, item displays and files) into `save_path`.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_backup_surveys(survey_names = 'training_diary', save_path = file.path(tempdir(), 'surveys'))

## End(Not run)
```

---

formr_connect	<i>Connect to formr</i>
---------------	-------------------------

---

**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://rforms.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 <a href="#">formr_last_host()</a> , which defaults to <a href="https://rforms.org">https://rforms.org</a>
keyring	a shorthand for the account you're using

**Value**

Invisibly TRUE on success; called for its side effect of establishing an authenticated cookie session with the formr server (stored in httr's cookie jar).

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_connect(keyring = "formr_diary_study_account" )

## End(Not run)
```

---

formr_default_dir	<i>Get or set the default directory for downloads and backups</i>
-------------------	---

---

### Description

formr's file-writing functions (e.g. `formr_backup_study()`, `formr_api_backup_run()`, `formr_api_pull_project()`) never write to your working directory by default. Instead they default their destination to the value stored here, which is unset (NULL) until you choose one — so nothing is ever written until you opt in. Call this function with a path to set a session-wide default, or without arguments to read the current value. The path is held in memory for the current R session only; nothing is written to disk to persist it. There is no separate reset: the value lasts until you overwrite it with another path or your R session ends.

### Usage

```
formr_default_dir(dir = NULL)
```

### Arguments

<code>dir</code>	a single directory path to use as the default. If NULL (the default) the stored value is returned unchanged. The directory itself is created on first write if it does not yet exist.
------------------	---

### Value

The current default directory as a length-1 character string, or NULL when none has been set.

### Examples

```
formr_default_dir(tempdir())  
formr_default_dir()
```

---

formr_disconnect	<i>Disconnect from formr</i>
------------------	------------------------------

---

### Description

Disconnects from formr if connected.

### Usage

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

### Arguments

<code>host</code>	defaults to <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>
-------------------	--

**Value**

Invisibly TRUE on a successful logout; called to log out and clear the active session.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
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\(\)](#)

**Value**

A length-1 character string of rendered inline HTML.

---

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://rforms.org`  
 path path to local JSON copy of the item table

**Value**

A list of class `formr_item_list` (item metadata per item, named by item name).

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
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://rforms.org`

**Value**

A data.frame (parsed JSON) of item-display records with timing and display counts.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_connect(email = 'you@example.net', password = 'zebrafinch' )
formr_item_displays(survey_name = 'training_diary' )

## End(Not run)
```

---

formr_knit	<i>knit rmarkdown to markdown for formr</i>
------------	---

---

**Description**

Render text

**Usage**

```
formr_knit(text)
```

**Arguments**

text                    rmarkdown that will be knit

**Value**

A length-1 character string of knitted markdown.

---

formr_last_host	<i>Get the last specified host</i>
-----------------	------------------------------------

---

**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://rforms.org>

**Value**

the last specified host

**Examples**

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

---

formr\_overview\_sankey *Render a participant-flow Sankey for an overview script*

---

## Description

Pulls the run's per-unit interaction history via `formr_api_unit_sessions()` and renders a plotly Sankey diagram of how participants are moving through the ordered units.

## Usage

```
formr_overview_sankey(
  run_name = .formr$run_name,
  testing = FALSE,
  orientation = "v",
  min_avg_visits_to_annotate = 1.1
)
```

## Arguments

<code>run_name</code>	Name of the run. Defaults to <code>.formr\$run_name</code> , which <code>rforms.org</code> sets when an <code>OverviewScriptPage</code> renders.
<code>testing</code>	If <code>FALSE</code> (default), only real participants are included. <code>TRUE</code> to include only test sessions, <code>NULL</code> for both.
<code>orientation</code>	Sankey orientation; <code>"v"</code> (default) renders top-to-bottom (readable on narrow admin pages), <code>"h"</code> renders left-to-right.
<code>min_avg_visits_to_annotate</code>	Threshold above which a node's label gets the "avg N visits" suffix. Default 1.1 – slightly above exactly-once so single-pass runs stay clean.

## Details

Sankey diagrams can only draw acyclic graphs, but diary / longitudinal studies revisit the same units across iterations. To stay readable without losing information, the helper collapses each position to a single node (counting only the first time a participant visits it) and surfaces the average per-participant visit count as an "avg N visits" suffix on the node label. Single-pass runs stay clutter-free (no suffix when the average is ~1); diary studies show e.g. "p20: Daily mood (avg 14.2 visits)" so the loop density is visible without drawing it.

Designed to be called from an `OverviewScriptPage` on `rforms.org`, where the server injects the per-token `.formr$access_token / .formr$host / .formr$run_name` environment and `formr_api_authenticate()` picks them up automatically. Outside an `Overview` render, set `run_name` explicitly and call `formr_api_authenticate()` first.

## Value

A plotly Sankey object, or `NULL` with a message when there are no rows to plot. Returning `NULL` lets the caller's knitr chunk gracefully display the message instead of erroring.

---

`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

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

## Value

A `data.frame`/`tibble` with recognised types, reverse-keyed items flipped, scales aggregated, and missing values tagged.

## 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	<i>Download data from formr</i>
-------------------	---------------------------------

---

## 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 <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>

## Value

The survey's results before processing: a data.frame (or the raw parsed list).

## Examples

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_raw_results(survey_name = 'training_diary' )

## End(Not run)
```

---

formr_recognise	<i>Recognise data types based on item table</i>
-----------------	---

---

## 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

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

## Value

The results data.frame with POSIXct timestamps and numeric/factor/haven::labelled columns set according to item types.

## 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	<i>render text for formr</i>
--------------	------------------------------

---

**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()`

**Value**

A length-1 character string: the path to the rendered HTML file.

---

formr_render_commonmark	<i>render inline text for formr</i>
-------------------------	-------------------------------------

---

**Description**

Render text

**Usage**

```
formr_render_commonmark(text)
```

**Arguments**

`text` that will be passed to knitr

**Value**

A length-1 character string of HTML rendered from CommonMark.

**Examples**

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

---

formr_results	<i>Download processed, aggregated results from formr</i>
---------------	--

---

### 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

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

### Value

A tibble of processed, aggregated survey results (the output of `formr_post_process_results()`).

### Examples

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_results(survey_name = 'training_diary' )

## End(Not run)
```

---

formr_reverse	<i>Reverse items based on item table or a fallback_max</i>
---------------	--

---

### 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

**Value**

The results data.frame with reverse-keyed (R-suffixed) items flipped and labelled items' value labels updated.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
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\_run\_structure     *Download run structure from formr*

---

**Description**

After connecting to formr using `formr_connect()` you can download the study/run structure using this command.

**Usage**

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

**Arguments**

run_name	case-sensitive name of a run your account owns
host	defaults to <code>formr_last_host()</code> , which defaults to <code>https://rforms.org</code>

**Value**

A list (parsed JSON) describing the run structure (its units).

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_run_structure(run_name = 'training_diary' )

## End(Not run)
```

---

formr_shuffled	<i>Download random groups</i>
----------------	-------------------------------

---

**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 <a href="#">formr_last_host()</a> , which defaults to <a href="https://rforms.org">https://rforms.org</a>

**Value**

A data.frame (parsed JSON) of random group assignments keyed by session.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
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

<code>item_list</code>	the result of a call to <code>formr_connect()</code>
<code>n</code>	defaults to 300

## Value

A data.frame of simulated survey data (columns `id`, `created`, `modified`, `ended`, plus sampled values for each item).

## Examples

```
## Not run:  
# Not run: needs a live formr server and an authenticated session.  
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 Credentials in Keyring*

---

### Description

Securely stores formr credentials in the system keyring. This function supports two modes:

1. **Classic Mode:** Stores email/password (and optional 2FA) for a specific account name.
2. **API Mode:** Stores OAuth credentials or Access Tokens for a specific host.

### Usage

```
formr_store_keys(
    account_name = NULL,
    email = NULL,
    password = NULL,
    secret_2fa = NULL,
    host = "https://rforms.org",
    client_id = NULL,
    client_secret = NULL,
    access_token = NULL,
    account = NULL,
    verbose = TRUE
)
```

### Arguments

account_name	(Classic) A shorthand name for the account. If provided, Classic mode is triggered.
email	(Classic) Email address for the account. Will be prompted if omitted.
password	(Classic) Optional. Provide to skip interactive prompt (useful for scripts/tests).
secret_2fa	(Classic) A 2FA secret. Set to NULL to be prompted, or "" if not used.
host	(API) The API URL (e.g., https://rforms.org). Defaults to rforms.org.
client_id	(API) OAuth Client ID.
client_secret	(API) OAuth Client Secret.
access_token	(API) Direct Personal Access Token (alternative to OAuth).
account	(API) Optional string identifier for multiple accounts on the same host.
verbose	Logical. If TRUE (default), reports progress via <a href="#">message()</a> .

### Value

Invisibly NULL; called for its side effect of storing the password and 2FA seed in the system keyring.

**Examples**

```
## Not run:
# Not run: prompts interactively and writes to the system keyring.
# --- Classic EXAMPLES ---
# Prompts for password interactively
formr_store_keys("formr_diary_study_account")

# --- NEW API EXAMPLES ---

# Store OAuth Credentials for a custom host
formr_store_keys(host = "http://localhost",
                 client_id = "my-id",
                 client_secret = "my-secret")

# Store token for a specific secondary account
formr_store_keys(host = "http://localhost",
                 client_id = "my-id",
                 client_secret = "my-secret",
                 account = "project_b")

## End(Not run)
```

---

formr\_uploaded\_files *Download uploaded files from formr*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A list (parsed JSON) of uploaded-file metadata.

**Examples**

```
## Not run:
# Not run: needs a live formr server and an authenticated session.
formr_uploaded_files(survey_name = 'training_diary' )

## 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://rforms.org>

### Value

Invisibly TRUE on success; called to upload an item-table file to the formr server.

### Examples

```
## Not run:  
# Not run: needs a live formr server and an authenticated session.  
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://rforms.org`

**Value**

A data.frame (parsed JSON) of detailed per-session progress through the run.

**Examples**

```
## Not run:  
# Not run: needs a live formr server and an authenticated session.  
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://rforms.org`

**Value**

A data.frame (parsed JSON) of per-session progress/overview data.

**Examples**

```
## Not run:  
# Not run: needs a live formr server and an authenticated session.  
formr_connect(email = 'you@example.net', password = 'zebrafinch' )  
formr_user_overview(run_name = 'different_drills' )  
  
## End(Not run)
```

---

get_opencpu_rds	<i>pass in the url to the RDS representation of a openCPU session object, get the object</i>
-----------------	--

---

### 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. <a href="https://public.opencpu.org/ocpu/tmp/x02a93ec/R/.val/rds">https://public.opencpu.org/ocpu/tmp/x02a93ec/R/.val/rds</a>
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

### Value

The R object deserialised from an OpenCPU session's RDS result (or loaded from a local .RData).

### Examples

```
## Not run:
# Not run: fetches a result from a remote OpenCPU server.
get_opencpu_rds('https://public.opencpu.org/ocpu/tmp/x02a93ec/R/.val/rds')

## End(Not run)
```

---

ifelsena	<i>Like <a href="#">ifelse()</a>, but allows you to assign a third value to missings.</i>
----------	---

---

### 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

**Value**

A vector like `ifelse()`'s result, with NA positions replaced by missing.

**Examples**

```
## Not run:
# Not run: ifelsena() is deprecated; use dplyr::if_else() instead.
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

**Value**

x with its NA values replaced by missing.

**Examples**

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

---

if_na_null	<i>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 rforms.org, where a field in a survey may either not exist, be missing or have a value to check.</i>
------------	---

---

### 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 rforms.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

### Value

A length-0-or-1 value: test when it is non-missing, na when test is NA, or null when test has length 0.

### 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	<i>checks whether the current time is in a certain time window</i>
----------------	--

---

**Description**

supply min,max as POSIXct

**Usage**

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

**Arguments**

min	POSIXct < max
max	POSIXct > min

**Value**

A length-1 logical: TRUE if the current time lies between min and max.

**Examples**

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

---

item	<i>get item from survey attribute</i>
------	---------------------------------------

---

**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

**Value**

The metadata list for a single item (from a survey variable's attributes), or NULL with a warning if not found.

**Examples**

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

---

items	<i>get item list from survey attributes</i>
-------	---

---

**Description**

get item list from survey attributes

**Usage**

```
items(survey)
```

**Arguments**

survey            survey with item\_list attribute

**Value**

A list of class formr\_item\_list extracted from a survey data.frame's variable attributes.

**Examples**

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

---

knit_prefixed	<i>knit prefixed</i>
---------------	----------------------

---

**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()`

**Value**

A length-1 character string of knitted output, with figure/cache paths prefixed by the input file name.

---

last	<i>Gives the last non-missing element</i>
------	---

---

**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

**Value**

A vector of the same type as x holding its last n elements (after dropping NAs when `na.rm = TRUE`); an empty vector if none remain.

**Examples**

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

---

markdown_custom_options	<i>custom markdown options for rmarkdown's pandoc</i>
-------------------------	---

---

**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 <code>-section-divs</code> (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 <code>rmarkdown::html_document()</code> Custom rmarkdown input format options based on the standard <code>rmarkdown::html_document()</code> , but with options that you can specify. Find the format options here in the pandoc documentation: <a href="https://pandoc.org/MANUAL.html#pandocs-markdown">https://pandoc.org/MANUAL.html#pandocs-markdown</a> . Pandoc's enhanced version of markdown includes syntax for footnotes, tables, flexible ordered lists, definition lists, fenced code blocks, superscript, subscript, strikethrough, title blocks, automatic tables of contents, embedded LaTeX math, citations, and markdown inside HTML block elements or spoken in options: <code>+escaped_line_breaks</code> , <code>+header_attributes</code> , <code>+yaml_metadata_block</code> , <code>+auto_identifiers</code> , <code>+implicit_header_references</code> , <code>+blank_before_blockquote</code> , <code>+fenced_code_blocks</code> , <code>+fenced_code_attributes</code> , <code>+line_blocks</code> , <code>+definition_lists</code> , <code>+startnum</code> , <code>+fancy_lists</code> , <code>+pipe_tables</code> , <code>+pandoc_title_block</code> , <code>+intraword_underscores</code> , <code>+strikethrough</code> , <code>+superscript</code> , <code>+subscript</code> , <code>+tex_math_dollars</code> , <code>+raw_html</code> , <code>+markdown_in_html_blocks</code> , <code>+implicit_figures</code> , <code>+footnotes</code> , <code>+inline_notes</code> , <code>+citations</code> . The current default rmarkdown additions to Pandoc's enhanced markdown are: <code>+autolink_bare_uris</code> , <code>+ascii_identifiers</code> , <code>+tex_math_single_backslash</code> , <code>-implicit_figures</code> .

**Value**

An rmarkdown `output_format` object (from `rmarkdown::html_document()/html_fragment()`) with customised pandoc/knitr options.

---

markdown_github	<i>github_markdown for rmarkdown</i>
-----------------	--------------------------------------

---

**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`, `+strikethrough`, `+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()`

**Value**

An rmarkdown output\_format object configured for GitHub-flavoured markdown.

---

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()`

**Value**

An rmarkdown output\_format object with hard line breaks enabled.

---

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 `rforms.org`. Will be coerced to `POSIXct`.

**Value**

A `POSIXct`: midnight at the start of the day after `date`.

**Examples**

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

---

```
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", collapse = "\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

**Value**

A length-1 character string of class `knit_asis` (the inputs concatenated).

**Examples**

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

---

```
print.formr_api_run_structure
```

*Print method for formr run structure*

---

**Description**

Print method for formr run structure

**Usage**

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

**Arguments**

x	The object.
...	Additional arguments.

**Value**

Invisibly returns x; called for its side effect of printing a formatted table of the run's units.

---

```
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

**Value**

Invisibly NULL; called for its side effect of printing the knit\_asis content.

---

qplot\_on\_bar

*Plot normed values as a barchart*


---

### Description

Pass in a data.frame with z-standardised values  $(x - \text{Mean})/\text{SD}$ , 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

### Value

A ggplot object: a bar chart, optionally with error bars.

### 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	<i>Plot a normed value on the standard normal</i>
-----------------	---

---

### Description

Pass in a z-standardised value  $(x - \text{Mean})/\text{SD}$ , 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

### Value

A ggplot object showing the standard normal distribution with a reference line at normed\_value.

### Examples

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

---

qplot_on_polar	<i>Time-polar plot</i>
----------------	------------------------

---

### Description

Pass in a data.frame with z-standardised values  $(x - \text{Mean})/\text{SD}$ , 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

**Value**

A ggplot object drawn in polar coordinates.

**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 <https://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

**Value**

A POSIXct vector of N random dates within the given range.

---

render_text	<i>render text</i>
-------------	--------------------

---

**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 <code>rmarkdown::render()</code>

**Value**

A length-1 character string of rendered HTML.

---

rescue_attributes	<i>Rescue lost attributes</i>
-------------------	-------------------------------

---

**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

**Value**

A data.frame: df with variable attributes restored from a reference data.frame.

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)
```

---

summary.formr\_results *Summarize Processing History*

---

### Description

Prints a human-readable audit trail of all data cleaning steps.

### Usage

```
## S3 method for class 'formr_results'
summary(object, ...)
```

### Arguments

object                A formr\_results object.  
...                    Additional arguments passed to summary (ignored).

### Value

Invisibly NULL; called for its side effect of printing the processing audit trail.

---

`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 = FALSE)
```

**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

**Value**

Logical TRUE/FALSE indicating send success, or (when return\_result = TRUE) the raw API response list.

**Examples**

```
## Not run:  
# Not run: sends a real SMS via the Clickatell gateway (needs an account).  
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 = "0A",
    test = "0",
    return_result = FALSE
)

```

**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

**Value**

Logical TRUE/FALSE indicating send success (or the raw character API response).

**Examples**

```

## Not run:
# Not run: sends a real SMS via the massenversand.de gateway (needs an account).
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 = FALSE)
```

## 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 Twilio token
return_result	whether to return simply TRUE/FALSE on success/failure or the whole result

## Value

Logical TRUE/FALSE indicating send success, or (when return\_result = TRUE) the raw API response list.

## Examples

```
## Not run:  
# Not run: sends a real SMS via the Twilio gateway (needs an account).  
text_message_twilio(  
  To = '492222',  
  From = '15005000',  
  Body = 'Hello friend',  
  Account = 'ID', Token = 'Tokentokentoken')  
  
## End(Not run)
```

---

time_passed	<i>checks how much time has passed relative to the user's last action</i>
-------------	---

---

### 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 <code>lubridate::dseconds()</code>
time	defaults to <code>.formr\$last_action_time</code> , a hidden variable that is automatically set by <code>rforms.org</code>

### Value

A length-1 logical: TRUE if at least the specified duration has elapsed since `time`.

### Examples

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

---

word_document	<i>word_document from rmarkdown, but has an added option not to break on error</i>
---------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

An rmarkdown `output_format` object like `rmarkdown::word_document()`, with an added option not to break on error.

---

<code>%begins_with%</code>	<i>check whether a character string begins with a string</i>
----------------------------	--

---

**Description**

Escapes any special RegEx 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

**Value**

A logical vector, TRUE where haystack begins with needle.

**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
```

---

<code>%contains%</code>	<i>check whether a character string contains another</i>
-------------------------	--

---

**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

**Value**

A logical vector, TRUE where haystack contains needle as a fixed substring.

**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
```

---

<code>%contains_word%</code>	<i>check whether a character string contains another as a word</i>
------------------------------	--

---

**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

**Value**

A logical vector, TRUE where haystack contains needle as a whole word.

**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
```

---

<code>%ends_with%</code>	<i>check whether a character string ends with a string</i>
--------------------------	--

---

**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

**Value**

A logical vector, TRUE where haystack ends with needle.

**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

.formr, 4  
%starts\_with% (%begins\_with%), 73  
%begins\_with%, 73  
%contains%, 74  
%contains\_word%, 74  
%ends\_with%, 75  
  
aggregate\_and\_document\_scale, 5  
as.data.frame.formr\_api\_run\_structure,  
6  
as.data.frame.formr\_item\_list, 6  
asis\_knit\_child, 7  
asis\_knit\_child(), 62  
  
cat(), 8  
choice\_labels\_for\_values, 8  
current, 9  
  
dplyr::if\_else(), 54  
  
email\_image, 10  
environment, 5  
expired, 10  
  
feedback\_chunk, 11  
finished, 12  
first, 12  
first(), 59  
formr\_aggregate, 13  
formr\_aggregate(), 42  
formr\_api\_aggregate, 14  
formr\_api\_authenticate, 15  
formr\_api\_authenticate(), 4, 41  
formr\_api\_backup\_run, 16  
formr\_api\_backup\_run(), 37  
formr\_api\_create\_run, 16  
formr\_api\_create\_session, 17  
formr\_api\_delete\_all\_files, 18  
formr\_api\_delete\_file, 18  
formr\_api\_delete\_run, 19  
formr\_api\_delete\_survey, 19  
  
formr\_api\_fetch\_results, 20  
formr\_api\_files, 21  
formr\_api\_is\_authenticated, 21  
formr\_api\_logout, 22  
formr\_api\_pull\_project, 22  
formr\_api\_pull\_project(), 37  
formr\_api\_push\_project, 23  
formr\_api\_recognise, 24  
formr\_api\_results, 24  
formr\_api\_results(), 4  
formr\_api\_reverse, 25  
formr\_api\_run\_settings, 26  
formr\_api\_run\_structure, 27  
formr\_api\_runs, 26  
formr\_api\_session, 27  
formr\_api\_session\_action, 29  
formr\_api\_sessions, 28  
formr\_api\_sessions(), 31  
formr\_api\_survey\_structure, 30  
formr\_api\_surveys, 29  
formr\_api\_token\_expiry, 30  
formr\_api\_unit\_sessions, 31  
formr\_api\_unit\_sessions(), 41  
formr\_api\_upload\_file, 32  
formr\_api\_upload\_survey, 33  
formr\_backup\_files, 33  
formr\_backup\_study, 34  
formr\_backup\_study(), 37  
formr\_backup\_surveys, 35  
formr\_connect, 36  
formr\_connect(), 33, 38, 39, 43, 46–49,  
51–53  
formr\_default\_dir, 37  
formr\_default\_dir(), 16, 22, 23, 34, 35  
formr\_disconnect, 37  
formr\_inline\_render, 38  
formr\_item\_displays, 39  
formr\_item\_displays(), 46  
formr\_items, 38

formr\_items(), 13, 44, 46, 49  
formr\_knit, 40  
formr\_last\_host, 40  
formr\_last\_host(), 13, 34–37, 39, 43, 44, 46–48, 51–53  
formr\_overview\_sankey, 41  
formr\_overview\_sankey(), 4  
formr\_post\_process\_results, 42  
formr\_post\_process\_results(), 46  
formr\_raw\_results, 43  
formr\_raw\_results(), 46  
formr\_recognise, 44  
formr\_recognise(), 42  
formr\_render, 45  
formr\_render\_commonmark, 45  
formr\_results, 46  
formr\_reverse, 46  
formr\_run\_structure, 47  
formr\_shuffled, 48  
formr\_simulate\_from\_items, 49  
formr\_store\_keys, 50  
formr\_upload\_items, 52  
formr\_uploaded\_files, 51  
formr\_user\_detail, 52  
formr\_user\_overview, 53  
  
get\_opencpu\_rds, 54  
grepl(), 73–75  
  
haven::labelled(), 68  
  
if\_na, 55  
if\_na\_null, 56  
ifelse(), 54, 55  
ifelsena, 54  
in\_time\_window, 57  
item, 57  
items, 58  
  
knit\_prefixed, 58  
knitr::knit(), 58  
knitr::knit\_child(), 7, 8  
knitr::opts\_knit, 10  
  
last, 59  
last(), 12  
lubridate::dseconds(), 72  
  
markdown\_custom\_options, 38, 45, 59  
markdown\_custom\_options(), 60  
  
markdown\_github, 60  
markdown\_hard\_line\_breaks, 61  
message(), 15–19, 21–23, 26–33, 50  
  
next\_day, 61  
  
paste(), 62  
paste.knit\_asis, 62  
paste.knit\_asis(), 8  
print.formr\_api\_run\_structure, 63  
print.knit\_asis, 63  
  
qplot\_on\_bar, 64  
qplot\_on\_normal, 65  
qplot\_on\_polar, 65  
  
random\_date\_in\_range, 66  
render\_text, 67  
rescue\_attributes, 67  
reverse\_labelled\_values, 68  
rmarkdown::html\_document(), 60, 61  
rmarkdown::render(), 38, 45, 67  
rmarkdown::word\_document(), 73  
  
stringr::str\_detect(), 73–75  
summary.formr\_results, 68  
  
text\_message\_clickatell, 69  
text\_message\_massenversand, 69  
text\_message\_twilio, 71  
time\_passed, 72  
time\_passed(), 4  
  
word\_document, 73