Package 'irw'

November 2, 2026

Type Package

Title Access and Analyze Data from the Item Response Warehouse

Version 1.4.0

Maintainer Ben Domingue <ben.domingue@gmail.com>

URL https://itemresponsewarehouse.github.io/Rpkg/

Contact mailto:itemresponsewarehouse@stanford.edu

Description Find, download, and analyze datasets from the Item Response Warehouse, a harmonized collection of item response data from educational and psychological measurement described in Nadela et al. (2026) <doi:10.59863/cijg4549>. Functions search tables by metadata, fetch them in a common long format, reshape between long and wide layouts, simulate item response data, and compare the predictive accuracy of fitted models with the InterModel Vigorish.

Depends R (>= 4.0.0)

Imports glue, httr, imv, jsonlite, tibble, tools, utils, stats

Suggests arrow, redivis, testthat (>= 3.0.0), withr, evaluate, mirt, gnm, BradleyTerry2

Additional_repositories https://redivis.r-universe.dev

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Config/testthat/edition 3

Config/build/clean-inst-doc FALSE

Config/Needs/website rmarkdown, mirt, ltm, psychotools, psych, mokken, lavaan, lme4, difR, irtoys

BugReports https://github.com/itemresponsewarehouse/Rpkg/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Hansol Lee [aut, cph],
Michael Hardy [aut, cph],
Ben Domingue [aut, cre, cph]

Repository CRAN

Date/Publication 2026-11-02 14:10:26 UTC

Contents

diff_long . . . 3
irw_cache_dir . . . 4
irw_cache_info . . . 4
irw_check_resp . . . 5
irw_clear_cache . . . 6
irw_collection . . . 6
irw_collections . . . 7
irw_collection_members . . . 8
irw_conj_long . . . 9
irw_covariates . . . 10
irw_covariate_labels . . . 12
irw_decode . . . 13
irw_download . . . 14
irw_fetch . . . 15
irw_filter . . . 17
irw_filter_comp . . . 22
irw_get_version . . . 23
irw_has_credentials . . . 24
irw_imv . . . 24
irw_info . . . 26
irw_itemtext . . . 27
irw_itemtext_licence . . . 28
irw_license_options . . . 28
irw_list_itemtext_tables . . . 29
irw_list_tables . . . 30
irw_long2resp . . . 31
irw_merge . . . 32
irw_metadata . . . 34
irw_predict . . . 35
irw_rank_compress . . . 36
irw_recode . . . 37
irw_recode_key . . . 38
irw_reset_version . . . 39
irw_resp2long . . . 39
irw_save_bibtex . . . 40
irw_set_version . . . 41
irw_simdata . . . 42
irw_simdata_comp . . . 43
irw_simu_diff . . . 45

irw_table_sets 46
irw_tags 47
irw_tag_options 48
irw_use_version 49
irw_version 50
swmd_mokken 51

Index **53**

`diff_long` *Simulated Item Difficulty Pool from IRW Datasets*

Description

A dataset of item difficulty estimates and associated standard errors compiled from a curated subset of IRW datasets. Used internally by `irw_simu_diff()` to generate empirically grounded item parameters for simulation.

Usage

```
diff_long
```

Format

A data frame with N rows and 4 variables:

dataset Dataset identifier (character). Indicates the source dataset in IRW.

item Item identifier (character). A label for each item.

difficulty Estimated item difficulty (numeric).

SE Standard error of the difficulty estimate (numeric).

Details

This pool includes binary-response items from IRW datasets that passed basic quality checks—specifically, datasets with less than 50% missing data and without repeated or duplicate responses. It is accessible to users and is used as the default source in `irw_simu_diff()`.

Source

Curated and compiled by the IRW package authors from selected datasets in the IRW repository.

irw_cache_dir *Folder where fetched tables are cached*

Description

`irw_fetch()` and `irw_itemtext()` keep each table they download in this folder, so later sessions do not export it again – every export counts against the Redivis 30-day export cap.

Usage

```
irw_cache_dir()
```

Details

The folder is the `IRW_CACHE_DIR` environment variable if set, otherwise `tools::R_user_dir("irw", "cache")` (e.g. `~/.cache/R/irw` on Linux). The Python package reads and writes the same files, but its default folder is `~/.cache/irw`; to share one cache between the two languages, set `IRW_CACHE_DIR` to the same folder for both.

Switch the cache off with `options(irw.cache = FALSE)` for a session, or `IRW_CACHE=0` for all of them.

Value

Path to the cache folder (it may not exist yet).

See Also

`irw_cache_info()`, `irw_clear_cache()`

irw_cache_info *List the cached tables*

Description

`irw_fetch()` and `irw_itemtext()` keep each table they download so that later sessions do not export it again. A copy is used for as long as the table is unchanged on Redivis, and replaced when it changes. See `irw_cache_dir()` for where the files are and how to switch this off.

Usage

```
irw_cache_info()
```

Value

A tibble with one row per cached file: `kind` (`"tables"` or `"itemtext"`), `table`, `hash` (Redivis' content hash, which decides whether the copy is current), `version` (the dataset release it was fetched from), `reference`, `size_mb`, `fetched_at` and `path`.

Examples

```
info <- irw_cache_info()
sum(info$size_mb)
```

`irw_check_resp` *Check response patterns in IRW data*

Description

Summarize basic response diagnostics in IRW data, including items with a single observed response category and sparse categories in polytomous items.

Usage

```
irw_check_resp(x, min_count = 5L, min_prop = 0.01, resp_col = "resp")
```

Arguments

`x`	A data frame in long format with columns `id`, `item`, `resp`, or the result of `irw_long2resp()`.
`min_count`	Minimum count for a category before it is considered sparse (only used when `x` is long-format). Default is 5.
`min_prop`	Minimum within-item proportion for a category before it is considered sparse (only used when `x` is long-format). Default is 0.01.
`resp_col`	Character string giving the response column to inspect when `x` is long-format. Defaults to `"resp"`.

Details

When `x` is long-format (with columns `id`, `item`, `resp`), diagnostics are recomputed using `min_count` and `min_prop`. When `x` is the wide-format result of `irw_long2resp()`, any attached diagnostics from `check_resp = TRUE` are returned directly.

A response category is treated as "sparse" if it has fewer than `min_count` observed responses OR a within-item proportion smaller than `min_prop`.

Value

A list with two elements: `single_category_items` (character vector of item IDs) and `sparse_category_items` (named list of data frames with columns `resp`, `count`, `prop`).

irw_clear_cache *Delete cached tables*

Description

Delete cached tables

Usage

```
irw_clear_cache(table = NULL)
```

Arguments

table	Character vector of table names to remove, matched case-insensitively; a table's item text goes with it. NULL (the default) removes everything.

Value

Invisibly, the number of bytes freed.

See Also

irw_cache_info(), irw_cache_dir()

irw_collection *Get the Tables in an IRW Collection*

Description

Returns the names of every IRW table in one collection, so they can be passed straight to irw_fetch(), irw_save_bibtex() or any other vectorised accessor.

Usage

```
irw_collection(name, quiet = FALSE)
```

Arguments

name	Character. A single collection name, e.g. "big_five".
quiet	Logical. Suppress the definition/coverage message.

Details

Prints the collection's definition and, when the collection does not cover the whole warehouse, says so — see the Coverage section of irw_collections(). Suppress with quiet = TRUE.

Value

A sorted character vector of table names.

See Also

`irw_collections()`, `irw_filter()`

Examples

```
tabs <- irw_collection("depression")
irw_save_bibtex(head(tabs, 3), output_file = file.path(tempdir(), "depression.bib"))

# Compose with the numeric filters
irw_filter(collection = "depression", n_participants = c(500, Inf))

# Cross-collection: math assessments administered in booklets
intersect(irw_collection("math"), irw_collection("multistage"))
```

`irw_collections` *List IRW Collections*

Description

Returns the collection registry: one row per collection, with its kind, definition, the rule that produced it, and how complete its coverage is.

Usage

```
irw_collections(kind = NULL)
```

Arguments

`kind`	Optional character vector. Filter to `"design"`, `"instrument"` and/or `"construct"`.

Details

Collections are labelled groupings of IRW tables — study designs (`rct`, `clustered`, `q_matrix`), instrument families (`big_five`, `promis`), exact instruments (`phq9`, `gad7`, `rses`, ...) and constructs (`depression`, `math`). A table can belong to any number of them.

Value

A tibble, one row per collection.

Coverage

`coverage` says how much of the warehouse the collection's rule actually searched, and it matters when you need "all" of something:

`metadata-complete` The rule read `irw_metadata()`, so it saw every documented table. Note this is not the same as the whole warehouse — `irw_metadata()` itself has gaps.

`tagged-subset-only` The rule read the `construct name` tag, which is filled for only about half of documented tables, unevenly across the warehouses — and NOT by age: in October 2026 the newest shard was 100% covered and the thinnest was shard 3 at 26%. The definition states the exact count searched. These collections should not be described as exhaustive, and `collections.html` on the site reports the current spread.

`curated-only` Every member was chosen by hand. The exact-instrument collections are of this kind; see `irw_collection_members()` for how each of their members was identified.

See Also

`irw_collection()`, `irw_collection_members()`, `irw_filter()`

Examples

```
irw_collections()
irw_collections(kind = "instrument")
```

`irw_collection_members` *IRW Collection Membership*

Description

The long membership table: one row per (table, collection) pair, with the `basis` on which that membership was decided — a rule expression, or a curator. Use this to ask the inverse question: what collections is a given table in?

Usage

```
irw_collection_members(tables = NULL, collection = NULL)
```

Arguments

`tables`	Optional character vector. Restrict to these table names.
`collection`	Optional character vector. Restrict to these collections.

Value

A tibble with columns `table`, `collection`, `basis`.

Basis in the exact-instrument collections

In the exact-instrument collections (phq9, gad7, rses, cesd, dass21, panas, tipi, pcl5, gse, erq, mspss, sd3), basis starts with how the table was identified as that instrument:

wording The study's own item text matches the instrument. Identity is observed.

supplied IRW identified the instrument and supplied its canonical wording, so a wording match only confirms that identification. Identity is asserted.

name There is no usable English item text; only the table's metadata names the instrument.

A note may follow after "; ": "supplied; subscale: anxiety (7/21)" for a table holding one whole subscale, "name; partial: 18/20 items" for one holding most of the scale. For the strictly verified set, keep rows whose basis starts with "wording", not rows equal to it.

See Also

irw_collections(), irw_collection()

Examples

```
irw_collection_members(tables = "frac20")
irw_collection_members(collection = "q_matrix")

# PHQ-9 tables whose own item text was checked against the instrument
m <- irw_collection_members(collection = "phq9")
m[startsWith(m$basis, "wording"), ]
```

---

irw_conj_long *Convert a conjoint table to the IRW long format*

Description

Conjoint tables (source = "conj") have one row per respondent, task and profile: id, task, profile, the outcomes (choice, rating, and any choice_<name> or rating_<name>), the profile's attribute levels in attr_* columns, and respondent covariates in cov_*. There is no item or resp, because every profile is a new random bundle of attributes.

Usage

```
irw_conj_long(df, outcomes = NULL)
```

Arguments

df	A conjoint table, as returned by irw_fetch(name, source = "conj").
outcomes	Optional character vector of outcome columns to keep. Default: all of them.

Details

This function stacks the outcomes into the core layout so that tools built for `id`/`item`/`resp` can be used: `item` is the name of the outcome question (`"choice"`, `"rating"`, ...), `resp` its value, and `task`, `profile` and the attributes become `trial_` columns. Rows with a missing outcome are dropped.

Value

A data frame with columns `id`, `item`, `resp`, `trial_task`, `trial_profile`, `trial_attr_*`, and the remaining columns of `df`.

Examples

```
## Not run:
d <- irw_fetch("kreps_2020_covid_vaccine", source = "conj")
long <- irw_conj_long(d)
table(long$item)

## End(Not run)
```

`irw_covariates` *Extract Person-Level Covariates from IRW Long-Format Data*

Description

Reduces IRW long-format data to one row per `id`, keeping the columns that describe the person rather than the response. Optionally aligns those rows to the row order of a wide response matrix, which is what most psychometric packages need when a grouping variable is passed alongside a response matrix.

Usage

```
irw_covariates(df, cols = NULL, align = NULL, labels = FALSE, table = NULL)
```

Arguments

`df`	A data frame in IRW long format, containing an `id` column.
`cols`	Character vector of person-level columns to extract. Defaults to `NULL`, meaning detect them automatically as described above.
`align`	Optional. A data frame with an `id` column (such as the result of `irw_long2resp()`), a matrix with `id` rownames, or a vector of ids. The returned rows are put in this order, one row per element, so the result lines up with the rows of a wide response matrix.
`labels`	Opt in to source value labels (ben-domingue/irw#1775). `FALSE` (default) returns the shipped codes. `TRUE` reads the labels for `table` with `irw_covariate_labels()`; a data frame from `irw_covariate_labels()` is used as given (no network). Each returned covariate that has labels becomes a factor whose levels are the labels in code order (unordered: the source does not say whether the codes are

ordinal). A code with no label keeps its code text as a level and is named in a message. A covariate whose codes share one label – institution names withheld – is left as codes, since decoding would merge distinct groups. Covariates without labels are untouched.

`table` The IRW table `df` came from, for `labels = TRUE`. `irw_fetch()` output does not carry its table name, so this must be given unless `df` has a `source_table` column naming a single table. Rows from several tables are refused: the same code means different things in different tables.

Details

`irw_long2resp()` keeps only `id`, `item`, and the response column, so person-level covariates are dropped and the row order of its result is not sorted. Aligning a covariate to that result by hand requires a `match()` on `id`, and getting it wrong silently misassigns people to groups. Passing the wide result as `align` does that match for you.

When `cols` is `NULL`, the person-level columns are detected automatically: every column other than `id` and the response-level columns (`item`, `resp`, `rater`, `rt`, `text`, `date`, `source_table`) that takes exactly one value within every `id`. A column with a value for some of a person's rows and `NA` for others counts as varying and is not selected. Note that `wave` is treated as person-level only when each person appears in a single wave.

Value

A data frame with one row per `id` and columns `id` followed by the person-level covariates. When `align` is supplied, rows follow that order and ids not present in `df` yield `NA` rows.

See Also

`irw_long2resp`, `irw_covariate_labels`

Examples

```
df <- data.frame(
  id = c(1, 1, 2, 2),
  item = c("i1", "i2", "i1", "i2"),
  resp = c(1, 0, 1, 1),
  cov_group = c("A", "A", "B", "B"),
  stringsAsFactors = FALSE
)
irw_covariates(df)

# Aligned to the row order of a wide response matrix
wide <- irw_long2resp(df, id_density_threshold = NULL)
irw_covariates(df, align = wide)

# With the source's value labels (cov_gender 1/2 -> female/male)

if (irw_has_credentials() || nzchar(Sys.getenv("IRW_RUN_EXAMPLES"))) {
  df <- irw_fetch("cucchi_2018_rfq")
  irw_covariates(df, labels = TRUE, table = "cucchi_2018_rfq")
```

```
}
```

`irw_covariate_labels` *Source Value Labels for an IRW Table's Coded Covariates*

Description

Many covariates ship as bare codes – `cov_gender` in {1, 2} – whose meaning was recorded in the source file (an SPSS or Stata value label) but is not in the IRW table, and the meaning differs from table to table: `cucchi_2018_rfq` codes 1 = female, `rowe_2016_cfs_strain` 1 = male. This returns the source's own labels, one row per code, from irw_meta's `covariate_labels` table. It does not touch the response data, and `irw_fetch()` output is unchanged.

Usage

```
irw_covariate_labels(table)
```

Arguments

`table`	Character vector of IRW table names (case-insensitive).

Value

A tibble with character columns `table`, `covariate`, `code`, `label`, one row per code, sorted by table, covariate and code. `code` is the shipped value written as text (a shipped `1.0` is `"1"`). `label` is the source's wording, verbatim and not harmonised across tables, or the literal `"[institution name withheld]"` where the codes name institutions. Only codes that occur in the shipped column are listed, and a covariate may be partly covered. Zero rows, with a message, when IRW has no labels for the table.

If the irw_meta version in use has no `covariate_labels` table (an older or pinned version), this is an error of class `irw_covariate_labels_unavailable`, not an empty result, so it cannot be mistaken for "no labelled covariates".

See Also

`irw_covariates`, whose `labels = TRUE` applies these labels.

Examples

```
irw_covariate_labels("cucchi_2018_rfq")
```

`irw_decode` *Restore Original Identifier Values*

Description

Inverse of `irw_recode()`: maps codes such as `I0001` back to the original identifier values using the recode key.

Usage

```
irw_decode(x, key = NULL, cols = NULL)
```

Arguments

`x`	A data frame returned by `irw_recode()`, or derived from one (including the wide result of `irw_long2resp()`).
`key`	A recode key as returned by `irw_recode_key()`. Defaults to the key attached to `x`; supply it explicitly if the attribute was dropped.
`cols`	Character vector of columns to decode. Defaults to all columns present in both `x` and the key.

Details

Works on long-format data (columns named in the key are decoded in place) and on the wide output of `irw_long2resp()`, where item codes appear as column names. An `item_` prefix on a column name is handled and preserved, which since #3221 matters only for a matrix built by an older version of this package – `irw_long2resp()` now returns the item ids themselves. Values or column names that are not found in the key are left untouched.

Value

`x` with codes replaced by the original values and the `"irw_recode_key"` attribute removed.

See Also

`irw_recode`, `irw_recode_key`

`irw_download` *Download a Dataset*

Description

Downloads a specified dataset table from the IRW datasource.

Usage

```
irw_download(
  table_name,
  path,
  overwrite = FALSE,
  source = "core",
  sim = FALSE,
  comp = FALSE,
  nom = FALSE
)
```

Arguments

`table_name`	A character string specifying the name of the Redivis table to download.
`path`	A string giving the file path the data are saved to. There is no default: name the file yourself, e.g. `"frac20.csv"`, or use `tempfile(fileext = ".csv")` for a throwaway copy.
`overwrite`	Logical. Whether to overwrite an existing file.
`source`	Character. The data family: `"core"` (default), `"nom"`, `"sim"`, or `"comp"`.
`sim`	Deprecated. Use `source = "sim"` instead.
`comp`	Deprecated. Use `source = "comp"` instead.
`nom`	Deprecated. Use `source = "nom"` instead.

Value

A message confirming the file download location.

irw_fetch *Fetch Table(s) from the Item Response Warehouse*

Description

Retrieves one or more tables from IRW and returns them as tibbles. By default this downloads every row of each table and counts against the Redivis export quota. Pass `max_rows` and/or `columns` to transfer a page rather than a table – both are pushed to Redivis, so the rest is never exported. If you only need the item or response value sets, or per-item counts and ranges, use `irw_table_sets()`, which answers with a server-side query. If the table includes a character `resp` column, the function attempts to coerce it into numeric. Strings like `"NA"`, `""`, and `NA` are treated as missing values. A warning is issued only if other non-numeric values are encountered.

Usage

```
irw_fetch(
  name,
  source = "core",
  dedup = FALSE,
  max_rows = NULL,
  columns = NULL,
  sim = FALSE,
  comp = FALSE,
  nom = FALSE,
  resp = FALSE
)
```

Arguments

`name`	Character vector of one or more table names (IRW table IDs).
`source`	Character. The data family: `"core"` (default), `"nom"`, `"sim"`, `"comp"`, `"conj"` (conjoint experiments; see `irw_conj_long`), or `"rank"` (rankings; see `irw_rank_compress`).
`dedup`	Logical, optional. If TRUE, deduplicates responses based on timing variables. Defaults to FALSE. If a `date` column is present, no deduplication is performed. If only a `wave` column is present, one random response is retained per (id, item) group within each wave. If neither `date` nor `wave` is present, one random response is retained per (id, item) pair.
`max_rows`	Integer, optional. Stop after this many rows. Pushed to Redivis, so only those rows are exported and counted against the account's export quota – which is the point: 326 IRW tables are over a million rows, and a cap applied after the download has already cost the quota. A slice is never written to the disk cache, because it cannot stand in for the table; a cached whole table does serve a slice.
`columns`	Character vector, optional. Fetch only these columns, also pushed to Redivis. A column the table does not have is an error. Ignored for the download when `dedup = TRUE` (deduplication needs every column); the columns are then dropped after deduplicating.

sim	Deprecated. Use source = "sim" instead.
comp	Deprecated. Use source = "comp" instead.
nom	Deprecated. Use source = "nom" instead.
resp	Logical, optional. If TRUE, returns response matrix via irw_long2resp(). Defaults to FALSE.

Details

Each table downloaded is also kept on disk (see irw_cache_dir()) and reused by later fetches, in this session or another, for as long as the table is unchanged on Redivis; the Python package shares the same copies. Switch this off with options(irw.cache = FALSE) or IRW_CACHE=0.

A table the IRW found through another collection, such as openESM, prints that collection's note and citation once per session per collection. Silence it with options(irw.source_note = FALSE) or IRW_SOURCE_NOTE=0.

Value

If a single name is provided, returns a tibble. If multiple, returns a named list of tibbles (or error messages, if retrieval failed).

See Also

irw_table_sets() for value sets and summaries without a download; irw_cache_info() for the tables kept on disk.

Examples

```
# Main IRW data
irw_fetch("environment_ltm")

# A look at a large table without exporting all of it
irw_fetch("neurips_2020", max_rows = 100)
irw_fetch("neurips_2020", columns = c("id", "item", "resp"), max_rows = 1000)

# Deduplicated
irw_fetch("pks_probability", dedup = TRUE)

# Simulation data
irw_fetch("gilbert_meta_3", source = "sim")

# Competition data
irw_fetch("collegefb_2021and2022", source = "comp")
```

irw_filter *Filter Available Datasets in IRW*

Description

Returns the names of datasets in the Item Response Warehouse (IRW) that match user-specified metadata, tag values, variable presence, and license criteria.

Usage

```
irw_filter(
  n_responses = NULL,
  n_categories = NULL,
  n_participants = NULL,
  n_items = NULL,
  responses_per_participant = NULL,
  responses_per_item = NULL,
  difficulty_spread = NULL,
  mean_item_rest_r = NULL,
  mean_abs_item_rest_r = NULL,
  prop_items_extreme = NULL,
  density = c(0.5, Inf),
  var = NULL,
  age_range = NULL,
  child_age__for_child_focused_studies_ = NULL,
  construct_type = NULL,
  construct_name = NULL,
  sample = NULL,
  measurement_tool = NULL,
  item_format = NULL,
  primary_language_s_ = NULL,
  longitudinal = NULL,
  n_actors = NULL,
  n_respondents = NULL,
  n_attributes = NULL,
  outcome = NULL,
  country = NULL,
  domain = NULL,
  choice_object = NULL,
  source = "core",
  comp = FALSE,
  sim = FALSE,
  nom = FALSE,
  license = NULL,
  collection = NULL
)
```

Arguments

`n_responses` Numeric vector of length 1 or 2. Filters datasets by total number of responses.

- Length 1: exact value (e.g., `n_responses = 1000`)
- Length 2: range (e.g., `n_responses = c(1000, Inf)`)

`n_categories` Numeric vector of length 1 or 2. Filters by number of unique response categories.

`n_participants` Numeric vector of length 1 or 2. Filters by number of unique participants (`id`).

`n_items` Numeric vector of length 1 or 2. Filters by number of unique items.

`responses_per_participant`
Numeric vector of length 1 or 2. Filters by average responses per participant.

`responses_per_item`
Numeric vector of length 1 or 2. Filters by average responses per item.

`difficulty_spread`
Numeric vector of length 1 or 2. Filters by the share of the table's observed response range that its item means cover – the range of item p-values on a dichotomous table, and normalised by the response range so a 1-5 Likert table is comparable. Near 0 means every item sits at the same difficulty; near 1 means they span the scale. `core` only.

`mean_item_rest_r`
Numeric vector of length 1 or 2. Filters by the mean item-rest correlation, SIGNED. IRW does not harmonise response direction, so a scale with reverse-keyed items averages toward zero here; use `mean_abs_item_rest_r` to ask whether the items hang together at all. `core` only.

`mean_abs_item_rest_r`
Numeric vector of length 1 or 2. Filters by the mean ABSOLUTE item-rest correlation, which is direction-invariant. A wide gap between this and `mean_item_rest_r` indicates reverse-keyed items. `core` only.

`prop_items_extreme`
Numeric vector of length 1 or 2. Filters by the share of items whose mean sits within 5% of either end of the response range, i.e. floor or ceiling items. `prop_items_extreme = c(0, 0)` keeps tables with none. `core` only.

`density` Numeric vector of length 1 or 2, or `NULL`. Filters on `density`, which is **responses per person-item cell** – `n_responses / (n_participants * n_items)` – and is therefore not bounded by 1. A value above 1 means the table measures a person-item pair more than once: 768 of 6,720 core tables are above 1 (measured 2026-10-25), 700 of them carrying an occasion column (`wave`, `rater`, `trial_*`, `session`, ...), and the largest is 3,624.6 for a go/no-go task. It is not the proportion of cells that are filled, which is a different quantity IRW does not publish (ben-domingue/irw#3222).

- Default is `c(0.5, Inf)`: a lower bound only, to exclude sparse matrices. The default used to cap at 1, which silently dropped every repeated-measures table in the corpus.
- Use `NULL` to disable this filter.

- The default is skipped for a source whose metadata has no density column ("nom"). Passing density, var, longitudinal or a numeric filter whose column that source lacks is an error.

var — Character vector. Filters datasets by presence of variables.

- Use exact names (e.g., "rt", "wave"), or
- Use a prefix (e.g., "cov_") to match any variable starting with that prefix.

age_range — Character vector. Filters by participant age group (e.g., "Adult (18+)"). See irw_tag_options("age_range") for values.

child_age__for_child_focused_studies_ — Character vector. Filters by child age subgroup. See irw_tag_options("child_age__for_child_focus for values.

construct_type — Character vector. Filters by high-level construct category (e.g., "Affective/mental health"). See irw_tag_options("construct_type").

construct_name — Character vector. Filters by specific construct (e.g., "Big Five"). See irw_tag_options("construct_na

sample — Character vector. Filters by sample type or recruitment method (e.g., "Educational", "Clinical"). See irw_tag_options("sample").

measurement_tool — Character vector. Filters by instrument type (e.g., "Survey/questionnaire"). See irw_tag_options("measurement_tool").

item_format — Character vector. Filters by item format (e.g., "Likert Scale/selected response"). See irw_tag_options("item_format").

primary_language_s_ — Character vector. Filters by language used (e.g., "eng"). See irw_tag_options("primary_language_s_

longitudinal — Logical or NULL. Filters longitudinal datasets with variables wave or date.

- TRUE: include only datasets flagged as longitudinal
- FALSE: exclude datasets flagged as longitudinal
- NULL (default): no filter

n_actors — Numeric vector of length 1 or 2. Competition-only filter for the number of actors. Only used when source = "comp".

n_respondents — Numeric vector of length 1 or 2. Conjoint-only: number of respondents. Only used when source = "conj".

n_attributes — Numeric vector of length 1 or 2. Conjoint-only: number of attributes varied on each profile.

outcome — Character, "choice" and/or "rating". Conjoint-only: keep tables that have every outcome type named (a table with both matches either).

country — Character vector of ISO 3166 alpha-2 codes (e.g. "US"). Conjoint-only: keep tables fielded in any of them; a table pooling several countries matches each.

domain — Character vector. Conjoint-only: what the study is about, its setting rather than the attribute it manipulates (a candidate conjoint that varies gender is "candidate_choice"). A table carries one or more domains, primary first; it matches when any value named is among them. The vocabulary: candidate_choice, immigration_asylum, climate_energy_environment, international_economy, democracy_institutions, foreign_policy_security, health_covid, tax_welfare_economy, consumer_products,

social_identity_interpersonal, labour_hiring, crime_justice_policing, technology_ai_privacy, political_communication, parties_platforms, public_administration, corruption_accountability, housing_neighbourhoods, other. A value no table carries is an error that lists the values in use. table(irw_metadata(source = "conj")$domain) gives the counts.

choice_object — Character vector. Conjoint-only: what one profile represents, one value per table: person (candidate, applicant, patient, neighbour), policy, scenario (a situation or case), product_service, place, organisation or message (a text or media artefact judged as such). Any value named matches.

source — Character. The data family: "core" (default), "nom", "sim", "comp", "conj", or "rank". For "rank" the usable filter is license; the other criteria name columns its metadata does not have and are refused by name. Filter on the design facts (method, seats, positions offered, ...) with irw_metadata(source = "rank") directly. For "conj" the filters are n_respondents, n_attributes, outcome, country, domain, choice_object and license; see irw_metadata() with source = "conj" for the other design facts. The two tag columns are defined in the irw repo's data/conjoint/README.md ("Topic tags"). Tag filters require "core" or "nom".

comp — Deprecated. Use source = "comp" instead.

sim — Deprecated. Use source = "sim" instead.

nom — Deprecated. Use source = "nom" instead.

license — Character vector. Filters datasets by license (e.g., "CC BY 4.0"). See irw_license_options() for available values.

collection — Character vector. Filters datasets by collection membership (e.g., "rct", "big_five", "depression"). Matching is **OR within the argument**: collection = c("rct", "response_time") returns the union. For the intersection, use intersect(irw_collection("rct"), irw_collection("response_time")). See irw_collections() for available collections and their coverage. Core source only.

Details

Filtering is based on:

- **Numeric metadata**: number of responses, participants, items, etc.
- **Tag metadata**: e.g., construct type, sample, measurement tool
- **Variable presence**: e.g., rt, wave, cov_
- **License type**: e.g., "CC BY 4.0", "CC0 1.0"

Psychometric Filtering:

Four columns describe how a table behaves rather than how big it is: difficulty_spread, mean_item_rest_r, mean_abs_item_rest_r and prop_items_extreme. They are classical and model-free – no IRT model is fitted anywhere – so they are defined for every core table, polytomous and sparse included, and they take no position on what the items measure. They are computed per item and published per table; the per-item rows behind them are the item_classical table of irw_meta.

Two cautions. Response direction is not harmonised across IRW, so a signed mean item-rest correlation near zero can mean a reverse-keyed scale rather than an incoherent one – compare it with the absolute version. And where a respondent sees a handful of items out of thousands, rest scores are nearly disjoint between respondents, so the correlation is a weaker statement about a sparse bank than about a 20-item questionnaire.

Metadata and Tag-Based Filtering:

To explore available metadata:

- `summary(irw_metadata())` — numeric summaries (e.g., `n_responses`, `density`)
- `irw_tags()` — full tag metadata table (1 row per dataset)
- `irw_tag_options("column_name")` — valid values (with counts) for any tag column
- `irw_license_options()` — available license values with frequencies
- `irw_collections()` — the 22 collections, with kind, definition and coverage
- `irw_collection("name")` — the table names in one collection

Tag-based metadata (e.g., `construct_type`, `sample`, `item_format`) can be passed directly as named arguments. See the parameter list below for supported tag columns. Tag filters are available for `source = "core"` and `source = "nom"`; `"comp"` and `"sim"` have no tags by design and reject them.

Value

A sorted character vector of dataset names that match all specified filters, or `character(0)` if no match is found.

Examples

```
# Numeric filters
irw_filter(n_responses = c(1000, Inf), n_items = c(10, 50))
irw_filter(n_participants = c(500, Inf), density = c(0.3, 0.9))

# Variable presence
irw_filter(var = "rt")
irw_filter(var = c("wave", "cov_"), density = NULL)

# Tag metadata filtering
irw_filter(construct_type = "Affective/mental health", sample = "Educational")
irw_tag_options("construct_type")  # view tag values

# License filtering
irw_license_options()
irw_filter(license = "CC BY 4.0")

# Collection filtering
irw_collections()                                  # what exists
irw_filter(collection = "rct")
irw_filter(collection = "depression", n_participants = c(500, Inf))
irw_filter(collection = c("rct", "response_time"))   # union, not intersection

# Filter by response category complexity
```

```
irw_filter(n_categories = 2, density = NULL)           # binary
irw_filter(n_categories = c(3, 5), density = NULL)     # small multi-category
irw_filter(n_categories = c(10, Inf), density = NULL)  # large category sets
# Filter by psychometric property, not just shape (irw_meta tier-0 columns)
irw_filter(mean_abs_item_rest_r = c(0.4, 1))            # items that hang together
irw_filter(prop_items_extreme = c(0, 0),                # no floor or ceiling items
           difficulty_spread = c(0.3, 1))               # spanning the scale
irw_filter(mean_item_rest_r = c(-1, 0.1),               # signed near zero but
           mean_abs_item_rest_r = c(0.3, 1))            # absolute high: reverse-keyed
irw_filter(source = "sim", n_items = c(10, Inf))
irw_filter(source = "comp", n_actors = c(2, 10))
irw_filter(source = "conj", outcome = "rating", country = c("US", "GB"))
irw_filter(source = "conj", domain = "immigration_asylum")       # the immigration conjoints
irw_filter(source = "conj", domain = c("candidate_choice", "parties_platforms"),
           choice_object = "person")
```

`irw_filter_comp` *Filter Available Competition Datasets in IRW*

Description

Returns the names of competition tables that match user-specified criteria. Competition filtering uses:

- Numeric metadata: `n_responses`, `n_actors`
- License: via `comp_biblio` (`Derived_License`)

Usage

```
irw_filter_comp(n_responses = NULL, n_actors = NULL, license = NULL)
```

Arguments

`n_responses`	Numeric vector length 1 or 2. Filters by number of responses. • Length 1: exact value (e.g., `n_responses = 300`) • Length 2: range (e.g., `n_responses = c(300, Inf)`)
`n_actors`	Numeric vector length 1 or 2. Filters by number of actors.
`license`	Character vector. Filters by license (e.g., "CC BY 4.0"). See `irw_license_options(comp = TRUE)` for available values.

Details

Notes:

- Competition datasets do not have tags or item text metadata.
- Competition metadata table names are stored in lowercase.

Value

A sorted character vector of competition table names (lowercase), or `character(0)` if no match is found.

`irw_get_version` *Report the IRW dataset versions in use*

Description

Shows which Redivis version each IRW dataset is being read at, and whether that version is pinned for the session or is simply the current release. Record these tags to make an analysis reproducible; replay them with `irw_set_version`.

Usage

```
irw_get_version(dataset = NULL)
```

Arguments

`dataset`	Optional character vector of dataset names to report on. Defaults to every IRW dataset.

Value

A data frame with columns `dataset`, `version`, and `pinned`.

See Also

`irw_set_version`, `irw_reset_version`

Examples

```
irw_get_version()
irw_get_version("item_response_warehouse")
```

irw_has_credentials *Check whether irw can read from the warehouse without a login prompt*

Description

Fetching and listing IRW tables needs no Redivis login: the warehouse is public, and **redivis** 0.12.14 and later send those requests anonymously. A login is still needed for the server-side queries behind `irw_table_sets`, and for every warehouse call on an older **redivis**, which authenticates unconditionally. In an interactive session irw opens the browser login when one is needed; in a script or other non-interactive session it stops with instructions instead. This function reports whether the **redivis** client and a login are already in place, so code can skip the calls that would otherwise stop.

Usage

```
irw_has_credentials()
```

Value

`TRUE` if **redivis** is installed and credentials are available (a `REDIVIS_API_TOKEN` environment variable or a cached login), otherwise `FALSE`.

Examples

```
irw_has_credentials()
```

irw_imv *InterModel Vigorish (IMV)*

Description

Compare two sets of predicted probabilities for the same binary outcomes. The IMV expresses how much better the second model predicts than the first, on the scale of a weighted coin: each model's geometric mean likelihood is converted to the weight of a coin with the same entropy, and the IMV is the proportional gain in that weight.

Usage

```
irw_imv(data, p1, p2, resp = "resp", eps = 1e-06)
```

Arguments

`data`	A data frame holding the outcomes and both sets of predicted probabilities, or a numeric vector of binary outcomes. When `data` is a vector, `p1` and `p2` must be numeric vectors of the same length.
`p1`	Predictions from the baseline model: a column name in `data`, or a numeric vector when `data` is a vector of outcomes.
`p2`	Predictions from the comparison model, in the same form as `p1`.
`resp`	The binary outcome. Ignored when `data` is a vector; otherwise a column name in `data`, defaulting to `"resp"`.
`eps`	Clamping tolerance for predicted probabilities. Default 1e-6.

Details

An IMV of 0 means the two models predict equally well; 0.05 means model 2 is equivalent to a coin 5\ is not symmetric: `irw_imv(data, "p1", "p2")` is the gain from moving from `p1` to `p2`, and swapping the arguments does not simply flip the sign.

Predicted probabilities are clamped to `[eps, 1 - eps]` before the likelihoods are computed, so predictions of exactly 0 or 1 do not produce an infinite log-likelihood.

The computation is `imv.binary` from the **imv** package; `irw_imv()` adds the data-frame interface and input checks.

Out-of-sample predictions are the intended use: comparing in-sample predictions will favour the more flexible model by construction. The "Comparing Models Out of Sample with the IMV" article works through a cross-validated comparison of a Rasch model against a 2PL.

Value

A single numeric value: the proportional gain in coin weight from `p1` to `p2`.

References

Domingue, B. W., Rahal, C., Faul, J., Freese, J., Kanopka, K., Rigos, A., Stenhaug, B., & Tripathi, A. (2025). The InterModel Vigorish (IMV) as a flexible and portable approach for quantifying predictive accuracy with binary outcomes. *PLOS ONE*, 20(3), e0316491. doi:10.1371/journal.pone.0316491

See Also

`irw_predict` for producing `p1` and `p2` from fitted **mirt** models.

Examples

```
set.seed(1)
truth <- rbinom(500, 1, 0.7)
# A model that knows the base rate beats one that guesses at random.
irw_imv(truth, p1 = rep(0.5, 500), p2 = rep(0.7, 500))
```

irw_info *Retrieve and Print IRW Database or Table Information*

Description

Prints information about the IRW datasets or a specific table.

Usage

```
irw_info(
  table_name = NULL,
  details = FALSE,
  source = "core",
  comp = FALSE,
  sim = FALSE,
  nom = FALSE
)
```

Arguments

table_name	Optional. Table name to describe; if NULL, prints database-level info.
details	Logical. When TRUE and table_name is NULL, also prints a breakdown by dataset. Defaults to FALSE.
source	Character. The data family: "core" (default), "nom", "sim", "comp", "conj" (conjoint experiments; see irw_conj_long), or "rank" (rankings; see irw_rank_compress).
comp	Deprecated. Use source = "comp" instead.
sim	Deprecated. Use source = "sim" instead.
nom	Deprecated. Use source = "nom" instead.

Details

If table_name is NULL, shows combined totals across all IRW datasets. Use details = TRUE to also print a per-dataset breakdown (main IRW only). If a table name is provided, shows details for that table.

Value

Invisibly returns NULL.

Examples

```
irw_info()                # Combined totals
irw_info(details = TRUE)   # Combined + per-dataset breakdown
irw_info("frac20")      # Specific IRW table
irw_info("t20_hyper", source = "comp")  # Competition table
```

```
irw_info("gilbert_meta_3", source = "sim")
```

`irw_itemtext` *Retrieve item text metadata for an IRW table*

Description

Returns item-level text metadata for a given IRW table, if available. If no item text table is available for the specified table, the function returns `NULL` and prints a message.

Usage

```
irw_itemtext(table_name)
```

Arguments

`table_name`	Character. The name of the IRW table to look up (e.g., `"gilbert_meta_49"`).

Value

A tibble containing item text metadata, or `NULL` if unavailable.

Disclaimer

IRW item text is reconstructed from published articles, codebooks, and supplementary materials using a largely automated pipeline with partial human review, and is provided **for research purposes only**. We make no guarantee that the text is accurate, complete, or correctly aligned with the `item` identifiers in the corresponding IRW response data; the original source document is always authoritative. Users are responsible for verifying item text against that source before drawing substantive conclusions. Documented discrepancies are listed at https://itemresponsewarehouse.org/itemtext_issues.html.

Note also that the licenses recorded for IRW response data do not extend to the instruments themselves. Inclusion of item text here implies no license to reuse an instrument; copyright and related rights remain with the original rights holders, and users are responsible for obtaining any permissions their intended use requires. The exception is a table whose wording its source released under an open licence: that licence is printed after the fetch and is returned by `irw_itemtext_licence`.

Examples

```
irw_itemtext("gilbert_meta_49")
```

irw_itemtext_licence *Licence on a table's item wording*

Description

Most IRW item text carries no licence of its own: the response-data licence does not extend to the instrument, and inclusion implies no licence to reuse it. A minority of tables are different: their wording was released by its source under an open licence (CC BY-SA 4.0, CC BY-NC 4.0, ...), and IRW republishes it under that licence, with attribution and a note of its changes. This function returns that record for one table, or `NULL` when the wording carries no licence of its own.

Usage

```
irw_itemtext_licence(table_name)
```

Arguments

`table_name`	Character. The IRW table name (case-insensitive).

Details

The record is the IRW repository's `itemtext/itemtext_licences.csv`, fetched once per session. `irw_itemtext()` prints the same line after fetching a licensed table.

Value

A one-row data frame with columns `table`, `licence`, `licence_url`, `attribution`, `changes` and `basis`, or `NULL` if the wording carries no licence of its own (or the record could not be fetched, in which case a warning says so).

Examples

```
irw_itemtext_licence("iowa_gambling_task")
```

irw_license_options *View Unique License Options with Frequencies*

Description

Returns a data frame showing the number of datasets associated with each license.

Usage

```
irw_license_options(source = "core", comp = FALSE, sim = FALSE, nom = FALSE)
```

Arguments

source	Character. The data family: "core" (default), "sim", "nom", "comp", "conj", or "rank".
comp	Deprecated. Use source = "comp" instead.
sim	Deprecated. Use source = "sim" instead.
nom	Deprecated. Use source = "nom" instead. "Custom" is not a licence: it marks a table whose terms are its own. When it appears, a message says how many of those tables record their terms, which irw_info() prints for a single table. Read the terms before relying on them; the package does not infer what they permit.

Value

A data.frame with 'license' and 'count' columns.

---

irw_list_itemtext_tables

List IRW tables with available item text metadata

Description

Returns a character vector of IRW table names for which item text is available.

Usage

```
irw_list_itemtext_tables()
```

Value

A sorted character vector of IRW table names with item text metadata.

Disclaimer

IRW item text is reconstructed from published articles, codebooks, and supplementary materials using a largely automated pipeline with partial human review, and is provided **for research purposes only**. We make no guarantee that the text is accurate, complete, or correctly aligned with the item identifiers in the corresponding IRW response data; the original source document is always authoritative. Users are responsible for verifying item text against that source before drawing substantive conclusions. Documented discrepancies are listed at https://itemresponsewarehouse.org/itemtext_issues.html.

Note also that the licenses recorded for IRW response data do not extend to the instruments themselves. Inclusion of item text here implies no license to reuse an instrument; copyright and related rights remain with the original rights holders, and users are responsible for obtaining any permissions their intended use requires.

Examples

```
irw_list_itemtext_tables()
```

`irw_list_tables` *List Available Tables*

Description

Retrieves a summary of available tables in the IRW database, including their name, number of rows, and variable count, sorted alphabetically by table name.

Usage

```
irw_list_tables(source = "core", sim = FALSE, comp = FALSE, nom = FALSE)
```

Arguments

`source`	Character. The data family: `"core"` (default), `"nom"`, `"sim"`, `"comp"`, `"conj"` (conjoint experiments; see `irw_conj_long`), or `"rank"` (rankings; see `irw_rank_compress`).
`sim`	Deprecated. Use `source = "sim"` instead.
`comp`	Deprecated. Use `source = "comp"` instead.
`nom`	Deprecated. Use `source = "nom"` instead.

Value

A data frame with the following columns:

`name`	The name of the table, sorted alphabetically.
`numRows`	The number of rows in the table.
`variableCount`	The number of variables in the table.

Examples

```
irw_list_tables()               # Main IRW database
irw_list_tables(source = "sim")   # Simulation dataset
irw_list_tables(source = "comp")  # Competition dataset
```

irw_long2resp *Convert IRW Long-Format Data to Wide-Format Response Matrix*

Description

Converts an IRW-compliant dataset from long format to wide format while handling optional metadata elements and filtering by wave.

Usage

```
irw_long2resp(
  df,
  wave = NULL,
  occasion = NULL,
  id_density_threshold = 0.1,
  agg_method = NULL,
  check_resp = FALSE,
  resp_col = "resp"
)
```

Arguments

df	A data frame containing IRW-compliant item response data in long format.
wave	(Optional) The occasion to keep, matched against df$wave as a string so that labelled waves ("pre"/"post") work as well as numbered ones. If the dataset does not have a wave column, this input is ignored. Defaults to the most frequent wave if NULL. A wave that is not present in the table is an error: reshaping without filtering would aggregate every occasion together.
occasion	(Optional) Named vector or list giving the occasion to keep for any other occasion column the standard defines – rater, session, trial_number, order, period, block, subtest, timepoint, occasion – e.g. occasion = c(rater = "2"). Matched as a string, exactly as wave is. An occasion column present in the table, not named here, and keying a repeated id-item pair is **never chosen for you**: either pass agg_method, which says aggregating across it is what you want, or this errors and names the column. Both halves matter – the aggregation step below would otherwise average occasions into a value that exists in no respondent's data, and picking one occasion unasked is the same defect wearing different clothes (on a 651-rater panel it returns one rater). The occasions kept are recorded on the result as the "occasion" attribute, and any aggregated across as "occasion_pooled".
id_density_threshold	A numeric value between 0.0 and 1.0 specifying the minimum response density required for an id to be included. Default is 0.1. Set to NULL to disable filtering. Density is the proportion of items with a non-missing value in the column given by resp_col.

agg_method	A string specifying how to handle multiple id-item pairs. Options: "mean", "mode", "median", "first". Default is "mean" for numeric response columns and "first" for non-numeric (e.g. text).
check_resp	Logical; if TRUE, perform basic response diagnostics on items using default thresholds and attach results as an attribute. Default is FALSE.
resp_col	Character string giving the column in df to use as the response value when building the wide matrix. Defaults to "resp". For nominal data stored in a text column, use e.g. resp_col = "text".

Details

This function applies a default sparsity filter with a threshold of 0.1 for id density to avoid excessive sparsity. Users can disable filtering by setting id_density_threshold = NULL.

If check_resp = TRUE, the function will run a basic diagnostic using default thresholds (min_count = 5, min_prop = 0.01), attach the results as an attribute "resp_checks" on returned object, and add a short NOTE with counts of flagged items. A response category is considered sparse if it has fewer than min_count responses OR a within-item proportion smaller than min_prop. For more control, use irw_check_resp() directly.

Value

A data frame in wide format where rows represent id values and each item column is named by the item id itself, as the long table spells it – so the matrix joins back to the item text (irw_itemtext()) and anything else keyed by item without renaming anything. (Before #3221 these names carried an item_ prefix.) A numeric item id becomes a column named by that number, which needs backticks in a formula and a rename for lavaan, whose model syntax rejects a name beginning with a digit even in backticks. The column used for response values is recorded in the "resp_col" attribute. If check_resp = TRUE, an attribute "resp_checks" is attached:

- single_category_items: character vector of item IDs with only one observed category.
- sparse_category_items: named list; each element is a data.frame with sparse categories for that item (columns: resp, count, prop).

---

irw_merge *Merge Tables Sharing the Same DOI or BibTex*

Description

Identifies and merges tables that share the same DOI or, if DOI is missing, the same BibTex entry. If tables do not have the same structure, only those with identical structures will be merged. This function performs several consistency checks before merging, and prompts the user for confirmation if inconsistencies are found.

Usage

```
irw_merge(table_name, add_source_column = TRUE)
```

Arguments

table_name	A character string specifying the name of the table to find merge candidates for.
add_source_column	A boolean value indicating whether to add the source_table column to the merged table (default is TRUE). This column keeps track of the original source of each row.

Details

irw_merge() performs the following checks before proceeding with the merge:

1. **Number of Respondents Check:**
   - The function checks if the number of respondents is consistent across all tables.
   - If the number of respondents is consistent, the merge proceeds.
   - If the number of respondents is inconsistent across tables, a **WARNING** message is displayed:

     ```
     === WARNING ===
     The number of respondents (N) is not consistent across tables.
     Merging tables with inconsistent N respondents may require human judgment.
     ```
2. **ID Consistency Check:**
   - Checks if all tables share the same set of IDs.
   - Checks if the IDs form a strict sequence (e.g., 1, 2, 3, ...).
   - If IDs are not consistent across tables, a **NOTE** message is displayed:

     ```
     === NOTE: ===
     - IDs do not match across tables.
     - IDs are sequential (1...n). You may need to manually verify the IDs,
       as there could be multiple studies with different subjects, where IDs
       are the same in both studies.
     ```
3. **Item Column Overlap Check:**
   - Verifies that there is no overlap of item columns across tables.
   - If overlapping items are found, a **NOTE** message is displayed:

     ```
     === NOTE: ===
     - There are items that overlap across tables.
     ```

Tables whose bibliography entry is missing (recorded as "NA" in the biblio table) are never treated as merge candidates for one another.

In a non-interactive session the confirmation prompts cannot be answered, so the merge proceeds by default; the checks above are still reported as messages.

Value

A merged data frame containing all tables with the same DOI or BibTex, or NULL if no merge candidates are found.

`irw_metadata` *Retrieve IRW Metadata Table*

Description

Fetches the metadata table from Redivis and returns it as a tibble. Automatically checks for updates and refreshes only when needed.

Usage

```
irw_metadata(source = "core", sim = FALSE, comp = FALSE, nom = FALSE)
```

Arguments

`source`	Character. The data family: `"core"` (default), `"nom"`, `"sim"`, `"comp"`, `"conj"` (conjoint experiments; see `irw_conj_long`), or `"rank"` (rankings; see `irw_rank_compress`).
`sim`	Deprecated. Use `source = "sim"` instead.
`comp`	Deprecated. Use `source = "comp"` instead.
`nom`	Deprecated. Use `source = "nom"` instead.

Details

Each source has its own columns. For `source = "conj"` a row is one conjoint experiment: `n_respondents`, `n_rows`, `n_tasks`, `n_profiles`, `n_attributes`, `outcomes`, `n_optout_tasks`, and the design facts `country`, `display_language`, `label_language`, `restrictions` (`none`, `yes`, `observed` or `unknown`), `restrictions_note`, `task_source` and `profile_source` (`recorded` or `inferred` from row order), and the topic tags `domain` (what the study is about; one or more values joined with `";"`, primary first) and `choice_object` (what one profile represents). Both are filters in `irw_filter`, which documents their vocabularies.

For `source = "rank"` a row is one rankings table: the counts `n_respondents`, `n_rows`, `n_tasks`, `n_items`, `max_resp` (highest position marked) and `n_duplicate_marks`, and the design record (`method`, `seats`, `n_positions`, `n_candidates`, `n_ballots_cast`, `n_ballots_with_marks`, `jurisdiction`, `date`, `office`, ...), whose codes are documented in the irw repo's `data/rankings/README.md`.

Value

A tibble containing metadata information.

`irw_predict` *Predicted Response Probabilities from a Fitted mirt Model*

Description

Attach per-response predicted probabilities from a fitted unidimensional, dichotomous **mirt** model to long-format IRW data. This is the piece needed to compare models out of sample with `irw_imv`.

Usage

```
irw_predict(model, resp, newdata = NULL, ability = "EAP")
```

Arguments

`model`	A fitted unidimensional **mirt** model with dichotomous items.
`resp`	The wide response frame used to fit `model`, including its `id` column, as returned by `irw_long2resp`. The item columns (everything but `id`) are what was passed to `mirt::mirt()`.
`newdata`	(Optional) A long-format data frame with columns `id` and `item` giving the responses to predict. Defaults to every `id`/`item` pair in `resp`.
`ability`	Scoring method passed to `mirt::fscores()`. Default `"EAP"`.

Details

Person abilities are taken from the fit itself with `mirt::fscores()`, so row *i* of the scored matrix is person `resp$id[i]`. Passing the wide frame that produced the fit is what lets the function recover that mapping; supplying ids in a different order than the rows given to `mirt::mirt()` would silently mispair people with abilities.

Probabilities come from `mirt::probtrace()`, so any dichotomous **mirt** item type (Rasch, 2PL, 3PL, 4PL, ...) is handled without assuming a particular parameterisation.

`newdata` may contain any `id`/`item` pairs, including ones held out of the fit, as long as each person appears in `resp` and each item was estimated. That is the cross-validation case: fit on part of the data, predict the rest.

Requires the **mirt** package.

Value

`newdata` with an added numeric column `p`, the predicted probability of a response of 1.

See Also

`irw_imv`, `irw_long2resp`

Examples

```
df <- irw_simdata(n_id = 200, n_item = 10, model = "2PL", seed = 1)
wide <- irw_long2resp(df)
fit <- mirt::mirt(wide[setdiff(names(wide), "id")], 1, "Rasch")
preds <- irw_predict(fit, wide)
```

---

irw_rank_compress *Compress the positions of a rankings table*

Description

Rankings tables (`source = "rank"`) keep the core column names, but `resp` is a *position*: one row per respondent (`id`), ranking (`task`) and marked position, `resp` = 1 for the first choice, **as recorded**. Two things in the stored table need care before a person-by-item pivot or a Plackett-Luce fit: a skipped position stays a gap (a ballot marked at positions 1 and 3 keeps `resp` 1 and 3), and a candidate marked twice keeps both rows. Counting rules for both differ by jurisdiction, so the table does not apply one; this function applies the common one.

Usage

```
irw_rank_compress(df, drop_repeats = TRUE)
```

Arguments

`df`	A rankings table, as returned by `irw_fetch(name, source = "rank")`: columns `id`, `item`, `resp`, and `task` (taken as a single task when absent).
`drop_repeats`	Logical. Drop an item's later marks within a ranking? Default `TRUE`.

Details

Within each `id` x `task`, rows are taken in `resp` order. With `drop_repeats = TRUE` an item that has already been marked at an earlier position is dropped at every later one (most counting rules treat the later mark as a skip). The reserved items (`write-in`) and (`overvote`) are never dropped: two write-in rows are usually two different people, and two overvote rows are two separate events. Then `resp` is renumbered `1..k` in rank order, so the gaps close. Every other column is kept as it was. The result is sorted by `id`, `task`, `resp`.

The Python package's `irw.rank_compress()` does exactly the same; the two are tested on one shared fixture.

Value

A data frame with the same columns as `df`, `resp` an integer running `1..k` within each `id` x `task`.

See Also

`irw_fetch`, `irw_metadata` with `source = "rank"` for each table's design record.

Examples

```
d <- data.frame(id = c(1, 1, 1, 2), task = 1, item = c("c01", "c03", "c01", "c02"),
                resp = c(1, 3, 4, 2))
irw_rank_compress(d)
irw_rank_compress(d, drop_repeats = FALSE)
## Not run:
d <- irw_fetch("eurovision_2016_final", source = "rank")
first_choices <- subset(irw_rank_compress(d), resp == 1)

## End(Not run)
```

`irw_recode` *Recode IRW Identifier Columns to Simple Sequential Codes*

Description

Replaces the values of identifier columns (by default `id` and `item`) with short ASCII codes such as `P0001` and `I0001`. This is useful when identifiers are in a script or use characters that are awkward to type on a US keyboard, which makes subsetting, writing model formulas, and reading wide-format column names difficult.

Usage

```
irw_recode(df, cols = c("id", "item"), prefix = NULL)
```

Arguments

`df`	A data frame in IRW long format.
`cols`	Character vector of columns to recode. Defaults to `c("id", "item")`.
`prefix`	Optional named character vector overriding the default code prefixes, e.g. `c(item = "Q")`. Defaults are `"P"` for `id`, `"I"` for `item`, and the uppercased first letter of the column name otherwise.

Details

Codes are assigned in sorted order of the unique non-missing values of each column, so they are deterministic for a given set of values. They are, however, only meaningful relative to their key: two different subsets of the same table will generally not produce the same codes.

The mapping is attached to the result as an attribute and can be retrieved with `irw_recode_key()`. Save it if you need it later, because most base R operations on a data frame drop attributes. The original values can be restored with `irw_decode()`.

Note that recoding `item` breaks manual joins against `irw_itemtext()` output, which is keyed on the original item identifiers. Use `irw_recode_key()` to map back before joining.

Value

`df` with the requested columns replaced by character codes and an `"irw_recode_key"` attribute holding a data frame with columns `column`, `original`, and `code`.

See Also

irw_decode, irw_recode_key

Examples

```
df <- data.frame(
  id = c("María-01", "María-01", "René-02"),
  item = c("Q_alpha", "Q_beta", "Q_alpha"),
  resp = c(1, 0, 1),
  stringsAsFactors = FALSE
)
d <- irw_recode(df)
head(d)
irw_recode_key(d)
identical(irw_decode(d)$item, df$item)
```

---

irw_recode_key *Retrieve the Recode Key from an Object Returned by irw_recode()*

Description

Retrieve the Recode Key from an Object Returned by irw_recode()

Usage

```
irw_recode_key(x)
```

Arguments

x	An object returned by `irw_recode()`.

Value

A data frame with columns `column`, `original`, and `code`.

See Also

irw_recode, irw_decode

irw_reset_version *Remove IRW version pins*

Description

Returns one dataset, or the whole session, to reading the current release.

Usage

```
irw_reset_version(dataset = NULL)
```

Arguments

dataset	Optional character vector of dataset names to unpin. Defaults to every pinned dataset.

Value

Invisibly, the named character vector of pins still in effect.

See Also

irw_set_version, irw_get_version

Examples

```
irw_reset_version("item_response_warehouse")
irw_reset_version()   # unpin everything
```

irw_resp2long *Convert Wide-Format Response Matrix to IRW Long Format*

Description

Inverse of irw_long2resp(): converts a wide response data frame/matrix into long format with columns id, item, resp.

Usage

```
irw_resp2long(x, id = TRUE)
```

Arguments

x	A wide response object (data.frame or matrix). If produced by irw_long2resp(), it will typically include an id column.
id	Logical. If TRUE (default), expects an id column in x. If FALSE, creates id = 1:nrow(x) and treats all columns as item columns.

Value

A data.frame with columns `id`, `item`, `resp`.

`irw_save_bibtex` *Save BibTeX Entries for IRW Tables*

Description

Saves BibTeX entries for one or more IRW tables to a specified output file. Updates the BibTeX key to match the table name.

Usage

```
irw_save_bibtex(
  table_names,
  output_file,
  source = "core",
  comp = FALSE,
  sim = FALSE,
  nom = FALSE
)
```

Arguments

`table_names`	A character vector of table names for which BibTeX entries are generated.
`output_file`	A character string giving the file path the BibTeX entries are written to, e.g. `"refs.bib"`. There is no default.
`source`	Character. The data family: `"core"` (default), `"nom"`, `"sim"`, `"comp"`, `"conj"`, or `"rank"`.
`comp`	Deprecated. Use `source = "comp"` instead.
`sim`	Deprecated. Use `source = "sim"` instead.
`nom`	Deprecated. Use `source = "nom"` instead. Only one of comp/sim/nom may be TRUE.

Details

A table the IRW found through another collection, such as openESM, also gets that collection's own entry (e.g. `siepe2026openesm`), added once however many of its tables are requested.

Value

Invisibly returns BibTeX entries as a character vector.

`irw_set_version` *Pin an IRW Redivis dataset to a released version*

Description

Fixes the version of one IRW dataset for the remainder of the R session, so that fetches, listings, and metadata are reproducible. Pin at the top of an analysis script and the script will return the same data when it is re-run, even after IRW has been updated.

Usage

```
irw_set_version(dataset, version)
```

Arguments

`dataset`	Character. Name of the IRW dataset to pin, e.g. `"item_response_warehouse"`. See `irw_get_version()` for the full list.
`version`	Character. A released version tag, e.g. `"v32.0"` (a leading `"v"` is optional).

Details

The pin covers every lookup that reads the dataset, not just fetches: `irw_fetch()`, `irw_filter()`, `irw_list_tables()`, and metadata all read the pinned version.

Each IRW dataset is versioned independently on Redivis, so pinning is per-dataset. `irw_get_version()` reports the versions currently in use, which is how you find the tags to write into a script.

A table that does not exist in the pinned version is an error rather than a silent fetch from the current release: quietly mixing versions would break the reproducibility the pin exists to provide.

Value

Invisibly, the named character vector of all pins in effect.

See Also

`irw_get_version`, `irw_reset_version`

Examples

```
irw_set_version("item_response_warehouse", "v32.0")
irw_set_version("item_response_warehouse_2", "v1.8")

df <- irw_fetch("kfcovid_fear_Li2020")   # served from v32.0

irw_reset_version()                      # back to the current release
```

irw_simdata *Simulate IRW-compliant dichotomous item response data*

Description

Simulates item response data from 1PL, 2PL, or 3PL logistic IRT models. Returns a long-format data frame with `id`, `item`, and `resp` columns.

Usage

```
irw_simdata(
  n_id = 1000,
  n_item = 20,
  model = "1PL",
  a = NULL,
  b = NULL,
  g = NULL,
  theta = NULL,
  theta_mean = 0,
  theta_sd = 1,
  seed = NULL,
  return_params = FALSE
)
```

Arguments

`n_id`	Number of respondents. Ignored if `theta` is provided. Default is 1000.
`n_item`	Number of items. Default is 20.
`model`	Character string: one of `"1PL"`, `"2PL"`, or `"3PL"`. Default is `"1PL"`.
`a`	Optional vector of item discriminations. If `NULL`, sampled from lognormal(0, 0.5), i.e. a median of 1 with roughly the middle half of items between 0.7 and 1.4 (ignored for 1PL).
`b`	Optional vector of item difficulties. If `NULL`, sampled from N(0, 1).
`g`	Optional vector of guessing parameters. Required for 3PL; if `NULL`, sampled from Beta(5, 17).
`theta`	Optional vector of person abilities. If provided, overrides `n_id`, `theta_mean`, and `theta_sd`.
`theta_mean`	Mean of theta distribution (if `theta` not supplied). Default is 0.
`theta_sd`	Standard deviation of theta distribution. Default is 1.
`seed`	Optional integer for reproducibility.
`return_params`	Logical. If `TRUE`, returns a list with data and parameters. Default is `FALSE`.

Value

A data frame with columns `id`, `item`, `resp`, or a list with additional parameters if `return_params = TRUE`.

Examples

```
# 1PL (default)
dat <- irw_simdata(n_item = 5)

# 3PL with latent traits drawn from N(-0.5, 1)
sim <- irw_simdata(n_item = 5, model = "3PL", theta_mean = -0.5, return_params = TRUE)
head(sim$data)
```

`irw_simdata_comp` *Simulate IRW-compliant pairwise-comparison data*

Description

Generates pairwise outcomes among `n_agent` agents using the Davidson model with ties. (See https://link.springer.com/article/10.3758/s13428-021-01714-2). Returns one row per comparison with `agent_a`, `agent_b`, and `winner`.

Usage

```
irw_simdata_comp(
  n_agent = 100,
  n_pairs = 10000,
  nu = 0,
  theta = NULL,
  theta_mean = 0,
  theta_sd = 1,
  seed = NULL,
  return_params = FALSE
)
```

Arguments

`n_agent`	Integer. Number of agents (ignored if `theta` provided). Default 100.
`n_pairs`	Integer. Number of pairwise comparisons to sample. Default 10000.
`nu`	Numeric. Tie propensity in the Davidson model. Larger values yield more ties. Default 0.
`theta`	Optional numeric vector of length `n_agent` with agent abilities. If provided, overrides `n_agent`. If NULL, sampled from N(0,1).
`theta_mean`	Mean of the theta distribution (used if `theta` is NULL). Default 0.
`theta_sd`	SD of the theta distribution (used if `theta` is NULL). Default 1.
`seed`	Optional integer for reproducibility.
`return_params`	Logical. If TRUE, return a list with `data`, `theta`, and `nu`. Default FALSE.

Value

A `data.frame` with columns:

- `agent_a` (int): index of first agent
- `agent_b` (int): index of second agent
- `winner` (chr): one of "agent_a","agent_b", or "draw".

If `return_params = TRUE`, returns a list with:

- `data`: the simulated data frame.
- `theta`: vector of simulated thetas used
- `nu`: tie parameter

Examples

```
# --- Basic usage ---
d <- irw_simdata_comp(n_agent = 100, n_pairs = 10000, nu = 0)
head(d)

# Simulate pairwise comparison data and recover abilities with a Davidson model

if (requireNamespace("gnm", quietly = TRUE) &&
    requireNamespace("BradleyTerry2", quietly = TRUE)) {
  th <- rnorm(n = 50, mean = 0.5, sd = 3)
  pairs <- irw_simdata_comp(
    n_agent = 50, n_pairs = 3000, nu = 0.1,
    theta = th, seed = 1, return_params = TRUE
  )

  x <- pairs$data
  x$winner <- ifelse(x$winner == "agent_b", -1, 1)
  x$winner <- ifelse(x$winner == "draw", 0, x$winner)
  x$agent_a <- as.factor(x$agent_a)
  x$agent_b <- as.factor(x$agent_b)
  x$pair <- seq_len(nrow(x))  # add pair id for expandCategorical

  library(gnm)
  library(BradleyTerry2)

  pairs.tri <- expandCategorical(x, "winner", idvar = "pair")

  dav <- gnm(
    count ~ GenDavidson(
      winner == 1, winner == 0, winner == -1,
      player1 = agent_a,
      player2 = agent_b
    ) - 1,
    eliminate = pair,
    family = poisson,
    data = pairs.tri
  )
```

```
  plot(
    pairs$theta, coef(dav)[-1],
    xlab = "True theta",
    ylab = "Estimated ability (shifted)",
    main = "Davidson recovery using gnm"
  )
  abline(lm(coef(dav)[-1] ~ pairs$theta), lty = 2)
}
```

`irw_simu_diff` *Simulate Item Difficulties from IRW or Custom Pools*

Description

This function generates simulated item difficulties by drawing from normal distributions centered around existing difficulty estimates and their associated standard errors. The result is a mixture distribution, from which new difficulties are sampled using inverse CDF sampling.

Usage

```
irw_simu_diff(
  num_items = 10,
  num_replications = 1,
  irw_names = NULL,
  difficulty_pool = NULL
)
```

Arguments

`num_items`	Number of item difficulties to simulate per replication.
`num_replications`	Number of replications to perform. If 1, returns a numeric vector. If >1, returns a data frame.
`irw_names`	Optional character vector of IRW dataset names to filter from `diff_long`.
`difficulty_pool`	Optional custom data frame with columns `dataset`, `difficulty`, and `SE`. If provided, overrides the default IRW difficulty pool (`diff_long`).

Details

By default, the function uses `diff_long`, a built-in dataset included in the `irw` package. This dataset contains item difficulty estimates and standard errors from a curated subset of IRW datasets. You can:

- Use the full IRW difficulty pool (`diff_long`)

- Filter to specific IRW datasets via `irw_names`
- Provide your own difficulty pool via `difficulty_pool`

This method is based on Zhang et al. (2025), which constructs realistic empirical distributions by accounting for uncertainty around item difficulty estimates.

Value

A numeric vector of difficulties (if `num_replications = 1`), or a data frame with `replication` and `difficulty` columns (if `num_replications > 1`).

References

Zhang, L., Liu, Y., Molenaar, D., & Domingue, B. (2025). *Realistic Simulation of Item Difficulties*. https://doi.org/10.31234/osf.io/jbhxy_v1

Examples

```
# Use all IRW data (default)
irw_simu_diff(num_items = 5)

# Filter to specific IRW datasets
irw_simu_diff(num_items = 5, irw_names = c("psychtools_epi", "psychtools_blot"))

# Use a custom difficulty pool
my_pool <- data.frame(dataset = "x",
                      difficulty = c(-0.2, 0.1),
                      SE = c(0.1, 0.2))
irw_simu_diff(num_items = 5, difficulty_pool = my_pool)

# Explore built-in IRW difficulty pool
head(diff_long)
unique(diff_long$dataset)
```

---

irw_table_sets *Summarize the value sets of an IRW table without downloading it*

Description

Answers set and summary questions about a table – which item codes it contains, which response values occur, how many rows each item has – using server-side aggregate queries rather than exporting the table. This matters for large tables: computing the item set of a 68-million-row table with `irw_fetch()` downloads all 68 million rows, while `irw_table_sets()` returns the same answer in seconds and does not consume the Redivis export quota.

Usage

```
irw_table_sets(name, source = "core", per_item = FALSE)
```

Arguments

name	Character vector of one or more IRW table names. Several names run the same queries once per table, still without an export, so this is the way to sweep many tables.
source	Character. The data family: "core" (default), "nom", "sim", or "comp".
per_item	Logical. If TRUE, also return a per-item summary (row count, response minimum, maximum, and number of distinct response values). One extra query; the result has one row per distinct item. Defaults to FALSE.

Value

For a single name, a list with elements:

table Fully qualified Redivis reference for the table.

n_rows Total number of rows.

items Sorted character vector of distinct item values.

resp Sorted vector of distinct resp values, numeric when all values are numeric and character otherwise. "NA" and empty strings are treated as missing, matching irw_fetch().

per_item Data frame of per-item summaries, or NULL when per_item = FALSE or the table has no item or resp column.

For several names, a named list of those lists, one per table, in the order given. Each element is exactly what the single-name call returns.

Examples

```
sets <- irw_table_sets("rosenberg_selfesteem")
sets$items
sets$resp

irw_table_sets("condon_2024_sapa_personality", per_item = TRUE)$per_item

# Several tables at once: no export, one set of queries per table
sweep <- irw_table_sets(c("rosenberg_selfesteem", "environment_ltm"))
lengths(lapply(sweep, `[[`, "items"))
```

---

irw_tags *Retrieve IRW Tags Table*

Description

Fetches the tags table from Redivis and returns it as a tibble. Link to Redivis table: https://redivis.com/datasets/bdxt-4fqe5tyf4/tables/7nkh-6m7wq01yv Automatically checks for updates and refreshes only when needed.

Usage

```
irw_tags(tables = NULL, source = "core")
```

Arguments

tables	Optional. A character vector of table name(s) to filter by.
source	Character. The data family: "core" (default) or "nom". "comp" and "sim" have no tags by design and error.

Value

A tibble containing tags information.

Examples

```
irw_tags()                  # core
irw_tags(source = "nom")    # nominal
```

---

irw_tag_options *View Available Tag Values with Frequencies*

Description

Returns a data frame of unique tag values from a given tag metadata column, along with the number of datasets each tag appears in. Multi-tag fields hold a comma-joined list; no tag value contains a comma (see .irw_split_tags).

Usage

```
irw_tag_options(column, source = "core")
```

Arguments

column	A character string specifying the tag column name.
source	Character. The data family: "core" (default) or "nom". "comp" and "sim" have no tags by design and error.

Value

A data.frame with columns: tag and count, sorted by descending frequency.

`irw_use_version` *Read the whole corpus at one IRW version*

Description

Pins every IRW dataset at once to what a single IRW version held, so that `irw_fetch()` and everything around it return the same data whenever the script is re-run – even after IRW has been corrected or extended. This is the reproducibility switch: put one call at the top of an analysis and the rest of the script is frozen.

Usage

```
irw_use_version(version = NULL, date = NULL, quiet = FALSE)
```

Arguments

`version`	Optional IRW version number, e.g. `332`. Defaults to the newest version.
`date`	Optional date or time to pin the corpus as it was then, e.g. `"2026-08-01"`. Mutually exclusive with `version`.
`quiet`	Logical. Suppress the per-dataset summary.

Details

```
irw_use_version(332)
df <- irw_fetch("gilbert_meta_12")   # v332's copy, today and next year
```

Called with no arguments it pins the newest version and reports its number: run it at the start of a project and record the number it prints.

The pin covers the metadata and item text datasets too, not just response data, and lasts for the session; `irw_reset_version()` lifts it.

Value

Invisibly, a data frame of `dataset` and `version` as pinned, with the IRW version number in the attribute `irw_version`.

Each dataset is pinned to its own version

The response data is spread over several Redivis warehouses ("shards"), each released on its own schedule, so one IRW version means a *different* Redivis version in each. That is handled: every dataset is pinned to the tag the manifest records for it, and a single `irw_fetch()` call may read several shards at different versions.

```
irw_use_version(200)   # shard 1 at v36.0, shard 2 at v3.3
L <- irw_fetch(c("idcr_martinez_2023_numSeries", "gcb5_2025"))
```

A dataset that did not exist yet

Not every IRW dataset has a release in every IRW version – the sixth warehouse shard is younger than the first. Rather than let those fall through to the current release and silently mix versions into a "reproduced" run, they are pinned to nothing and error if something reads them. Fetches are unaffected: a table that did not exist then is simply not found.

Prefer a version number to a date

A date has to be resolved to a version, and before 21 July 2026 that resolution is approximate for the older warehouse shards – see `irw_version()`. Pinning by date therefore warns; pinning by number never has to.

See Also

`irw_version()`, `irw_get_version()`, `irw_reset_version()`

Examples

```
irw_use_version(332)              # read the corpus as v332 held it
irw_use_version()                 # freeze today's version; record the number
irw_use_version(date = "2026-08-01")
irw_reset_version()               # back to the current release
```

`irw_version` *Which IRW version was live, and what every dataset was pinned to*

Description

IRW is made of eleven Redivis datasets that are versioned independently, so no single Redivis version describes the corpus. The IRW version number does: it increments whenever any dataset is published and names one exact combination of the eleven. Cite it in a paper and a reader can reconstruct the data you used.

Usage

```
irw_version(date = NULL, version = NULL)
```

Arguments

`date`	Optional. A date or time to look up, as `"2026-08-01"`, `"2026-08-01 12:00:00"`, or a `Date`/`POSIXct`.
`version`	Optional. An IRW version number, e.g. `332`. Mutually exclusive with `date`. Defaults, with `date`, to the newest version.

Details

Called with no argument, reports the newest IRW version. Called with a `version`, reports exactly what that version held. Called with a `date`, reports the version that was live then – which is how you recover what an analysis run months ago was actually reading.

Use `irw_use_version()` to make the session read one of these versions.

Value

A data frame with one row per dataset and columns `dataset`, `version`, `released_at`, and `approximate`. The IRW version number is attached as the attribute `irw_version` and printed as a message.

Dates before 21 July 2026 are approximate

Redivis overwrote its own release timestamps for the older warehouse shards during a platform migration: 142 of the corpus' 332 released versions claim to have been released inside one 80-minute window on 2026-07-21. For those, the manifest records the earliest date the version could have been live (its creation date) and marks the row `bracketed`. A lookup that lands on one warns, because the *tag* may then be wrong too – a later version could already have been released inside the bracket. IRW version numbers themselves are always exact; only the mapping from a date to a version is affected. This is why a paper should cite the number, not the date.

See Also

`irw_use_version()` to read a whole IRW version, `irw_set_version()` to pin one dataset, `irw_get_version()` for the versions the current session is actually reading.

Examples

```
irw_version()                # the newest IRW version, and its eleven pins
irw_version(version = 332)   # exactly what v332 held
irw_version("2026-08-01")    # what was live on 1 August 2026
```

---

swmd_mokken *Example Data from IRW Database*

Description

A sample dataset sourced from the IRW database to illustrate package functionality and enable users to experiment without needing authentication.

Usage

```
swmd_mokken
```

Format

A data frame with 4557 rows and 4 variables:

id Unique identifier for each subject.

item Item identifier, specifying the question or measurement.

resp Response variable, indicating the response provided.

rater Rater identifier, specifying the rater who provided the response.

Details

This dataset is part of the IRW database and serves as an example for using package functions. It includes basic variables typically present in psychometric datasets such as subject identifiers, item identifiers, and response variables.

Source

IRW Data Warehouse

Index

∗ **datasets**
    diff_long, 3
    swmd_mokken, 51

diff_long, 3

imv.binary, *25*
irw_cache_dir, 4
irw_cache_dir(), *4*, *6*, *16*
irw_cache_info, 4
irw_cache_info(), *4*, *6*, *16*
irw_check_resp, 5
irw_clear_cache, 6
irw_clear_cache(), *4*
irw_collection, 6
irw_collection(), *8*, *9*
irw_collection_members, 8
irw_collection_members(), *8*
irw_collections, 7
irw_collections(), *6*, *7*, *9*
irw_conj_long, 9, *15*, *26*, *30*, *34*
irw_covariate_labels, *10*, *11*, 12
irw_covariates, 10, *12*
irw_decode, 13, *38*
irw_download, 14
irw_fetch, 15, *36*
irw_fetch(), *6*
irw_filter, 17, *34*
irw_filter(), *7*, *8*
irw_filter_comp, 22
irw_get_version, 23, *39*, *41*
irw_get_version(), *50*, *51*
irw_has_credentials, 24
irw_imv, 24, *35*
irw_info, 26
irw_info(), *29*
irw_itemtext, 27
irw_itemtext_licence, *27*, 28
irw_license_options, 28
irw_list_itemtext_tables, 29
irw_list_tables, 30
irw_long2resp, *11*, 31, *35*
irw_merge, 32
irw_metadata, 34, *36*
irw_metadata(), *20*
irw_predict, *25*, 35
irw_rank_compress, *15*, *26*, *30*, *34*, 36
irw_recode, *13*, 37, *38*
irw_recode_key, *13*, *38*, 38
irw_reset_version, *23*, 39, *41*
irw_reset_version(), *49*, *50*
irw_resp2long, 39
irw_save_bibtex, 40
irw_save_bibtex(), *6*
irw_set_version, *23*, *39*, 41
irw_set_version(), *51*
irw_simdata, 42
irw_simdata_comp, 43
irw_simu_diff, 45
irw_table_sets, *24*, 46
irw_table_sets(), *15*, *16*
irw_tag_options, 48
irw_tags, 47
irw_use_version, 49
irw_use_version(), *51*
irw_version, 50
irw_version(), *50*

swmd_mokken, 51