

Package: midfieldr (via r-universe)

October 2, 2026

Title Tools and Methods for Working with MIDFIELD Data in 'R'

Version 1.0.3.9046

Maintainer Richard Layton <graphdoctor@gmail.com>

Description Provides tools in 'R' for working with undergraduate, longitudinal, student-level records modeled on the MIDFIELD database. Tools facilitate identifying academic program codes, excluding post-baccalaureate terms, excluding records for insufficient data, and assessing timely completion. The tools support the workflow of collecting programs, refining the population, constructing blocs of records for aggregation, and calculating quantitative metrics. 'midfieldr' interacts with practice data provided in the 'midfelddata' package or with any data modeled on the MIDFIELD database. The development of 'midfieldr' and 'midfelddata' was supported by the US National Science Foundation through grant numbers 1545667 and 2142087.

License MIT + file LICENSE

URL <https://midfieldr.github.io/midfieldr/>,
<https://midfieldr.github.io/midfelddata/>

BugReports <https://github.com/MIDFIELDR/midfieldr/issues>

Depends R (>= 4.1.0)

Imports checkmate, data.table (>= 1.9.8), stats, wrapr

Suggests knitr, rmarkdown, tinytest

VignetteBuilder knitr, rmarkdown

Config/Needs/website rmarkdown

Config/roxygen2/version 8.0.0

Encoding UTF-8

Language en-US

LazyData TRUE

LazyDataCompression bzip2

Roxygen list(markdown = TRUE)

Repository <https://midfieldr.r-universe.dev>
Date/Publication 2026-10-02 13:58:58 UTC
RemoteUrl <https://github.com/midfieldr/midfieldr>
RemoteRef HEAD
RemoteSha b77eba21771b410c80fb580bac02116b5193f9ba

Contents

act_sat_scale	2
add_completion_status	3
case_blocs	5
case_results	5
catch_error	6
cip	7
cip2010	8
completion_status	9
data_sufficiency	10
filter_programs	12
fye_proxy	13
grade_scale	14
initialize_fye_proxies	15
look_at	17
order_multiway	18
population_baseline	20
pre_or_post_bacc	21
rm_redundant_cols	22
select_basic_cols	24
sort_uniq	25
timely_term	26
toy_course	28
toy_degree	29
toy_student	30
toy_term	31
Index	32

act_sat_scale	<i>ACT-SAT conversion scale</i>
---------------	---------------------------------

Description

Data frame for converting between ACT and SAT scores. A range of SAT scores converts to a single ACT score; an ACT score converts to a single value equivalent SAT score.

Usage

```
act_sat_scale
```

Format

data.table with 28 rows and 4 columns:

act_comp Numerical, ACT composite score.

sat_lower Numerical, total SAT, lower limit of range corresponding to the ACT composite score.

sat_equiv Numerical, total SAT, value to use when converting ACT score to a single SAT score.

sat_upper Numerical, total SAT, upper limit of range corresponding to the ACT composite score.

Source

ACT/SAT Concordance (2018) ACT Education Corp. <https://www.act.org/content/dam/act/unsecured/documents/ACT-SAT-Concordance-Tables.pdf>

See Also

Other scales: [grade_scale](#)

add_completion_status *midfieldr deprecated functions*

Description

These functions were deprecated in `midfieldr` 1.0.4.

Usage

```
add_completion_status(dframe, midfield_degree = degree)
```

```
add_data_sufficiency(dframe, midfield_term = term)
```

```
filter_cip(keep_text = NULL, drop_text = NULL, cip = NULL, select = NULL)
```

```
prep_fye_mice(midfield_student, midfield_term, fye_codes = NULL)
```

```
select_required(midfield_x, select_add = NULL)
```

```
add_timely_term(
  dframe,
  midfield_term = term,
  ...,
  sched_span = NULL,
  span = NULL
)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble)
midfield_degree	*degree* data frame
midfield_term	*term* data frame
keep_text	Deprecated filter_cip(). Character vector of search text to keep.
drop_text	Deprecated filter_cip(). Character vector of search text to drop.
cip	Deprecated filter_cip(). Data frame of programs to be searched.
select	Deprecated filter_cip(). Character vector of column names to select.
midfield_student	*student* data frame
fye_codes	Character, one 6-digit CIP code used for FYE programs. Default "140102"
midfield_x	Deprecated select_required(). Data frame from which columns are selected.
select_add	Defunct select_required(). Character vector of patterns to search dframe column names.
...	Not used for passing values; forces subsequent arguments to be referable only by name.
sched_span	Integer scalar
span	Integer scalar

Details

add_completion_status() is deprecated in favor of [completion_status\(\)](#). Update midfieldr file names and argument names, dropping columns not used by the function, and preserving data frame class.

add_data_sufficiency() is deprecated in favor of [data_sufficiency\(\)](#). Update midfieldr file names and argument names, dropping columns not used by the function, and preserving data frame class.

add_timely_term() is deprecated in favor of [timely_term\(\)](#). Update midfieldr file names and argument names, dropping columns not used by the function, and preserving data frame class.

filter_cip() is deprecated in favor of [filter_programs\(\)](#). The new function is similar but with the CIP data frame as the first argument, enabling chained functions like those encountered using dplyr and friends.

select_required() is deprecated in favor of [select_basic_cols\(\)](#). The new functionality is similar but with exact matching to the default column names plus preserving data frame class.

case_blocs	<i>Case-study outcome by ID and grouping variables</i>
------------	--

Description

Data table of student IDs, program, race/ethnicity, sex, and bloc (graduates or ever-enrolled) of the case study (Civil, Electrical, Industrial/Systems, and Mechanical Engineering) just prior to grouping and summarizing to develop the stickiness metric.

Usage

case_blocs

Format

data.table with 8868 rows and 4 columns:

mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."

people Character. Merged values of race/ethnicity and sex.

program Character. Abbreviation of the case-study program name.

bloc Character. Distinguish between program graduates and those ever enrolled in the programs.

See Also

Other case-study-data: [case_results](#), [population_baseline](#)

case_results	<i>Case-study results</i>
--------------	---------------------------

Description

Data table of longitudinal stickiness for the four programs of the case study (Civil, Electrical, Industrial/Systems, and Mechanical Engineering) grouped by program, race/ethnicity, and sex. Provided for the convenience of vignette users.

Usage

case_results

Format

data.table with 43 rows and 4 columns:

program Character. Academic program label.

people Character. Race/ethnicity and sex as self-reported by the student, e.g., "Asian Male", "Black Female", etc.

ever Numerical. The number of students ever enrolled in a program.

grad Numerical. Number of students completing a program.

stick Numerical. Program stickiness, the ratio of the number of graduates to the number ever enrolled, in percent.

Details

Longitudinal stickiness is the ratio of the number of students graduating from a program to the number of students ever enrolled in the program over the time span of available data. Results are based on data that have been filtered for data sufficiency, degree seeking, undergraduate terms, and timely completion.

See Also

Other case-study-data: [case_blocs](#), [population_baseline](#)

catch_error

Error handling

Description

A wrapper on `base::tryCatch()` for previewing an error message, if any.

Usage

```
catch_error(f)
```

Arguments

f Function with arguments expecting an error

Value

Does not return anything. The side effect is to output to the terminal.

Examples

```
# setup
selected_ids <- toy_student[14:18, (mcid)]
s <- toy_student[mcid %chin% selected_ids, .(mcid, sex)]
t <- toy_term[mcid %chin% selected_ids, .(mcid, term)]
d <- toy_degree[mcid %chin% selected_ids, .(mcid, term_degree)]

# No error
catch_error(pre_or_post_bacc(dframe = t, midf_table = d))

# Error, no term variable in dframe
catch_error(pre_or_post_bacc(dframe = s, midf_table = d))

# Error, missing dframe argument
catch_error(pre_or_post_bacc(midf_table = d))

# Error, missing degree value in environment
catch_error(pre_or_post_bacc(dframe = t))
```

cip

Table of academic programs

Description

A data table based on the US National Center for Education Statistics (NCES), Integrated Post-secondary Education Data System (IPEDS), 2010 CIP. The data are codes and names for 1582 instructional programs organized on three levels: a 2-digit series, a 4-digit series, and a 6-digit series.

Usage

```
cip
```

Format

A data table with 1582 rows and 6 columns keyed by the 6-digit CIP code:

`cip6name` Character, program name at the 6-digit level

`cip6` Character, 6-digit code representing "specific instructional programs" (US National Center for Education Statistics).

`cip4name` Character, program name at the 4-digit level.

`cip4` Character, 4-digit code (the first 4 digits of `cip6`) representing "intermediate groupings of programs that have comparable content and objectives."

`cip2name` Character, program name at the 2-digit level.

`cip2` Character, 2-digit code (the first 2 digits of `cip6`) representing "the most general groupings of related programs."

Details

The midfielddata taxonomy includes one non-IPEDS code (999999) for Undecided or Unspecified, instances in which institutions reported no program information or that students were not enrolled in a program.

Source

<https://nces.ed.gov/ipeds/cipcode/>

See Also

Other cip-data: [cip2010](#), [fye_proxy](#)

cip2010	<i>Alternate table of academic programs</i>
---------	---

Description

A data table of the 2010 Classification of Instructional Programs (CIP) accessed in 2026 from the US National Center for Education Statistics (NCES). Like the cip data set originally included with midfielldr, cip2010 provides codes and names for instructional programs organized on three levels: a 2-digit series, a 4-digit series, and a 6-digit series.

Usage

cip2010

Format

data.table with 1849 rows and 6 columns keyed by the 6-digit CIP code:

cip6name Character, program name at the 6-digit level

cip6 Character, 6-digit code representing "specific instructional programs" (US National Center for Education Statistics).

cip4name Character, program name at the 4-digit level.

cip4 Character, 4-digit code (the first 4 digits of cip6) representing "intermediate groupings of programs that have comparable content and objectives."

cip2name Character, program name at the 2-digit level.

cip2 Character, 2-digit code (the first 2 digits of cip6) representing "the most general groupings of related programs."

Details

The midfielddata taxonomy includes one non-IPEDS code (999999) for Undecided or Unspecified, instances in which institutions reported no program information or that students were not enrolled in a program.

Source

<https://nces.ed.gov/ipeds/cipcode/>

See Also

Other cip-data: [cip](#), [fye_proxy](#)

completion_status	<i>Determine completion status</i>
-------------------	------------------------------------

Description

Identifies students completing a program in a timely manner. Adds columns to the data frame that support the finding.

Usage

```
completion_status(dframe, midf_table = degree)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble) with required variables {mcid, timely_term}.
midf_table	degree data frame with required variables {mcid, term_degree}.

Details

If a population has been filtered for data sufficiency, then determining every student's *completion status* is feasible. Completing an academic program in a timely manner means that a student completes the requirements for a degree within a set time span, typically 4, 6, or 8 years after admission depending on the definition adopted in a particular study. The term at the end of that span is the *timely completion term*.

If the student's degree term is no later than their timely completion term, then their completion status is "timely"; if later, their status is "late". For students with no degree, completion status is NA.

Redundant columns. To prevent overwriting, a variable such as bacc_term is not added to the data frame if it duplicates an existing variable. The test for redundancy is managed internally by calling rm_redundant_cols() before the final data frame is returned.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Row order is preserved. Rows with NA values in any of the required variables are removed. Duplicated rows are removed.

- New columns are added unless they are redundant (see Details). The new variables are:
 - `bacc_term` Character. Term of a student's first baccalaureate, encoded YYYYT or, if no degree recorded, NA. Joined from the `term_degree` variable in `midf_table`.
 - `completion` Character. Completion status, possible values of "timely", "late", and "NA".

Examples

```
# Assign toy data sets
student <- toy_student
term <- toy_term
degree <- toy_degree

# Start with a selected population
x <- student[c(9:11, 21:30, 344:345), .(mcid)]
x

# timely_term() to add required columns
x <- timely_term(x, midf_table = term)
x <- x[, .(mcid, timely_term)]
x

# Add completion status columns
x <- completion_status(x, midf_table = degree)
x

# No change if added columns duplicate existing
y <- completion_status(x, midf_table = degree)
check_equiv_frames(x, y)

# Filter to retain "timely" rows only
x[completion == "timely"]
```

data_sufficiency	<i>Determine data sufficiency</i>
------------------	-----------------------------------

Description

Identifies records to exclude due to insufficient data. Adds columns to the data frame that support the finding.

Usage

```
data_sufficiency(dframe, midf_table = term)
```

Arguments

<code>dframe</code>	Data frame or data frame extension (e.g., <code>data.table</code> or <code>tibble</code>) with required variables <code>{mcid, entry_term, timely_term}</code> .
<code>midf_table</code>	term data frame with required variables <code>{mcid, term, institution}</code> .

Details

In most studies, the population must satisfy the *data sufficiency* criterion, developed as follows:

- Program *completion* means satisfying the requirements for a first baccalaureate degree.
- Completion *status* is "timely" if accomplished within a set time span, typically 4, 6, or 8 years after admission depending on the definition one adopts. The *timely-completion term* is the term at the end of that span.
- The *data sufficiency* test identifies students whose actual admission term and projected timely completion term both lie within their institution's data range. These are the students for whom completion status—timely or otherwise—can be positively asserted, and are therefore the only students included a population.

To apply this criterion, our heuristic labels a row (keyed by student ID) "exclude-upper" when a student's timely completion term exceeds the upper limit of their institution's data range; "exclude-lower" when their initial term matches the non-summer, lower limit of the data range; and "include" otherwise. The rationale for these specific filters is explained in our data sufficiency article (see references).

Redundant columns. To prevent overwriting, a variable such as sufficiency is not added to the data frame if it duplicates an existing variable. The test for redundancy is managed internally by calling `rm_redundant_cols()` before the final data frame is returned.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Row order is preserved. Rows with NA values in any of the required variables are removed. Duplicated rows are removed.
- New columns are added unless they are redundant (see Details). The new variables are:
 - `data_range` Character. Institution data range, encoded YYYYT-YYYYT, indicating the institution's first and last term in the database. Extracted from `midf_table`.
 - `sufficiency` Character. Data sufficiency results. Possible values are "satisfied", "fail-lower," and "fail-upper."

References

R. Layton, R. Long, M. Ohland, M. Orr, and S. Lord (2026), "Data sufficiency," <https://midfieldr.github.io/midfieldr/articles/art-020-data-sufficiency.html>

Examples

```
# Assign toy data sets
student <- toy_student
term <- toy_term

# Start with a selected population
x <- student[c(9:11, 21:30, 344:345), .(mcid)]
x
```

```

# timely_term() to add required columns
x <- timely_term(x, midf_table = term)
x <- x[, .(mcid, entry_term, timely_term)]
x

# Add data sufficiency columns
x <- data_sufficiency(x, midf_table = term)
x

# No change if added columns duplicate existing
y <- data_sufficiency(x, midf_table = term)
check_equiv_frames(x, y)

# Filter to retain "satisfied" rows only
x[sufficiency == "satisfied"]

```

filter_programs	<i>Choose rows of CIP data</i>
-----------------	--------------------------------

Description

Helps in finding 6-digit program codes. Subsets a CIP data frame, retaining rows that match or partially match any string in a vector of character strings.

Usage

```
filter_programs(dframe, pattern, ..., negate = NULL)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble) with CIP program names and codes, e.g., the cip dataset.
pattern	Character vector of search strings, including regular expressions.
...	Not used for passing values; forces subsequent arguments to be referable only by name.
negate	Logical (default FALSE). If TRUE, inverts the resulting Boolean vector.

Details

Each element of the pattern vector is matched row-wise to every value in dframe using `grepl()`. If `negate = FALSE` (default), a match retains the full row; if `negate = TRUE`, a match removes the full row.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Rows are a subset of the input; row order is preserved. Duplicated rows are removed.
- Columns are not modified.

Examples

```
# Subset using keywords
filter_programs(cip, pattern = "history")

# Subset using codes
filter_programs(cip, pattern = "^54")

# Multiple passes to narrow the results
first_pass <- filter_programs(cip, "math")
first_pass[, .(cip6name, cip6)]

second_pass <- filter_programs(first_pass, c("bio", "educ"), negate = TRUE)
second_pass[, .(cip6name, cip6)]

third_pass <- filter_programs(second_pass, c("^27", "^30"))
third_pass[, .(cip6name, cip6)]

# Multiple passes by chaining
chain_pass <- cip |>
  filter_programs("math") |>
  filter_programs(c("bio", "educ"), negate = TRUE) |>
  filter_programs(c("^27", "^30"))
chain_pass[, .(cip6name, cip6)]
```

fye_proxy

Starting program proxies for FYE students

Description

Proxies are the degree-granting engineering programs we estimate that First-Year Engineering (FYE) students would have declared had they not been required to enroll in FYE. Keyed by student ID. Proxies are provided for all students in the midfielddata practice data who ever enroll in FYE.

Usage

```
fye_proxy
```

Format

data.table with 5789 rows and 2 columns keyed by student ID:

mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."

proxy Character. The 6-digit CIP code of the estimated proxy program.

Details

The proxy variable contains 6-digit CIP codes of degree-granting engineering programs, e.g., Electrical Engineering, Mechanical Engineering, etc., that are substituted for the FYE CIP code when an analysis requires degree-granting starting programs. The most common application is a graduation rate calculation.

The estimation is based on students' earliest non-FYE, degree-granting programs and a multiple imputation suitable for categorical variables using the mice package. The predictor variables are institution, race, and sex. The estimated variable is the 6-digit CIP code of a degree-granting engineering program at their institution.

fye_proxy holds only for the practice data in midfielddata—these values cannot be commingled with the MIDFIELD research database.

See Also

Other cip-data: [cip](#), [cip2010](#)

grade_scale	<i>Grade scale</i>
-------------	--------------------

Description

Data frame of letter grades and conventional point assignments used for computing grade point averages.

Usage

grade_scale

Format

data.table with 12 rows and 2 columns:

letter_grade Character, letter grades using the conventional US scale from A to F.

points Numerical, 4.0 scale of points assigned to letter grades.

See Also

Other scales: [act_sat_scale](#)

initialize_fye_proxies

Initialize FYE proxies for imputing missing data

Description

Conditions data for imputing starting majors of First-Year Engineering (FYE) students. The result is suitably formatted for input to the R mice package for multiple imputation.

Usage

```
initialize_fye_proxies(m_student, m_term, fye_cip = NULL, ..., alt_fye = NULL)
```

Arguments

m_student	Data frame or data frame extension (e.g., data.table or tibble) with required character variables {mcid, race, sex}. Typically the original, unfiltered student source data.
m_term	Data frame or data frame extension (e.g., data.table or tibble) with required character variables {mcid, term, cip6, institution}. Typically the original, unfiltered term source data.
fye_cip	Character, one 6-digit CIP code used for FYE programs. Default "140102", applied to all institutions except those (if any) optionally defined by user in alt_fye.
...	Not used for passing values; forces subsequent arguments to be referable only by name.
alt_fye	Data frame or data frame extension (e.g., data.table or tibble) with character variables {institution, alt_cip}. For users with institutions that use a 6-digit CIP code other than the value in fye_cip for their FYE programs. One FYE code only per institution.

Details

Assembles a data frame of students ever enrolled in First-Year Engineering (FYE) programs. Where practicable, a 6-digit CIP code is added to the data frame as a proxy for the student's preferred engineering major. If indeterminate, the proxy is NA and treated as missing data.

At some US institutions, engineering students are required to complete a First-Year Engineering (FYE) curriculum before they can be admitted to a degree-granting major such as Civil, Electrical, or Mechanical Engineering. This poses a problem when trying to count the number of students starting in one of these programs: the students don't start in Civil, Electrical, or Mechanical Engineering; they start in FYE.

For some metrics—graduation rate for example—correctly identifying starters is imperative. The problem posed by FYE programs is that we don't know the engineering starting majors these students would have selected had FYE not been required. We address the problem by constructing an *FYE proxy* variable.

The FYE proxy has one of two values:

1. If the record of an FYE student includes a degree-granting engineering program, the 6-digit CIP code of the first such program is returned as the student's FYE proxy.
2. If not, the proxy is NA and is treated as a missing value to be imputed later using the R mice package.

This function does not perform the imputation. It produces a data frame suitably formatted for input to the R mice package for multiple imputation.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Rows: One row for every degree-seeking FYE student. Rows in `m_student` or `m_term` with NA values in any of the required variables are removed.
- Columns: Conditioned for later use as an input to the mice R package for multiple imputation as follows:
 - `mcid` Character, anonymized student identifier.
 - `race` Factor, race/ethnicity from the student input data frame. An imputation predictor variable.
 - `sex` Factor, sex from the student input data frame. An imputation predictor variable.
 - `institution` Factor, anonymized institution name from the term data frame. An imputation predictor variable.
 - `proxy` Factor, 6-digit CIP code of a student's known, first degree-granting engineering program or NA representing missing values to be imputed.

Examples

```
library(data.table)

# Subset student and term data using selected IDs
IDs <- c("MCID3112319668", "MCID3112214437", "MCID3112328548",
        "MCID3111447797", "MCID3111566004", "MCID3111697452",
        "MCID3112268500", "MCID3112320295")
student <- select_basic_cols(toy_student[mcid %chin% IDs])
term <- select_basic_cols(toy_term[mcid %chin% IDs])

# Obtain results
proxy <- initialize_fye_proxies(student, term)
proxy

# ----- Examine details
# Note: the CIP code and name for FYE is 140102 Pre-Engineering

# Join program names to term data for display
term_seq <- cip[term, .(mcid, term, cip6, cip6name), on = "cip6", nomatch = NULL]

# Function to display results for individual students
f <- function(IDs, i) {
  cat(paste("Student", i, "record\n"))
}
```

```

    print(term_seq[mcid == IDs[i]])
    cat("\ninitialize_fye_proxies() results\n")
    print(proxy[mcid == IDs[i]])
}

# Example 1: Non-Engineering -> FYE -> Engineering
# 400501 (Chemistry) -> FYE -> 140701 (Chemical Engng)
# FYE proxy is 140701
f(IDs, 1)

# Example 2: FYE -> Engineering -> Non-Engineering
# FYE -> 140901 (Computer Engng) -> 450601 (Economics)
# FYE proxy is 140901
f(IDs, 2)

# Example 3: FYE -> Engineering
# FYE -> 141001 (Electrical Engng)
# FYE proxy is 141001
f(IDs, 3)

# Example 4: FYE -> Engineering -> Engineering
# FYE -> 141901 (Mechanical Engng) -> 143501 (Industrial Engng)
# FYE proxy is 141901
f(IDs, 4)

# Example 5: Non-Engineering -> FYE -> Leaves the database
# 240102 (General Studies) -> FYE
# FYE proxy is NA
f(IDs, 5)

# Example 6: FYE -> Leaves the database
# FYE proxy is NA
f(IDs, 6)

# Example 7: Non-Engineering -> FYE -> Non-Engineering
# 240102 (General Studies) -> FYE -> 110101 (Computer Science)
# FYE proxy is NA
f(IDs, 7)

# Example 8: FYE -> Non-Engineering
# FYE -> 230101 (English Literature)
# FYE proxy is NA
f(IDs, 8)

```

look_at

*Display structure***Description**

A wrapper on `base::str()` with arguments set to not show attributes, to not show length, and to cut the width.

Usage

```
look_at(x)
```

Arguments

x Any R object.

Value

Does not return anything. The side effect is to output to the terminal.

Examples

```
# data frames
look_at(cip)
look_at(toy_degree)

# character vectors
x <- sort(unique(toy_degree$institution))
look_at(x)
```

order_multiway	<i>Order multiway categories</i>
----------------	----------------------------------

Description

Conditions multiway data for Cleveland multiway charts. Two independent categorical variables are converted to factors with their levels ordered by the single quantitative response variable.

Usage

```
order_multiway(
  dframe,
  quantity,
  categories,
  ...,
  method = NULL,
  ratio_of = NULL
)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble) with the following required variables: two independent categorical variables, one quantitative response variable, and, if method = percent, its dividend and divisor variables.
quantity	Character. Name of the single multiway quantitative variable.

categories	Character. Vector of names of the two multiway categorical variables, in any order.
...	Not used for passing values; forces subsequent arguments to be referable only by name.
method	Character. Method of ordering the levels of the categories; possible values are "median" (default) or "percent". The median method determines medians of the quantitative column grouped by category. The percent method sums dividends and divisors by category and calculates their quotients by category.
ratio_of	Character. Vector of column names of the dividend and the divisor that produced the quantitative variable. Names must be in order, as in <code>c(dividend, divisor)</code> . Required when <code>method = "percent,"</code> ignored otherwise.

Details

Multiway data comprise a single quantitative value (or response) for every combination of levels of two categorical variables. The ordering of the rows and panels, based on the response quantity, is crucial to the perception of effects (Cleveland, 1993).

Multiway data comprise three variables: a categorical variable of m levels; a second independent categorical variable of n levels; and a quantitative variable (or *response*) of length $m \times n$ that cross-classifies the categories, that is, there is a value of the response for each combination of levels of the two categorical variables. If a response value is missing, it is assumed that a response for every combination is at least feasible.

In a multiway dot plot, one category is encoded by the panels, the second category is encoded by the rows of each panel, and the quantitative variable is encoded along identical horizontal scales.

Redundant columns. To prevent overwriting, a variable such as `program_median` is not added to the data frame if it duplicates an existing variable. The test for redundancy is managed internally by calling `rm_redundant_cols()` before the final data frame is returned.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Row order is preserved. Duplicated rows are removed.
- Column specified by `quantity` is converted to type double. Columns specified by `categories` are converted to factors and ordered.
- New columns are added unless they are redundant (see Details). The new columns, with capital letters denoting placeholders, have the form:
 - CATEGORY_1_LABEL
 - CATEGORY_2_LABEL

The CATEGORY placeholder is replaced with the column names from `categories`. The LABEL placeholder depends on the method. For example, suppose `categories = c("program", "people")`. For the median method, the new column names would be `program_median` and `people_median`. For the percent method, the new column names would be `program_metric` and `people_metric`, indicating that the metric had been calculated for each category independently.

References

Cleveland WS (1993). *Visualizing Data*. Hobart Press, Summit, NJ.

Examples

```
# Reconfigure built-in data set
DT <- case_results[grepl("^Elec|^Mech", program)]
DT <- DT[grepl("^Asian|^Black|^Hispanic|^White", people)]
DT

# Order factor levels by median
DT1 <- data.table::copy(DT)
DT1 <- DT1[, c("ever", "grad") := NULL]
mw1 <- order_multiway(DT1,
                      quantity = "stick",
                      categories = c("program", "people"))
data.table::setorderv(mw1, c("program_median", "people_median"))
mw1

# Levels in increasing order
levels(mw1$program)
levels(mw1$people)

# No change if added columns are redundant
mw1a <- order_multiway(mw1,
                      quantity = "stick",
                      categories = c("program", "people"))
check_equiv_frames(mw1, mw1a)

# Retain the `ratio_of` variables and order by percentage
mw2 <- order_multiway(DT,
                      quantity = "stick",
                      categories = c("program", "people"),
                      method = "percent",
                      ratio_of = c("grad", "ever"))
data.table::setorderv(mw2, c("program_metric", "people_metric"))
mw2

# Order of factor levels depends on the method. Here, for example,
# program levels are the same for median and percent methods,
all.equal(levels(mw1$program), levels(mw2$program))

# but people levels do not have the same order.
all.equal(levels(mw1$people), levels(mw2$people))
levels(mw1$people)
levels(mw2$people)
```

Description

Data frame of IDs after processing the practice data for data sufficiency and degree seeking. Provides a convenient bloc to start many of the analysis illustrated in the package articles.

Usage

```
population_baseline
```

Format

data.table with 76875 rows and 1 column:

mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."

See Also

Other case-study-data: [case_blocs](#), [case_results](#)

pre_or_post_bacc	<i>Label terms as pre- or post-baccalaureate</i>
------------------	--

Description

Distinguishes between pre- and post-baccalaureate terms. Adds columns to the data frame that support the finding.

Usage

```
pre_or_post_bacc(dframe, midf_table = degree)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble) with required variables {mcid} and one of the following: {term, term_course, term_degree}.
midf_table	degree data frame with required variables {mcid, term_degree}.

Details

The dividing event between pre- and post-baccalaureate terms is the student's first bachelor's degree. Post-baccalaureate records are usually dropped as a step in constructing baseline records for further analysis. The data frame for the principal argument is the term, course, or degree data table with all columns. Can be applied however to any data frame containing the required variables.

Redundant columns. To prevent overwriting, a variable such as bacc_term is not added to the data frame if it duplicates an existing variable. The test for redundancy is managed internally by calling rm_redundant_cols() before the final data frame is returned.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Row order is preserved. Rows with NA values in any of the required variables are removed. Duplicated rows are removed.
- New columns are added unless they are redundant (see Details). The new variables are:
 - `bacc_term` Character. Term of a student's first baccalaureate, encoded YYYYT or, if no degree recorded, NA. Joined from the `term_degree` variable in `midf_table`.
 - `pre_or_post` Character. Term label. Possible values are "pre-bacc" (pre-baccalaureate/undergraduate) and "post-bacc."

Examples

```
# For illustration only, choose a minimum set of columns
term <- toy_term[, .(mcid, term)]
course <- toy_course[, .(mcid, term_course)]
degree <- toy_degree[, .(mcid, term_degree)]

# Example result
term

# Add term ID columns
x <- pre_or_post_bacc(term, midf_table = degree)
x[order(-pre_or_post)]

# No change if added columns duplicate existing
y <- pre_or_post_bacc(x, midf_table = degree)
check_equiv_frames(x, y)

# Filter to retain "undergrad" rows only
x[pre_or_post == "pre-bacc"]

# Function is applied to all tables containing a term-value
term <- pre_or_post_bacc(term, midf_table = degree)
course <- pre_or_post_bacc(course, midf_table = degree)
degree <- pre_or_post_bacc(degree, midf_table = degree)
```

rm_redundant_cols	<i>Remove redundant columns</i>
-------------------	---------------------------------

Description

Subset a data frame to remove redundant columns. The goal of this function is to prevent over-writing existing columns and to avoid duplicating columns. Primarily used internally in `midfieldr` functions that add columns to a data frame.

Usage

```
rm_redundant_cols(dframe)
```

Arguments

dframe Data frame or data frame extension (e.g., data.table or tibble)

Details

Several `midfieldr` functions add columns to a working data frame. If the name of a new variable matches that of an existing variable, the new variable name acquires a suffix ".1", ".2", etc., provided by `base::make.unique()`. Before the final data frame is returned, all variable names (including existing variables) are temporarily split from their dot suffixes, if any. Resulting columns are dropped if they duplicate a previous column (where 'previous' means to the left of the duplicate in the data frame).

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Rows are preserved.
- Unique columns preserved. Redundant columns dropped.

Examples

```
# Construct a sample data frame
mcid <- c("mc_01", "mc_02", "mc_03")
term <- c("19911", "19912", "19913")
x <- data.frame(mcid, term)

# In the following, column names "var" and "var.i"
# represent identical names except for the suffix ".i"

# No effect if columns are unique
x
x <- rm_redundant_cols(x)
x

# Desired effect: drop new "var.i" if it duplicates existing "var"
x$term.1 <- x$term
x
x <- rm_redundant_cols(x)
x

# Retain "var.i" when "var" does not exist
x <- data.frame(mcid, term.1 = term)
x
x <- rm_redundant_cols(x)
x
```

```

# The leftmost of the redundant columns is retained
x$term <- term
x
x <- rm_redundant_cols(x)
x

# Redundancy is checked for the dot-separator only
x <- data.frame(mcid, term, term_1 = term, term.1 = term)
x
x <- rm_redundant_cols(x)
x

# When a midfieldr function adds a variable, e.g., `bacc_term`, and
# a subsequent operation adds it again, `bacc_term.1`, the second
# addition is usually expected to be redundant and dropped.
x$term_1 <- NULL
x$bacc_term <- c("19951", "19951", "19951")
x$bacc_term.1 <- c("19951", "19951", "19951")
x
x <- rm_redundant_cols(x)
x

# However, if the suffixed column remains, then the two columns
# have different values---likely indicating an error in the
# operations leading up to this point.
x$bacc_term.1 <- c("19953", "19951", "19951")
x
x <- rm_redundant_cols(x)
x

```

select_basic_cols	<i>Choose columns of student records</i>
-------------------	--

Description

Subset a MIDFIELD data table to retain the variables required by one or more midfieldr functions. Variables that constitute the key or composite key for a table are retained as well. A convenience function to reduce the number of columns displayed.

Usage

```
select_basic_cols(dframe)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble) equivalent to or derived from one of the MIDFIELD data tables: {student, term, course, degree}.
--------	---

Details

Functions in `midfieldr` with a `MIDFIELD` dataset argument—such as `student`, `term`, etc.—typically require only a few of the columns available in the table. Depending on which table is input, the following columns are returned if present:

- `student`: {`mcid`, `race`, `sex`}
- `term`: {`mcid`, `term`, `cip6`, `institution`, `level`}
- `course`: {`mcid`, `term_course`, `abbrev`, `number`}
- `degree`: {`mcid`, `term_degree`, `cip6`}
- Combination of the above if `dframe` contains columns from multiple tables.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Rows are not modified.
- Columns are a subset of the input, appearing in the same order.

Examples

```
# Basic usage
select_basic_cols(toy_student[1:5])
select_basic_cols(toy_term[1:5])
select_basic_cols(toy_course[1:5])
select_basic_cols(toy_degree[1:5])

# If the input is not strictly one of the four MIDFIELD data
# tables, all possible required columns are returned.
x <- toy_student[toy_degree, on = c("mcid")][1:5]
select_basic_cols(x)

# Required columns can only be returned if present,
# e.g., consider the result for a full table:
select_basic_cols(toy_term)

# Compared to the result for a subset of the same table:
y <- toy_term[, .(mcid, term, cip6, hours_term, gpa_term)]
select_basic_cols(y)
```

sort_uniq

Extract unique elements and sort

Description

A strict version of `sort()` and `unique()` (without ...) applied to vectors only.

Usage

```
sort_uniq(x, ..., na.rm = FALSE, decreasing = FALSE, na.last = FALSE)
```

Arguments

x	Vector of values to be sorted with any duplicate values removed.
...	Not used for passing values; forces subsequent arguments to be referable only by name.
na.rm	Logical. Indicates if missing values (including NaN) should be removed. Passed to unique().
decreasing	Logical. Should the sort be increasing or decreasing? Passed to sort().
na.last	Logical. Position of NA values. Passed to sort().

Value

A vector of unique values, sorted.

Examples

```
# Character vector
x <- toy_student$race
sort_uniq(x)

# Numeric vector
x <- toy_term$hours_term_attempt
sort_uniq(x)
```

timely_term	<i>Determine timely completion terms</i>
-------------	--

Description

Determines the latest term by which program completion would be considered timely. Adds columns to the data frame that support the finding.

Usage

```
timely_term(dframe, midf_table = term, ..., span = NULL, sched_span = NULL)
```

Arguments

dframe	Data frame or data frame extension (e.g., data.table or tibble) with required variable {mcid}.
midf_table	term data frame with required variables {mcid, term, level}.
...	Not used for passing values; forces subsequent arguments to be referable only by name.
span	Integer scalar (default 6), number of years to define timely completion, typically 4, 6, or 8 years (100%, 150%, 200% respectively of sched_span).#'
sched_span	Integer scalar (default 4), the number of years an institution officially schedules for completing a program.

Details

Completing an academic program in a "timely" manner means that a student completes the requirements for a degree within a set time span, typically 4, 6, or 8 years after admission depending on the definition adopted in a particular study. The final term of that span is the *timely completion term*.

Our heuristic assigns a time span of 6 academic years for timely completion (other values can be assigned via the span argument). For students admitted at second-year level or higher, the span value is reduced by one academic year for each full year the student is assumed to have completed. The adjusted span is added to their initial term at an institution to create the `timely_term` value for each observation.

Redundant columns. To prevent overwriting, a variable such as `timely_term` is not added to the data frame if it duplicates an existing variable. The test for redundancy is managed internally by calling `rm_redundant_cols()` before the final data frame is returned.

Value

Data frame with the following properties:

- Data frame class is preserved. Groups and keys are not preserved.
- Row order is preserved. Rows with NA values in any of the required variables are removed. Duplicated rows are removed.
- New columns are added unless they are redundant (see Details). The new variables are:
 - `entry_term` Character. Initial term of a student's longitudinal record, encoded YYYYT. Extracted from `midf_table`.
 - `entry_level` Character. Student level (01 Freshman, 02 Sophomore, etc.) in their initial term. Extracted from `midf_table`.
 - `adj_span` Numeric. Integer span of years for timely completion adjusted for a student's initial level.
 - `timely_term` Character. Latest term by which program completion would be considered timely. Encoded YYYYT.

Examples

```
# Assign toy data sets
student <- toy_student
term <- toy_term

# Start with a selected population
x <- student[c(9:11, 21:30, 344:345), .(mcid)]
x

# Add timely term columns
x <- timely_term(x, midf_table = term)
x

# No change if added columns duplicate existing
y <- timely_term(x, midf_table = term)
check_equiv_frames(x, y)
```

toy_course	<i>Small 'course' dataset for examples</i>
------------	--

Description

A subset of rows from the midfelddata course table matching the IDs in toy_student. A small dataset for use in examples.

Usage

```
toy_course
```

Format

- Data frame with 8950 rows and 12 columns (data.table class). Composite key: {mcid, term_course, abbrev, number}.
- mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."
 - term_course Character. Academic year and term, encoded YYYYT.
 - abbrev Character. Course alphabetical identifier, e.g. "ENGR", "MATH", "ENGL."
 - number Character. Course numeric identifier, e.g. "101", "3429."
 - institution Character. The anonymized name of the institution the student attended in a given term, e.g., "Institution A", "Institution B", etc.
 - course Character. Course name, e.g., "Astrophysics III", "Calculus For Social Science And Business", "Corp Financial Rprtnng 1", "Environmental Sanitation II", "Fitness and Wellness", "Introductory Astronomy 2", "Our Changing Environment", etc.
 - section Character. Course section identifier, from one to four characters, e.g., "1", "2", "01", "14", "001", "040", "785", "H02", "R01", "300E", "888R", etc.
 - type Character. Predominant delivery method for this section, e.g., "Blended", "Distance Education", "Face-to-Face", "Online", etc.

faculty_rank Character. The academic rank of the person teaching the course, e.g., "Assistant Professor", "Associate Professor", "Graduate Assistant", "Visiting Faculty", etc.

hours_course Numeric. Number of credit-hours for successful course completion.

grade Character. Course grade, e.g., "A+", "A", "A-", "B+", "I", "NG", etc.

discipline_midfield Character. A variable for grouping courses by academic discipline assigned by the pre-2023 MIDFIELD data curator, e.g., "Anthropology", "Business", "Computer Science", "Engineering", "Language and Literature", "Mathematics", "Visual and Performing Arts", etc.

See Also

Other toy-data: [toy_degree](#), [toy_student](#), [toy_term](#)

toy_degree

Small 'degree' dataset for examples

Description

A subset of rows from the midfelddata degree table that comprises those students from the toy_student dataset who complete a program. A small dataset used in examples.

Usage

toy_degree

Format

Data frame with 193 rows and 4 columns (data.table class). Composite key: {mcid, term_degree, cip6}.

mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."

term_degree Character. Academic year and term in which a student completes their program, encoded YYYYT.

cip6 Character. The 6-digit CIP code of the program that the student completes in this term.

institution Character. The anonymized name of the institution the student attended in a given term, e.g., "Institution A", "Institution B", etc.

degree Character. Type of degree awarded, e.g., "Bachelor of Arts in Geography", "Bachelor of Science in Finance," etc.

See Also

Other toy-data: [toy_course](#), [toy_student](#), [toy_term](#)

toy_student

*Small 'student' dataset for examples***Description**

A subset of rows from the midfielddata student table. A small dataset for use in examples.

Usage

```
toy_student
```

Format

Data frame with 351 rows and 13 columns (data.table class). Key: {mcid}.

mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."

race Character. Race/ethnicity as self-reported by the student, e.g., "Asian", "Black", "Hispanic", etc.

sex Character. Sex as self-reported by the student, possible values are "Female", "Male", and "Unknown."

institution Character. The anonymized name of the institution the student attended in a given term, e.g., "Institution A", "Institution B", etc.

transfer Character. Transfer status, possible values are "First-Time in College", "First-Time Transfer."

hours_transfer Numeric. Number of credit hours transferred (or NA).

age_desc Character. Age group, possible values are "25 and Older", "Under 25."

us_citizen Character. US citizenship, possible values are "No", "Yes."

home_zip Character. Home ZIP code (or NA), e.g., "02056", "20170", "51301", "80129", etc.

high_school Character. Code for the last high school attended before admission (or NA), e.g., "060075", "210512", "431800", "502195", etc.

sat_math Numeric. SAT mathematics test score (or NA).

sat_verbal Numeric. SAT reading test score (or NA).

act_comp Numeric. ACT composite test score (or NA).

See Also

Other toy-data: [toy_course](#), [toy_degree](#), [toy_term](#)

toy_term	<i>Small 'term' dataset for examples</i>
----------	--

Description

A subset of rows from the midfielddata term table matching the IDs in toy_student. A small dataset for use in examples.

Usage

```
toy_term
```

Format

Data frame with 1821 rows and 13 columns (data.table class). Composite key: {mcid, term}.

mcid Character. Anonymized student identifier that connects the four data tables, e.g., "MCID3111142897."

term Character. Academic year and term the student attended, encoded YYYYT.

cip6 Character. The 6-digit CIP code of the program in which a student is enrolled in this term.

institution Character. The anonymized name of the institution the student attended in a given term, e.g., "Institution A", "Institution B", etc.

level Character. Academic level of the student at the end of this term, e.g., "01 First-Year", "02-Second Year", etc.

standing Character. Academic standing during the reported term, e.g., "Good Standing", "Academic Warning", etc.

coop Character. Cooperative education term, possible values are "Yes", "No."

hours_term Numeric. Credit hours earned in the term.

hours_term_attempt Numeric. Credit hours attempted in the term.

hours_cumul Numeric, cumulative credit hours earned.

hours_cumul_attempt Numeric. Cumulative credit hours attempted.

gpa_term Numeric. Term grade point average.

gpa_cumul Numeric. Cumulative grade point average.

See Also

Other toy-data: [toy_course](#), [toy_degree](#), [toy_student](#)

Index

- * **case-study-data**
 - case_blocs, 5
 - case_results, 5
 - population_baseline, 20
- * **cip-data**
 - cip, 7
 - cip2010, 8
 - fye_proxy, 13
- * **datasets**
 - act_sat_scale, 2
 - case_blocs, 5
 - case_results, 5
 - cip, 7
 - cip2010, 8
 - fye_proxy, 13
 - grade_scale, 14
 - population_baseline, 20
 - toy_course, 28
 - toy_degree, 29
 - toy_student, 30
 - toy_term, 31
- * **scales**
 - act_sat_scale, 2
 - grade_scale, 14
- * **toy-data**
 - toy_course, 28
 - toy_degree, 29
 - toy_student, 30
 - toy_term, 31
- act_sat_scale, 2, 14
- add_completion_status, 3
- add_data_sufficiency
 - (add_completion_status), 3
- add_timely_term
 - (add_completion_status), 3
- case_blocs, 5, 6, 21
- case_results, 5, 5, 21
- catch_error, 6
- cip, 7, 9, 14
- cip2010, 8, 8, 14
- completion_status, 9
- completion_status(), 4
- data_sufficiency, 10
- data_sufficiency(), 4
- filter_cip (add_completion_status), 3
- filter_programs, 12
- filter_programs(), 4
- fye_proxy, 8, 9, 13
- grade_scale, 3, 14
- initialize_fye_proxies, 15
- look_at, 17
- midfieldr-deprecated
 - (add_completion_status), 3
- order_multiway, 18
- population_baseline, 5, 6, 20
- pre_or_post_bacc, 21
- prep_fye_mice (add_completion_status), 3
- rm_redundant_cols, 22
- select_basic_cols, 24
- select_basic_cols(), 4
- select_required
 - (add_completion_status), 3
- sort_uniq, 25
- timely_term, 26
- timely_term(), 4
- toy_course, 28, 29–31
- toy_degree, 29, 29–31
- toy_student, 29, 30, 31
- toy_term, 29, 30, 31