Package: grouper (via r-universe)

July 9, 2026

Type Package

Title Optimal Group Assignment and Workload Allocation

Version 0.7.3

Description Integer programming models to assign students to groups by maximising diversity or topic preferences, and to allocate multi-role teaching workloads while balancing role demand, preferences, fairness, and cohort protection.

License MIT + file LICENSE

URL https://Zimmy313.github.io/grouper/, https://github.com/Zimmy313/grouper

BugReports https://github.com/Zimmy313/grouper/issues

Encoding UTF-8

LazyData true

Suggests knitr, ompr.roi, pkgdown, rmarkdown, ROI.plugin.glpk, ROI.plugin.highs, testthat (>= 3.0.0)

VignetteBuilder knitr

Imports cluster, dplyr, magrittr, ompr, rlang, yaml

Depends R (>= 3.5)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/roxygen2/markdown TRUE

RoxygenNote 7.3.3

Repository https://zimmy313.r-universe.dev

Date/Publication 2026-07-08 05:09:44 UTC

RemoteUrl https://github.com/zimmy313/grouper

RemoteRef HEAD

RemoteSha 1f8eb7206050c05d54a8f7c7f9fd53c60c2f9a55

Contents

assign_groups . . . 2
assign_job . . . 3
compute_diversity . . . 4
convert_pref_mat . . . 4
dba_gc_ex001 . . . 5
dba_gc_ex003 . . . 5
dba_gc_ex004 . . . 6
extract_info . . . 6
extract_multirole_info . . . 7
extract_params_yaml . . . 9
extract_student_info . . . 9
get_group_pref_score . . . 11
multirole_demand_ex001 . . . 12
multirole_prefmat_ex001 . . . 12
multirole_students_ex001 . . . 13
pba_gc_ex002 . . . 13
pba_prefmat_ex002 . . . 14
prepare_diversity_model . . . 14
prepare_model . . . 15
prepare_multirole_model . . . 16
prepare_preference_model . . . 17
solve_assignment . . . 18
summary_dba . . . 19
summary_pba . . . 20

Index **21**

`assign_groups` *Assigns model result to the original data frame.*

Description

From the result of [ompr::solve_model()], this function attaches the derived groupings to the original dataframe comprising students.

Usage

```
assign_groups(
  model_result,
  assignment = c("diversity", "preference"),
  dframe,
  params_list,
  group_names
)
```

Arguments

model_result	The output solution objection.
assignment	Character string indicating the type of model that this dataset is for. The argument is either 'preference' or 'diversity'. Partial matching is fine.
dframe	The original dataframe used in [extract_student_info()].
params_list	The list of parameters from the YAML file, i.e. the output of [extract_params_yaml()]. This is only required for the preference-based assignment.
group_names	A character string. It denotes the column name in the original dataframe containing the self-formed groups. Note that we need the string here, not the integer position, since we are going to join with it.

Value

A data frame with the group assignments attached to the original group composition dataframe.

---

assign_job *Convert workload allocation to a manual-style wide table*

Description

Creates one row per student and one column per course-role pair, with units allocated by the solver.

Usage

```
assign_job(model_result, student_df, course_codes, name_col = "Name")
```

Arguments

model_result	Result object from 'ompr::solve_model()' for a PhD or multi-role workload model.
student_df	A data frame that contains student name information. Every row is a unique student.
course_codes	Character vector of course codes in the same order as preference-matrix columns and 'd_mat' rows.
name_col	Student name column name in 'student_df'.

Value

A data frame with columns: 'Name', then all '<course>-t', all '<course>-g', all '<course>-e'.

compute_diversity *Compute total pairwise diversity for a set of students*

Description

Compute total pairwise diversity for a set of students

Usage

```
compute_diversity(id, dmat)
```

Arguments

id	Integer vector of student indices into ‘dmat‘.
dmat	Numeric distance matrix (students × students).

Value

Scalar: sum of upper-triangle distances among ‘id‘.

convert_pref_mat *Convert a preference matrix to rank-based scores*

Description

Transforms raw preference ranks so that higher values indicate stronger preference. A rank of 1 maps to ‘n_topics * B‘, rank 2 to ‘n_topics * B - 1‘, and so on.

Usage

```
convert_pref_mat(pref_mat, n_topics, B)
```

Arguments

pref_mat	Numeric matrix of preference ranks (groups × columns).
n_topics	Integer. Number of base topics.
B	Integer. Number of subtopic subgroups per topic.

Value

Numeric matrix of the same dimensions as ‘pref_mat‘.

dba_gc_ex001 *DBA Group Composition Data Example 001*

Description

An example dataset to use with the diversity-based assignment model.

Usage

```
dba_gc_ex001
```

Format

'dba_gc_ex001' A data frame with 4 rows and 4 columns.

* id: the student id of each students, simply the integers 1 to 4. * major: the primary major of each student. * skill: the skill level of each student. * groups: the self-formed groups submitted by each student. In this case, student is in his/her own group.

Source

This dataset was constructed by hand.

dba_gc_ex003 *DBA Group Composition Data Example 003*

Description

An example dataset to use with the diversity-based assignment model. It is used to demonstrate the use of a custom dissimilarity matrix.

Usage

```
dba_gc_ex003
```

Format

'dba_gc_ex003' A matrix with 4 rows and 4 columns

* id: the student id of each students, simply the integers 1 to 4. * self_groups: The self-formed groups * year, major: demographics used in computing dissimilarities

Source

This dataset was constructed by hand.

dba_gc_ex004 *DBA Group Composition Data Example 004*

Description

An example dataset to use with the diversity-based assignment model. It is used to demonstrate the use of a vectors to indicate individual group size constraints for specific topics.

Usage

```
dba_gc_ex004
```

Format

‘dba_gc_ex004‘ A matrix with 5 rows and 4 columns

* id: the student id of each students, simply the integers 1 to 4. * self_groups: The self-formed groups * python: Python skill level - 1 is lowest, 3 is highest.

Source

This dataset was constructed by hand.

extract_info *Extract model inputs (wrapper)*

Description

Wrapper around [extract_student_info()], [extract_phd_info()], and [extract_multirole_info()].

Usage

```
extract_info(
  assignment = c("diversity", "preference", "phd", "multirole"),
  ...
)
```

Arguments

assignment	Character string indicating model type. Must be one of ‘"diversity"‘, ‘"preference"‘, ‘"phd"‘, or ‘"multirole"‘.
...	Additional arguments for the underlying extraction functions. See Details.

Details

Explicit argument guide by assignment:

- For ‘assignment = "diversity"‘, ‘extract_info()‘ forwards ‘...‘ to [extract_student_info()].

Required arguments: - ‘dframe‘ - ‘self_formed_groups‘ - either: - ‘d_mat‘, or - ‘demographic_cols‘, so Gower dissimilarity is computed internally

Optional arguments: - ‘skills‘, which can be supplied or set to ‘NULL‘

- For ‘assignment = "preference"‘, ‘extract_info()‘ forwards ‘...‘ to [extract_student_info()].

Required arguments: - ‘dframe‘ - ‘self_formed_groups‘ - ‘pref_mat‘

- For ‘assignment = "phd"‘, ‘extract_info()‘ forwards ‘...‘ to [extract_phd_info()].

Required arguments: - ‘student_df‘ - ‘p_mat‘ - ‘d_mat‘

Optional arguments: - ‘e_mode‘, which uses the default from [extract_phd_info()] - ‘C‘, which uses the default from [extract_phd_info()] - ‘s‘, which uses the default from [extract_phd_info()]

- For ‘assignment = "multirole"‘, ‘extract_info()‘ forwards ‘...‘ to [extract_multirole_info()].

Required arguments: - ‘student_df‘ - ‘d_mat‘

Optional arguments: - ‘p_ta_mat‘ and ‘p_gr_mat‘ - ‘e_mode‘, ‘C‘, ‘s‘, and ‘single_semester‘

This wrapper does not parse YAML files. YAML-based parameter extraction remains available via [extract_params_yaml()].

Value

A model input list from the corresponding extraction function.

---

extract_multirole_info

Extract inputs for the multi-role workload allocation model

Description

Converts individual-level data, role-specific preference matrices, and role demand into the list expected by [prepare_multirole_model()].

Usage

```
extract_multirole_info(
  student_df,
  d_mat,
  p_ta_mat = NULL,
  p_gr_mat = NULL,
  e_mode = c("rr", "none"),
  C = 4,
  s = c(-1, 0, 1, 2),
  single_semester = FALSE
)
```

Arguments

student_df	A data frame with one row per individual. By default, its first four columns must be named ‘student_id‘, ‘year‘, ‘past_ta‘, and ‘past_gr‘, in that order. With ‘single_semester = TRUE‘, only ‘student_id‘ and ‘year‘ are required as the first two columns. ‘year‘ is capped to the range 1-4.
d_mat	Numeric demand matrix with ‘Nj‘ rows and two or three columns. Columns are interpreted as TA, GR, and optional E.
p_ta_mat	Optional numeric TA preference matrix with dimensions ‘Ns x Nj‘.
p_gr_mat	Optional numeric GR preference matrix with dimensions ‘Ns x Nj‘.
e_mode	How to handle E demand when ‘d_mat‘ has no E column. ‘"rr"‘ computes E demand by round-robin allocation from highest to lowest GR demand; ‘"none"‘ sets E demand to zero.
C	Semester workload capacity per individual. It is stored in the extracted input and used by [prepare_multirole_model()] to set annual workload to ‘2 * C‘. It also determines E demand when ‘e_mode = "rr"‘.
s	Numeric vector containing E-allocation scores for Years 1, 2, 3, and 4. Larger values make E allocation more attractive when the ‘phi‘ term is active.
single_semester	Logical flag. When ‘TRUE‘, supplied past-workload columns are ignored and extraction returns synthetic prior workloads ‘t1 = 0‘ and ‘g1 = C‘ for every individual.

Details

Preference matrices are optional during extraction because their objective terms can be disabled in [prepare_multirole_model()]. When ‘beta_ta‘ or ‘beta_gr‘ is active, the corresponding matrix must be present.

Input order must already be aligned: row ‘i‘ in each preference matrix must correspond to row ‘i‘ in ‘student_df‘, and demand row ‘j‘ must correspond to preference column ‘j‘.

In single-semester mode, the uniform synthetic GR workload does not change the GR workload spread. It fills the prior-semester half of annual capacity, leaving ‘C‘ units per individual for current allocation.

Value

A list containing ‘Ns‘, ‘Nj‘, ‘C‘, ‘P_ta‘, ‘P_gr‘, ‘d‘, ‘s‘, ‘year‘, ‘t1‘, and ‘g1‘.

Examples

```
inputs <- extract_multirole_info(
  student_df = multirole_students_ex001,
  d_mat = multirole_demand_ex001,
  p_ta_mat = multirole_prefmat_ex001,
  p_gr_mat = multirole_prefmat_ex001,
  e_mode = "none"
)
```

extract_params_yaml *Extract parameters from a YAML file*

Description

The remaining parameters for the models are retrieved from a YAML file, so as not to clutter the argument list for [extract_student_info()].

Usage

```
extract_params_yaml(fname, assignment = c("diversity", "preference"))
```

Arguments

fname	A YAML file containing the remaining parameters.
assignment	Character string indicating the type of model that this dataset is for. The argument is either 'preference' or 'diversity'. Partial matching is fine.

Value

For the diversity+skill-based assignment, this function returns a list containing:

* n_topics: the number of topics * R: the optimally desired number of repetitions per topic * nmin: the minimum number of students per topic, * nmax: the maximum number of students per topic, * rmin: the minimum number of repetitions per topic, * rmax: the maximum number of repetitions per topic.

For the preference-based assignment, this function returns a list containing:

* n_topics: the number of topics * R: the optimally desired number of repetitions per topic * nmin: the minimum number of students per topic, * nmax: the maximum number of students per topic, * rmin: the minimum number of repetitions per topic, * rmax: the maximum number of repetitions per topic.

extract_student_info *Extract student information*

Description

Converts a dataframe with information on students to a list of parameters. This list forms one half of the inputs to prepare_model(). The remaining model parameters can come from [extract_params_yaml()] or be supplied directly to [prepare_model()] for non-YAML workflows.

Usage

```
extract_student_info(
  dframe,
  assignment = c("diversity", "preference"),
  self_formed_groups,
  demographic_cols,
  skills,
  pref_mat,
  d_mat
)
```

Arguments

dframe	A dataframe with one row for each student. The columns could possibly contain demographic variables, an overall skill measure, and a column indicating self-formed groups. It is best to have an id column to identify each student.
assignment	Character string indicating the type of model that this dataset is for. The argument is either 'preference' or 'diversity'. Partial matching is fine.
self_formed_groups	An integer column that identifies the self-formed groups, submitted by students.
demographic_cols	A set of integers indicating the columns corresponding to demographic information, e.g. major, year of study, gender, etc. This argument is only used by the diversity-based assignment.
skills	A numeric measure of overall skill level (higher means more skilled). This argument is only used by the diversity-based assignment. This argument can be set to NULL. If this is done, then the model used only maximises the diversity.
pref_mat	The preference matrix with dimensions equal to the num of groups x B*T, where T is the number of topics and B is the number of sub-groups per topic. This argument is only used in the preference-based assignment. See the Details section for more information.
d_mat	The dissimilarity matrix with number of rows equal to the number of students. This matrix should be symmetric, with diagonals equal to 0. This argument is only used in the diversity-based assignment. If it is not provided, the "Gower" distance from the cluster package is used. If this is provided, then demographic_cols is ignored.

Details

For the diversity-based assignment, the demographic variables are converted into an NxN dissimilarity matrix. By default, the dissimilarity metric used is the Gower distance [cluster::daisy()].

For the preference-based assignment, the preference matrix indicates the preference that each group has for the project topics. For this model, each topic has possibly B sub-groups. The number of columns of this matrix must be B*T. Suppose there are T=3 topics and B=2 sub-groups per topic. Then the order of the sub-topics should be:

T1S1, T2S1, T3S1, T1S2, T2S2, and T3S2.

Note that higher values in the preference matrix reflect a greater preference for a particular topic-subtopic combination, since the objective function is set to be maximised.

Value

For the diversity-based assignment model, this function returns a list containing:

* N: number of students * G: number of self-formed groups * m: a (student x groups) matrix, indicating group membership for each student. * d: dissimilarity matrix, NxN * s: skills vector for each individual student (possibly NULL)

For the preference-based assignment model, this function returns a list containing:

* N: number of students * G: number of self-formed groups * m: a (student x groups) matrix, indicating group membership for each student. * n: a vector of length G, with the number of students in each self-formed group. * p: The preference matrix from the input argument.

---

get_group_pref_score *Look up a group's preference score for a topic-subtopic combination*

Description

Look up a group's preference score for a topic-subtopic combination

Usage

```
get_group_pref_score(group_num, topic, subtopic, pref_mat, n_topics)
```

Arguments

group_num	Integer. Group index (row of 'pref_mat').
topic	Integer. Base topic index.
subtopic	Integer. Subtopic (subgroup) index.
pref_mat	Numeric preference matrix (groups × topic-subtopic columns).
n_topics	Integer. Number of base topics.

Value

Scalar preference score.

multirole_demand_ex001

Multi-role Demand Matrix Example 001

Description

An example demand matrix for the multi-role workload allocation model.

Usage

```
multirole_demand_ex001
```

Format

'multirole_demand_ex001' A matrix with 4 rows and 2 columns.

Columns are in the order 'TA', 'GR'. Row names store the course codes.

Source

This dataset was constructed by hand.

multirole_prefmat_ex001

Multi-role Preference Matrix Example 001

Description

An example preference matrix for the multi-role workload allocation model. It can be used for either TA or GR preferences.

Usage

```
multirole_prefmat_ex001
```

Format

'multirole_prefmat_ex001' A matrix with 4 rows and 4 columns.

Rows correspond to individuals in 'multirole_students_ex001', and columns correspond to rows of 'multirole_demand_ex001'.

Preference scores are encoded as 3 (first choice), 2 (second choice), and 1 (third choice). Unranked courses are encoded as -99.

Source

This dataset was constructed by hand.

multirole_students_ex001 *Multi-role Individual Data Example 001*

Description

An example individual table for the multi-role workload allocation model.

Usage

```
multirole_students_ex001
```

Format

'multirole_students_ex001' A data frame with 4 rows and 5 columns.

* student_id: unique individual id. * year: cohort or year, encoded from 1 to 4. * past_ta: previous-semester TA workload units. * past_gr: previous-semester GR workload units. * Name: individual name.

In this toy dataset, 'past_ta + past_gr = 4' for every individual.

Source

This dataset was constructed by hand.

pba_gc_ex002 *PBA Group Composition Data Example 002*

Description

An example dataset to use with the preference-based assignment model.

Usage

```
pba_gc_ex002
```

Format

'pba_gc_ex002' A data frame with 8 rows and 2 columns.

* id: the student id of each students, simply the integers 1 to 8. * grouping: the self-formed groups submitted by each student. In this case, each self-formed group is of size 2.

Source

This dataset was constructed by hand.

pba_prefmat_ex002 *PBA Group Preference Data Example 002*

Description

An example dataset to use with the preference-based assignment model.

Usage

```
pba_prefmat_ex002
```

Format

‘pba_prefmat_ex002‘ A matrix with 4 rows and 4 columns

Each row represents the preferences of each self-formed group in the dataset ‘pba_gc_ex002‘.

Source

This dataset was constructed by hand.

prepare_diversity_model *Prepare the diversity-based assignment model*

Description

Prepare the diversity-based assignment model

Usage

```
prepare_diversity_model(df_list, yaml_list, w1 = 0.5, w2 = 0.5)
```

Arguments

df_list	The output list from [extract_student_info()] for ‘assignment = "diversity"‘.
yaml_list	The output list from [extract_params_yaml()] for ‘assignment = "diversity"‘.
w1, w2	Numeric values between 0 and 1. Should sum to 1. These weights correspond to the importance given to the diversity- and skill-based portions in the objective function.

Value

An ompr model.

`prepare_model` *Initialise optimisation model (wrapper)*

Description

Initialise optimisation model (wrapper)

Usage

```
prepare_model(
  df_list,
  yaml_list = NULL,
  assignment = c("diversity", "preference", "phd", "multirole"),
  w1 = 0.5,
  w2 = 0.5,
  ...
)
```

Arguments

`df_list`	Model input list.
`yaml_list`	Parameter list from [extract_params_yaml()]. Optional for ‘assignment = "diversity"‘ and ‘assignment = "preference"‘ for backward compatibility. If supplied, this list is used directly. Ignored for ‘assignment = "phd"‘ and ‘assignment = "multirole"‘.
`assignment`	Character string indicating model type. Must be one of ‘"diversity"‘, ‘"preference"‘, ‘"phd"‘, or ‘"multirole"‘.
`w1, w2`	Numeric values between 0 and 1. Should sum to 1. Used only for ‘assignment = "diversity"‘.
`...`	Additional arguments: * For ‘assignment = "diversity"‘ when ‘yaml_list‘ is ‘NULL‘: supply ‘n_topics‘, ‘nmin‘, ‘nmax‘, ‘rmin‘, and ‘rmax‘. * For ‘assignment = "preference"‘ when ‘yaml_list‘ is ‘NULL‘: supply ‘n_topics‘, ‘B‘, ‘nmin‘, ‘nmax‘, ‘rmin‘, and ‘rmax‘. * For ‘assignment = "phd"‘: passed to [prepare_phd_model()], including ‘protected_year‘ when a cohort other than Year 1 should receive the soft TA-load protection. * For ‘assignment = "multirole"‘: passed to [prepare_multirole_model()]. Multi-role semester capacity is supplied during extraction and read from ‘df_list$C‘.

Value

An ompr model.

`prepare_multirole_model` *Prepare the multi-role workload allocation model*

Description

Builds a mixed-integer model for assigning TA, GR, and lighter E duties while balancing role-specific workload, preferences, and cohort protection.

Usage

```
prepare_multirole_model(
  df_list,
  ta_protected_max = 1,
  gr_protected_max = 1,
  e_max = NULL,
  ta_min = NULL,
  ta_max = NULL,
  gr_min = NULL,
  gr_max = NULL,
  e_min = NULL,
  alpha_ta = 2,
  alpha_gr = NULL,
  beta_ta = 1,
  beta_gr = NULL,
  phi = 1,
  rho_ta = 10,
  rho_gr = NULL,
  protected_year_ta = 1,
  protected_year_gr = 1
)
```

Arguments

`df_list`	A model input list from [extract_multirole_info()].
`ta_protected_max, gr_protected_max`	Non-negative soft upper limits on current-semester TA or GR workload for the corresponding protected cohort. A value may be ‘NULL‘ when the corresponding ‘rho_*‘ term is disabled.
`e_max`	Optional upper bound on per-individual E units.
`ta_min, ta_max`	Optional lower and upper bounds on per-individual TA units.
`gr_min, gr_max`	Optional lower and upper bounds on per-individual GR units.
`e_min`	Optional lower bound on per-individual E units.
`alpha_ta, alpha_gr`	Non-negative weights for annual TA and GR workload spread.

beta_ta, beta_gr
: Non-negative weights for TA and GR preferences.

phi
: Non-negative weight for score-guided E allocation.

rho_ta, rho_gr
: Non-negative penalties for TA and GR protected-cohort slack.

protected_year_ta, protected_year_gr
: Whole numbers from 1 to 4 identifying the TA- and GR-protected cohorts.

Details

Any objective weight set to 'NULL' or zero is disabled. Disabled preference and E terms are omitted from the objective. Disabling a spread term also omits its two spread variables and fairness constraints. Disabling a protection penalty omits that role's slack variables and soft-limit constraints, and includes every individual in that role's fairness spread.

When a preference term is active, the corresponding 'P_ta' or 'P_gr' element must be present in 'df_list'. Semester capacity is read from 'df_list$C', as supplied to [extract_multirole_info()]. Annual total workload is fixed at '2 * C'.

Value

An 'ompr' model.

Examples

```
inputs <- extract_multirole_info(
  student_df = multirole_students_ex001,
  d_mat = multirole_demand_ex001,
  p_ta_mat = multirole_prefmat_ex001,
  p_gr_mat = multirole_prefmat_ex001,
  e_mode = "rr"
)
model <- prepare_multirole_model(
  inputs,
  alpha_gr = 1,
  beta_gr = 1,
  rho_gr = 10
)
```

---

prepare_preference_model

Prepare the preference-based assignment model

Description

Prepare the preference-based assignment model

Usage

```
prepare_preference_model(df_list, yaml_list)
```

Arguments

df_list	The output list from [extract_student_info()] for ‘assignment = "preference"‘.
yaml_list	The output list from [extract_params_yaml()] for ‘assignment = "preference"‘.

Value

An ompr model.

---

solve_assignment *Solve a prepared model and post-process the assignment*

Description

Solves an existing ‘ompr‘ model with an ROI-backed solver, then routes the solver result through [assign_groups()] or [assign_job()] depending on the assignment type.

Usage

```
solve_assignment(
  model,
  assignment = c("diversity", "preference", "phd", "multirole"),
  solver = c("glpk", "highs", "gurobi"),
  dframe = NULL,
  params_list = NULL,
  group_names = NULL,
  student_df = NULL,
  course_codes = NULL,
  name_col = "Name",
  verbose = TRUE,
  time_limit = NULL,
  iteration_limit = NULL,
  solver_args = list()
)
```

Arguments

model	A prepared ‘ompr‘ model, usually from [prepare_model()].
assignment	Character string indicating model type. Must be one of ‘"diversity"‘, ‘"preference"‘, ‘"phd"‘, or ‘"multirole"‘.
solver	Solver to use through ‘ompr.roi‘. Must be one of ‘"glpk"‘, ‘"highs"‘, or ‘"gurobi"‘.
dframe	The original dataframe used in [extract_student_info()]. Required for ‘assignment = "diversity"‘ and ‘assignment = "preference"‘.
params_list	The list of parameters from [extract_params_yaml()]. Required for ‘assignment = "preference"‘.

group_names	A character string denoting the self-formed group column in ‘dframe‘. Required for ‘assignment = "diversity"‘ and ‘assignment = "preference"‘.
student_df	A data frame that contains individual name information. Required for ‘assignment = "phd"‘ and ‘assignment = "multirole"‘.
course_codes	Character vector of course or task codes in model order. Required for ‘assignment = "phd"‘ and ‘assignment = "multirole"‘.
name_col	Student name column name in ‘student_df‘.
verbose	Logical value passed to ‘ompr.roi::with_ROI()‘.
time_limit, iteration_limit	Optional Gurobi controls. These are applied only when ‘solver = "gurobi"‘.
solver_args	Additional named arguments passed to ‘ompr.roi::with_ROI()‘.

Value

A list with two elements:

- model_result: the raw result from [ompr::solve_model()]
- output: the post-processed assignment table

---

summary_dba *Summarise a DBA result by topic-repetition group*

Description

Summarise a DBA result by topic-repetition group

Usage

```
summary_dba(df_result, df_list, id_col)
```

Arguments

df_result	Data frame returned by [solve_assignment()] for a diversity model. Must contain ‘topic‘, ‘rep‘, and the column named by ‘id_col‘.
df_list	Input list from [extract_student_info()]. Must contain ‘d‘ (distance matrix) and optionally ‘s‘ (skill vector).
id_col	Character. Name of the student-ID column in ‘df_result‘.

Value

A grouped summary tibble with columns ‘topic‘, ‘rep‘, ‘n‘, and ‘total_diversity‘ (plus ‘total_skill‘ when ‘df_list$s‘ is present).

summary_pba *Summarise a PBA result by topic-subtopic-repetition group*

Description

Summarise a PBA result by topic-subtopic-repetition group

Usage

```
summary_pba(df_result, df_list, n_topics)
```

Arguments

df_result	Data frame returned by [solve_assignment()] for a preference model. Must contain columns ‘group‘, ‘topic2‘, ‘subtopic‘, and ‘rep‘.
df_list	Input list from [extract_student_info()] for ‘assignment = "preference"‘. Must contain ‘p‘ (preference matrix).
n_topics	Integer. Number of base topics.

Value

A grouped summary tibble with columns ‘topic2‘, ‘subtopic‘, ‘rep‘, ‘n‘, and ‘total_pref_score‘.

Index

∗ **datasets**
dba_gc_ex001, 5
dba_gc_ex003, 5
dba_gc_ex004, 6
multirole_demand_ex001, 12
multirole_prefmat_ex001, 12
multirole_students_ex001, 13
pba_gc_ex002, 13
pba_prefmat_ex002, 14

assign_groups, 2
assign_job, 3

compute_diversity, 4
convert_pref_mat, 4

dba_gc_ex001, 5
dba_gc_ex003, 5
dba_gc_ex004, 6

extract_info, 6
extract_multirole_info, 7
extract_params_yaml, 9
extract_student_info, 9

get_group_pref_score, 11

multirole_demand_ex001, 12
multirole_prefmat_ex001, 12
multirole_students_ex001, 13

pba_gc_ex002, 13
pba_prefmat_ex002, 14
prepare_diversity_model, 14
prepare_model, 15
prepare_multirole_model, 16
prepare_preference_model, 17

solve_assignment, 18
summary_dba, 19
summary_pba, 20