Package: grouper (via r-universe)

May 15, 2026

Type Package

Title Optimal Assignment of Students to Groups

Version 0.6.2

Description Integer programming models to assign students to groups by maximising diversity within groups, or by maximising preference scores for topics.

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, testthat (>= 3.0.0)

VignetteBuilder knitr

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

Roxygen list(markdown = TRUE)

Depends R (>= 3.5)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-05-15 03:29:07 UTC

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

RemoteRef HEAD

RemoteSha e0160718a750afda01254886948dd36433e4dac3

Contents

assign_groups . . . 2
assign_job . . . 3
dba_gc_ex001 . . . 3
extract_info . . . 4
extract_params_yaml . . . 5
extract_phd_info . . . 6
extract_student_info . . . 7
pba_gc_ex002 . . . 9
pba_prefmat_ex002 . . . 10
phd_demand_ex001 . . . 10
phd_prefmat_ex001 . . . 11
phd_students_ex001 . . . 11
prepare_diversity_model . . . 12
prepare_model . . . 12
prepare_phd_model . . . 13
prepare_preference_model . . . 14

Index **16**

`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 PhD solver allocation to 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 the PhD 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 p_mat 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.

---

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.

`extract_info` *Extract model inputs (wrapper)*

Description

Wrapper around `extract_student_info()` and `extract_phd_info()`.

Usage

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

Arguments

`assignment`	Character string indicating model type. Must be one of `"diversity"`, `"preference"`, or `"phd"`.
`...`	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()`

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

Value

A model input list from `extract_student_info()` or `extract_phd_info()`.

`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_phd_info` *Extract inputs for the PhD workload allocation model*

Description

Converts student-level data and input matrices into the list expected by `prepare_phd_model()`.

Usage

```
extract_phd_info(student_df, p_mat, d_mat, e_mode = c("rr", "none"), C = 4)
```

Arguments

`student_df`	A data frame with one row per student. Required columns are: `student_id`, `year`, `past_ta`, and `past_gr`. Note that it has to be in this order and name. `year` is capped to 1-4.
`p_mat`	Preference matrix with dimensions `Ns x Nj`. Row order must match `student_df` row order.
`d_mat`	Demand matrix with dimensions `Nj x 2` or `Nj x 3`. Columns are interpreted in order as `TA`, `GR`, and optional `E`. Row order must match the column order of `p_mat`.
`e_mode`	How to handle E demand when `d_mat` does not include E. `"rr"` computes E using round-robin allocation from highest to lowest GR demand; `"none"` leaves E at 0.
`C`	Semester workload capacity per student. Used when `e_mode = "rr"` to compute total semester capacity as `Ns * C`.

Details

This function assumes input order is already aligned:

- `student_df` row `i` corresponds to `P[i, ]`, `s[i]`, `t1[i]`, and `g1[i]`.
- `d_mat` row `j` corresponds to `P[, j]`.

If E is computed (`e_mode = "rr"`), total E is set to:

`Ns * C - sum(TA) - sum(GR)`.

Value

A list containing:

- `Ns`: number of students
- `Nj`: number of courses
- `P`: preference matrix (`Ns x Nj`)
- `d`: demand matrix (`Nj x 3`) with columns `TA`, `GR`, `E`
- `s`: seniority vector (`year - 2`)
- `t1`: past TA workload vector
- `g1`: past GR workload vector

`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.

`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.

phd_demand_ex001 *PhD Demand Matrix Example 001*

Description

An example demand matrix to use with the PhD workload allocation model.

Usage

```
phd_demand_ex001
```

Format

`phd_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.

phd_prefmat_ex001 *PhD Preference Matrix Example 001*

Description

An example preference matrix to use with the PhD workload allocation model.

Usage

```
phd_prefmat_ex001
```

Format

`phd_prefmat_ex001`:
A matrix with 4 rows and 4 columns.
Rows correspond to students in `phd_students_ex001`, and columns correspond to rows of `phd_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.

phd_students_ex001 *PhD Student Data Example 001*

Description

An example student table to use with the PhD workload allocation model.

Usage

```
phd_students_ex001
```

Format

`phd_students_ex001`:
A data frame with 4 rows and 5 columns.

- student_id: unique student id.
- year: PhD year, encoded from 1 to 4.
- past_ta: previous-semester TA workload units.
- past_gr: previous-semester GR workload units.
- Name: student name.

In this toy dataset, `past_ta` + `past_gr` = 4 for every student.
For a one-semester sanity-check variant, reuse this dataset with `past_ta` = `0` and `past_gr` = `C` for all students before extraction.

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"),
  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".
assignment	Character string indicating model type. Must be one of "diversity", "preference", or "phd".
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, R, nmin, nmax, rmin, and rmax. • For assignment = "preference" when yaml_list is NULL: supply n_topics, B, R, nmin, nmax, rmin, and rmax. • For assignment = "phd": passed to prepare_phd_model().

Value

An ompr model.

---

prepare_phd_model *Prepare the PhD workload allocation model*

Description

Builds a mixed-integer optimisation model for assigning TA, GR, and E units across students and courses.

Usage

```
prepare_phd_model(
  df_list,
  t_max_y1 = 1,
  e_max = NULL,
  ta_min = NULL,
  ta_max = NULL,
  gr_min = NULL,
  gr_max = NULL,
  e_min = NULL,
  alpha = 2,
  beta = 1,
  phi = 1,
  rho = 10,
  C = 4
)
```

Arguments

df_list	A list of model inputs, typically from extract_phd_info(). Required elements are: • Ns: number of students • Nj: number of courses • P: preference matrix [i, j] • d: demand matrix [j, r] where r = 1:3 for TA, GR, E • s: seniority vector (offset form; e.g. year - 2) • t1: past TA workload vector • g1: past GR workload vector
t_max_y1	Maximum current-semester TA load for Year-1 students (s == -1) before slack is used.
e_max	Optional upper bound on per-student E units in current semester.
ta_min, ta_max	Optional lower/upper bounds on per-student TA units in current semester.
gr_min, gr_max	Optional lower/upper bounds on per-student GR units in current semester.
e_min	Optional lower bound on per-student E units in current semester.
alpha	Objective weight on TA spread (Tmax - Tmin).
beta	Objective weight on TA preference term.
phi	Objective weight on seniority-weighted E term.
rho	Objective weight on Year-1 TA slack penalties.
C	Semester workload capacity per student. The model fixes annual workload at 2 * C via T_i + G_i + e_i^(2) == 2 * C. Default is 4.

Details

Index alignment is critical: P[i, j], d[j,], s[i], t1[i], and g1[i] must refer to the same student/course ordering.

Value

An ompr model object ready for ompr::solve_model().

---

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.

Index

∗ **datasets**
    dba_gc_ex001, 3
    pba_gc_ex002, 9
    pba_prefmat_ex002, 10
    phd_demand_ex001, 10
    phd_prefmat_ex001, 11
    phd_students_ex001, 11

assign_groups, 2
assign_job, 3

cluster::daisy(), *8*

dba_gc_ex001, 3

extract_info, 4
extract_params_yaml, 5
extract_params_yaml(), *2*, *5*, *7*, *12, 13*, *15*
extract_phd_info, 6
extract_phd_info(), *4, 5*, *14*
extract_student_info, 7
extract_student_info(), *2*, *4, 5*, *12*, *15*

ompr::solve_model(), *2*

pba_gc_ex002, 9
pba_prefmat_ex002, 10
phd_demand_ex001, 10
phd_prefmat_ex001, 11
phd_students_ex001, 11
prepare_diversity_model, 12
prepare_model, 12
prepare_model(), *7*
prepare_phd_model, 13
prepare_phd_model(), *13*
prepare_preference_model, 14