Package: socialroulette (via r-universe)

September 15, 2024

Type Package

Title Generate social roulette partitions, i.e. divide people into groups of predefined size

Version 0.1.0

Author Michael Höhle [aut, cre], Jin-Kao Hao [aut], Xiangjing Lai [aut]

Maintainer Michael Höhle `<hoehle@math.su.se>`

Description Partition individuals into groups of a predefined size. Different ways of partitioning are provided, one based on the maximally diverse grouping problem solver by Lai and Hao (2016) tries to maximize the overall distance of how far back in time individuals have been in the same group previously.

License GPL-3

Encoding UTF-8

LazyData true

Imports magrittr, dplyr, purrr, readr, stringr, tidyr, tibble, rlang

RoxygenNote 7.1.1

Suggests rmarkdown, knitr, igraph, ggplot2

VignetteBuilder rmarkdown

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

RemoteUrl https://github.com/mhoehle/socialroulette

RemoteRef HEAD

RemoteSha b05c039e0de1f588b5fa87fa71bc8b89bc0b3a94

Contents

`socialroulette-package`
socialroulette: A package to generate social groupings

Description

A package for partitioning individuals into groups of a pre-specified size. This can be as simple as using simple random sampling (srs) to divide n individuals into groups of size at least m. If one keeps track of the past partitions then an additional aim can be to try to maximize the time that people have a reunion, i.e. end up in the same group. This boils down to an instance of the maximally diverse grouping problem. See the [package website](https://hoehleatsu.github.io/socialroulette/) for more information, documentation and examples.

Details

A side effect of the package is that it provides a maximally diverse grouping problem solver, which can in principle be used for other purposes. As a consequence, internal functionality is exposed using export statements.

`mdgp_read_solutionfile`
Read output from the Lai and Hao (2016) MDGP solver

Description

Read output from the Lai and Hao (2016) MDGP solver

Usage

```
mdgp_read_solutionfile(file_name)
```

Arguments

`file_name`	Name of the solution file

Value

data.frame with two clumns, idx (index) and group

`mdgp_solver` *Maximally diverse grouping problem solver based on Lai and Hao (2016)*

Description

Given a distance metric for each possible pair in the frame of n individuals, find a partition into as many groups as possible with group size at least m. This is an instance of the maximallz diverse grouping problem, which we use the algorithm by Lai and Hao (2016) to solve.

Usage

```
mdgp_solver(mdgp_format_file, time_limit = 15)
```

Arguments

`mdgp_format_file`	Path to the file containing the MDGP specification in mdgplib format
`time_limit`	Number of seconds to iteratively optimize each run. The larger the number of participants to group, the larger this value should be. Rule of thumb: time_limit = exp(0.5 + 0.0025*n)

Details

The maximally diverse grouping problem is about partitioning n individuals into groups of size at least m while maximizing a sum of utility values computed by summing the utility $u(i,j)$ over all individuals i,j partitioned into the same group. and summing this quantity over all groups. More formally, let d_{ij} denote the number of time unit (typically days) ago, that individual i and j were in the same group. Note: This distance is derived by looking at the previous partitions and it is a matter of definition what this value should be, if i and j have not previously been in the same group. Let $G = n$ div m denote the resulting number of groups where div denotes integer division. For a given partition let x_{ig} be an indicator variable, which is 1, if i is assigned into group g and zero otherwise. A solver of the maximally diverse grouping problem now tries to maximize

$$\sum_{g=1}^{G}\sum_{i=1}^{n}\sum_{j=i+1}^{n} d_{ij}x_{ig}x_{jg},$$

subject to the conditions

$$\sum_{g=1}^{G} x_{ig} = 1, \quad i = 1, \ldots, n,$$

$$\sum_{i=1}^{n} x_{ig} = n_g, \quad g = 1, \ldots, G,$$

$$x_{ig} \in \{0,1\}, \quad i = 1, \ldots, n; g = 1, \ldots, G,$$

where n_g is the size of group g, i.e. $\sum_{g=1}^{G} n_g = n$. We shall adopt the convention that group sizes are determined by assigning the labels $1, \ldots, G$ to all individuals and then count how often each label occurs. This means, e.g., that for $n = 7$ and $m = 4$ we get $n_g = 7$ div $4 = 1$ group with 7 members.

Note: The code calls C++ code by Xiangjing Lai and Jin-Kao Hao. This is an R adapted version of the C++ code available from http://www.info.univ-angers.fr/ Note: several temporary files are generated using ‘tempfile()‘.

Value

File name of the solution file

Author(s)

Xiangjing Lai and Jin-Kao Hao, R interface by M. Höhle

References

Xiangjing Lai and Jin-Kao Hao (2016). *Iterated maxima search for the maximally diverse grouping problem*. European Journal of Operational Research, 254(3), pp. 780-800, https://doi.org/10.1016/j.ejor.2016.05.018

`mdgp_write_specfile` *Convert partition specification to Lai and Hao (2016) MDGP solver input specification.*

Description

Generate a `tempfile()` containing the appropriate specification formed from `current_frame` and `past_partitions`.

Usage

```
mdgp_write_specfile(current_frame, past_partitions, m)
```

Arguments

`current_frame`	A pair-distance tibble. The frame needs to contain a column denoted ‘id‘
`past_partitions`	A named list of partitions, where the names correspond to the date when the partition was used.
`m`	The minimum group size, i.e. all groups have size at least m

Details

The specification format has the following header line

$$n\ m\ ds\ n_1\ n_1\ \ldots n_G\ n_G$$

Here, each group size $n_g, g = 1, \ldots, G$ is repeated twice, because the solver has the opportunity to specify a lower as well as an upper boundary for each group size. The keyword 'ds' for the solver in the above means that the groups can be of different sizes as determined by the respective n_g.

Value

File name of the generated specification file

Examples

```
frame  <- tibble::tibble(id=sprintf("id%.2d", 1:5), date=as.Date("2021-04-28"))
past_partitions <- list("2021-04-21"=list(c("id02", "id03", "id04"), c("id01", "id05")))
spec_file <- socialroulette:::mdgp_write_specfile(frame, past_partitions=past_partitions, m=2)
cat(stringr::str_c(readLines(spec_file), collapse="\n"))
```

pairs_to_partitions *Convert a pairs data.frame into a partition list*

Description

The function works by splitting the pairs data.frame by date and then apply the internal function pairs_to_partition to the resulting sub data.frame.

Usage

```
pairs_to_partitions(pairs)
```

Arguments

pairs	A data.frame with columns date and id1 and id2.

See Also

pairs_to_partition

Examples

```
partitions <- list("2021-04-21"=list(c("id02", "id03", "id04"), c("id01", "id05")),
                   "2021-04-28"=list(c("id03", "id04", "id05"), c("id01", "id02")))
partitions2 <- partitions %>% socialroulette::partitions_to_pairs() %>%
                              socialroulette::pairs_to_partitions()
all.equal(partitions, partitions2)
```

partitions_to_distance

Convert a list of partitions into pairs format including a distance metric

Description

Convert a list of partitions into pairs format including a distance metric

Usage

```
partitions_to_distance(current_frame, past_partitions)
```

Arguments

`current_frame`	The current frame
`past_partitions`	List of previous partitions

Examples

```
frame  <- tibble::tibble(id=sprintf("id%.2d", 1:5), date=as.Date("2021-04-28"))
partitions <- list("2021-04-14"=list(c("id02", "id03", "id04"), c("id01", "id05")),
                   "2021-04-21"=list(c("id03", "id04", "id05"), c("id01", "id02")))
socialroulette::partitions_to_distance(frame, partitions)
```

partitions_to_pairs

Convert a list of partitions into a data.frame with all pairs

Description

The name of the list contain the date at which each partition was used.

Usage

```
partitions_to_pairs(partitions)
```

Arguments

`partitions`	A list of past partitions, i.e. a named list where each entry is a partition and the names reflect the dates the partition were used

Examples

```
partitions <- list("2021-04-21"=list(c("id02", "id03", "id04"), c("id05", "id01")),
                   "2021-04-28"=list(c("id05", "id03", "id04"), c("id02", "id01")))
socialroulette::partitions_to_pairs(partitions)
```

`partition_to_frame` *Take a partition and convert it to frame representation*

Description

The resulting `tibble` will have an additional column `group`

Usage

```
partition_to_frame(l)
```

Arguments

`l`	A partition, i.e. list of vectors, where each vector contains all individuals in the corresponding group

Value

frame The frame with ‘id‘ and ‘group‘ columns to convert

Examples

```
round1 <- list(c("id02", "id03", "id04"), c("id05", "id01"))
socialroulette::partition_to_frame(round1)
```

`rsocialroulette` *Make a new lunch roulette partition maximizing the gossip to exchange*

Description

One can either use simple random sampling (srs) to generate the partition or solve the maximally diverse grouping problem (mdgp) using the algorithm by Lai and Hao (2016) in order to maximize time since last meets over all groups.

Usage

```
rsocialroulette(
  current_frame,
  past_partitions = NULL,
  m,
  algorithm = c("mdgp", "srs"),
  ...
)
```

Arguments

`current_frame`	A tibble containing the participants of the current round, i.e. it has a column ‘id‘ containing a unique identifier and a ‘date‘ column representing the date of the partition.
`past_partitions`	A list of partition lists each named by the date the partition was used (this is used to determine temporal distance to last meeting)
`m`	minimum group size, i.e. all groups will be at least size m.
`algorithm`	String specifying either "srs" (simple random sampling - DEFAULT) or "mdgp" (maximally diverse grouping problem)
`...`	Additional arguments to be sent to the solver

Value

A partitioning of current_frame maximizing the overall sum of gossip to be exchanged.

References

Höhle M (2021), Long time, no see: Virtual Lunch Roulette, Blog post, https://staff.math.su.se/hoehle/blog/2021/04/04/socialsamp.html

Xiangjing Lai and Jin-Kao Hao (2016). *Iterated maxima search for the maximally diverse grouping problem*. European Journal of Operational Research, 254(3), pp. 780-800, https://doi.org/10.1016/j.ejor.2016.05.018

See Also

mdgp_solver

Examples

```
today <- Sys.Date()
frame <- tibble::tibble( id=sprintf("id%.02d",1:4), date=today)
round1 <- rsocialroulette(current_frame = frame, m=2, algorithm="srs")
round1

#Generate list of past partitions
past_partitions <- list(round1) %>% setNames(today)
frame2 <- frame %>% dplyr::mutate(date = today+7)
round2 <- rsocialroulette(current_frame = frame2,
                          past_partitions=past_partitions, m=2, algorithm="mdgp")
round2
```

Index