Package 'PopulateR'

July 21, 2025

Type Package

Title Create Data Frames for the Micro-Simulation of Human Populations

Version 1.13

Maintainer Michelle Gosse <michelle.a.gosse@gmail.com>

Description Tools for constructing detailed synthetic human populations from frequency tables. Add ages based on age groups and sex, create households, add students to education facilities, create employers, add employers to employees, and create interpersonal networks.

Depends R (>= 4.0)

Imports brainGraph (>= 3.1.0), data.table (>= 1.16.2), dplyr (>= 1.1.4), igraph (>= 2.1.1), magrittr (>= 2.0.3), PearsonDS (>= 1.3.1), plyr (>= 1.8.9), rlang (>= 1.1.4), sn (>= 2.1.1), tidyr (>= 1.3.1), tidyselect (>= 1.2.1), withr (>= 3.0.2)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://github.com/programgirl/PopulateR

BugReports https://github.com/programgirl/PopulateR/issues

NeedsCompilation no

Author Michelle Gosse [aut, cre, cph], Jonathan Marshall [aut], Mark Bebbington [ctb]

Repository CRAN

Date/Publication 2025-01-29 18:30:05 UTC

Contents

`ABMToCova` *Creates the four data frames of weighted contact pairs for use in Covasim*

Description

Creates the household, school, workplace, and contacts layers, from ABMPop, for use with the Python package Covasim. A 1xn data frame of ages is also created.

Usage

```
ABMToCova(
  ABMPop,
  ABMID,
  ABMAge,
  place1,
  place2,
  ECE = TRUE,
  PSchool = TRUE,
  SSchool = TRUE,
  contacts = NULL,
  excludeDF = NULL
)
```

Arguments

ABMPop	The agent-based modelling data frame.
ABMID	The variable containing the unique identifier for each person, in the ABMPop data frame.
ABMAge	The variable containing the ages, in the in the ABMPop data frame.
place1	The variable containing the Household ID.
place2	The variable containing the school and workplace IDs.
ECE	Are ECE centres open? Default is TRUE, change to FALSE if ECEs are to close.
PSchool	Are primary schools open? Default is TRUE, change to FALSE if primary schools are to close.
SSchool	Are secondary schools open? Default is TRUE, change to FALSE if secondary schools are to close.
contacts	A data frame consisting of existing contact pairs. The first two variables define the two people in the pair.
excludeDF	A data frame of industries to exclude. This must be the relevant IndNum variable in the ABMPop data frame. If this data frame is not included, all industries will be represented in the output data frame.

Details

There are three restrictions for use. First, the place2 codes for preschool, primary school, and secondary school must be set to "P801000", "P802100", and "P802200", respectively. Second, at least one school type must be "TRUE" as Covasim requires a school layer. Third, the place2 value for people who are not in school, and not in a workplace, must be "Not employed".

Value

A data frame of the household, school, workplace, contact layers, and people's ages, for use in Covasim.

addemp *Add employers to people in employment*

Description

Creates a data frame of people and matching employers, if employed. Two data frames are required: one for the people and one for the employers. For people not in employment, a user-supplied missing value is used instead of the employer information. A numeric or ordered factor for working hours is required. The minimum value for being in employment must be specified. Anyone coded under this value will be treated as unemployed. Thus, pre-cleaning the people data frame is not required. The employer data frame can be either a summary in the form of the number of employees by employer. The other option is that each row represents a vacancy for an employee. Thus, an employer with 5 employees may be represented as either: a single row with an employee count of 5, or 5 rows with an employee count of 1 in each row.

Usage

```
addemp(
  employers,
  empid,
  empcount,
  people,
  pplid,
  wrkhrs,
  hoursmin,
  missval = NA,
  userseed = NULL
)
```

Arguments

employers	The data frame containing employer data.
empid	The variable containing the unique identifier for each employer.
empcount	The variable containing the count of employees for each employer.
people	The data frame containing the people that require employers.
pplid	The variable containing the unique ID for each person, in the people data frame.
wrkhrs	The variable containing the hours worked by each person. Must be an ordered factor or numeric. If the variable is an ordered factor, the levels/values must be ascending for hours worked. This is output as an ordered factor.
hoursmin	The wrkhrs value representing the minimum number of hours worked (numeric) or lowest factor level/number. Any wrkhrs value lower than this number/level will be treated as unemployed.
missval	The value that will be used to replace any NA results in the output data frame. If not supplied, NA will be used for all employer-related variables for the non-working people.

userseed — The user-defined seed for reproducibility. If left blank the normal set.seed() function will be used.

Value

A data frame of the people, with an employer ID attached to each person. Unemployed people will have an employer ID of NA, or the value specified by missval. All columns in the employers data frame, except for the employee counts, are included in the output data frame.

Examples

```
library("dplyr")

EmployedPeople <- addemp(EmployerSet, empid = "Company", empcount = "NumEmployees", Township,
                      pplid = "ID", wrkhrs = "HoursWorked", hoursmin = 2, missval = "NA",
                         userseed = 4)
```

---

addind — *Add a variable indicating whether the person is in education, or has left education*

Description

Creates a data frame with a variable indicating whether the person is a student, or is not in education. This is an factor with two levels. Pre-cleaning so that only people inside the student age range is not required. Three data frames are required. The first is the data frame that contains the people ("people") to whom the indicator will be applied. The other two data frames are counts: school leaver counts ("leavers"), and the sex/age pyramid counts ("pyramid") that apply to the school leaver counts. As cumulative proportions of school leavers are calculated, the leavers data frames must contain multiple years of data. For example, if the minimum school leaving age is 17 and the maximum age is 18, then there must be two years of data in the leavers data frame. The pyramid data frame contains the sex/age counts for the relevant year. For example, if the people data frame is based on 2021 data frame, then the pyramid data frame should be the counts for 2021, and the value for pplyear would be 2021. The variables specifying sex can be numeric, character, or factor. The sole requirement is that the same code is used in all three data frames. For example, if "F" and "M" are used in the adolescents data frame to denote sex, then "F" and "M" are the codes required in both the leavers and pyramid data frames. Any number of values can be used, so long as they are unique.

Usage

```
addind(
  people,
  pplid,
  pplsx,
  pplage,
  pplyear,
  minedage = NULL,
```

```
  maxedage = NULL,
  leavers,
  lvrsx,
  lvrage,
  lvryear,
  lvrcount,
  pyramid,
  pyrsx,
  pyrage,
  pyrcount,
  stvarname = "Status",
  verbose = FALSE,
  userseed = NULL
)
```

Arguments

people	A data frame containing individual people.
pplid	The variable containing the unique identifier for each person, in the people data frame
pplsx	The variable containing the codes for sex, in the people data frame.
pplage	The variable containing the ages, in the people data frame.
pplyear	The year associated with the people data frame.
minedage	The minimum age that a person, normally a child, can enter education.
maxedage	The maximum age that a person, normally an adolescent, can leave education.
leavers	A data frame containing the counts, by sex, age, and year, of the people who have left education.
lvrsx	The variable containing the codes for sex, in the leavers data.
lvrage	The variable containing the codes for sex, in the leavers data.
lvryear	The variable containing the year for the lvrcount.
lvrcount	The variable containing the counts for each sex/age combination in the leavers data.
pyramid	A data frame containing the sex/age pyramid to be used.
pyrsx	The variable containing the codes for sex, in the pyramid data.
pyrage	The variable containing the ages, in the pyramid data.
pyrcount	The variable containing the counts for each sex/age combination, in the pyramid data
stvarname	The name of the variable to contain the education status. The output is "Y" for those still in education and "N" for those not in education.
verbose	If TRUE, the proportion of students who have left school by age and sex will be printed to the console. Default is FALSE
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.

Details

The proportion of people, by age and sex, who have left school is printed to the console.

Value

A data frame of an observations, with an added column that contains the education status of each person.

Examples

```
WithInd <- addind(Township, pplid = "ID", pplsx = "Sex", pplage = "Age", pplyear = 2018,
                  minedage = 5, maxedage = 18, LeftSchool, lvrsx = "Sex", lvrage = "Age",
                   lvryear = "YearLeft", lvrcount = "Total", RegionalStructure,
                  pyrsx = "Sex", pyrage = "Age", pyrcount = "Value", stvarname = "Status",
                   verbose = TRUE, userseed = 4)
```

---

addnetwork *Create a social network for people in a population*

Description

Creates social networks between people, based on age differences. A data frame of people with ages is required. These are the people who will have social relationships between each other. A a 1x n matrix of counts must also be supplied, where n is the number of rows in the people data frame. As person-to-person pairs are constructed, the sum of the matrix counts must be even. If it is not, the function will randomly select one person's social network size from the matrix and add 1 to it. If this correction happens, an explanation, including the index position of the count, will be printed to the console.

Usage

```
addnetwork(
  people,
  pplid,
  pplage,
  netmax,
  sdused = 0,
  probsame = 0.5,
  userseed = NULL,
  numiters = 1e+06,
  usematrix = FALSE,
  verbose = FALSE
)
```

Arguments

people	A data frame containing people to be matched to each other using social networks.
pplid	The variable for each person's unique ID.
pplage	The variable for each person's age.
netmax	A data frame containing the 1-dimensional matrix of network sizes. Must contain only integers and be the same length as the people data frame.
sdused	The standard deviation for the age differences between two people.
probsame	The probability that a friend of a friend is also a friend. For example, if A and B and friends, and B and C are friends, this is the probability that C is also a friend of A.
userseed	The user-defined seed for reproducibility. If left blank, the normal set.seed() function will be used.
numiters	The maximum number of iterations used to construct the coupled data frame. This has a default value of 100, and is the stopping rule if the algorithm does not converge.
usematrix	If an adjacency matrix is output instead of an igraph object. Default is FALSE so an igraph object is output. If TRUE is used, the n x n dgCMatrix is output.
verbose	Whether a notification is printed to the console if the number of contacts must be increased by one. Notification is that it has occurred, where the value has been increased, and the original and new number of contacts. The default is FALSE, so no information will be printed to the console.

Details

A normal distribution is used, using the age differences between the pairs. This is centred on 0, i.e. the people in the pair are the same age. If people B and C are in person A's network, the value of probsame is used to determine the likelihood that people B and C know each other. The larger this probability, the more likely that people in one person's network know each other, compared to random construction of a network between them.

The two options for output are a dgCMatrix or an igraph. The dgCMatrix is output as n x n. For a large data frame of people, this will be a large and sparse matrix, which may not be completed due to RAM limitations. The igraph output only contains the pairs, and should be a smaller object compared to the dgCMatrix.

Value

Either an igraph of social networks, or a dgCMatrix of n x n.

Examples

```
library("dplyr")

# smaller sample for visualisation
set.seed(2) # small datasets can cause problems if a random seed is used for sampling
 SmallDemo <- Township %>%
```

```
  filter(between(Age, 20, 29)) %>%
  slice_sample(n = 20)
  Smallnetwork <- rpois(n = nrow(SmallDemo), lambda = 1.5)
  NetworkSmallN <- addnetwork(SmallDemo, "ID", "Age", Smallnetwork, sdused=2,
                              probsame = .5, userseed=4, numiters = 10)
# plot(NetworkSmallN)
```

---

addschool *Match school children to schools*

Description

Creates a data frame of people and matching schools. By default, all similarly-aged students in the same household will be matched to the same school. If one student is matched to a same-sex school, then all similarly aged students will also be matched to a same-sex school. This includes opposite-sex children, with boys attending a same-sex boys school, and girls attending a same-sex girls school. Two data frames are required: one for the people ("people) and one for the schools ("schools"). In the "people" data frame, a numeric or ordered factor for school status is required. The smallest value/level will be treated as the code for non-students. If one value is used, everyone in the data frame will be allocated a school. Thus, pre-cleaning a data frame is not required. The "schools" data frame must be a summary in the form of roll counts by age within school. Each row is one age only. For example, if a school has children aged 5 to 9 years, there should be 5 rows. Any combination of co-educational and single-sex schools can be used, and the relevant value must be on each row of the schools" data frame.

Usage

```
addschool(
  people,
  pplid,
  pplage,
  pplsx,
  pplst = NULL,
  hhid = NULL,
  schools,
  schid,
  schage,
  schroll,
  schtype,
  schmiss = 0,
  sameprob = 1,
  userseed = NULL
)
```

Arguments

people	A data frame containing individual people.

pplid	The variable containing the unique identifier for each person, in the people data frame.
pplage	The variable containing the ages, in the people data frame.
pplsx	The variable containing the codes for sex, in the people data frame.
pplst	The school status variable in the people data frame. Only two numeric values/factor levels can be used. The smallest number/level is the code for people not in school.
hhid	The household identifier variable, in the people data frame.
schools	A data frame containing the schools.
schid	The variable containing the unique identifier for each school, in the schools data frame.
schage	The variable containing the ages, in the schools data frame.
schroll	The variable containing the number of places available for people at that school age, within the school.
schtype	The variable that indicates whether the school is co-educational or single-sex. The expected value for a co-educational school is "C". The codes for female and male must be the same as in the people data frame.
schmiss	The school identifier value that will be given to those people not in school. If left blank, the default value is 0. If the school IDs are numeric in the schools data frame, a numeric missing value must be supplied.
sameprob	The probability that students from the same household will be at the same school, given age (and sex if there are same-sex schools). Results depend on the number of students in each household, and student ages, combined with the sizes of the school rolls. Value must be between 0 and 1. The default value is 1.
userseed	If specified, this will set the seed to the number provided. If not,the normal set.seed() function will be used.

Value

Two data frames, as a list. $Population contains the synthetic population with the schools added. $Schools contains the remaining roll counts for the schools.

Examples

```
library(dplyr)

# children in the same household will be added to the same school, if possible with a .8 probability
SchoolsAdded <- addschool(IntoSchools, pplid = "ID", pplage = "Age", pplsx = "SexCode",
                          pplst = "SchoolStatus", hhid = "HouseholdID", SchoolsToUse,
                      schid = "School.Name", schage = "AgeInRoll", schroll = "RollCount",
                          schtype = "Gender", schmiss = 0, sameprob = .8, userseed = 4)

Population <- SchoolsAdded$Population
Schools <- SchoolsAdded$Schools
```

AdultsNoID *Non-partnered synthetic people*

Description

A subset of people from the Township data frame, aged 20 years and older with a relationship status of "NonPartnered".

Usage

```
AdultsNoID
```

Format

A data frame of 2,213 rows and 5 variables:

Sex SEither Male or Female

Relationship Relationship status of the person

ID The unique identifier for the person

Age The age of the person

HoursWorked The number of hours worked in employment, per week

agedis *Add a sex/age structure to a data frame of grouped ages*

Description

Adds an age variable to a data frame that contains age groups, based on age group within sex. Two data frames are required: the data frame that contains individuals with age bands ("individuals"), and a data frame used as the basis for constructing a sex/age pyramid ("pyramid"). The individuals data frame requires two columns relating to the age groups. One is the minimum age in the age group. The second is the maximum age in the age group. For example, the age group 0 - 4 years would have 0 as the minimum age value and 4 as the maximum age value. Each person in the individuals data frame must have both the minimum and maximum age variables populated. The pyramid data frame must contain counts by sex/age in the population of interest. The variables specifying sex can be numeric, character, or factor. The sole requirement is that the codes must match. For example, if "F" and "M" are used in the individuals data frame to denote sex, then "F" and "M" are the codes required in the pyramid data frame. Any number of sex code values can be used, so long as they are unique.

Usage

```
agedis(
  individuals,
  indsx,
  minage,
  maxage,
  pyramid,
  pyrsx,
  pyrage,
  pyrcount,
  agevarname,
  userseed = NULL
)
```

Arguments

individuals	A data frame containing observations with grouped ages. These are the observations to which the sex/age pyramid is applied.
indsx	The variable containing the codes for sex, in the individuals data frame.
minage	The variable containing the minimum age for the age group, in the individuals data frame.
maxage	The variable containing the maximum age for the age group, in the individuals data frame.
pyramid	A data frame containing the sex/age pyramid to be used.
pyrsx	The variable containing the codes for sex, in the pyramid data frame.
pyrage	The variable containing the ages, in the pyramid data frame.
pyrcount	The variable containing the counts for each sex/age combination, in the pyramid data frame.
agevarname	The name to use for the constructed age variable in the output data frame. For each row, this will contain one integer.
userseed	The user-defined seed for reproducibility. If left blank the normal set.seed() function will be used.

Value

A data frame of an observations, with an added column that contains the age.

Examples

```
library("dplyr")

ReducedDF <- InitialDataframe %>%
  slice_sample(n=200, replace = FALSE)
DisaggregateAge <- agedis(ReducedDF, indsx = "Sex", minage = "LowerAge", maxage = "UpperAge",
                    pyramid = SingleAges, pyrsx = "Sex", pyrage = "Age", pyrcount = "Value",
                           agevarname = "TheAge", userseed = 4)
```

AllEmployers *Employers and employees, by industry*

Description

The number of businesses and employees by industry, Timaru District, for 2018.

Usage

```
AllEmployers
```

Format

A data frame of 183 rows and 7 variables:

ANZSIC06 The code and associated name for each industry

BusinessCount The random-rounded count of employers in the industry

EmployeeCount The random-rounded count of employees in the industry

minCo The minimum number of employers in the industry

maxCo The maximum number of employers in the industry

minStaff The minimum number of people employed in the industry

maxStaff The maximum number of people employed in the industry

Source

Statistics New Zealand. Statistics New Zealand data are licensed by Stats NZ for reuse under the Creative Commons Attribution 4.0 International licence. The data has been modified by adding in four additional variables, representing the estimated minimum and maximum counts of businesses and employees.

BadRels *Synthetic people restricted to an age range*

Description

A subset of people from the Township data frame, aged between 20 and 91 years. Age bands, and the associated minimum and maximum ages, have been added.

Usage

```
BadRels
```

Format

A data frame of 7,568 rows and 8 variables:

Sex Either Male or Female

Relationship Relationship status of the person

ID The unique identifier for the person

Age The age of the person

HoursWorked The number of hours worked in employment, per week

AgeBand The ten-year age band for the age

MinAge The minimum age in the age band

MaxAge The maximum age in the age band

`createemp` *Create employers, each with employee counts*

Description

Constructs individual employers from aggregate counts, such as number of employers per employer type. Employer type is often industry, such as "Sheep, Beef Cattle and Grain Farming". Within each employer type, the number of employers is extracted. The number of employees is then randomly assigned to each of those employers, using the total employee count for that industry. A randomisation method is used to ensure that the company counts can be quite dissimilar across the employers within a type. However, this is constructed by the ratio of employers to employees. If the number of employers is similar to the number of employees, the number of employees will tend to be 1 for each employer.

Usage

```
createemp(
  employers,
  industry,
  indsmin,
  indsmax,
  pplmin,
  pplmax,
  stffname = NULL,
  cpyname = NULL,
  userseed = NULL
)
```

Arguments

employers	A data frame containing aggregate data on employers.
industry	The variable containing the types of employers. This can be an industry code.
indsmin	The variable containing the minimum number of employees in each industry.
indsmax	The variable containing the maximum number of employees in each industry.
pplmin	The variable containing the minimum number of staff in each industry.
pplmax	The variable containing the maximum number of staff in each industry.
stffname	The variable name to use for the staff counts for each employer.
cpyname	The variable name to use for the companies.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.

Value

#'A data frames of synthetic companies, with the number of employees and a mock company name.

Examples

```
library("dplyr")

TownshipEmployment <- createemp(AllEmployers, industry = "ANZSIC06", indsmin = "minCo",
                                indsmax = "maxCo", pplmin = "minStaff", pplmax = "maxStaff",
                                  stffname="NumEmployees", cpyname="Company", userseed = 4)
```

---

diffsample *Sample from groups, when the sample size for each group is different*

Description

Produces samples by group, enabling different sample sizes to be specified for each group. Sampling without replacement is used. While the function example is based on sampling by age, in practice sampling can be performed using any variable of choice. Only one grouping variable is used.

Usage

```
diffsample(people, pplage, sampledf, smplage, smplcounts, userseed = NULL)
```

Arguments

people	A data frame containing individual people.
pplage	The variable containing the ages, in the people data frame.
sampledf	A data frame containing ages and sample size counts.
smplage	The variable containing the ages, in the sampledf data frame.
smplcounts	The variable containing the sample size counts, in the sampledf data frame.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.

Value

A data frame of people sampled according to the age sample sizes required.

Examples

```
SampleNeeded <- data.frame(Age = c(16, 17, 18),
                           NumNeeded = c(5, 10, 15))
SampledAdolescents <- diffsample(WorkingAdolescents, pplage = "Age", sampledf = SampleNeeded,
                                 smplage = "Age", smplcounts = "NumNeeded", userseed = 4)

table(SampledAdolescents$Age)
```

EmployerSet *Synthetic employers and their employee counts*

Description

Synthetic employers and their associated number of employees, randomly constructed using the "AllEmployers" data frame.

Usage

```
EmployerSet
```

Format

A data frame of 225 rows and 3 variables:

ANZSIC06 The code and associated name for the industry associated with the employer

NumEmployees The count of employees for the employer

Company The name of the employer

---

fastmatch *Create couples using a weighted age group structure*

Description

Creates couples when the only information is the proportions of people in couples, by age group. If there is an age range that should be up-sampled compared to other ages, this can be specified using the uwProp, uwLA, and uwUA variables. If uwProp is not provided, a simple random sampling without replacement is used. The number of couples that are output is determined by probSS. At least one same-sex couple will be output.

Usage

```
fastmatch(
  people,
  pplage,
  probSS = NULL,
  uwProp = NULL,
  uwLA = NULL,
  uwUA = NULL,
  HHStartNum = NULL,
  HHNumVar = NULL,
  userseed = NULL
)
```

Arguments

people	A data frame containing individual people.
pplage	The variable containing the ages.
probSS	The probability of a person being in a same-sex couple.
uwProp	The proportion of individuals who are to be over-sampled. By default, no age group is up-sampled, and people are selected based on simple random sampling, without replacement.
uwLA	The youngest age for the over-sampling. Required if uwProp value is provided.
uwUA	The oldest age for the over-sampling. Required if uwProp value is provided.
HHStartNum	The starting value for HHNumVar Must be numeric.
HHNumVar	The name for the household variable.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.

Value

A data frame of an even number of observations for allocation into same-sex couples. If HHStartNum is specified, household allocation will be performed.

Examples

```
library(dplyr)

PersonDataframe <- data.frame(cbind(PersonID = c(1:1000),
                                    PersonAge = c(round(runif(200, min=18, max=23),0),
                                    round(runif(300, min=24, max=50),0),
                                    round(runif(500, min=51, max=90),0))))

# unweighted example, probability of being in a same-sex couple is 0.03
Unweighted <- fastmatch(PersonDataframe, pplage = "PersonAge", probSS = 0.03, HHStartNum = 1,
                        HHNumVar = "Household", userseed = 4)
NumUnweighted <- Unweighted %>%
  filter(between(PersonAge, 25, 54))
```

```
# prop is
nrow(NumUnweighted)/nrow(Unweighted)

# weighted example, same probability, 66% of people in a same-sex relationship are aged between 25
# and 54
Weighted <- fastmatch(PersonDataframe, pplage = "PersonAge", probSS = 0.03, uwProp = .66,
                uwLA = 25, uwUA = 54, HHStartNum = 1, HHNumVar = "Household", userseed = 4)
NumWeighted <- Weighted %>%
  filter(between(PersonAge, 25, 54))
# prop is
nrow(NumWeighted)/nrow(Weighted)
```

`fixhours` *Reallocates working hours between people in education and people not in education*

Description

Reallocates working hours so that people in education work fewer hours than people not in education. Pre-cleaning so that only people inside the student age range is not required. The hours of work are reallocated so that shorter hours worked are prioritised to those in education. The variables provided in the grpdef vector define the marginal totals that must be retained.

Usage

```
fixhours(people, pplid, pplstat, pplhours, hoursmax, grpdef, userseed = NULL)
```

Arguments

`people`	A data frame containing individual people.
`pplid`	The variable containing the unique identifier for each person, in the people data frame.
`pplstat`	The variable containing the indicator of whether a person is in education, in the people data frame. This must consist of only two values, and can be either an ordered factor or numeric. If this is a factor, factor level 2 must be for those in education. If it is a numeric variable, the lowest number must be for those in education.
`pplhours`	The variable containing the hours worked by each adolescent. Must be a factor or numeric. If this is a factor, it is assumed to be ordered. The levels/values must be ascending for hours worked.
`hoursmax`	The maximum hours worked by people in education. Must be the relevant factor level/number from pplhours.
`grpdef`	The vector containing any grouping variable to be used. If this is used, the changes to the working hours will be performed using grouped data. Marginal totals for the cross-tabulations of the grouping variables are retained.
`userseed`	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.

Value

A data of observations, with working hours reallocated so that people's working hours are compatible with their education status.

Examples

```
# table of hours by schoolstatus
table(WorkingAdolescents$HoursWorked, WorkingAdolescents$SchoolStatus)

# one grouping variable
Group1 <- "Age"
OneGroup <- fixhours(WorkingAdolescents, pplid = "ID", pplstat = "SchoolStatus",
                  pplhours = "HoursWorked", hoursmax = 3, grpdef = Group1, userseed = 4)
table(OneGroup$HoursWorked, OneGroup$SchoolStatus)

# two grouping variables
Group2 <- c("Age", "Sex")
TwoGroups <- fixhours(WorkingAdolescents, pplid = "ID", pplstat = "SchoolStatus",
                   pplhours = "HoursWorked", hoursmax = 3, grpdef = Group2, userseed = 4)
table(TwoGroups$HoursWorked, TwoGroups$SchoolStatus)
```

---

fixrelations *Provide an age structure to relationship status, estimated from age groups*

Description

Redistributes a user-defined relationship status value between ages, using age groups and other variables (if specified). Within the group definition provided, the marginal totals of the relationship status values are retained. The data frame can include groups where all people have the same relationship status. In this situation, there is no need to restrict the data frame to only those whose relationship status must be redistributed.

Usage

```
fixrelations(
  people,
  pplid,
  pplage,
  pplstat,
  stfixval,
  props,
  propcol,
  grpdef,
  matchdef,
  userseed = NULL
)
```

Arguments

people	A data frame containing individual people.
pplid	The variable containing the unique identifier for each person.
pplage	The variable containing the ages.
pplstat	The relationship status variable in the people data frame.
stfixval	The value of the relationship status, in the people data frame, that will be adjusted for age. If there are only two relationship status values, the choice does not matter. But if there are three or more values, this is the one value that will be age-corrected.
props	The data frame containing the proportions of people with the stfixval value, by the grpdef.
propcol	The variable in the props data frame that contains the proportions for the relationship status value of interest.
grpdef	A vector containing the combination of grouping variables, in the people dataframe, that defines the marginal totals for relationship status counts. This can be one variable or a string of multiple variables. Include the age-group variable, but not the age variable.
matchdef	A vector containing the same variables as grpdef, except the age variable is substituted for the age-group variable.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.

Value

A data frame of observations, with one relationship status redistributed so that an age, rather than age group, structure is created.

Examples

```
library("dplyr")
thegroups <- as.vector("Sex")
GroupInfo <- rbind(GroupInfo, list("Male", "Under 20 Years", 19, 19, "Partnered", 0, 19),
                   list("Female", "Under 20 Years", 19, 19, "Partnered", 0, 19))
RelProps <- interdiff(GroupInfo, pplage = "MidPoints", pplprop = "RelProps", endmin = "MinAge",
                      endmax = "MaxAge", grpdef = thegroups)
# add in the age groups
RelProps <- RelProps %>%
  mutate(AgeBand = ifelse(Age==19, "Under 20 Years",
                   ifelse(between(Age, 20, 29), "20-29 Years",
                   ifelse(between(Age, 30, 39), "30-39 Years",
                   ifelse(between(Age, 40, 49), "40-49 Years",
                   ifelse(between(Age, 50, 59), "50-59 Years",
                   ifelse(between(Age, 60, 69), "60-69 Years",
                   ifelse(between(Age, 70, 79), "70-79 Years", "80-90 Years")))))))

# perform separately by sex
thejoindef <- c("Age", "Sex")
```

```
thegroups <- c("Sex", "AgeBand")
FinalRels <- fixrelations(BadRels, pplid = "ID", pplage = "Age", pplstat = "Relationship",
                          stfixval = "Partnered", props = RelProps, propcol = "Fits",
                          grpdef = thegroups, matchdef = thejoindef, userseed = 4)
```

`GroupInfo` *The proportion of people in a relationship, by age band within sex*

Description

The estimated proportion of people in a relationship, by age band within sex, for people aged between 20 and 90 years.

Usage

```
GroupInfo
```

Format

A data frame of 14 rows and 7 variables:

Sex Either Male or Female

AgeBand The 10-year age band

MinAge The minimum age of the age band

MaxAge The maximum age of the age band

Relationship All people are Partnered

RelProps The proportion of people who have a relationship status of "Partnered"

MidPoints The median age in the age band

`InitialDataframe` *People in age groups, in the Timaru District*

Description

A data frame of 46,293 synthetic people. Age groups are present, but not ages.

Usage

```
InitialDataframe
```

Format

A data frame with 46,293 rows and 6 variables:

Sex Either Male or Female

Age.group Age group in five-year age bands

Relationship Relationship status of the person

LowerAge The youngest age in the Age.group

UpperAge The oldest age in the Age.group

ID The unique identifier for the person

Source

Timaru District 2018 census data (tablecodes 8277 and 8395), sourced from Statistics New Zealand. Statistics New Zealand data are licensed by Stats NZ for reuse under the Creative Commons Attribution 4.0 International licence.

`interdiff` *Interpolate ages from age group medians*

Description

The node ages for each age group are defined by the user, along with the age group values. The ages are then imputed from these nodes. Zero values at both extremes must be included. For example, for the age group 20-24 years, the pplprop value is for pplage. if the first non-zero relationship probability is for the age group 20-24 years, and the previous age group is 15-19 years, pplprop==0 for pplage==19. For each age group, there must be a minimum and maximum age specified. This provides the interpolation range for each age group. For the anchoring 0 values, the minimum and maximum ages are the same. In this example, for pplage==19, endmin==19, and endmax==19. If there is no zero for older ages, as the final node value occurs inside the age group, the function assumes that the last node-to-node should be used to extrapolate for the ages older than the oldest node value. For example, if the last node value is for 90 years of age, but the oldest age is 95 years, the function will assume the same slope for ages 91 through 95 years. The function can perform a separate interpolation for groups, for example, a separate interpolation can be performed for each sex. The function is flexible for the number of variables that can be used to define groups. If only one interpolation is required, the same grpdef value should be used for each row in the data frame.

Usage

```
interdiff(nodes, pplage, pplprop, endmin, endmax, grpdef)
```

Arguments

nodes	A data frame containing all grouping variables, the node ages for each group, and the associated node values.
pplage	The variable containing the node ages.
pplprop	The variable containing the node values.
endmin	The variable that contains the minimum age for each group.
endmax	The variable that contains the maximum age for each group.
grpdef	A character vector containing the names of the grouping variables.

Details

While the function is designed to interpolate proportions, in practice it can interpolate any values. The limitation is that the function performs no rounding. Integer node values may produce non-integer estimates.

Value

A data frame containing the fitted values, by age within group.

Examples

```
library("dplyr")

# create the expected proportion of people in relationships, by age within sex
thegroups <- as.vector("Sex")
RelProps <- interdiff(GroupInfo, pplage = "MidPoints", pplprop = "RelProps", endmin = "MinAge",
                      endmax = "MaxAge", grpdef = thegroups)
```

IntoSchools *Four person households, with a school status for each person*

Description

Four-person households, consisting of one parent and three children, with a combination of people in school and not in school. Ages 15 through 18 contain a mixture of people in school and those who have left school. This has been constructed from the Township data frame.

Usage

```
IntoSchools
```

Format

A data frame of 980 rows and 8 variables:

Sex Either Male or Female

Relationship Relationship status of the person

ID The unique identifier for the person

Age The age of the person

HoursWorked The number of hours worked in employment, per week

SchoolStatus The indicator of whether the person is in school (Y) or not (N)

HouseholdID The household identifier for the person

SexCode Either (F)emale or (M)ale

LeftSchool *School leavers*

Description

School leavers in the Canterbury Region, counts by age and sex, for the period 2009 to 2018.

Usage

```
LeftSchool
```

Format

A data frame with 120 rows and 4 variables:

YearLeft The year for the school leaver count

Sex The sex for the school leaver count

Age The age for the school leaver count

Total The count of adolescents who left school in that year, of that age and sex

Source

Ministry of Education. The Ministry of Education's data are licensed by the Ministry of Education for reuse under the Creative Commons Attribution 4.0 International licence.

NetworkMatrix *The number of contacts for 5000 person*

Description

A matrix of 1,000 integers constricted using a Poisson distribution. Each value is the number of contacts for a person.

Usage

```
NetworkMatrix
```

Format

A list of 1,000 integers

other *Match people into new households*

Description

This function creates a data frame of household inhabitants, with the specified number of inhabitants. One data frame, containing the people to match, is required. The use of an age distribution for the matching ensures that an age structure is present in the households. A less correlated age structure can be produced by entering a larger standard deviation. The output data frame of matches will only contain households of the required size. If the number of rows in the people data frame is not divisible by household size, the overcount will be output to a separate data frame.

Usage

```
other(
  people,
  pplid,
  pplage,
  numppl = NULL,
  sdused,
  HHStartNum,
  HHNumVar,
  userseed = NULL,
  ptostop = NULL,
  numiters = 1e+06,
  verbose = FALSE
)
```

Arguments

people	A data frame containing the people to be matched into households.
pplid	The variable containing the unique ID for each person.
pplage	The age variable.
numppl	The number of people in the households.
sdused	The standard deviation of the normal distribution for the distribution of ages in a household.
HHStartNum	The starting value for HHNumVar. Must be numeric.
HHNumVar	The name for the household variable.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
ptostop	The critical p-value stopping rule for the function. If this value is not set, the critical p-value of .01 is used.
numiters	The maximum number of iterations used to construct the output data frame ($Matched) containing the household inhabitants. The default value is 1000000, and is the stopping rule if the algorithm does not converge.
verbose	Whether the number of iterations used, the critical chi-squared value, and the final chi-squared value are printed to the console. The information will be printed for each set of pairs. For example, if there are three people in each household, the information will be printed twice. The default is FALSE, so no information will be printed to the console.

Value

A list of two data frames $Matched contains the data frame of households containing matched people. All households will be of the specified size. $Unmatched, if populated, contains the people that were not allocated to households. If the number of rows in the people data frame is divisible by the household size required, $Unmatched will be an empty data frame.

Examples

```
library(dplyr)

# creating three-person households toy example with few iterations
NewHouseholds <- other(AdultsNoID, pplid = "ID", pplage = "Age", numppl = 3, sdused = 3,
                       HHStartNum = 1, HHNumVar = "Household", userseed=4, ptostop = .05,
                        numiters = 500, verbose = TRUE)

PeopleInHouseholds <- NewHouseholds$Matched
PeopleNot <- NewHouseholds$Unmatched      # 2213 not divisible by 3
```

otherNum *Match people into existing households*

Description

Creates a data frame of household inhabitants, with the specified number of inhabitants. Two data frames are required. The 'existing' data frame contains the people already in households. The 'additions' data frame contains the people. The use of an age distribution for the matching ensures that an age structure is present in the households. A less correlated age structure can be produced by entering a larger standard deviation. The output data frame of matches will only contain households of the required size.

Usage

```
otherNum(
  existing,
  exsid,
  exsage,
  HHNumVar = NULL,
  additions,
  addid,
  addage,
  numadd = NULL,
  sdused = NULL,
  userseed = NULL,
  attempts = 10,
  numiters = 10000,
  verbose = FALSE
)
```

Arguments

existing	A data frame containing the people already in households.
exsid	The variable containing the unique ID for each person, in the existing data frame.
exsage	The age variable, in the existing data frame.
HHNumVar	The household identifier variable. This must exist in only one data frame.
additions	A data frame containing the people to be added to the existing households.
addid	The variable containing the unique ID for each person, in the additions data frame.
addage	The age variable, in the additions data frame.
numadd	The number of people to be added to the household.
sdused	The standard deviation of the normal distribution for the distribution of ages in a household.
userseed	The user-defined seed for reproducibility. If left blank the normal set.seed() function will be used.

attempts	The number of times the function will randomly change two matches to improve the fit.
numiters	The maximum number of iterations used to construct the household data frame. This has a default value of 10000, and is the stopping rule if the algorithm does not converge.
verbose	Whether the number of iterations used, the critical chi-squared value, and the final chi-squared value are printed to the console. The information will be printed for each set of pairs. For example, if there are two people being added to each household, the information will be printed twice. The default is FALSE, so no information will be printed to the console.

Value

A list of three data frames $Matched contains the data frame of households containing matched people. All households will be of the specified size. $Existing, if populated, contains the excess people in the existing data frame, who could not be allocated additional people. $Additions, if populated, contains the excess people in the additions data frame who could not be allocated to an existing household.

Examples

```
library("dplyr")

AdultsID <- IntoSchools %>%
filter(Age > 20) %>%
select(-c(SchoolStatus, SexCode))
set.seed(2)
NoHousehold <- Township %>%
  filter(Age > 20, Relationship == "NonPartnered", !(ID %in% c(AdultsID$ID))) %>%
  slice_sample(n = 1500)

# toy example with few iterations
OldHouseholds <- otherNum(AdultsID, exsid = "ID", exsage = "Age", HHNumVar = "HouseholdID",
                        NoHousehold, addid = "ID", addage = "Age", numadd = 2, sdused = 3,
                           userseed=4, attempts= 10, numiters = 80)
CompletedHouseholds <- OldHouseholds$Matched # will match even if critical p-value not met
IncompleteHouseholds <- OldHouseholds$Existing # no-one available to match in
UnmatchedOthers <- OldHouseholds$Additions # all people not in households were matched
```

---

pairbeta4 *Pair two people, using a four-parameter beta distribution, into households*

Description

Creates a data frame of paired people, based on a distribution of age differences. The function uses a four-parameter beta distribution to create the pairs. Two data frames are required. One person from each data frame will be matched, based on the age difference distribution specified. If the data

frames are different sizes, the "smalldf" data frame must be the smaller of the two. In this situation, a random subsample of the "largedf" data frame will be used. Both data frames must be restricted to only those people that will be paired.

Usage

```
pairbeta4(
  smalldf,
  smlid,
  smlage,
  largedf,
  lrgid,
  lrgage,
  shapeA = NULL,
  shapeB = NULL,
  locationP = NULL,
  scaleP = NULL,
  HHStartNum,
  HHNumVar,
  userseed = NULL,
  ptostop = NULL,
  attempts = 10,
  numiters = 1e+06,
  verbose = FALSE
)
```

Arguments

smalldf	The data frame containing one set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the smallest number.
smlid	The variable containing the unique ID for each person, in the smalldf data frame.
smlage	The age variable, in the smalldf data frame.
largedf	A data frame containing the second set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the largest number.
lrgid	The variable containing the unique ID for each person, in the largedf data frame.
lrgage	The age variable, in the largedf data frame.
shapeA	This is the first shape parameter of the four-parameter beta distribution If this value is negative, smalldf has the oldest ages. If this value is positive, smalldf has the youngest ages.
shapeB	This is the second shape parameter of the four-parameter beta distribution This value must be positive.
locationP	The location parameter of the four-parameter beta distribution.
scaleP	The scale parameter of the four-parameter beta distribution.
HHStartNum	The starting value for HHNumVar. Must be numeric.

HHNumVar	The column name for the household variable.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
ptostop	The critical p-value stopping rule for the function. If this value is not set, the critical p-value of .01 is used.
attempts	The maximum number of times largedf will be sampled to draw an age match from the correct distribution, for each observation in the smalldf. The default number of attempts is 10.
numiters	The maximum number of iterations used to construct the output data frame ($Matched) containing the pairs. The default value is 1000000, and is the stopping rule if the algorithm does not converge.
verbose	Whether the number of iterations used, the critical chi-squared value, and the final chi-squared value are printed to the console. The default value is FALSE.

Value

A list of three data frames. $Matched contains the data frame of pairs. $Smaller contains the unmatched observations from smalldf. $Larger contains the unmatched observations from largedf.

Examples

```
library(dplyr)

# the children data frame is smaller
set.seed(1)
# sample a combination of females and males to be parents
Parents <- Township %>%
  filter(Relationship == "Partnered", Age > 18) %>%
  slice_sample(n = 500)
  Children <- Township %>%
    filter(Relationship == "NonPartnered", Age < 20) %>%
    slice_sample(n = 200)

ChildAllMatched <- pairbeta4(Children, smlid = "ID", smlage = "Age", Parents, lrgid = "ID",
                             lrgage = "Age", shapeA = 2.2, shapeB = 3.7, locationP = 16.5,
                               scaleP = 40.1, HHStartNum = 1, HHNumVar = "Household",
                               userseed=4, ptostop = .01, attempts = 2, numiters = 8)

MatchedPairs <- ChildAllMatched$Matched
UnmatchedChildren <- ChildAllMatched$Smaller
UnmatchedAdults <- ChildAllMatched$Larger

# children data frame is larger, the locationP and scaleP values are negative

Parents2 <- Township %>%
 filter(Relationship == "Partnered", Age > 18) %>%
 slice_sample(n = 100)
Children2 <- Township %>%
 filter(Relationship == "NonPartnered", Age < 20) %>%
 slice_sample(n = 500)
```

```
 ChildMatched <- pairbeta4(Parents2, smlid = "ID", smlage = "Age", Children2, lrgid = "ID",
                           lrgage = "Age", shapeA = 2.2, shapeB = 3.7, locationP = -16.5,
                            scaleP = -40.1, HHStartNum = 1, HHNumVar = "Household",
                            userseed=4, ptostop = .05, attempts = 2, numiters = 8)

MatchedPairs2 <- ChildMatched$Matched
UnmatchedChildren2 <- ChildMatched$Smaller
UnmatchedAdults2 <- ChildMatched$Larger
```

`pairbeta4Num` *Pair two people, using a four-parameter beta distribution, households already exist*

Description

This function creates a data frame of pairs, based on a distribution of age differences. The function will use either a skew normal or normal distribution, depending on whether a skew ("locationP") parameter is provided. The default value for the skew is 0, and using the default will cause a normal distribution to be used. Two data frames are required. One person from each data frame will be matched, based on the age difference distribution specified. If the data frames are different sizes, the smalldf data frame must be the smaller of the two. In this situation, a random subsample of the largedf data frame will be used. The household identifier variable can exist in either data frame. The function will apply the relevant household identifier once each pair is constructed. Both data frames must be restricted to only those people that are successfully paired. At least 30 matched pairs are required for the function to run. This is to reduce the proportion of empty cells.

Usage

```
pairbeta4Num(
  smalldf,
  smlid,
  smlage,
  largedf,
  lrgid,
  lrgage,
  shapeA = NULL,
  shapeB = NULL,
  locationP = NULL,
  scaleP = NULL,
  HHNumVar,
  userseed = NULL,
  attempts = 10,
  numiters = 1e+06,
  verbose = FALSE
)
```

Arguments

smalldf	The data frame containing one set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the smallest number.
smlid	The variable containing the unique ID for each person, in the smalldf data frame.
smlage	The age variable, in the smalldf data frame.
largedf	A data frame containing the second set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the largest number.
lrgid	The variable containing the unique ID for each person, in the largedf data frame.
lrgage	The age variable, in the largedf data frame.
shapeA	This is the first shape parameter of the four-parameter beta distribution If this value is negative, smalldf has the oldest ages. If this value is positive, smalldf has the youngest ages.
shapeB	This is the second shape parameter of the four-parameter beta distribution This value must be positive.
locationP	The location parameter of the four-parameter beta distribution
scaleP	The scale parameter of the four-parameter beta distribution
HHNumVar	The household identifier variable. This must exist in only one data frame.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
attempts	The maximum number of times largedf will be sampled to draw an age match from the correct distribution, for each observation in the smalldf. The default number of attempts is 10.
numiters	The maximum number of iterations used to construct the output data frame ($Matched) containing the pairs. The default value is 1000000, and is the stopping rule if the algorithm does not converge.
verbose	Whether the number of iterations used, the critical chi-squared value, and the final chi-squared value are printed to the console. The default value is FALSE.

Value

A list of three data frames $Matched contains the data frame of pairs. $Smaller contains the unmatched observations from smalldf. $Larger contains the unmatched observations from largedf.

Examples

```
library(dplyr)

# demonstrate matched dataframe sizes first
set.seed(1)
# sample a combination of females and males to be parents
Parents <- Township %>%
  filter(Relationship == "Partnered", Age > 18) %>%
  slice_sample(n = 500) %>%
```

```
  mutate(Household = row_number())
Children <- Township %>%
  filter(Relationship == "NonPartnered", Age < 20) %>%
  slice_sample(n = 200)

# match the children to the parents, toy example with few iterations
ChildAllMatched <- pairbeta4Num(Children, smlid = "ID", smlage = "Age", Parents, lrgid = "ID",
                            lrgage = "Age", shapeA = 2.2, shapeB = 3.7, locationP = 16.5,
                          scaleP = 40.1, HHNumVar = "Household", userseed=4, attempts = 8,
                                numiters = 90)

MatchedPairs <- ChildAllMatched$Matched
UnmatchedChildren <- ChildAllMatched$Smaller # all children matched
UnmatchedAdults <- ChildAllMatched$Larger

# # children data frame is larger, the locationP and scaleP values are negative
#
# Parents2 <- Township %>%
#   filter(Relationship == "Partnered", Age > 18) %>%
#  slice_sample(n = 200) %>%
#   mutate(Household = row_number())
# Children2 <- Township %>%
#   filter(Relationship == "NonPartnered", Age < 20) %>%
#   slice_sample(n = 500)
#
# ChildMatched <- pairbeta4Num(Parents2, smlid = "ID", smlage = "Age", Children2, lrgid = "ID",
#                            lrgage = "Age", shapeA = 2.2, shapeB = 3.7, locationP = -16.5,
#                                scaleP = -40.1, HHNumVar = "Household", userseed=4,
#                                attempts = 10, numiters = 80)
#
# MatchedPairs2 <- ChildMatched$Matched
# UnmatchedChildren2 <- ChildMatched$Smaller
# UnmatchedAdults2 <- ChildMatched$Larger
```

`pairmult` *Create many-to-one pairs of people and place them into households*

Description

Creates a data frame of many-to-one pairs, based on a distribution of age differences. Designed to match multiple children to the same parent, the function can be used for any situation where a many-to-one match is required based on a range of age differences. For clarity and brevity, the terms "children" and "parents" will be used. Two data frames are required: the first contains the people representing the many (e.g children). The second contains the people that will be paired with multiple others (e.g. the parents of two or more children). The minimum and maximum ages of parents must be specified. This ensures that there are no parents who were too young (e.g. 11 years) or too old (e.g. 70 years) at the time the child was born. The presence of too young and too old parents is tested throughout this function. Thus, pre-cleaning the parents data frame is not required. Both data frames must be restricted to only those people that will be paired.

Usage

```
pairmult(
  children,
  chlid,
  chlage,
  numchild = 2,
  twinprob = 0,
  parents,
  parid,
  parage,
  minparage = NULL,
  maxparage = NULL,
  HHStartNum = NULL,
  HHNumVar = NULL,
  userseed = NULL,
  maxdiff = 1000
)
```

Arguments

children	The data frame containing the children to be paired with a parent/guardian.
chlid	The variable containing the unique ID for each person,in the children data frame.
chlage	The age variable, in the children data frame.
numchild	The number of children that are required in each household.
twinprob	The probability that a person is a twin.
parents	The data frame containing the potential parents.(This data frame must contain at least the same number of observations as the children data frame.)
parid	The variable containing the unique ID for each person,in the parents data frame.
parage	The age variable, in the parent data frame.
minparage	The youngest age at which a person becomes a parent. The default value is NULL, which will cause the function to stop.
maxparage	The oldest age at which a person becomes a parent. The default value is NULL, which will cause the function to stop.
HHStartNum	The starting value for HHNumVar. Must be numeric.
HHNumVar	The name for the household variable.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
maxdiff	The maximum age difference for the children in a household ages. This is applied to the first child randomly selected for the household, so overall age differences may be 2* maxdiff. Default value is no constraints on child age differences in the household.

Value

A list of three data frames. $Matched contains the data frame of child-parent matches. $Adults contains any unmatched observations from the parents data frame. $Children contains any unmatched observations from the children data frame. $Adults and/or $Children may be empty data frames.

Examples

```
library(dplyr)

set.seed(1)
Parents <- Township %>%
  filter(Relationship == "Partnered", Age > 18) %>%
  slice_sample(n = 500)
Children <- Township %>%
  filter(Relationship == "NonPartnered", Age < 20) %>%
  slice_sample(n = 400)

# example with assigning two children to a parent
# the same number of children is assigned to all parents
# adding two children to each parent
ChildMatched <- pairmult(Children, chlid = "ID", chlage = "Age", numchild = 2, twinprob = 0.03,
                    Parents, parid = "ID", parage = "Age", minparage = 18, maxparage = 54,
                        HHStartNum = 1, HHNumVar = "Household", userseed=4, maxdiff = 3)
MatchedFamilies <- ChildMatched$Matched
```

---

pairmultNum *Create many-to-one pairs, when there are existing households*

Description

Creates a data frame of many-to-one pairs, based on a distribution of age differences. Designed to match multiple children to the same parent, the function can be used for any situation where a many-to-one match is required based on a range of age differences. For clarity and brevity, the terms "children" and "parents" will be used. Two data frames are required: one for children and one for potential parents. The data frame of potential parents must contain household identifiers The minimum and maximum ages of parents must be specified. This ensures that there are no parents who were too young (e.g. 11 years) or too old (e.g. 70 years) at the time the child was born. The presence of too young and too old parents is tested throughout this function. Thus, pre-cleaning the parents data frame is not required. Both data frames must be restricted to only those people that will be paired.

Usage

```
pairmultNum(
  children,
  chlid,
  chlage,
```

```
  numchild = 2,
  twinprob = 0,
  parents,
  parid,
  parage,
  minparage = NULL,
  maxparage = NULL,
  HHNumVar = NULL,
  userseed = NULL,
  maxdiff = 1000
)
```

Arguments

children	The data frame containing the children to be paired with a parent/guardian.
chlid	The variable containing the unique ID for each person,in the children data frame.
chlage	The age variable, in the children data frame.
numchild	The number of children that are required in each household.
twinprob	The probability that a person is a twin.
parents	The data frame containing the potential parents.(This data frame must contain at least the same number of observations as the children data frame.)
parid	The variable containing the unique ID for each person,in the parents data frame.
parage	The age variable, in the parent data frame.
minparage	The youngest age at which a person becomes a parent. The default value is NULL, which will cause the function to stop.
maxparage	The oldest age at which a person becomes a parent. The default value is NULL, which will cause the function to stop.
HHNumVar	The name of the household identifier variable in the parents data frame.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
maxdiff	The maximum age difference for the children in a household ages. This is applied to the first child randomly selected for the household, so overall age differences may be 2* maxdiff. Default value is no constraints on child age differences in the household.

Value

A list of three data frames. $Matched contains the data frame of child-parent matches. $Adults contains any unmatched observations from the parents data frame. $Children contains any unmatched observations from the children data frame. $Adults and/or $Children may be empty data frames.

Examples

```
library(dplyr)

set.seed(1)
```

```
Parents <- Township %>%
  filter(Relationship == "Partnered", Age > 18) %>%
  slice_sample(n = 500) %>%
  mutate(Household = row_number())
Children <- Township %>%
  filter(Relationship == "NonPartnered", Age < 20) %>%
  slice_sample(n = 400)

# example with assigning two children to a parent
# the same number of children is assigned to all parents
# adding two children to each parent

ChildMatched <- pairmultNum(Children, chlid = "ID", chlage = "Age", numchild = 2, twinprob = 0.03,
                     Parents, parid = "ID", parage = "Age", minparage = 18, maxparage = 54,
                            HHNumVar = "Household", userseed =4, maxdiff = 3)
MatchedFamilies <- ChildMatched$Matched
UnmatchedChildren <- ChildMatched$Children
UnmatchedAdults <- ChildMatched$Adults
```

`pairnorm` *Pair two people, using either a normal or skew-normal distribution, into households*

Description

Creates a data frame of couples, based on a distribution of age differences. The function will use either a skew normal or normal distribution, depending on whether a skew ("alphaused") parameter is provided. The default value for the skew is 0, and using the default will cause a normal distribution to be used. Two data frames are required. One person from each data frame will be matched, based on the age difference distribution specified. If the data frames are different sizes, the smalldf data frame must be the smaller of the two. In this situation, a random subsample of the largedf data frame will be used. Both data frames must be restricted to only those people that will have a couples match performed.

Usage

```
pairnorm(
  smalldf,
  smlid,
  smlage,
  largedf,
  lrgid,
  lrgage,
  directxi = NULL,
  directomega = NULL,
  alphaused = 0,
  HHStartNum,
  HHNumVar,
  userseed = NULL,
```

```
  ptostop = NULL,
  numiters = 1e+06,
  verbose = FALSE
)
```

Arguments

smalldf	A data frame containing one set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the smallest number.
smlid	The variable containing the unique ID for each person, in the smalldf data frame.
smlage	The age variable, in the smalldf data frame.
largedf	A data frame containing the second set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the largest number.
lrgid	The variable containing the unique ID for each person, in the largedf data frame.
lrgage	The age variable, in the largedf data frame.
directxi	If a skew-normal distribution is used, this is the location value. If the default alphaused value of 0 is used, this defaults to the mean value for the normal distribution.
directomega	If a skew-normal distribution is used, this is the scale value. If the default alphaused value of 0 is used, this defaults to the standard deviation value for the normal distribution.
alphaused	The skew. If a normal distribution is to be used, this can be omitted as the default value is 0 (no skew).
HHStartNum	The starting value for HHNumVar Must be numeric.
HHNumVar	The name for the household variable.
userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
ptostop	The critical p-value stopping rule for the function. If this value is not set, the critical p-value of .01 is used.
numiters	The maximum number of iterations used to construct the output data frame ($Matched) containing the couples. The default value is 1000000, and is the stopping rule if the algorithm does not converge.
verbose	Whether the distribution used, number of iterations used, the critical chi-squared value, and the final chi-squared value are printed to the console. The default value is FALSE.

Value

A list of two data frames. $Matched contains the data frame of pairs. $Unmatched contains the unmatched observations from largedf. If there are no unmatched people, $Unmatched will be an empty data frame.

Examples

```
library(dplyr)

# matched dataframe sizes first, using a normal distribution
# females younger by a mean of -2 and a standard deviation of 3
set.seed(1)
PartneredFemales1 <- Township %>%
  filter(Sex == "Female", Relationship == "Partnered") %>%
  slice_sample(n=120, replace = FALSE)
PartneredMales1 <- Township %>%
 filter(Sex == "Male", Relationship == "Partnered") %>%
 slice_sample(n = nrow(PartneredFemales1), replace = FALSE)

# partners females and males, using a normal distribution, with the females
# being younger by a mean of -2 and a standard deviation of 3
OppSexCouples1 <- pairnorm(PartneredFemales1, smlid = "ID", smlage = "Age", PartneredMales1,
                           lrgid = "ID", lrgage = "Age", directxi = -2, directomega = 3,
                       HHStartNum = 1, HHNumVar = "HouseholdID", userseed = 4, ptostop=.3)
Couples1 <- OppSexCouples1$Matched

# different size dataframes
# there are more partnered males than partnered females
# so all partnered males will have a matched female partner
# but not all females will be matched
# being the smallest data frame, the female one must be the first
#
# PartneredFemales2 <- Township %>%
#   filter(Sex == "Female", Relationship == "Partnered") %>%
#   slice_sample(n=120, replace = FALSE)
# PartneredMales2 <- Township %>%
#   filter(Sex == "Male", Relationship == "Partnered") %>%
#   slice_sample(n=140, replace = FALSE)
#
# OppSexCouples2 <- pairnorm(PartneredFemales2, smlid = "ID", smlage = "Age", PartneredMales2,
#                            lrgid = "ID", lrgage = "Age", directxi = -2, directomega = 3,
#                        HHStartNum = 1, HHNumVar="HouseholdID", userseed = 4, ptostop=.3)
# Couples2 <- OppSexCouples2$Matched
```

`pairnormNum` *Pair two people, using either a normal or skew-normal distribution, households already exist*

Description

Creates a data frame of pairs, based on a distribution of age differences. The function will use either a skew normal or normal distribution, depending on whether a skew ("locationP") parameter is provided. The default value for the skew is 0, and using the default will cause a normal distribution to be used. Two data frames are required. One person from each data frame will be matched, based on the age difference distribution specified. If the data frames are different sizes, the smalldf data

frame must be the smaller of the two. In this situation, a random subsample of the largedf data frame will be used. The household identifier variable can exist in either data frame. The function will apply the relevant household identifier once each pair is constructed. Both data frames must be restricted to only those people that are successfully paired. At least 30 matched pairs are required for the function to run. This is to reduce the proportion of empty cells.

Usage

```
pairnormNum(
  smalldf,
  smlid,
  smlage,
  largedf,
  lrgid,
  lrgage,
  directxi = NULL,
  directomega = NULL,
  alphaused = 0,
  HHNumVar,
  userseed = NULL,
  attempts = 10,
  numiters = 1e+06,
  verbose = FALSE
)
```

Arguments

smalldf	The data frame containing one set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the smallest number.
smlid	The variable containing the unique ID for each person, in the smalldf data frame.
smlage	The age variable, in the smalldf data frame.
largedf	A data frame containing the second set of people to be paired. If the two data frames contain different numbers of people, this must be the data frame containing the largest number.
lrgid	The variable containing the unique ID for each person, in the largedf data frame.
lrgage	The age variable, in the largedf data frame.
directxi	If a skew-normal distribution is used, this is the location value. If the default alphaused value of 0 is used, this defaults to the mean value for the normal distribution. Use a positive value if the older ages are in smldf.
directomega	If a skew-normal distribution is used, this is the scale value. If the default alphaused value of 0 is used, this defaults to the standard deviation value for the normal distribution.
alphaused	The skew. If a normal distribution is to be used, this can be omitted as the default value is 0 (no skew).
HHNumVar	The household identifier variable. This must exist in only one data frame.

userseed	If specified, this will set the seed to the number provided. If not, the normal set.seed() function will be used.
attempts	The maximum number of times largedf will be sampled to draw an age match from the correct distribution, for each observation in the smalldf. The default number of attempts is 10.
numiters	The maximum number of iterations used to construct the output data frame ($Matched) containing the pairs. The default value is 1000000, and is the stopping rule if the algorithm does not converge.
verbose	Whether the distribution used, number of iterations used, the critical chi-squared value, and the final chi-squared value are printed to the console. The default value is FALSE.

Value

A list of three data frames $Matched contains the data frame of pairs. $Smaller contains the unmatched observations from smalldf. $Larger contains the unmatched observations from largedf.

Examples

```
library(dplyr)

# parents are older than the children using a normal distribution of mean = 30,
# standard deviation of 5
set.seed(1)
Parents <- Township %>%
  filter(between(Age, 24, 60)) %>%
  slice_sample(n=120, replace = FALSE) %>%
  mutate(HouseholdID = row_number())
Children <- Township %>%
  filter(Age < 20) %>%
  slice_sample(n = nrow(Parents), replace = FALSE)

PrntChld <- pairnormNum(Parents, smlid = "ID", smlage = "Age", Children, lrgid = "ID",
                  lrgage = "Age", directxi = 30, directomega = 5, HHNumVar = "HouseholdID",
                        userseed = 4, attempts=10, numiters = 80)
Matched <- PrntChld$Matched  # all matched but not the specified distribution
UnmatchedAdults <- PrntChld$Smaller
UnmatchedChildren <- PrntChld$Larger
```

Ppl4networks *Synthetic people living in the Timaru District*

Description

1000 synthetic people, to match the number of people in the NetworkMatrix data frame.

Usage

```
Ppl4networks
```

Format

A data frame with 1,000 rows and 5 variables:

Sex Either Male or Female

Relationship Relationship status of the person

ID The unique identifier for the person

Age The age of the person

HoursWorked The number of hours worked in employment, per week

Source

Timaru District 2018 census data (tablecodes 8277, 8395, and 8460), sourced from Statistics New Zealand. Statistics New Zealand data are licensed by Stats NZ for reuse under the Creative Commons Attribution 4.0 International licence.

RegionalStructure *Sex/Age pyramid for teenagers in the Canterbury Region*

Description

The number of people, by age and sex, living in the Canterbury region, restricted to ages 13 to 19 years.

Usage

```
RegionalStructure
```

Format

A data frame with 14 observations and 4 variables:

Sex The sex relating to the count

Age.group String variable of age plus the text " years"

Value The count of adolescents

Age The age relating to that count

Source

Canterbury region 2018 census data (tablecode 8277), sourced from Statistics New Zealand. Statistics New Zealand data are licensed by Stats NZ for reuse under the Creative Commons Attribution 4.0 International licence.

SchoolsToUse *Schools and their roll counts*

Description

Nineteen schools in the Canterbury region, with their 2018 roll counts.

Usage

```
SchoolsToUse
```

Format

A data frame with 266 rows and 5 variables:

School.ID The numeric ID for the school

School.Name The name for the school

Gender Indicator of whether the school is (C)o-ed, (F)emale-only, or (M)ale-only

AgeInRoll The age of possible students

RollCount The number of students. The value is 0 if no students that age attend.

Source

The Ministry of Education. The Ministry of Education's data are licensed by the Ministry of Education for reuse under the Creative Commons Attribution 4.0 International licence.

SingleAges *Sex/Age pyramid data for Timaru District*

Description

The number of people, by age and sex, living in the Timaru District.

Usage

```
SingleAges
```

Format

A data frame with 190 rows and 4 variables:

Age.group Age group, in five-year age bands

Sex Either Male or Female

Value The number of people that age and sex

Age Age at last birthday

Source

Timaru District 2018 census data (tablecode 8277), sourced from Statistics New Zealand. Statistics New Zealand data are licensed by Stats NZ for reuse under the Creative Commons Attribution 4.0 International licence.

Township *Simulated township*

Description

10,000 simulated people.

Usage

```
Township
```

Format

A data frame with 10,000 rows and 5 variables

Sex Sex of the person

Relationship Relationship status of the person

ID The unique identifier for the person

Age The age of the person

HoursWorked The number of hours worked in employment, per week

Source

Timaru District 2018 census data, using tablecodes 8277, 8395, and 8460, sourced from Statistics New Zealand. Statistics New Zealand data are licensed by Stats NZ for reuse under the Creative Commons Attribution 4.0 International licence.

WorkingAdolescents *Adolescents with a school status and employment hours*

Description

A set of synthetic adolescents aged between 15 and 18.

Usage

```
WorkingAdolescents
```

Format

A data frame of 478 observations and 6 variables:

Sex Either Male or Female

Relationship Relationship status of the person

ID The unique identifier for the person

Age Age of the person

HoursWorked The number of hours worked in employment, per week

SchoolStatus The indicator of whether the person is in school (Y) or not (N)

Source

Timaru District 2018 census data (tablecodes 8277, 8395, and 8460). School status was added using school leavers data produced by the Ministry of Education. Statistics New Zealand and the Ministry of Education's data are licensed, separately, for reuse under the Creative Commons Attribution 4.0 International licence.

Index