Package 'socialmixr'

May 9, 2026

Title Social Mixing Matrices for Infectious Disease Modelling

Version 0.6.0

Description Methods for sampling contact matrices from diary data for use in infectious disease modelling, as discussed in Mossong et al. (2008) <doi:10.1371/journal.pmed.0050074>.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports checkmate, countrycode, curl, data.table, grDevices, httr, jsonlite, lifecycle, lubridate, memoise, purrr, oai, wpp2017, xml2, cli, rlang, methods

Suggests contactsurveys, ggplot2, here, knitr, quarto, reshape2, rmarkdown, roxyglobals (>= 1.0.0), testthat, withr

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.3.3

URL https://github.com/epiforecasts/socialmixr, https://epiforecasts.io/socialmixr/

BugReports https://github.com/epiforecasts/socialmixr/issues

Config/testthat/edition 3

Author Sebastian Funk [aut, cre],
Lander Willem [aut],
Hugo Gruson [aut],
Nicholas Tierney [aut] (ORCID: <https://orcid.org/0000-0003-1460-8722>),
Maria Bekker-Nielsen Dunbar [ctb],
Carl A. B. Pearson [ctb],
Sam Clifford [ctb],
Christopher Jarvis [ctb],
Alexis Robert [ctb],

Niel Hens [ctb],
Pietro Coletti [col, dtm],
Lloyd Chapman [ctb]

Maintainer Sebastian Funk <sebastian.funk@lshtm.ac.uk>

Repository CRAN

Date/Publication 2026-04-29 06:40:18 UTC

Contents

agegroups_to_limits *Convert age groups to lower age limits*

Description

Inverse of `limits_to_agegroups()`. Extracts lower age limits from age group labels.

Usage

```
agegroups_to_limits(x)
```

Arguments

x	age groups (a factor, as produced by `limits_to_agegroups()` or `assign_age_groups()`)

Value

a numeric vector of lower age limits

Examples

```
agegroups_to_limits(limits_to_agegroups(c(0, 5, 10), notation = "brackets"))
```

assign_age_groups *Assign age groups in survey data*

Description

This function processes age data in a survey object. It imputes ages from ranges, handles missing values, and assigns age groups.

Usage

```
assign_age_groups(
  survey,
  age_limits = NULL,
  estimated_participant_age = c("mean", "sample", "missing"),
  estimated_contact_age = c("mean", "sample", "missing"),
  missing_participant_age = c("remove", "keep"),
  missing_contact_age = c("remove", "sample", "keep", "ignore")
)
```

Arguments

survey	a survey() object
age_limits	lower limits of the age groups over which to construct the matrix. Defaults to NULL. If NULL, age limits are inferred from participant and contact ages.
estimated_participant_age	if set to "mean" (default), people whose ages are given as a range (in columns named "..._est_min" and "..._est_max") but not exactly (in a column named "..._exact") will have their age set to the mid-point of the range; if set to "sample", the age will be sampled from the range; if set to "missing", age ranges will be treated as missing
estimated_contact_age	if set to "mean" (default), contacts whose ages are given as a range (in columns named "..._est_min" and "..._est_max") but not exactly (in a column named "..._exact") will have their age set to the mid-point of the range; if set to "sample", the age will be sampled from the range; if set to "missing", age ranges will be treated as missing
missing_participant_age	if set to "remove" (default), participants without age information are removed; if set to "keep", participants with missing age are kept and treated as a separate age group
missing_contact_age	if set to "remove" (default), participants that have contacts without age information are removed; if set to "sample", contacts without age information are sampled from all the contacts of participants of the same age group; if set to "keep", contacts with missing age are kept and treated as a separate age group; if set to "ignore", contact with missing age are ignored in the contact analysis

Value

The survey object with processed age data.

Examples

```
polymod_grouped <- assign_age_groups(polymod)
polymod_grouped
polymod_custom <- assign_age_groups(polymod, age_limits = c(0, 5, 10, 15))
polymod_custom
```

---

as_contact_survey *Check contact survey data*

Description

Checks that a survey fulfills all the requirements to work with the 'contact_matrix' function

Usage

```
as_contact_survey(
  x,
  id_column = "part_id",
  country_column = NULL,
  year_column = NULL,
  ...,
  id.column = deprecated(),
  country.column = deprecated(),
  year.column = deprecated()
)
```

Arguments

`x`	list containing • an element named 'participants', a data frame containing participant information • an element named 'contacts', a data frame containing contact information • (optionally) an element named 'reference, a list containing information information needed to reference the survey, in particular it can contain$a "title", "bibtype", "author", "doi", "publisher", "note", "year"
`id_column`	the column in both the `participants` and `contacts` data frames that links contacts to participants
`country_column`	the column in the `participants` data frame containing the country in which the participant was queried; if NULL (default), will use "country" column if present
`year_column`	the column in the `participants` data frame containing the year in which the participant was queried; if NULL (default), will use "year" column if present
`...`	additional arguments (currently ignored)
`id.column`, `country.column`, `year.column`	**[Deprecated]** Use the underscore versions (e.g., `id_column`) instead.

Value

invisibly returns a character vector of the relevant columns

Examples

```
data(polymod)
check(polymod)
```

check *Check contact survey data*

Description

Checks that a survey fulfills all the requirements to work with the 'contact_matrix' function

Usage

```
## S3 method for class 'contact_survey'
check(
  x,
  id.column = "part_id",
  participant.age.column = "part_age",
  country.column = "country",
  year.column = "year",
  contact.age.column = "cnt_age",
  ...
)
```

Arguments

x	A survey() object
id.column	the column in both the participants and contacts data frames that links contacts to participants
participant.age.column	the column in the participants data frame containing participants' age; if this does not exist, at least columns "..._exact", "..._est_min" and "..._est_max" must exist (see the estimated.participant.age option in contact_matrix())
country.column	the column in the participants data frame containing the country in which the participant was queried
year.column	the column in the participants data frame containing the year in which the participant was queried
contact.age.column	the column in the contacts data frame containing the age of contacts; if this does not exist, at least columns "..._exact", "..._est_min" and "..._est_max" must exist (see the estimated.contact.age option in contact_matrix())
...	ignored

Value

invisibly returns a character vector of the relevant columns

Examples

```
data(polymod)
check(polymod)
```

clean *Clean contact survey data*

Description

Cleans survey data to work with the 'contact_matrix' function

Usage

```
## S3 method for class 'contact_survey'
clean(
  x,
  participant_age_column = "part_age",
  ...,
  participant.age.column = deprecated()
)
```

Arguments

x	A survey() object
participant_age_column	the column in x$participants containing participants' age
...	ignored
participant.age.column	**[Deprecated]** Use participant_age_column instead.

Value

a cleaned survey in the correct format

Examples

```
data(polymod)
cleaned <- clean(polymod) # not really necessary, polymod is clean
```

compute_matrix *Compute contact matrix from prepared survey data*

Description

Computes a contact matrix from a contact_survey that has been processed by assign_age_groups() and optionally weigh(). This is the final step in the pipeline workflow.

For post-processing, pipe the result into symmetrise(), split_matrix(), or per_capita().

Usage

```
compute_matrix(survey, counts = FALSE, weight_threshold = NULL)
```

Arguments

survey	a survey() object with age groups assigned (via assign_age_groups())
counts	whether to return counts instead of means
weight_threshold	numeric; if provided, weights above this threshold are capped to the threshold value and then re-normalised (default NULL)

Value

a list with elements matrix and participants

Examples

```
data(polymod)
polymod |>
  assign_age_groups(age_limits = c(0, 5, 15)) |>
  compute_matrix()
```

---

contact_age_distribution

Extract the empirical age distribution of contacts from a survey

---

Description

Returns a data.frame of (age, proportion) pairs representing how contact ages are distributed in the survey. This can be passed to assign_age_groups() as estimated_contact_age to impute ages from ranges using this distribution rather than uniform sampling.

Usage

```
contact_age_distribution(survey)
```

Arguments

survey	a survey() object

Value

a data.frame with columns age (integer) and proportion (numeric, summing to 1)

Examples

```
data(polymod)
dist <- contact_age_distribution(polymod)
head(dist)
plot(dist$age, dist$proportion, type = "h",
     xlab = "Age", ylab = "Proportion")
```

`contact_matrix` *Generate a contact matrix from diary survey data*

Description

Samples a contact survey

Usage

```
contact_matrix(
  survey,
  countries = NULL,
  survey_pop = NULL,
  age_limits = NULL,
  filter = NULL,
  counts = FALSE,
  symmetric = FALSE,
  split = FALSE,
  sample_participants = FALSE,
  estimated_participant_age = c("mean", "sample", "missing"),
  estimated_contact_age = c("mean", "sample", "missing"),
  missing_participant_age = c("remove", "keep"),
  missing_contact_age = c("remove", "sample", "keep", "ignore"),
  weights = NULL,
  weigh_dayofweek = FALSE,
  weigh_age = FALSE,
  weight_threshold = NA,
  symmetric_norm_threshold = 2,
  sample_all_age_groups = FALSE,
  sample_participants_max_tries = 1000,
  return_part_weights = FALSE,
  return_demography = NA,
  per_capita = FALSE,
  ...,
  survey.pop = deprecated(),
  age.limits = deprecated(),
  sample.participants = deprecated(),
  estimated.participant.age = deprecated(),
  estimated.contact.age = deprecated(),
```

```
  missing.participant.age = deprecated(),
  missing.contact.age = deprecated(),
  weigh.dayofweek = deprecated(),
  weigh.age = deprecated(),
  weight.threshold = deprecated(),
  symmetric.norm.threshold = deprecated(),
  sample.all.age.groups = deprecated(),
  sample.participants.max.tries = deprecated(),
  return.part.weights = deprecated(),
  return.demography = deprecated(),
  per.capita = deprecated()
)
```

Arguments

survey	a survey() object.
countries	limit to one or more countries; if NULL (default), will use all countries in the survey; these can be given as country names or 2-letter (ISO Alpha-2) country codes.
survey_pop	survey population – either a data frame with columns 'lower.age.limit' and 'population', or a character vector giving the name(s) of a country or countries from the list that can be obtained via wpp_countries; if NULL (default), will use the country populations from the chosen countries, or all countries in the survey if countries is NULL.
age_limits	lower limits of the age groups over which to construct the matrix. If NULL (default), age limits are inferred from participant and contact ages.
filter	any filters to apply to the data, given as list of the form (column=filter_value) - only contacts that have 'filter_value' in 'column' will be considered. If multiple filters are given, they are all applied independently and in the sequence given. Default value is NULL; no filtering performed.
counts	whether to return counts (instead of means).
symmetric	whether to make matrix symmetric, such that $c_{ij}N_i = c_{ji}N_j$.
split	whether to split the contact matrix into the mean number of contacts, in each age group (split further into the product of the mean number of contacts across the whole population (mean.contacts), a normalisation constant (normalisation) and age-specific variation in contacts (contacts)), multiplied with an assortativity matrix (assortativity) and a population multiplier (demography). For more detail on this, see the "Getting Started" vignette.
sample_participants	whether to sample participants randomly (with replacement); done multiple times this can be used to assess uncertainty in the generated contact matrices. See the "Bootstrapping" section in the vignette for how to do this.
estimated_participant_age	if set to "mean" (default), people whose ages are given as a range (in columns named "..._est_min" and "..._est_max") but not exactly (in a column named

"..._exact") will have their age set to the mid-point of the range; if set to "sample", the age will be sampled from the range; if set to "missing", age ranges will be treated as missing

`estimated_contact_age`
: if set to "mean" (default), contacts whose ages are given as a range (in columns named "..._est_min" and "..._est_max") but not exactly (in a column named "..._exact") will have their age set to the mid-point of the range; if set to "sample", the age will be sampled from the range; if set to "missing", age ranges will be treated as missing.

`missing_participant_age`
: if set to "remove" (default), participants without age information are removed; if set to "keep", participants with missing age are kept and will appear in the contact matrix in a row labelled "NA".

`missing_contact_age`
: if set to "remove" (default), participants that have contacts without age information are removed; if set to "sample", contacts without age information are sampled from all the contacts of participants of the same age group; if set to "keep", contacts with missing age are kept and will appear in the contact matrix in a column labelled "NA"; if set to "ignore", contacts without age information are removed from the analysis (but the participants that made them are kept).

`weights`
: column name(s) of the participant data of the `survey()` object with user-specified weights (default = empty vector).

`weigh_dayofweek`
: whether to weigh social contacts data by the day of the week (weight (5/7 / N_week / N) for weekdays and (2/7 / N_weekend / N) for weekends).

`weigh_age`
: whether to weigh social contacts data by the age of the participants (vs. the populations' age distribution).

`weight_threshold`
: threshold value for the standardized weights before running an additional standardisation (default 'NA' = no cutoff).

`symmetric_norm_threshold`
: threshold value for the normalization weights when `symmetric = TRUE` before showing a warning that that large differences in the size of the sub-populations are likely to result in artefacts when making the matrix symmetric (default 2).

`sample_all_age_groups`
: what to do if sampling participants (with `sample_participants = TRUE`) fails to sample participants from one or more age groups; if FALSE (default), corresponding rows will be set to NA, if TRUE the sample will be discarded and a new one taken instead.

`sample_participants_max_tries`
: maximum number of attempts when `sample_all_age_groups = TRUE`; defaults to 1000.

`return_part_weights`
: boolean to return the participant weights.

`return_demography`
: boolean to explicitly return demography data that corresponds to the survey data (default 'NA' = if demography data is requested by other function parameters).

per_capita — whether to return a matrix with contact rates per capita (default is FALSE and not possible if 'counts=TRUE' or 'split=TRUE').

... — further arguments to pass to `get_survey()`, `check()` and `pop_age()` (especially column names).

`survey.pop`, `age.limits`, `sample.participants`, `estimated.participant.age`, `estimated.contact.age`, `missing.participant.age`, `missing.contact.age`, `weigh.dayofweek`, `weigh.age`, `weight.threshold`, `symmetric.norm.threshold`, `sample.all.age.groups`, `sample.participants.max.tries`, `return.part.weights`, `return.demography`, `per.capita`

[Deprecated] Use the underscore-separated versions of these arguments instead.

Value

a contact matrix, and the underlying demography of the surveyed population

Author(s)

Sebastian Funk

Examples

```
data(polymod)
contact_matrix(
  survey = polymod,
  countries = "United Kingdom",
  age_limits = c(0, 1, 5, 15)
)
```

---

download_survey *Download a survey from its Zenodo repository*

Description

[Deprecated]

`download_survey()` has been deprecated in favour of `contactsurveys::download_survey()`.

`download_survey()` downloads survey data from Zenodo.

Usage

```
download_survey(survey, dir = NULL, sleep = 1)
```

Arguments

survey — a URL (see `contactsurveys::list_surveys()`)

dir — a directory to save the files to; if not given, will save to a temporary directory

sleep — time to sleep between requests to avoid overloading the server (passed on to `Sys.sleep`)

Value

a vector of filenames that can be used with load_survey

See Also

load_survey

Examples

```
# we recommend using the contactsurveys package for download_survey()
## Not run:
# if needed, discover surveys with:
contactsurveys::list_surveys()
peru_survey <- download_survey("https://doi.org/10.5281/zenodo.1095664")
# -->
peru_survey <- contactsurveys::download_survey(
  "https://doi.org/10.5281/zenodo.1095664"
)

## End(Not run)
```

`get_citation` *Citation for a survey*

Description

[Deprecated]

`get_citation()` has been deprecated in favour of `contactsurveys::get_citation()`.

Gets a full citation for a `survey()`.

Usage

```
get_citation(x)
```

Arguments

`x`	a character vector of surveys to cite

Value

citation as bibentry

Examples

```
# we recommend using the contactsurveys package for get_citation()
## Not run:
data(polymod)
citation <- contactsurveys::get_citation(polymod)
print(citation)
print(citation, style = "bibtex")

## End(Not run)
```

`get_survey` *Get a survey, either from its Zenodo repository, a set of files, or a survey variable*

Description

[Deprecated]

`get_survey()` has been deprecated in favour of using `contactsurveys::download_survey()` and then `load_survey()`.

Downloads survey data, or extracts them from files, and returns a clean data set. If a survey URL is accessed multiple times, the data will be cached (unless `clear_cache` is set to `TRUE`) to avoid repeated downloads.

If survey objects are used repeatedly the downloaded files can be saved and reloaded between sessions then survey objects can be saved/loaded using `base::saveRDS()` and `base::readRDS()`, or via the individual survey files that can be downloaded using `download_survey()` and subsequently loaded using `load_survey()`.

Usage

```
get_survey(survey, clear_cache = FALSE, ...)
```

Arguments

`survey`	a DOI or url to get the survey from, or a `survey()` object.
`clear_cache`	logical, whether to clear the cache before downloading the survey; by default, the cache is not cleared and so multiple calls of this function to access the same survey will not result in repeated downloads.
`...`	currently unused

Value

a survey in the correct format

Examples

```
## Not run:
list_surveys()
peru_doi <- "https://doi.org/10.5281/zenodo.1095664"
peru_survey <- get_survey(peru_doi)
## --> We now recommend:
peru_survey <- contactsurveys::download_survey(peru_doi)
peru_data <- load_survey(peru_survey)

## End(Not run)
```

`is_doi` *Checks if a character string is a DOI*

Description

Checks if a character string is a DOI

Usage

```
is_doi(x)
```

Arguments

`x`	Character vector; the string or strings to check

Value

Logical; `TRUE` if `x` is a DOI, `FALSE` otherwise

Author(s)

Sebastian Funk

`limits_to_agegroups` *Convert lower age limits to age groups.*

Description

Mostly used for plot labelling

Usage

```
limits_to_agegroups(
  x,
  limits = sort(unique(x)),
  notation = c("dashes", "brackets")
)
```

Arguments

x	age limits to transform
limits	lower age limits; if not given, will use all limits in x
notation	whether to use bracket notation, e.g. [0,4) or dash notation, e.g. 0-4)

Value

Age groups as specified in notation

Examples

```
limits_to_agegroups(c(0, 5, 10))
```

---

list_surveys *List all surveys available for download*

Description

[Deprecated]

list_surveys() has been deprecated in favour of contactsurveys::list_surveys().

Usage

```
list_surveys(clear_cache = FALSE)
```

Arguments

clear_cache	logical, whether to clear the cache before downloading the survey; by default, the cache is not cleared and so multiple calls of this function to access the same survey will not result in repeated downloads.

Value

character vector of surveys

Examples

```
# we recommend using the contactsurveys package now for listing surveys.
## Not run:
contactsurveys::list_surveys()

## End(Not run)
```

load_survey *Load a survey from local files*

Description

Loads a survey from a local file system. Tables are expected as csv files, and a reference (if present) as JSON.

Usage

```
load_survey(files, participant_key = NULL, ...)
```

Arguments

files	a vector of file names as returned by download_survey()
participant_key	character vector specifying columns that uniquely identify participant observations. For cross-sectional surveys this is typically just "part_id" (the default). For longitudinal surveys with multiple observations per participant, specify additional columns like c("part_id", "wave"). When NULL (the default), the function will auto-detect if additional columns are needed and inform you.
...	options for clean(), which is called at the end of this

Value

a survey in the correct format. For longitudinal surveys with multiple observations per participant, the returned object includes an observation_key field containing the column names (excluding part_id) that distinguish observations for the same participant.

Examples

```
## Not run:
list_surveys()
peru_files <- download_survey("https://doi.org/10.5281/zenodo.1095664")
peru_survey <- load_survey(peru_files)

# For longitudinal surveys, specify the unique key explicitly:
france_files <- download_survey("https://doi.org/10.5281/zenodo.1157918")
france_survey <- load_survey(france_files,
  participant_key = c("part_id", "wave", "studyDay")
)

## End(Not run)
```

`matrix_plot` *Draws an image plot of a contact matrix with a legend strip and the numeric values in the cells.*

Description

This function combines the R image.plot function with numeric contact rates in the matrix cells.

Usage

```
matrix_plot(
  mij,
  min.legend = 0,
  max.legend = NA,
  num.digits = 2,
  num.colors = 50,
  main,
  xlab,
  ylab,
  legend.width,
  legend.mar,
  legend.shrink,
  cex.lab,
  cex.axis,
  cex.text,
  color.palette = heat.colors
)
```

Arguments

`mij`	a contact matrix containing contact rates between participants of age i (rows) with contacts of age j (columns). This is the default matrix format of `contact_matrix()`.
`min.legend`	the color scale minimum (default = 0). Set to NA to use the minimum value of `mij`.
`max.legend`	the color scale maximum (default = NA). Set to NA to use the maximum value of `mij`.
`num.digits`	the number of digits when rounding the contact rates (default = 2). Use NA to disable this.
`num.colors`	the number of color breaks (default = 50)
`main`	the figure title
`xlab`	a title for the x axis (default: "Age group (year)")
`ylab`	a title for the y axis (default: "Contact age group (year)")
`legend.width`	width of the legend strip in characters. Default is 1.
`legend.mar`	width in characters of legend margin. Default is 5.1.

legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot. Default is 0.9.
cex.lab	size of the x and y labels (default: 1.2)
cex.axis	size of the axis labels (default: 0.8)
cex.text	size of the numeric values in the matrix (default: 1)
color.palette	the color palette to use (default: heat.colors()). Other examples are topo.colors(), terrain.colors() and hcl.colors(). User-defined functions are also possible if they take the number of colors to be in the palette as function argument.

Details

This is a function using basic R graphics to visualise a social contact matrix.

Author(s)

Lander Willem

Examples

```
## Not run:
data(polymod)
mij <- contact_matrix(
  polymod,
  countries = "United Kingdom",
  age_limits = c(0, 18, 65)
)$matrix
matrix_plot(mij)

## End(Not run)
```

---

per_capita *Convert a contact matrix to per-capita rates*

Description

Divides each column of the contact matrix by the population of the corresponding age group, giving the contact rate of age group i with one individual of age group j.

Usage

```
per_capita(x, survey_pop, ...)
```

Arguments

x	a list as returned by compute_matrix(), with elements matrix and participants
survey_pop	a data frame with columns lower.age.limit and population (e.g. from wpp_age())
...	passed to pop_age() for interpolation

Value

x with $matrix replaced by the per-capita version

Examples

```
data(polymod)
pop <- wpp_age("United Kingdom", 2005)
polymod |>
  (\(s) s[country == "United Kingdom"])() |>
  assign_age_groups(age_limits = c(0, 5, 15)) |>
  compute_matrix() |>
  per_capita(survey_pop = pop)
```

---

polymod *Social contact data from 8 European countries*

Description

A dataset containing social mixing diary data from 8 European countries: Belgium, Germany, Finland, Great Britain, Italy, Luxembourg, The Netherlands and Poland. The Data are fully described in Mossong J, Hens N, Jit M, Beutels P, Auranen K, Mikolajczyk R, et al. (2008) Social Contacts and Mixing Patterns Relevant to the Spread of Infectious Diseases. PLoS Med 5(3): e74.

Usage

```
polymod
```

Format

A list of two data frames:

participants the study participant, with age, country, year and day of the week (starting with 1 = Monday)

contacts reported contacts of the study participants. The variable phys_contact has two levels (1 denotes physical contact while 2 denotes non-physical contact), duration_multi has five levels (1 is less than 5 minutes while 5 is more than 4 hours, increasing in the order found in Figure 1 in Mossong et al.), and frequency_multi has five levels (1 is daily, 2 is weekly, 3 is monthly, 4 is less often, and 5 is first time) All other variables are described on the Zenodo repository of the data, available at doi:10.5281/zenodo.1043437

Source

doi:10.1371/journal.pmed.0050074

pop_age *Change age groups in population data*

Description

This changes population data to have age groups with the given age_limits, extrapolating linearly between age groups (if more are requested than available) and summing populations (if fewer are requested than available)

Usage

```
pop_age(
  pop,
  age_limits = NULL,
  pop_age_column = "lower.age.limit",
  pop_column = "population",
  ...,
  age.limits = deprecated(),
  pop.age.column = deprecated(),
  pop.column = deprecated()
)
```

Arguments

pop	a data frame with columns indicating lower age limits and population sizes (see 'pop_age_column' and 'pop_column')
age_limits	lower age limits of age groups to extract; if NULL (default), the population data is returned unchanged
pop_age_column	column in the 'pop' data frame indicating the lower age group limit
pop_column	column in the 'pop' data frame indicating the population size
...	ignored
age.limits, pop.age.column, pop.column	**[Deprecated]** Use the underscore versions (e.g., age_limits) instead.

Value

data frame of age-specific population data

Examples

```
ages_it_2015 <- wpp_age("Italy", 2015)

# Modify the age data.frame to get age groups of 10 years instead of 5
pop_age(ages_it_2015, age_limits = seq(0, 100, by = 10))

# The function will also automatically interpolate if necessary
pop_age(ages_it_2015, age_limits = c(0, 18, 40, 65))
```

reduce_agegroups *Reduce the number of age groups given a broader set of limits*

Description

Operates on lower limits

Usage

```
reduce_agegroups(x, limits)
```

Arguments

x	vector of limits
limits	new limits

Value

vector with the new age groups

Examples

```
reduce_agegroups(seq_len(20), c(0, 5, 10))
```

split_matrix *Decompose a contact matrix into mean contacts, normalisation and assortativity*

Description

Splits the contact matrix into the mean number of contacts across the whole population (`mean.contacts`), a normalisation constant (`normalisation`), age-specific contact rates (`contacts`), and an assortativity matrix (replacing `$matrix`). For details, see the "Getting Started" vignette.

Usage

```
split_matrix(x, survey_pop, ...)
```

Arguments

x	a list as returned by `compute_matrix()`, with elements `matrix` and `participants`
survey_pop	a data frame with columns `lower.age.limit` and `population` (e.g. from `wpp_age()`)
...	passed to `pop_age()` for interpolation

Value

x with $matrix replaced by the assortativity matrix, plus additional elements $mean.contacts, $normalisation, and $contacts

Examples

```
data(polymod)
pop <- wpp_age("United Kingdom", 2005)
polymod |>
  (\(s) s[country == "United Kingdom"])() |>
  assign_age_groups(age_limits = c(0, 5, 15)) |>
  compute_matrix() |>
  split_matrix(survey_pop = pop)
```

---

survey *Contact survey*

Description

Deprecated. Use as_survey instead.

Usage

```
survey(participants, contacts, reference = NULL)
```

Arguments

participants	a data.frame containing information on participants
contacts	a data.frame containing information on contacts
reference	a list containing information needed to reference the survey, in particular it can contain a "title", "bibtype", "author", "doi", "publisher", "note", "year"

Value

a new survey object

Author(s)

Sebastian Funk

survey_countries *List all countries contained in a survey*

Description

[Deprecated]

Usage

```
survey_countries(survey, country.column = "country", ...)
```

Arguments

survey	a DOI or url to get the survey from, or a survey() object.
country.column	column in the survey indicating the country
...	further arguments for get_survey()

Details

survey_countries() has been deprecated in favour of using contactsurveys::download_survey(), and load_survey(), and then exploring the country column yourself.

Value

list of countries

Examples

```
data(polymod)
survey_countries(polymod)
## --> we now recommend
## Not run:
doi_peru <- "10.5281/zenodo.1095664" # nolint
# download the data with the contactsurveys package
peru_survey <- contactsurveys::download_survey(doi_peru)
# load the survey with socialmixr
peru_data <- socialmixr::load_survey(peru_survey)
# find the unique country - assuming your data has a "country" column:
unique(peru_data$participants$country)

## End(Not run)
```

survey_country_population *Get survey country population data*

Description

Looks up the country and year inside a survey, or a provided "countries" value, and determines the corresponding demographics in the world population prospects data using `wpp_age()`.

Usage

```
survey_country_population(survey, countries = NULL)
```

Arguments

survey	A `survey()` object, with column "country" in "participants".
countries	Optional. A character vector of country names. If specified, this will be used instead of the potential "country" column in "participants".

Value

A data table with population data by age group for the survey countries, aggregated by lower age limit. The function will error if no country information is available from either the survey or countries argument.

Examples

```
survey_country_population(polymod)
survey_country_population(polymod, countries = "Belgium")
survey_country_population(polymod, countries = c("Belgium", "Italy"))
```

symmetrise *Symmetrise a contact matrix*

Description

Makes a contact matrix symmetric so that $c_{ij}N_i = c_{ji}N_j$, where c_{ij} is the (i, j) entry and N_i is the population of age group i. This is done by replacing each pair with half their sum, weighted by population size.

Usage

```
symmetrise(x, survey_pop, symmetric_norm_threshold = 2, ...)
```

Arguments

x	a list as returned by compute_matrix(), with elements matrix and participants
survey_pop	a data frame with columns lower.age.limit and population (e.g. from wpp_age())
symmetric_norm_threshold	threshold for the normalisation factor before issuing a warning (default 2)
...	passed to pop_age() for interpolation

Value

x with $matrix replaced by the symmetrised version

Examples

```
data(polymod)
pop <- wpp_age("United Kingdom", 2005)
polymod |>
  (\(s) s[country == "United Kingdom"])() |>
  assign_age_groups(age_limits = c(0, 5, 15)) |>
  compute_matrix() |>
  symmetrise(survey_pop = pop)
```

---

weigh *Weigh survey participants*

Description

Applies weights to participants in a contact_survey object. Weights are always multiplied into an existing weight column (or one is created with value 1), making multiple calls composable.

The behaviour depends on the combination of arguments:

target = NULL Numeric column: multiply weight by column values directly.

Unnamed target + groups Map column values to groups, assign target[g] / n_in_group per participant.

Named target Names match column values, assign target[val] / n_with_val per participant.

Data frame target Post-stratify against population data (expanded to single-year ages via pop_age()).

Usage

```
weigh(survey, by, target = NULL, groups = NULL, ...)
```

Arguments

survey	a survey() object (must have been processed by assign_age_groups() if using data frame target)
by	column name in the participant data to weigh by
target	target weights: NULL for direct numeric weighting, an unnamed numeric vector (with groups), a named numeric vector, or a data frame with columns lower.age.limit and population
groups	a list of value sets mapping column values to groups (used with unnamed target vector); must be the same length as target
...	further arguments passed to pop_age() when target is a data frame

Value

the survey object with updated participant weights

Examples

```
data(polymod)
# Direct numeric weighting
if ("survey_weight" %in% names(polymod$participants)) {
  polymod |> weigh("survey_weight")
}

# Dayofweek weighting with groups (POLYMOD uses 0 = Sunday, 6 = Saturday)
polymod |>
  weigh("dayofweek", target = c(5, 2), groups = list(1:5, c(0, 6)))
```

---

wpp_age — *Get age-specific population data according to the World Population Prospects 2017 edition*

Description

[Deprecated]

This function is deprecated in favour of passing population data directly to contact_matrix() via the survey_pop argument. Additionally, the underlying wpp2017 data is outdated. For more recent population data, use the wpp2024 package from GitHub.

Usage

```
wpp_age(countries, years)
```

Arguments

countries	countries, will return all if not given
years	years, will return all if not given

Details

This uses data from the wpp2017 package but combines male and female, and converts age groups to lower age limits. If the requested year is not present in the historical data, WPP projections are used.

Value

data frame of age-specific population data

Examples

```
wpp_age("Italy", c(1990, 2000))

# For more recent data, use wpp2024 from GitHub:
# remotes::install_github("PPgp/wpp2024")
# library(wpp2024)
# data(popAge1dt)
# uk_pop <- popAge1dt[name == "United Kingdom" & year == 2020,
#                     .(lower.age.limit = age, population = pop * 1000)]
# contact_matrix(polymod, countries = "United Kingdom", survey_pop = uk_pop)
```

---

wpp_countries *List all countries and regions for which socialmixr has population data*

Description

[Deprecated]

This function is deprecated in favour of passing population data directly to contact_matrix() via the survey_pop argument, which removes the need for a country list. Additionally, the underlying wpp2017 data is outdated. For countries available in more recent WPP editions, use the wpp2024 package from GitHub.

Usage

```
wpp_countries()
```

Details

Uses the World Population Prospects data from the wpp2017 package.

Value

list of countries

Examples

```
if (requireNamespace("wpp2017", quietly = TRUE)) {
  wpp_countries()
}
```

`[.contact_survey` *Subset a contact survey*

Description

Filters a `contact_survey` object using an expression. The expression is evaluated against whichever table(s) contain the referenced columns (participants, contacts, or both). When participants are filtered, contacts are automatically pruned to matching `part_id`s.

Usage

```
## S3 method for class 'contact_survey'
x[i, ...]
```

Arguments

`x`	a `contact_survey` object
`i`	an expression to evaluate as a row filter (e.g. `country == "United Kingdom"`)
`...`	ignored

Value

a filtered `contact_survey` object

Examples

```
data(polymod)
polymod[country == "United Kingdom"]
```

Index