Package: conmat (via r-universe)

August 26, 2024

Title Builds contact matrices using GAMs and population data

Version 0.0.2.9000

Description Builds contact matrices using GAMs and population data. This package incorporates data that is copyright Commonwealth of Australia (Australian Electoral Commission and Australian Bureau of Statistics) 2020.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Suggests covr, knitr, vdiffr, testthat (>= 3.0.0), rmarkdown, future, spelling, deSolve

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports mgcv, dplyr (>= 1.0.9), tidyr (>= 1.2.0), cli, stats, socialmixr (>= 0.2.0), ggplot2, tibble (>= 3.1.8), patchwork, magrittr, rlang (>= 1.0.4), glue (>= 1.6.2), readr, furrr, purrr (>= 1.0.1), tidyselect, vctrs, scales, english, waldo, stringr

Repository https://idem-lab.r-universe.dev

RemoteUrl https://github.com/idem-lab/conmat

RemoteRef HEAD

RemoteSha 86a8702ab6bf4b840b6177ccc62ae67fb7698cb9

Contents

`abs_abbreviate_states` *Abbreviate Australian State Names*

Description

Given a full name (Title Case) of an Australian State or Territory, produces the abbreviated state name.

Usage

```
abs_abbreviate_states(state_names)
```

Arguments

`state_names`	vector of state names in long form

Value

shortened state names

See Also

abs_unabbreviate_states()

Examples

```
abs_abbreviate_states("Victoria")
abs_abbreviate_states(c("Victoria", "Queensland"))
```

---

abs_age_data *Return Australian Bureau of Statistics (ABS) age population data for a given Local Government Area (LGA) or state*

Description

Return Australian Bureau of Statistics (ABS) age population data for a given Local Government Area (LGA) or state

Usage

```
abs_age_lga(lga_name)

abs_age_state(state_name)
```

Arguments

lga_name	lga name - can be a partial match, e.g., although the official name might be "Albury (C)", "Albury" is fine.
state_name	shortened state name

Value

a conmat_population dataset containing: lga (or state), lower.age.limit, year, and population.

Examples

```
abs_age_lga(c("Albury (C)", "Fairfield (C)"))
abs_age_state(c("NSW", "VIC"))
```

abs_age_education_state *Return data on educated population for a given age and state or lga of Australia.*

Description

Return data on educated population for a given age and state or lga of Australia.

Usage

```
abs_age_education_state(state = NULL, age = NULL)

abs_age_education_lga(lga = NULL, age = NULL)
```

Arguments

state	target Australian state name or a vector with multiple state names in its abbreviated form, such as "QLD", "NSW", or "TAS"
age	a numeric or numeric vector denoting ages between 0 to 115. The default is to return all ages.
lga	target Australian local government area (LGA) name, such as "Fairfield (C)" or a vector with multiple lga names. See abs_lga_lookup() for list of lga names.

Value

dataset with information on the number of educated people belonging to a particular age, its total population and the corresponding proportion.

Examples

```
abs_age_education_state(state = "VIC")
abs_age_education_state(state = "WA", age = 1:5)
abs_age_education_state(state = c("QLD", "TAS"), age = 5)
abs_age_education_lga(lga = c("Albury (C)", "Barcoo (S)"), age = 10)
```

abs_age_work_lga *Return data on employed population for a given age and state or lga of Australia*

Description

Return data on employed population for a given age and state or lga of Australia

Usage

```
abs_age_work_lga(lga = NULL, age = NULL)

abs_age_work_state(state = NULL, age = NULL)
```

Arguments

lga	target Australian local government area (LGA) name, such as "Fairfield (C)" or a vector with multiple lga names. See `abs_lga_lookup()` for list of lga names.
age	a numeric or numeric vector denoting ages between 0 to 115. The default is to return all ages.
state	target Australian state name or a vector with multiple state names in its abbreviated form, such as "QLD", "NSW", or "TAS"

Value

data set with information on the number of employed people belonging to a particular age, its total population and the corresponding proportion.

Examples

```
abs_age_work_state(state = "NSW")
abs_age_work_state(state = c("QLD", "TAS"), age = 5)
abs_age_work_lga(lga = "Albany (C)", age = 1:5)
abs_age_work_lga(lga = c("Albury (C)", "Barcoo (S)"), age = 39)
```

---

abs_avg_school *ABS education data for 2016*

Description

An internal dataset containing Australian Bureau of Statistics education data for each age in 2016. The data is averaged across each state to provide an overall average, and is used to provide estimated education populations for model fitting in `add_school_work_participation()`, which is used in `fit_single_contact_model()`. The data is summarised from `data_abs_state_education`, see `?data_abs_state_education` for more details.

Usage

```
abs_avg_school
```

Format

A data frame with 116 rows and 2 variables:

age 0 to 115

school_fraction fraction of population at school

Source

Census of Population and Housing, 2016, TableBuilder

`abs_avg_work` *ABS work data for 2016*

Description

An internal dataset containing Australian Bureau of Statistics work data for each age in 2016. The data is averaged across each state to provide an overall average, and is used to provide estimated work populations for model fitting in `add_school_work_participation()`, which is used in `fit_single_contact_model()`. The data is summarised from `data_abs_state_work`, see `?data_abs_state_work` for more details.

Usage

```
abs_avg_work
```

Format

A data frame with 116 rows and 2 variables:

age 0 to 115

work_fraction fraction of population working.

Source

Census of Population and Housing, 2016, TableBuilder

`abs_education_state` *ABS education by state for 2006-2020*

Description

A dataset containing Australian Bureau of Statistics education data by state for 2006 to 2020

Usage

```
abs_education_state
```

Format

A data frame with 4194 rows and 5 variables:

year year - 2020

state state - short state or territory name

aboriginal_and_torres_strait_islander_status "Aboriginal and Torres Strait Islander" or "Non-Indigenous"

age 4 through to 21. Note that "4" is 4 or younger and "21" is actually 21+ (21 or older)

n_full_and_part_time number of people full and part time

Source

https://www.abs.gov.au/statistics/people/education/schools/2020#data-download (table 42B)

`abs_education_state_2020`

2020 ABS education population data, interpolated into 1 year bins, by state.

Description

A dataset containing Australian Bureau of Statistics education data by state for 2020. These were interpolated into 1 year age bins. There are still some issued with the methods used, as the interpolated values are sometimes higher than the population.

Usage

```
abs_education_state_2020
```

Format

A data frame with 808 rows and 6 variables:

year year - 2020

state state - short state or territory name

age 0 to 100

population number of people full and part time

population_interpolated "Government" or "Non-government"

prop population / population_interpolated

Source

https://www.abs.gov.au/statistics/people/education/schools/2020#data-download (table 42B)

abs_employ_age_lga *ABS employment by age and LGA for 2016*

Description

A dataset containing Australian Bureau of Statistics employment data by state for 2016

Usage

```
abs_employ_age_lga
```

Format

A data frame with 5600 rows and 8 variables:

year year - 2016

state state - short state or territory name

lga local government area name

age_group age groups are as follows: 15-19, 20-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75-84, 85+, total

total_employed total number of people employed

total_unemployed total number of people unemployed

total_labour_force total number of people in the labour force

total sum of these totals...or thereabouts??

Note

still need to finalise these columns

Source

ABS.stat https://stat.data.abs.gov.au/Index.aspx? LABOUR > Employment and Unemployment > Labour force status > Census 2016, G43 labour status by age and sex (LGA)

abs_household_lga *ABS household data for 2016*

Description

A dataset containing Australian Bureau of Statistics household data for 2016. The data is filtered to "Total Households". Contains information on the number of people typically in a residence in the region and the number of households associated with those number of residents. This data is typically used to obtain the household size distributions to compute the per capita household size of a particular region.

Usage

```
abs_household_lga
```

Format

A data frame with 4986 rows and 6 variables:

year year - 2016

state state - long state or territory name

lga name of LGA

n_persons_usually_resident Number of people typically in residence

n_households number of households with that number of people

Note

still need to clean this

Source

https://stat.data.abs.gov.au/Index.aspx?Datasetcode=ABS_FAMILY_PROJ (downloaded the CSV) PEOPLE > People and Communities > Household Composition > Census 2016, T23 Household Composition By Number Of Persons Usually Resident (LGA)

abs_lga_lookup *ABS lookup table of states, lga code and lga name*

Description

A dataset containing Australian Bureau of Statistics official short state names, lga_code, and lga name.

Usage

```
abs_lga_lookup
```

Format

A data frame with 544 rows and 3 variables, arrange by state then LGA

state state - short state or territory name

lga_code official lga code

lga lga name

`abs_pop_age_lga_2016` *ABS population by age for 2016 for LGAs*

Description

A dataset containing Australian Bureau of Statistics population data by local government area (LGA) for age for 2016

Usage

```
abs_pop_age_lga_2016
```

Format

A data frame with 9918 rows and 6 variables:

year year - 2020

state state - short state or territory name

lga LGA name

age_group age age band, 0-4, in 5 year increments up to 80-84, then 85+

population number of people in a given lga in an age band

Source

https://www.abs.gov.au/statistics/people/population/regional-population-age-and-sex

abs_pop_age_lga_2020 *ABS population by age for 2020 for LGAs*

Description

A dataset containing Australian Bureau of Statistics population data by local government area (LGA) for age for 2020

Usage

```
abs_pop_age_lga_2020
```

Format

A data frame with 9900 rows and 6 variables:

year year - 2020

state state - long state or territory name

lga LGA name

age_group age group band, 0-4, in 5 year increments up to 80-84, then 85+

population number of people in a given lga in an age band

Source

https://www.abs.gov.au/statistics/people/population/regional-population-age-and-sex

abs_state_age *ABS state population data for 2020*

Description

Dataset containing Australian Bureau of Statistics state level population data for 2020

Usage

```
abs_state_age
```

Format

A data frame with 168 rows and 3 variables:

state state - short state or territory name

age_group age group in five year bins from 0 to 99, then 100+

population population size

Source

https://www.abs.gov.au/statistics/people/population/national-state-and-territory-population/dec-2020#data-downloads-data-cubes

---

abs_unabbreviate_states *Un-abbreviate Australian state names*

Description

Un-abbreviate Australian state names

Usage

```
abs_unabbreviate_states(state_names)
```

Arguments

state_names	vector of state names in short form

Value

Longer state names

See Also

abs_abbreviate_states()

Examples

```
abs_unabbreviate_states("VIC")
abs_unabbreviate_states(c("VIC", "QLD"))
```

---

add_intergenerational *Add column, "intergenerational"*

Description

For modelling purposes it is useful to have a feature that is the absolute difference between age_from and age_to columns.

Usage

```
add_intergenerational(data)
```

Arguments

data	data.frame with columns age_from, and age_to

Value

data.frame with extra column, intergenerational

Examples

```
polymod_contact <- get_polymod_contact_data()

polymod_contact %>% add_intergenerational()
```

---

add_modelling_features

Add features required for modelling to the dataset

Description

This function adds three main groups of features to the data. It is used internally in fit_single_contact_model() and predict_contacts_1y(). It requires columns named age_to and age_from. The three types of features it adds are described below:

1. Population distribution of contact ages from the function add_population_age_to(), which requires a column called "age_to" representing the age of the person who had contact. It creates a column called pop_age_to. add_population_age_to() takes an extra argument for population, which defaults to get_polymod_population(), but needs to be a conmat_population object, which specifies the age and population characteristics, or a data frame with columns, lower.age.limit, and population.
2. School work participation, which is from the function add_school_work_participation(). This requires columns age_to and age_from, but will operate on any column starting with age and adds columns: school_probability, work_probability, school_year_probability, and school_weighted_pop_fraction.
3. Offset is added on to the data using add_offset(). This requires variables school_weighted_pop_fraction (from add_school_work_participation()) and pop_age_to (from add_school_work_participation()). It adds two columns, log_contactable_population_school, and log_contactable_population.

Usage

```
add_modelling_features(
  contact_data,
  school_demographics = NULL,
  work_demographics = NULL,
  population = get_polymod_population()
)
```

Arguments

contact_data	contact data with columns age_to and age_from
school_demographics	(optional) defaults to census average proportion at school. You can provide a dataset with columns, "age" (numeric), and "school_fraction" (0-1), if you would like to specify these details. See abs_avg_school for the default values. If you would like to use the original school demographics used in conmat, these are provided in the dataset, conmat_original_school_demographics.
work_demographics	(optional) defaults to census average proportion employed. You can provide a dataset with columns, "age" (numeric), and "work_fraction", if you would like to specify these details. See abs_avg_work for the default values. If you would like to use the original work demographics used in conmat, these are provided in the dataset, conmat_original_work_demographics.
population	the population argument of add_population_age_to()

Value

data frame with 11 extra columns - the contents of contact_data, plus: pop_age_to, school_fraction_age_from, work_fraction_age_from, school_fraction_age_to, work_fraction_age_to, school_probability, work_probability, school_year_probability, school_weighted_pop_fraction, log_contactable_population_school, and log_contactable_population.

Examples

```
age_min <- 10
age_max <- 15
all_ages <- age_min:age_max
library(tidyr)
example_df <- expand_grid(
  age_from = all_ages,
  age_to = all_ages,
)
add_modelling_features(example_df)
add_modelling_features(
  example_df,
  school_demographics = conmat_original_school_demographics,
  work_demographics = conmat_original_work_demographics
)
```

---

add_offset *Adds offset variables*

Description

Mostly used internally in add_modelling_features(). Adds two offset variables to be used in fit_single_contact_model():

1. log_contactable_population_school, and
2. log_contactable_population. These two variables require variables school_weighted_pop_fraction (from add_school_work_participation()) and pop_age_to (from add_school_work_participation()). This provides separate offsets for school setting when compared to the other settings such as home, work and other. The offset for school captures cohorting of students for schools and takes the logarithm of the weighted combination of contact population age distribution & school year probability calculated in add_school_work_participation(). See "details" for more information.

Usage

```
add_offset(contact_data)
```

Arguments

contact_data	contact data - must contain columns age_to, age_from, pop_age_to (from add_population_age_to(), and school_weighted_pop_fraction (from add_school_work_partici

Details

why double offsets? There are two offsets specified, once in the model formula, and once in the "offset" argument of mgcv::bam. The offsets get added together when the model first fit. In addition, the setting specific offset from offset_variable, which is included in the GAM model as ... + offset(log_contactable_population) is used in prediction, whereas the other offset, included as an argument in the GAM as offset = log(participants) is only included when the model is initially created. See more detail in fit_single_contact_model().

Value

data.frame of contact_data with two extra columns: log_contactable_population_school and log_contactable_population

Author(s)

Nick Golding

Examples

```
age_min <- 10
age_max <- 15
all_ages <- age_min:age_max
library(tidyr)
example_df <- expand_grid(
  age_from = all_ages,
  age_to = all_ages,
)
```

```
example_df %>%
  add_population_age_to() %>%
  add_school_work_participation() %>%
  add_offset()
```

---

add_population_age_to *Add the population distribution for contact ages.*

Description

Adds the population distribution of contact ages. Requires a column called "age_to", representing the contact age - the age of the person who had contact. It creates a column, pop_age_to. The population argument defaults to get_polymod_population(), which is a conmat_population object, which has age and population specified. But this can also be a data frame with columns, lower.age.limit, and population. If population is 'polymod' then use the participant-weighted average of POLYMOD country/year distributions. It adds the population via interpolation, using get_age_population_function() to create a function that generates population from ages.

Usage

```
add_population_age_to(contact_data, population = get_polymod_population())
```

Arguments

contact_data	contact data containing columns age_to and age_from
population	Defaults to get_polymod_population(), a conmat_population object, which specifies the age and population columns. But it can optionally be any data frame with columns, lower.age.limit, and population.

Value

data frame

Examples

```
age_min <- 10
age_max <- 15
all_ages <- age_min:age_max
library(tidyr)
example_df <- expand_grid(
  age_from = all_ages,
  age_to = all_ages,
)
add_population_age_to(example_df)
```

`add_school_work_participation` *Add columns describing the fractions of the population in each age group that attend school/work (average FTE)*

Description

Add fractions of the population in each age group that attend school/work (average FTE) to compute the probability that both participant and contact attend school/work. Requires columns `age_to` and `age_from`. Note that it will operate on any column starting with `age`. Adds columns: `school_probability`, `work_probability`, `school_year_probability`, and `school_weighted_pop_fraction`. The columns `school_probability` and `work_probability` represent the probability a person of the other age goes to the same work/school. `school_year_probability` represents the probability that a person of the other age would be in the same school year. `school_weighted_pop_fraction` represents the weighted combination of contact population age distribution & school year probability, so that if the contact is in the same school year, the weight is 1, and otherwise it is the population age fraction. This can be used as an offset, so that population age distribution can be used outside the classroom, but does not affect classroom contacts (which due to cohorting and regularised class sizes are unlikely to depend on the population age distribution).

Usage

```
add_school_work_participation(
  contact_data,
  school_demographics = NULL,
  work_demographics = NULL
)
```

Arguments

`contact_data`	contact data containing columns: `age_to`, `age_from`, and `pop_age_to` (from `add_population_age_to()`)
`school_demographics`	(optional) defaults to census average proportion at school. You can provide a dataset with columns, "age" (numeric), and "school_fraction" (0-1), if you would like to specify these details. See `abs_avg_school` for the default values. If you would like to use the original school demographics used in conmat, these are provided in the dataset, `conmat_original_school_demographics`.
`work_demographics`	(optional) defaults to census average proportion employed. You can provide a dataset with columns, "age" (numeric), and "work_fraction", if you would like to specify these details. See `abs_avg_work` for the default values. If you would like to use the original work demographics used in conmat, these are provided in the dataset, `conmat_original_work_demographics`.

Value

dataset with 9 extra columns: school_fraction_age_from, work_fraction_age_from, school_fraction_age_to, work_fraction_age_to, school_probability, work_probability, school_year_probability, and school_weighted_pop_fraction.

Note

To use previous approach input the arguments `school_demographics` and `work_demographics` with `conmat_original_school_demographics` and `conmat_original_work_demographics`, respectively.

Examples

```
age_min <- 10
age_max <- 15
all_ages <- age_min:age_max
library(tidyr)
example_df <- expand_grid(
  age_from = all_ages,
  age_to = all_ages,
)

example_df %>%
  add_population_age_to() %>%
  add_school_work_participation()

example_df %>%
  add_population_age_to() %>%
  add_school_work_participation(
    school_demographics = conmat_original_school_demographics,
    work_demographics = conmat_original_work_demographics
  )
```

---

add_symmetrical_features

Add symmetrical, age based features

Description

This function adds 6 columns to assist with describing various age based interactions for model fitting. Requires that the age columns are called "age_from", and "age_to"

Usage

```
add_symmetrical_features(data)
```

Arguments

`data`	data.frame with columns, `age_from`, and `age_to`

Value

data.frame with 6 more columns, gam_age_offdiag, gam_age_offdiag_2, gam_age_diag_prod, gam_age_diag_sum, gam_age_pmax, gam_age_pmin,

Examples

```
vec_age <- 0:2
dat_age <- expand.grid(
  age_from = vec_age,
  age_to = vec_age
)

add_symmetrical_features(dat_age)
```

---

age *Accessing conmat attributes*

Description

Accessing conmat attributes

Usage

```
age(x)

age_label(x)

## Default S3 method:
age_label(x)

## S3 method for class 'conmat_population'
age_label(x)

population_label(x)

## Default S3 method:
population_label(x)

## S3 method for class 'conmat_population'
population_label(x)

population(x)
```

Arguments

x	conmat_population data frame

Value

age or population symbol or label

Examples

```
## Not run:
perth <- abs_age_lga("Perth (C)")
age(perth)
age_label(perth)
population(perth)
population_label(perth)

## End(Not run)
```

---

age_breaks *Extract age break attribute information*

Description

Extract age break attribute information

Usage

```
age_breaks(x)

## S3 method for class 'conmat_age_matrix'
age_breaks(x)

## S3 method for class 'conmat_setting_prediction_matrix'
age_breaks(x)

## S3 method for class 'setting_data'
age_breaks(x)

## S3 method for class 'ngm_setting_matrix'
age_breaks(x)

## S3 method for class 'setting_vaccination_matrix'
age_breaks(x)

## S3 method for class 'numeric'
age_breaks(x)

## S3 method for class 'matrix'
age_breaks(x)

## S3 method for class 'array'
```

```
age_breaks(x)

## S3 method for class 'predicted_contacts'
age_breaks(x)

## S3 method for class 'transmission_probability_matrix'
age_breaks(x)

## S3 method for class 'setting_contact_model'
age_breaks(x)

## Default S3 method:
age_breaks(x)
```

Arguments

x	an object containing age break information

Value

age breaks character vector

Methods (by class)

- `age_breaks(conmat_age_matrix)`: Get age break information
- `age_breaks(conmat_setting_prediction_matrix)`: Get age break information
- `age_breaks(setting_data)`: Get age break information
- `age_breaks(ngm_setting_matrix)`: Get age break information
- `age_breaks(setting_vaccination_matrix)`: Get age break information
- `age_breaks(numeric)`: Get age break information
- `age_breaks(matrix)`: Get age break information
- `age_breaks(array)`: Get age break information
- `age_breaks(predicted_contacts)`: Get age break information
- `age_breaks(transmission_probability_matrix)`: Get age break information
- `age_breaks(setting_contact_model)`: Get age break information
- `age_breaks(default)`: Get age break information

Examples

```
age_breaks <- c(0, 5, 19, 15)
age_break_names <- c("[0,5)", "[5,10)", "[10, 15)")
age_mat <- matrix(
  runif(9),
  nrow = 3,
  ncol = 3,
  dimnames = list(
    age_break_names,
```

```
    age_break_names
  )
)

age_mat <- new_age_matrix(age_mat, age_breaks)

age_breaks(age_mat)
```

`age_group_lookup` *Lookup table of age groups in 5 year bins*

Description

A dataset containing age lower and upper levels with age group

Usage

```
age_group_lookup
```

Format

A data frame with 21 rows and 3 variables:

lower Lower age

upper upper age

age_group age group as a factor

`age_population` *Get cleaned population data with lower and upper limits of age.*

Description

This function helps clean up datasets of population data, which might be similar to `socialmixr::wpp_age()` or a dataset with columns representing: population, location, age, and year. If age is numeric, it groups ages into age groups with 5 year bins (0-4, 5-9, etc). It then separates age groups into two column of these lower and upper limits. Finally, it filters data passed to the specified year and location. If no year or location is provided then all years or locations are used.

Usage

```
age_population(
  data,
  location_col = NULL,
  location = NULL,
  age_col,
  year_col = NULL,
  year = NULL
)
```

Arguments

data	dataset containing information on population for a given age, country, and year
location_col	bare variable name for the column with location information. If using, both location_col & location must be specified.
location	character vector with location names. If using, both location_col & location must be specified.
age_col	bare variable name for the column with age information
year_col	bare variable name for the column with year information. If using, both year_col & year must be specified.
year	numeric vector representing the desired year(s). If using, both year_col & year must be specified.

Value

tidy dataset with information on population of different age bands

Examples

```
world_data <- socialmixr::wpp_age()
world_data
# Tidy data for multiple locations across different years
age_population(
  data = world_data,
  location_col = country,
  location = c("Asia", "Afghanistan"),
  age_col = lower.age.limit,
  year_col = year,
  year = c(2010:2020)
)

# Tidy data for a given location irrespective of year
age_population(
  data = world_data,
  location_col = country,
  location = "Afghanistan",
  age_col = lower.age.limit
)

# Tidy data for a given location irrespective of location
age_population(
  data = world_data,
  age_col = lower.age.limit
)
age_population(
  data = world_data,
  age_col = lower.age.limit,
  year_col = year,
  year = c(2011:2015)
)
```

```
# Tidy datasets with age groups
population_age_groups <- abs_pop_age_lga_2020
population_age_groups
age_population(
  data = population_age_groups,
  age_col = age_group,
  year_col = year,
  year = 2020
)

# Tidy datasets with numeric age
population_numeric_age <- abs_age_state("WA")
population_numeric_age
age_population(
  data = population_numeric_age,
  age_col = lower.age.limit,
  year_col = year,
  year = 2020
)
```

`aggregate_predicted_contacts` *Aggregate predicted contacts to specified age breaks*

Description

Aggregates contacts rate from, say, a 1 year level into provided age breaks, weighting the contact rate by the specified age population. For example, if you specify breaks as c(0, 5, 10, 15, Inf), it will return age groups as 0-5, 5-10, 10-15, and 15+ (Inf). Used internally within `predict_contacts()`, although can be used by users.

Usage

```
aggregate_predicted_contacts(
  predicted_contacts_1y,
  population,
  age_breaks = c(seq(0, 75, by = 5), Inf)
)
```

Arguments

`predicted_contacts_1y`	contacts in 1 year breaks (could technically by in other year breaks). Data must contain columns, `age_from`, `age_to`, `contacts`, and `se_contacts`, which is the same output as `predict_contacts_1y()` - see examples below.
`population`	a `conmat_population` object, which has the `age` and `population` columns specified, or a dataframe with columns `lower.age.limit`, and `population`. See examples below.
`age_breaks`	vector of ages. Default: c(seq(0, 75, by = 5), Inf)

Value

data frame with columns, `age_group_from`, `age_group_to`, and `contacts`, which is the aggregated model.

Examples

```
fairfield <- abs_age_lga("Fairfield (C)")

fairfield

# We can predict the contact rate for Fairfield from the existing contact
# data, say, between the age groups of 0-15 in 5 year bins for school:

fairfield_contacts_1 <- predict_contacts_1y(
  model = polymod_setting_models$home,
  population = fairfield,
  age_min = 0,
  age_max = 15
)

fairfield_contacts_1

aggregated_fairfield <- aggregate_predicted_contacts(
  predicted_contacts_1y = fairfield_contacts_1,
  population = fairfield,
  age_breaks = c(0, 5, 10, 15, Inf)
)

aggregated_fairfield
```

---

apply_vaccination *Apply vaccination effects to next generation contact matrices*

Description

Applies the effect of vaccination on the next generation of infections, to understand and describe the reduction of acquisition and transmission in each age group.

Usage

```
apply_vaccination(ngm, data, coverage_col, acquisition_col, transmission_col)
```

Arguments

`ngm`	next generation matrices. See `generate_ngm()` for creating next generation matrices of a state or a local government area for specific age groups
`data`	data frame with location specific information on vaccine coverage, efficacy of acquisition/susceptibility and efficacy of transmission/infectiousness for the ordered age groups from lowest to highest of the next generation matrix

coverage_col bare variable name for the column with information on vaccine coverage by age groups

acquisition_col bare variable name for the column with information on efficacy of acquisition

transmission_col bare variable name for the column with information on efficacy of transmission

Details

Vaccination improves a person's immunity from a disease. When a sizeable section of the population receives vaccinations or when vaccine coverage is sufficient enough, the likelihood that the unvaccinated population will contract the disease is decreased. This helps to slow infectious disease spread as well as lessen its severity. For this reason, it is important to understand how much of a reduction in probability of acquisition (the likelihood that an individual will contract the disease), and probability of transmission (the likelihood that an individual will spread the disease after contracting it), has occurred as an the effect of vaccination, in other words the effect of vaccination on the next generation of infections.

`apply_vaccination` returns the percentage reduction in acquisition and transmission in each age group. It does this by taking the outer product of these reductions in acquisition and transmission by age group, creating a transmission reduction matrix. The next generation matrices with the vaccination effects applied are then produced using the obtained transmission reduction matrix and the next generation matrices passed to the function as an argument.

Value

list of contact matrices, one for each setting with reduction in transmission matching the next generation matrices

Examples

```
# examples take 20 second to run so skipping
## Not run:
# example data frame with vaccine coverage, acquisition and transmission
# efficacy of different age groups
vaccination_effect_example_data

# Generate next generation matrices

perth <- abs_age_lga("Perth (C)")
perth_hh <- get_abs_per_capita_household_size(lga = "Perth (C)")

age_breaks_0_80 <- c(seq(0, 80, by = 5), Inf)

# refit the model - note that the default if age_breaks isn't specified is
# 0 to 75
perth_contact_0_80 <- extrapolate_polymod(
  perth,
  per_capita_household_size = perth_hh,
  age_breaks = age_breaks_0_80
)
```

```
perth_ngm_0_80 <- generate_ngm(perth_contact_0_80,
  age_breaks = age_breaks_0_80,
  per_capita_household_size = perth_hh,
  R_target = 1.5
)

# In the old way we used to be able to pass age_breaks_0_80 along
generate_ngm_oz(
  lga_name = "Perth (C)",
  age_breaks = age_breaks_0_80,
  R_target = 1.5
)


# another way to do this using the previous method for generating NGMs
# The number of age breaks must match the vaccination effect data
ngm_nsw <- generate_ngm_oz(
  state_name = "NSW",
  age_breaks = c(seq(0, 80, by = 5), Inf),
  R_target = 1.5
)

# Apply vaccination effect to next generation matrices
ngm_nsw_vacc <- apply_vaccination(
  ngm = ngm_nsw,
  data = vaccination_effect_example_data,
  coverage_col = coverage,
  acquisition_col = acquisition,
  transmission_col = transmission
)

## End(Not run)
```

`as_conmat_population` *Convert to conmat population*

Description

Convert to conmat population

Usage

```
as_conmat_population(data, ...)

## Default S3 method:
as_conmat_population(data, ...)

## S3 method for class 'data.frame'
as_conmat_population(data, age, population, ...)
```

```
## S3 method for class 'list'
as_conmat_population(data, age, population, ...)

## S3 method for class 'grouped_df'
as_conmat_population(data, age, population, ...)
```

Arguments

data	data.frame
...	extra arguments
age	age column - an unquoted variable of numeric integer ages
population	population column - an unquoted variable, numeric value

Examples

```
some_age_pop <- data.frame(
  age = 1:10,
  pop = 101:110
)

some_age_pop

as_conmat_population(
  some_age_pop,
  age = age,
  population = pop
)
```

---

as_setting_prediction_matrix

Coerce object to a setting prediction matrix

Description

This will also calculate an `all` matrix, if `all` is not specified. This is the sum of all other matrices.

Usage

```
as_setting_prediction_matrix(list_matrix, age_breaks, ...)
```

Arguments

list_matrix	list of matrices
age_breaks	numeric vector of ages
...	extra arguments (currently not used)

Value

object of class setting prediction matrix

Examples

```
age_breaks_0_80_plus <- c(seq(0, 80, by = 10), Inf)
one_by_nine <- matrix(1, nrow = 9, ncol = 9)

mat_list <- list(
  home = one_by_nine,
  work = one_by_nine
)

mat_list

mat_set <- as_setting_prediction_matrix(
  mat_list,
  age_breaks = age_breaks_0_80_plus
)

mat_set
```

---

autoplot-conmat *Plot setting matrices using ggplot2*

Description

Plot setting matrices using ggplot2

Usage

```
## S3 method for class 'conmat_age_matrix'
autoplot(object, ..., title = "Contact Matrices")

## S3 method for class 'conmat_setting_prediction_matrix'
autoplot(object, ..., title = "Setting-specific synthetic contact matrices")

## S3 method for class 'transmission_probability_matrix'
autoplot(
  object,
  ...,
  title = "Setting-specific transmission probability matrices"
)

## S3 method for class 'ngm_setting_matrix'
autoplot(object, ..., title = "Setting-specific NGM matrices")
```

```
## S3 method for class 'setting_vaccination_matrix'
autoplot(object, ..., title = "Setting-specific vaccination matrices")
```

Arguments

object	A matrix or a list of square matrices, with row and column names indicating the age groups.
...	Other arguments passed on
title	Title to give to plot setting matrices. Defaults are provided for certain objects

Value

a ggplot visualisation of contact rates

Examples

```
## Not run:
if (interactive()) {
  polymod_contact_data <- get_polymod_setting_data()
  polymod_survey_data <- get_polymod_population()

  setting_models <- fit_setting_contacts(
    contact_data_list = polymod_contact_data,
    population = polymod_survey_data
  )

  fairfield <- abs_age_lga("Fairfield (C)")

  fairfield_hh_size <-
    get_abs_per_capita_household_size(lga = "Fairfield (C)")

  synthetic_settings_5y_fairfield_hh <- predict_setting_contacts(
    population = fairfield,
    contact_model = setting_models,
    age_breaks = c(seq(0, 85, by = 5), Inf),
    per_capita_household_size = fairfield_hh_size
  )

  # Plotting synthetic contact matrices across all settings

  autoplot(
    object = synthetic_settings_5y_fairfield_hh,
    title = "Setting specific synthetic contact matrices"
  )

  # Work setting specific synthetic contact matrices
  autoplot(
    object = synthetic_settings_5y_fairfield_hh$work,
    title = "Work"
  )
}
```

```
## End(Not run)
```

`conmat_original_school_demographics`
Original school demographics for conmat

Description

An internal dataset containing the original estimates of which fraction of ages were attending school in Australia. These can be used inside of `fit_single_contact_model()` and `fit_setting_contacts()`.

Usage

```
conmat_original_school_demographics
```

Format

A data frame with 121 rows and 2 variables:

age 0 to 120

school_fraction fraction of population at school

Source

Census of Population and Housing, 2016, TableBuilder

`conmat_original_work_demographics`
Original work demographics for conmat

Description

An internal dataset containing the original estimates of which fraction of ages were working in Australia. These can be used inside of `fit_single_contact_model()` and `fit_setting_contacts()`.

Usage

```
conmat_original_work_demographics
```

Format

A data frame with 121 rows and 2 variables:

age 0 to 120

work_fraction fraction of population working.

Source

Census of Population and Housing, 2016, TableBuilder

conmat_population *Define a conmat population*

Description

A conmat population is a dataframe that stores which columns represent the age and population information. This is useful as it means we can refer to this information throughout other functions in the conmat package without needing to specify or hard code which columns represent the age and population information.

Usage

```
conmat_population(data, age, population)
```

Arguments

data	data.frame
age	bare name representing the age column
population	bare name representing the population column

Value

a data frame with age and population attributes

Examples

```
perth <- abs_age_lga("Perth (C)")
```

data_abs_lga_education *LGA wise ABS education population data on different ages for year 2016*

Description

A dataset containing Australian Bureau of Statistics education data by lga for 2016. The data sourced from 2016 Census - Employment, Income and Education through TableBuilder have been randomly adjusted by the ABS to avoid the release of confidential data. As a result of this, there are some cases where the estimated number of people being educated is higher than the population of those people. Such cases have been flagged under the `anomaly_flag` variable.

Usage

```
data_abs_lga_education
```

Format

A data frame with 64,264 rows and 8 variables:

year 2016, data is based on 2016 Census of Population and Housing.

state String denoting abbreviated name of state or territory, for example, 'NSW', 'VIC', and 'QLD'.

lga String denoting the official name of Local Government Area. For example, 'Albury (C).'

age Ages from 0 to 115.

population_educated Number of people educated including students with full-time, part-time status, as well as the people who mentioned just the type of educational institution they attend and not their student status.

total_population Number depicting the total population belonging to the age.

proportion Number denoting the measure of the ratio of educated population and total population belonging to the age i.e, population_educated / total_population

anomaly_flag Logical variable flagging abnormal observations. E.g., total population lesser than population_educated as TRUE.

Source

Census of Population and Housing, 2016, TableBuilder

`data_abs_lga_work` *LGA wise ABS work population data on different ages for year 2016*

Description

A dataset containing Australian Bureau of Statistics labour force population data by lga for 2016. The data sourced from 2016 Census - Employment, Income and Education through TableBuilder have been randomly adjusted by the ABS to avoid the release of confidential data. As a result of this, there are some cases where the estimated number of people being employed is higher than the population of those people. Such cases have been flagged under the `anomaly_flag` variable.

Usage

```
data_abs_lga_work
```

Format

A data frame with 64,496 rows and 8 variables:

year 2016, as data is from 2016 Census of Population and Housing.

state String denoting the abbreviated name of state or territory name such as 'NSW', 'VIC', 'QLD' etc.

lga String denoting the official name of Local Government Area. For example, 'Albury (C).'

age Ages from 0 to 115.

employed_population Number of people employed including people with full-time, part-time employment status.

total_population Total population of age in row.

proportion The ratio of employed population and total population belonging to the age i.e, employed_population/ total_population.

anomaly_flag Logical variable flagging abnormal observations, such as total population lesser than employed_population as TRUE.

Source

Census of Population and Housing, 2016, TableBuilder

`data_abs_state_education`

State wise ABS education population data on different ages for year 2016

Description

A dataset containing Australian Bureau of Statistics education data by state for 2016. The data sourced from 2016 Census - Employment, Income and Education through TableBuilder have been randomly adjusted by the ABS to avoid the release of confidential data.

Usage

```
data_abs_state_education
```

Format

A data frame with 1044 rows and 6 variables:

year 2016, as data is from 2016 Census of Population and Housing.

state String of abbreviated name of state or territory names, e.g., 'NSW', 'VIC', 'QLD' and so on.

age Ages from 0 to 115.

population_educated Number of people educated, including students with full-time, part-time status, and people who mentioned only the type of educational institution they attend and not their student status.

total_population Total population belonging to age in a row.

proportion The ratio of educated population and total population belonging to the age i.e, population_educated / total_population

Source

Census of Population and Housing, 2016, TableBuilder

data_abs_state_work *State wise ABS work population data on different ages for year 2016*

Description

A dataset containing Australian Bureau of Statistics labour force population data by state for 2016. The data sourced from 2016 Census - Employment, Income and Education through TableBuilder have been randomly adjusted by the ABS to avoid the release of confidential data.

Usage

```
data_abs_state_work
```

Format

A data frame with 1044 rows and 6 variables:

year 2016, as data is from 2016 Census of Population and Housing.

state String. Abbreviated name of state or territory, e.g., 'NSW', 'VIC', 'QLD' and so on.

age Ages from 0 to 115.

employed_population Number of people employed including people with full-time, part-time employment status.

total_population Total population belonging to the age.

proportion The ratio of employed population and total population belonging to the age i.e, employed_population/ total_population

Source

Census of Population and Housing, 2016, TableBuilder

davies_age_extended *Susceptibility and clinical fraction parameters from Davies et al.*

Description

A dataset containing data from https://www.nature.com/articles/s41591-020-0962-9#code-availability When using this data, ensure that you cite the original authors at:

Usage

```
davies_age_extended
```

Format

A data frame of the probability of transmission from a case to a contact. There are 101 rows and 4 variables.

age from 0 to 100

clinical_fraction Estimate of fraction with clinical symptoms, or the age-specific proportion of infections resulting in clinical symptoms inferred by applying a smoothing spline to the mean estimates from Davies et al.

davies_original Age specific parameters of the relative susceptibility to infection inferred from a smoothing-spline estimate of the mean relative susceptibility estimate from Davies et al.

davies_updated Re-estimated parameter of the susceptibility profile for under-16s that is estimated in a similar way but to the age-distribution of infections in England from the UK ONS prevalence survey rather than case counts which may undercount children

Details

"Davies, N.G., Klepac, P., Liu, Y. et al. Age-dependent effects in the transmission and control of COVID-19 epidemics. Nat Med 26, 1205–1211 (2020). https://doi.org/10.1038/s41591-020-0962-9"

---

estimate_setting_contacts

Get predicted setting specific as well as combined contact matrices

---

Description

Given a named list of contact datasets (with names giving the setting, and assumed to together make up the full set of contacts for individuals in the survey), a representative population distribution for the survey, and a set of age breaks at which to aggregate contacts, return a set of predicted contact matrices for each setting, and for all combined. Note that this function is parallelisable with `future`, and will be impacted by any `future` plans provided.

Usage

```
estimate_setting_contacts(
  contact_data_list,
  survey_population,
  prediction_population = survey_population,
  age_breaks,
  per_capita_household_size = NULL,
  symmetrical = TRUE,
  school_demographics = NULL,
  work_demographics = NULL
)
```

Arguments

`contact_data_list`
: list of data sets with information on the contacts of individuals at different settings

`survey_population`
: representative population distribution for the survey

`prediction_population`
: population for prediction. The default value set is survey_population

`age_breaks`
: vector depicting age values. For example, `c(seq(0, 75, by = 5), Inf)`

`per_capita_household_size`
: Optional (defaults to NULL). When set, it adjusts the household contact matrix by some per capita household size. To set it, provide a single number, the per capita household size. More information is provided below in Details. See `get_abs_per_capita_household_size()` function for a helper for Australian data with a workflow on how to get this number.

`symmetrical`
: whether to enforce symmetrical terms in the model. Defaults to TRUE. See `details` of `fit_single_contact_model` for more information.

`school_demographics`
: (optional) defaults to census average proportion at school. You can provide a dataset with columns, "age" (numeric), and "school_fraction" (0-1), if you would like to specify these details. See `abs_avg_school` for the default values. If you would like to use the original school demographics used in conmat, these are provided in the dataset, `conmat_original_school_demographics`.

`work_demographics`
: (optional) defaults to census average proportion employed. You can provide a dataset with columns, "age" (numeric), and "work_fraction", if you would like to specify these details. See `abs_avg_work` for the default values. If you would like to use the original work demographics used in conmat, these are provided in the dataset, `conmat_original_work_demographics`.

Value

predicted setting specific contact matrices, and for all combined

Examples

```
## Not run:
# takes a long time to run
settings_estimated_contacts <- estimate_setting_contacts(
  contact_data_list = get_polymod_setting_data(),
  survey_population = get_polymod_population(),
  prediction_population = get_polymod_population(),
  age_breaks = c(seq(0, 85, by = 5), Inf),
  per_capita_household_size = NULL
)

# or predict to fairfield
fairfield_hh <- get_abs_per_capita_household_size(lga = "Fairfield (C)")
```

```
contact_model_pred_est <- estimate_setting_contacts(
  contact_data_list = get_polymod_setting_data(),
  survey_population = get_polymod_population(),
  prediction_population = abs_age_lga("Fairfield (C)"),
  age_breaks = c(seq(0, 85, by = 5), Inf),
  per_capita_household_size = fairfield_hh
)

# or use different populations in school or work demographics
fairfield_hh <- get_abs_per_capita_household_size(lga = "Fairfield (C)")
contact_model_pred_est <- estimate_setting_contacts(
  contact_data_list = get_polymod_setting_data(),
  survey_population = get_polymod_population(),
  prediction_population = abs_age_lga("Fairfield (C)"),
  age_breaks = c(seq(0, 85, by = 5), Inf),
  per_capita_household_size = fairfield_hh,
  school_demographics = conmat_original_school_demographics,
  work_demographics = conmat_original_work_demographics
)

# or use non-symmetric model terms
contact_model_pred_est <- estimate_setting_contacts(
  contact_data_list = get_polymod_setting_data(),
  survey_population = get_polymod_population(),
  prediction_population = abs_age_lga("Fairfield (C)"),
  age_breaks = c(seq(0, 85, by = 5), Inf),
  per_capita_household_size = fairfield_hh,
  symmetrical = FALSE
)

## End(Not run)
```

`extrapolate_polymod` *Fit all-of-polymod model and extrapolate to a given population an age breaks*

Description

Uses `estimate_setting_contacts()` to fit a contact model on the data from polymod and later extrapolate on to a desired population. Note that this function is parallelisable with `future`, and will be impacted by any `future` plans provided.

Usage

```
extrapolate_polymod(
  population,
  age_breaks = c(seq(0, 75, by = 5), Inf),
  per_capita_household_size = NULL
)
```

Arguments

population	a conmat_population object, specifying the age and population characteristics. Or a data frame with lower.age.limit and population columns. See get_polymod_population() for an example of this data.
age_breaks	vector depicting age values. Default value is c(seq(0, 75, by = 5), Inf)
per_capita_household_size	Optional (defaults to NULL). When set, it adjusts the household contact matrix by some per capita household size. To set it, provide a single number, the per capita household size. More information is provided below in Details. See get_abs_per_capita_household_size() function for a helper for Australian data with a workflow on how to get this number.

Details

Also note that since this model uses the already fit polymod_setting_models data, which has been fit using symmetrical model terms, if you want to fit a model with asymmetric model terms, you will need to go through the full process of building new models. You can find this detail in last section of the vignette "example pipeline".

Value

Returns setting-specific and combined contact matrices for the desired ages.

Examples

```
## Not run:
polymod_population <- get_polymod_population()
synthetic_settings_5y_polymod <- extrapolate_polymod(
  population = polymod_population
)
synthetic_settings_5y_polymod
synthetic_settings_5y_fairfield <- extrapolate_polymod(
  population = abs_age_lga("Fairfield (C)")
)
synthetic_settings_5y_fairfield

## End(Not run)
```

---

eyre_transmission_probabilities

Transmission probabilities of COVID19 from Eyre et al.

Description

A dataset containing data digitised from "The impact of SARS-CoV-2 vaccination on Alpha & Delta variant transmission", by David W Eyre, Donald Taylor, Mark Purver, David Chapman, Tom Fowler, Koen B Pouwels, A Sarah Walker, Tim EA Peto (doi:10.1101/2021.09.28.21264260. The figures were taken from https://www.medrxiv.org/content/10.1101/2021.09.28.21264260v1.full-text, and the code to digitise these figures is in `data-raw` under "read_eyre_transmission_probabilities.R". When using this data, ensure that you cite the original authors at 'Eyre, D. W., Taylor, D., Purver, M., Chapman, D., Fowler, T., Pouwels, K. B., Walker, A. S., & Peto, T. E. (2021). The impact of SARS-CoV-2 vaccination on Alpha & Delta variant transmission (Preprint). Infectious Diseases (except HIV/AIDS). https://doi.org/10.1101/2021.09.28.21264260'

Usage

```
eyre_transmission_probabilities
```

Format

A data frame of the probability of transmission from a case to a contact. There are 40,804 rows and 6 variables.

setting "household", "household_visitor", "work_education", or "events_activities"

case_age from 0 to 100

contact_age from ages 0 to 100

case_age_5y If case is between ages 0-4, in 5 year bins up to 100

contact_age_5y If contact is between ages 0-4, in 5 year bins up to 100

probability probability of transmission. Value is 0 - 1

Examples

```
## Not run:

# plot this
library(ggplot2)
library(stringr)
eyre_transmission_probabilities %>%
  group_by(
    setting,
    case_age_5y,
    contact_age_5y
  ) %>%
  summarise(
    across(
      probability,
      mean
    ),
    .groups = "drop"
  ) %>%
  rename(
    case_age = case_age_5y,
```

```
  contact_age = contact_age_5y
) %>%
mutate(
  across(
    ends_with("age"),
    ~ factor(.x,
      levels = str_sort(
        unique(.x),
        numeric = TRUE
      )
    )
  )
) %>%
ggplot(
  aes(
    x = case_age,
    y = contact_age,
    fill = probability
  )
) +
facet_wrap(~setting) +
geom_tile() +
scale_fill_viridis() +
coord_fixed() +
theme_minimal() +
theme(
  axis.text = element_text(angle = 45, hjust = 1)
)

## End(Not run)
```

`fit_setting_contacts` *Fit a contact model to a survey population*

Description

fits a gam model for each setting on the survey population data & the setting wise contact data. The underlying method is described in more detail in `fit_single_contact_model()`. The models can be fit in parallel, see the examples. Note that this function is parallelisable with `future`, and will be impacted by any `future` plans provided.

Usage

```
fit_setting_contacts(
  contact_data_list,
  population,
  symmetrical = TRUE,
  school_demographics = NULL,
  work_demographics = NULL
)
```

Arguments

contact_data_list	A list of dataframes, each containing information on the setting (home, work, school, other), age_from, age_to, the number of contacts, and the number of participants. Example data can be retrieved with get_polymod_setting_data().
population	conmat_population object or dataset with columns lower.age.limit and population. Example data can be retrieved with get_polymod_population().
symmetrical	whether to enforce symmetrical terms in the model. Defaults to TRUE. See details of fit_single_contact_model for more information.
school_demographics	(optional) defaults to census average proportion at school. You can provide a dataset with columns, "age" (numeric), and "school_fraction" (0-1), if you would like to specify these details. See abs_avg_school for the default values. If you would like to use the original school demographics used in conmat, these are provided in the dataset, conmat_original_school_demographics.
work_demographics	(optional) defaults to census average proportion employed. You can provide a dataset with columns, "age" (numeric), and "work_fraction", if you would like to specify these details. See abs_avg_work for the default values. If you would like to use the original work demographics used in conmat, these are provided in the dataset, conmat_original_work_demographics.

Value

list of fitted gam models - one for each setting provided

Author(s)

Nicholas Tierney

Examples

```
# These aren't being  run as they take too long to fit
## Not run:
contact_model <- fit_setting_contacts(
  contact_data_list = get_polymod_setting_data(),
  population = get_polymod_population()
)

# can fit the model in parallel
library(future)
plan(multisession, workers = 4)

polymod_setting_data <- get_polymod_setting_data()
polymod_population <- get_polymod_population()

contact_model <- fit_setting_contacts(
  contact_data_list = polymod_setting_data,
  population = polymod_population
```

```
)

# you can specify your own population data for school and work demographics
contact_model_diff_data <- fit_setting_contacts(
  contact_data_list = polymod_setting_data,
  population = polymod_population,
  school_demographics = conmat_original_school_demographics,
  work_demographics = conmat_original_work_demographics
)

## End(Not run)
```

`fit_single_contact_model` *Fit a single GAM contact model to a dataset*

Description

This is the workhorse of the `conmat` package, and is typically used inside `fit_setting_contacts()`. It predicts the contact rate between all age bands (the contact rate between ages 0 and 1, 0 and 2, 0 and 3, and so on), for a specified setting, with specific terms being added for given settings. See "details" for further information.

Usage

```
fit_single_contact_model(
  contact_data,
  population,
  symmetrical = TRUE,
  school_demographics = NULL,
  work_demographics = NULL
)
```

Arguments

`contact_data`	dataset with columns `age_to`, `age_from`, `setting`, `contacts`, and `participants`. See `get_polymod_contact_data()` for an example dataset - or the dataset in examples below.
`population`	`conmat_population` object, or data frame with columns `lower.age.limit` and `population`. See `get_polymod_population()` for an example.
`symmetrical`	whether to enforce symmetrical terms in the model. Defaults to TRUE. See `details` for more information.
`school_demographics`	(optional) defaults to census average proportion at school. You can provide a dataset with columns, "age" (numeric), and "school_fraction" (0-1), if you would like to specify these details. See `abs_avg_school` for the default values. If you would like to use the original school demographics used in conmat, these are provided in the dataset, `conmat_original_school_demographics`.

work_demographics
: (optional) defaults to census average proportion employed. You can provide a dataset with columns, "age" (numeric), and "work_fraction", if you would like to specify these details. See abs_avg_work for the default values. If you would like to use the original work demographics used in conmat, these are provided in the dataset, conmat_original_work_demographics.

Details

The model fit is a Generalised Additive Model (GAM). We provide two "modes" for model fitting. Either using "symmetric" or "non-symmetric" model predictor terms with the logical variance "symmetrical", which is set to TRUE by default. We recommend using the "symmetrical" terms as it reflects the fact that contacts are symmetric - person A having contact with person B means person B has had contact with person A. We've included a variety of terms to account for assortativity with age, where people of similar ages have more contact with each other. And included terms to account for intergenerational contact patterns, where parents and grandparents will interact with their children and grand children. These terms are fit with a smoothing function. Specifically, the relevant code looks like this:

```
# abs(age_from - age_to)
s(gam_age_offdiag) +
# abs(age_from - age_to)^2
s(gam_age_offdiag_2) +
# abs(age_from * age_to)
s(gam_age_diag_prod) +
# abs(age_from + age_to)
s(gam_age_diag_sum) +
# pmax(age_from, age_to)
s(gam_age_pmax) +
# pmin(age_from, age_to)
s(gam_age_pmin)
```

We also include predictors for the probability of attending school, and attending work. These are computed as the probability that a person goes to the same school/work, proportional to the increase in contacts due to attendance. These terms are calculated from estimated proportion of people in age groups attending school and work. See add_modelling_features() for more details.

Finally, we include two offset terms so that we estimate the contact rate, that is the contacts per capita, instead of the number of contacts. These offset terms are log(contactable_population), and log(contactable_population_school) when the model is fit to a school setting. The contactable population is estimated as the interpolated 1 year ages from the data. For schools this is the contactable population weighted by the proportion of the population attending school.

This leaves us with a model that looks like so:

```
mgcv::bam(
  formula = contacts ~
    # abs(age_from - age_to)
    s(gam_age_offdiag) +
    # abs(age_from - age_to)^2
```

```
    s(gam_age_offdiag_2) +
    # abs(age_from * age_to)
    s(gam_age_diag_prod) +
    # abs(age_from + age_to)
    s(gam_age_diag_sum) +
    # pmax(age_from, age_to)
    s(gam_age_pmax) +
    # pmin(age_from, age_to)
    s(gam_age_pmin) +
    school_probability +
    work_probability +
    offset(log_contactable_population) +
    # or for school settings
    # offset(log_contactable_population_school)
    family = stats::poisson,
  offset = log(participants),
  data = population_data
)
```

But if the term `symmetrical = FALSE` is used, you get:

```
mgcv::bam(
  formula = contacts ~
    s(age_to) +
    s(age_from) +
    s(abs(age_from - age_to)) +
    s(abs(age_from - age_to), age_from) +
    school_probability +
    work_probability +
    offset(log_contactable_population) +
    # or for school settings
    # offset(log_contactable_population_school)
    family = stats::poisson,
  offset = log(participants),
  data = population_data
)
```

Value

single model

Examples

```
example_contact <- get_polymod_contact_data(setting = "home")
example_contact
example_population <- get_polymod_population()

library(dplyr)
```

```
example_contact_20 <- example_contact %>%
  filter(
    age_to <= 20,
    age_from <= 20
  )

my_mod <- fit_single_contact_model(
  contact_data = example_contact_20,
  population = example_population
)

# you can specify your own population data for school and work demographics
my_mod_diff_data <- fit_single_contact_model(
  contact_data = example_contact_20,
  population = example_population,
  school_demographics = conmat_original_school_demographics,
  work_demographics = conmat_original_work_demographics
)
```

`generate_ngm` *Calculate next generation contact matrices*

Description

Once infected, a person can transmit an infectious disease to another, creating generations of infected individuals. We can define a matrix describing the number of newly infected individuals in given categories, such as age, for consecutive generations. This matrix is called a "next generation matrix" (NGM). We can generate an NGM from two sources - a `conmat_population` object (such as the output from `abs_age_lga()`), or a `conmat_setting_prediction_matrix`, which is the output from `extrapolate_polymod()` or `predict_setting_contacts()`.

Usage

```
generate_ngm(x, age_breaks, R_target, setting_transmission_matrix, ...)

## S3 method for class 'conmat_setting_prediction_matrix'
generate_ngm(
  x,
  age_breaks,
  R_target,
  setting_transmission_matrix = NULL,
  per_capita_household_size = NULL,
  ...,
  lga_name,
  state_name
)

## S3 method for class 'conmat_population'
```

```
generate_ngm(
  x,
  age_breaks,
  R_target,
  setting_transmission_matrix = NULL,
  per_capita_household_size = NULL,
  ...,
  lga_name,
  state_name
)
```

Arguments

x	data input - could be a conmat_population (such as the output from abs_age_lga()), or a conmat_setting_prediction_matrix, which is the output from extrapolate_polymod() or predict_setting_contacts().
age_breaks	vector depicting age values with the highest age depicted as Inf. For example, c(seq(0, 85, by = 5), Inf)
R_target	target reproduction number
setting_transmission_matrix	default is NULL, which calculates the transmission matrix using get_setting_transmission_matrices You can provide your own transmission matrix, but its rows and columns must match the number of rows and columns, and must be a list of one matrix for each setting. See the output for get_setting_transmission_matrices(age_breaks) to get a sense of the structure. See get_setting_transmission_matrices() for more detail.
...	extra arguments, currently not used
per_capita_household_size	default is NULL - which defaults to get_polymod_per_capita_household_size(), which gives 3.248971
lga_name	now defunct, but capturing arguments for informative error
state_name	now defunct, but capturing arguments for informative error

Details

The NGM can be used to calculate the expected number of secondary infections in a given age group. Given certain age breaks, we compute the unscaled next generation matrices for that location across different settings & age groups using the contact rates extrapolated from POLYMOD survey data on the specified location, adjusted by the per capita household size and the setting-specific relative per-contact transmission probability matrices for the same age groups. These NGMs are then scaled according to a target reproduction number (which is provided as an argument) using the ratio of the desired R0 and the R0 of the NGM for the combination of all settings. The R0 of the combination of all settings is obtained by calculating the unique, positive eigen value of the combination NGM. This ratio is then used to scale all the setting specific NGMs.

Note

When using a setting prediction contact matrix (such as one generated by `extrapolate_polymod`, with class `conmat_setting_prediction_matrix`), the age breaks specified in `generate_ngm` must be the same as the age breaks specified in the synthetic contact matrix, otherwise it will error as it is trying to multiple incompatible matrices.

Examples

```
## Not run:
perth <- abs_age_lga("Perth (C)")
perth_hh <- get_abs_per_capita_household_size(lga = "Perth (C)")

age_breaks_0_80_plus <- c(seq(0, 80, by = 5), Inf)

# you can also run this without `per_capita_household_size`
perth_ngm_lga <- generate_ngm(
  perth,
  age_breaks = age_breaks_0_80_plus,
  per_capita_household_size = perth_hh,
  R_target = 1.5
)

perth_contact <- extrapolate_polymod(
  perth,
  per_capita_household_size = perth_hh
)

perth_ngm <- generate_ngm(
  perth_contact,
  age_breaks = age_breaks_0_80_plus,
  R_target = 1.5
)

# using our own transmission matrix
new_transmission_matrix <- get_setting_transmission_matrices(
  age_breaks = age_breaks_0_80_plus,
  # is normally 0.5
  asymptomatic_relative_infectiousness = 0.75
)

new_transmission_matrix

perth_ngm_0_80_new_tmat <- generate_ngm(
  perth_contact,
  age_breaks = age_breaks_0_80_plus,
  R_target = 1.5,
  setting_transmission_matrix = new_transmission_matrix
)

## End(Not run)
# examples not run as they take a long time
## Not run:
```

```
perth <- abs_age_lga("Perth (C)")
perth_contact <- extrapolate_polymod(perth)
generate_ngm(perth_contact, age_breaks = c(seq(0, 85, by = 5), Inf))

## End(Not run)
```

---

generate_ngm_oz *Calculate next generation contact matrices from ABS data*

Description

This function calculates a next generation matrix (NGM) based on state or LGA data from the Australian Bureau of Statistics (ABS). For full details see generate_ngm().

Usage

```
generate_ngm_oz(
  state_name = NULL,
  lga_name = NULL,
  age_breaks,
  R_target,
  setting_transmission_matrix = NULL
)
```

Arguments

state_name	target Australian state name in abbreviated form, such as "QLD", "NSW", or "TAS"
lga_name	target Australian local government area (LGA) name, such as "Fairfield (C)". See abs_lga_lookup() for list of lga names.
age_breaks	vector depicting age values with the highest age depicted as Inf. For example, c(seq(0, 85, by = 5), Inf)
R_target	target reproduction number
setting_transmission_matrix	default is NULL, which calculates the transmission matrix using get_setting_transmission_matrices You can provide your own transmission matrix, but its rows and columns must match the number of rows and columns, and must be a list of one matrix for each setting. See the output for get_setting_transmission_matrices(age_breaks) to get a sense of the structure. See get_setting_transmission_matrices() for more detail.

Examples

```
# don't run as both together takes a long time to run
## Not run:
ngm_nsw <- generate_ngm_oz(
  state_name = "NSW",
```

```
  age_breaks = c(seq(0, 85, by = 5), Inf),
  R_target = 1.5
)
ngm_fairfield <- generate_ngm_oz(
  lga_name = "Fairfield (C)",
  age_breaks = c(seq(0, 85, by = 5), Inf),
  R_target = 1.5
)

## End(Not run)
```

`get_abs_household_size_distribution` *Get household size distribution based on state or LGA name*

Description

Get household size distribution based on state or LGA name

Usage

```
get_abs_household_size_distribution(state = NULL, lga = NULL)
```

Arguments

`state`	target Australian state name in abbreviated form, such as "QLD", "NSW", or "TAS"
`lga`	target Australian local government area (LGA) name, such as "Fairfield (C)". See `abs_lga_lookup()` for list of lga names

Value

returns a data frame with household size distributions of a specific state or LGA

Examples

```
get_abs_household_size_distribution(lga = "Fairfield (C)")
get_abs_household_size_distribution(state = "NSW")
## Not run:
# cannot specify both state and LGA
get_abs_household_size_distribution(state = "NSW", lga = "Fairfield (C)")

## End(Not run)
```

get_abs_household_size_population

Get population associated with each household size in an LGA or a state

Description

Get population associated with each household size in an LGA or a state

Usage

```
get_abs_household_size_population(state = NULL, lga = NULL)
```

Arguments

state	target Australian state name in abbreviated form, such as "QLD", "NSW", or "TAS"
lga	target Australian local government area (LGA) name, such as "Fairfield (C)". See abs_lga_lookup() for list of lga names

Value

returns a data frame with household size and the population associated with it in each LGA or state.

Examples

```
get_abs_household_size_population(state = "NSW")
```

get_abs_per_capita_household_size

Get per capita household size based on state or LGA name

Description

Get per capita household size based on state or LGA name

Usage

```
get_abs_per_capita_household_size(state = NULL, lga = NULL)
```

Arguments

state	state name
lga	lga name

Value

Numeric of length 1 - the per capita household size for a given state or LGA.

Author(s)

Nick Golding

Examples

```
get_abs_per_capita_household_size(lga = "Fairfield (C)")
get_abs_per_capita_household_size(state = "NSW")
## Not run:
# cannot specify both state and LGA
get_abs_per_capita_household_size(state = "NSW", lga = "Fairfield (C)")

## End(Not run)
```

---

get_abs_per_capita_household_size_lga

Get household size distribution based on LGA name

Description

Get household size distribution based on LGA name

Usage

```
get_abs_per_capita_household_size_lga(lga = NULL)
```

Arguments

lga	target Australian local government area (LGA) name, such as "Fairfield (C)". See abs_lga_lookup() for list of lga names

Value

returns a numeric value depicting the per capita household size of the specified LGA

Examples

```
get_abs_per_capita_household_size_lga(lga = "Fairfield (C)")
```

`get_abs_per_capita_household_size_state` *Get household size distribution based on state name*

Description

Get household size distribution based on state name

Usage

```
get_abs_per_capita_household_size_state(state = NULL)
```

Arguments

`state`	target Australian state name in abbreviated form, such as "QLD", "NSW", or "TAS"

Value

returns a numeric value depicting the per capita household size of the specified state

Examples

```
get_abs_per_capita_household_size_state(state = "NSW")
```

`get_age_population_function` *Return an interpolating function for populations in 1y age increments*

Description

This function returns an interpolating function to get populations in 1y age increments from chunkier distributions produced by `socialmixr::wpp_age()`.

Usage

```
get_age_population_function(data, ...)

## S3 method for class 'conmat_population'
get_age_population_function(data = population, ...)

## S3 method for class 'data.frame'
get_age_population_function(
  data = population,
  age_col = lower.age.limit,
  pop_col = population,
  ...
)
```

Arguments

data	dataset containing information on population of a given age/age group
...	extra arguments
age_col	bare variable name for the column with age information
pop_col	bare variable name for the column with population information

Details

The function first prepares the data to fit a smoothing spline to the data for ages below the maximum age. It arranges the data by the lower limit of the age group to obtain the bin width/differences of the lower age limits. The mid point of the bin width is later added to the ages and the population is scaled as per the bin widths. The maximum age is later obtained and the populations for different above and below are filtered out along with the sum of populations with and without maximum age. A cubic smoothing spline is then fitted to the data for ages below the maximum with predictor variable as the ages with the mid point of the bins added to it where as the response variable is the log-scaled population. Using the smoothing spline fit, the predicted population of ages 0 to 200 is obtained and the predicted population is adjusted further using a ratio of the sum of the population across all ages from the data and predicted population. The ratio is based on whether the ages are under the maximum age as the total population across all ages differs for ages above and below the maximum age. The maximum age population is adjusted further to drop off smoothly, based on the weights. The final population is then linearly extrapolated over years past the upper bound from the data. For ages above the maximum age from data, the population is calculated as a weighted population of the maximum age that depends on the years past the upper bound. Older ages would have lower weights, therefore lower population.

Value

An interpolating function to get populations in 1y age increments

Examples

```
polymod_pop <- get_polymod_population()

polymod_pop

# But these ages and populations are binned every 5 years. So we can now
# provide a specified age and get the estimated population for that 1 year
# age group. First we create the new function like so

age_pop_function <- get_age_population_function(
  data = polymod_pop
)
# Then we pass it a year to get the estimated population for a particular age
age_pop_function(4)

# Or a vector of years, to get the estimated population for a particular age
# range
age_pop_function(1:4)
```

```
# Notice that we get a _pretty similar_ number of 0-4 if we sum it up, as
# the first row of the table:
head(polymod_pop, 1)
sum(age_pop_function(age = 0:4))

# Usage in dplyr
library(dplyr)
example_df <- slice_head(abs_education_state, n = 5)
example_df %>%
  mutate(population_est = age_pop_function(age))
```

`get_polymod_contact_data` *Format POLYMOD data and filter contacts to certain settings*

Description

Provides contact and participant POLYMOD data from selected countries. It impute missing contact ages via one of three methods:

1. imputing contact ages from a random uniform distribution from the range of ages. 2) using the average of the ages, 3) removal of those participants. The contact settings are then classified as "home", "school", "work" and "others", where "others" include locations such as leisure, transport or other places. The participants with missing contact ages or settings are removed, and the number of contacts per participant and contact age from ages 0-100 are obtained for various countries and settings.

Usage

```
get_polymod_contact_data(
  setting = c("all", "home", "work", "school", "other"),
 countries = c("Belgium", "Finland", "Germany", "Italy", "Luxembourg", "Netherlands",
    "Poland", "United Kingdom"),
  ages = 0:100,
  contact_age_imputation = c("sample", "mean", "remove_participant")
)
```

Arguments

`setting`	Which setting to extract data from. Default is all settings. Options are: "all", "home", "work", "school", and "other".
`countries`	countries to extract data from. Default is all countries from this list: "Belgium", "Finland", "Germany", "Italy", "Luxembourg", "Netherlands", "Poland", and "United Kingdom".
`ages`	Which ages to return. Default is ages 0 to 100.

contact_age_imputation
: How to handle age when it is missing. Choose one of three methods: 1) "sample", which imputes contact ages from a random uniform distribution from the range of ages. 2) "mean", use the average of the ages, 3) "remove_participant" removal of those participants. Default is "sample".

Value

A data.frame with columns: "setting" (all, work, home, etc. as specified in "setting" argument); "age_from" - the age of the participant; "age_to" - the age of the person the participant had contact with; "contacts" the number of contacts that person had; "participants" the number of participants in that row.

Examples

```
get_polymod_contact_data()
get_polymod_contact_data(setting = "home")
get_polymod_contact_data(countries = "Belgium")
get_polymod_contact_data(countries = c("Belgium", "Italy"))
get_polymod_contact_data(ages = 0:50)
get_polymod_contact_data(contact_age_imputation = "sample")
get_polymod_contact_data(contact_age_imputation = "mean")
get_polymod_contact_data(contact_age_imputation = "remove_participant")
```

---

get_polymod_per_capita_household_size

Get polymod per capita household size.

Description

Convenience function to help get the per capita household size. This is calculated as `mean(socialmixr::polymod$particip`

Usage

```
get_polymod_per_capita_household_size()
```

Value

number, 3.248971

Author(s)

Nicholas Tierney

get_polymod_population

Return the polymod-average population age distribution in 5y

Description

returns the polymod-average population age distribution in 5y increments (weight country population distributions by number of participants). Note that we don't want to weight by survey age distributions for this, since the total number of *participants* represents the sampling. It uses the participant data from the polymod survey as well as the age specific population data from `socialmixr` R package to return the age specific average population of different, countries weighted by the number of participants from those countries who participated in the polymod survey.

Usage

```
get_polymod_population(
 countries = c("Belgium", "Finland", "Germany", "Italy", "Luxembourg", "Netherlands",
    "Poland", "United Kingdom")
)
```

Arguments

`countries`	countries to extract data from. Default is to get: Belgium, Finland, Germany, Italy, Luxembourg, Netherlands, Poland, and United Kingdom.

Value

A `conmat_population` data frame with two columns: `lower.age.limit` and `population`

Examples

```
get_polymod_population()
get_polymod_population("Belgium")
get_polymod_population("United Kingdom")
get_polymod_population("Italy")
```

get_polymod_setting_data

Get polymod setting data

Description

`get_polymod_setting_data()` acts as an extension of `get_polymod_contact_data()`, and extracts the setting wise contact data on the desired country, as a list.

Usage

```
get_polymod_setting_data(
 countries = c("Belgium", "Finland", "Germany", "Italy", "Luxembourg", "Netherlands",
    "Poland", "United Kingdom")
)
```

Arguments

countries	countries to extract data from

Value

A list of data frames, of the polymod data. One list per setting: "home", "work", "school", and "other".

Examples

```
get_polymod_setting_data()
get_polymod_setting_data("Belgium")
```

---

get_setting_transmission_matrices
Get Setting Transmission Matrices

Description

Given some age breaks, return a named list of matrices containing age-specific relative per-contact transmission probability matrices for each of 4 settings: home, school, work, other. These can be combined with contact matrices to produce setting-specific relative next generation matrices (NGMs). These can be scaled to match a required reproduction number based on the dominant eigenvalue of the all-settings NGM (the elementwise sum of all setting NGMs).

Usage

```
get_setting_transmission_matrices(
  age_breaks = c(seq(0, 80, by = 5), Inf),
  asymptomatic_relative_infectiousness = 0.5,
  susceptibility_estimate = c("davies_updated", "davies_original")
)
```

Arguments

age_breaks	vector of age breaks, defaults to `c(seq(0, 80, by = 5), Inf)`
asymptomatic_relative_infectiousness	the assumed ratio of onward infectiousness between asymptomatic and symptomatic cases. This represents the infectiousness of asymptomatic relative to symptomatic. Default value is 0.5, which means the asymptomatic cases are 50% less infectious than symptomatic cases.

susceptibility_estimate
: Which estimate to use for susceptibility by age. Either, the smoothed original Davies et al estimates, "davies_original" or, the set updated to match UK under-16 infections (the default), "davies_updated".

Details

These matrices are created from: an estimate of the clinical fraction for each age (inferred by applying a smoothing spline to the mean estimates from Davies et al.); an assumption of the infectiousness of asymptomatics relative to symptomatics (provided as an argument); estimates of the relative susceptibility to infection of individuals of different ages, inferred from a smoothing-spline estimate of the mean relative susceptibility estimate from Davies et al., combined with a re-estimation of the susceptibility profile for under-16s, estimated in a similar way but to the age-distribution of infections in England from the UK ONS prevalence survey (rather than case counts with may undercount children), assuming the above clinical fraction estimates, and accounting for vaccination, reduced mixing, and reduced transmissibility in work and other settings due to hygiene behaviour; and estimates of the relative transmissibility in household vs non-household settings - scaled linearly for non-household transmission and binomially for household transmission (so that onward infections do not to exceed the number of other household members).

When using this data, ensure that you cite this package, and the original authors of the paper from which these estimates were derived:

Davies, N.G., Klepac, P., Liu, Y. et al. Age-dependent effects in the transmission and control of COVID-19 epidemics. Nat Med 26, 1205–1211 (2020). https://doi.org/10.1038/s41591-020-0962-9

Value

list of matrices, containing the relative per-contact transmission probability for each setting

Examples

```
## Not run:
# fit polymod model
setting_models <- fit_setting_contacts(
  contact_data_list = get_polymod_setting_data(),
  population = get_polymod_population()
)

# define age breaks for prediction
age_breaks <- c(seq(0, 80, by = 5), Inf)

# define a new population age distribution to predict to
fairfield <- abs_age_lga("Fairfield (C)")

# predict setting-specific contact matrices to a new population
contact_matrices <- predict_setting_contacts(
  population = fairfield,
  contact_model = setting_models,
  age_breaks = age_breaks
)
```

```
# remove the 'all' matrix, keep the other four settings
contact_matrices <- contact_matrices[c("home", "school", "work", "other")]

# get setting-specific per-contact transmission rate matrices for the same
# age aggregations
transmission_matrices <- get_setting_transmission_matrices(
  age_breaks = age_breaks
)

# combine them to get setting-specific (unscaled) next-generation matrices
next_generation_matrices <- mapply(
  FUN = `*`,
  contact_matrices,
  transmission_matrices,
  SIMPLIFY = FALSE
)

# get the all-settings NGM
ngm_overall <- Reduce("+", next_generation_matrices)

## End(Not run)
```

`matrix_to_predictions` *Convert a contact matrix as output into a long-form tibble*

Description

This function is the opposite of `predictions_to_matrix()`. It converts a wide matrix into a long data frame. It is mostly used within plotting functions.

Usage

```
matrix_to_predictions(contact_matrix)
```

Arguments

`contact_matrix`	square matrix with age group to and from information in the row and column names.

Value

data.frame with columns `age_group_to`, `age_group_from`, and `contacts`.

Examples

```
fairfield <- abs_age_lga("Fairfield (C)")

# We can convert the predictions into a matrix
```

```
fairfield_school_contacts <- predict_contacts(
  model = polymod_setting_models$school,
  population = fairfield,
  age_breaks = c(0, 5, 10, 15, Inf)
)

fairfield_school_contacts

fairfield_school_mat <- predictions_to_matrix(fairfield_school_contacts)

fairfield_school_mat

matrix_to_predictions(fairfield_school_mat)
```

`new_age_matrix` *Build new age matrix*

Description

A matrix that knows about its age breaks - which are by default provided as its rownames. Mostly intended for internal use.

Usage

```
new_age_matrix(matrix, age_breaks)
```

Arguments

`matrix`	numeric matrix
`age_breaks`	character vector of age breaks, by default the rownames.

Value

matrix with age breaks attribute

Examples

```
age_break_names <- c("[0,5)", "[5,10)", "[10, 15)")
age_mat <- matrix(
  runif(9),
  nrow = 3,
  ncol = 3,
  dimnames = list(
    age_break_names,
    age_break_names
  )
)

new_age_matrix(
```

```
  age_mat,
  age_breaks = age_break_names
)
```

`new_ngm_setting_matrix` *Establish new BGM setting data*

Description

Establish new BGM setting data

Usage

```
new_ngm_setting_matrix(list_matrix, raw_eigenvalue, scaling, age_breaks)
```

Arguments

`list_matrix`	list of matrices
`raw_eigenvalue`	the raw eigenvalue
`scaling`	scaling factor
`age_breaks`	vector of age breaks

Value

object with additional (primary) class "ngm_setting_matrix", and attributes for "age_breaks", "scaling", and "raw_eigenvalue".

`new_setting_data` *Establish new setting data*

Description

Establish new setting data

Usage

```
new_setting_data(list_df)
```

Arguments

`list_df`	list of data frames

Value

object with additional (primary) class "setting data" and an "age_breaks attribute.

`per_capita_household_size` *Get per capita household size with household size distribution*

Description

Returns the per capita household size for a location given its household size distribution. See `get_abs_household_size_distribution()` function for retrieving household size distributions for a given place.

Usage

```
per_capita_household_size(
  household_data,
  household_size_col = household_size,
  n_people_col = n_people
)
```

Arguments

`household_data`	data set with information on the household size distribution of specific state or LGA.
`household_size_col`	bare variable name of the column depicting the household size. Default is 'household_size' from `get_abs_per_capita_household_size_lga()`.
`n_people_col`	bare variable name of the column depicting the total number of people belonging to the respective household size. Default is 'n_people' from `get_abs_per_capita_household_size_lga`

Value

Numeric of length 1 - the per capita household size for a given state or LGA.

Author(s)

Nick Golding

Examples

```
demo_data <- get_abs_household_size_population(lga = "Fairfield (C)")
demo_data
per_capita_household_size(
  household_data = demo_data,
  household_size_col = household_size,
  n_people_col = n_people
)
```

`polymod` *Social contact data from 8 European countries (imported from* `socialmixr`*)*

Description

A dataset containing social mixing diary data from 8 European countries: Belgium, Germany, Finland, Great Britain, Italy, Luxembourg, The Netherlands and Poland.

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

Details

This data has been sourced from the socialmixr package.

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.

Source

doi:10.1371/journal.pmed.0050074

polymod_setting_models *Polymod Settings models*

Description

A data object containing a list of fitted gam models predicting the number of contacts in each of the four settings which are "home","work","school" and "other". For more details on model fitting, see `fit_setting_contacts()`. This object has been provided as data to avoid recomputing a relatively common type of model for use with `conmat`.

Usage

```
polymod_setting_models
```

Format

An object of class `list` of length 4.

See Also

`fit_setting_contacts()`

Examples

```
## Not run:
# code used to produce this data
library(conmat)
set.seed(2022 - 08 - 26)
polymod_contact_data <- get_polymod_setting_data()
polymod_survey_data <- get_polymod_population()
polymod_setting_models <- fit_setting_contacts(
  contact_data_list = polymod_contact_data,
  # population = polymod_survey_data
)

## End(Not run)
```

predictions_to_matrix *Convert dataframe of predicted contacts into matrix*

Description

Helper function to convert predictions of contact rates in data frames to matrix format with the survey participant age groups as columns and contact age groups as rows.

Usage

```
predictions_to_matrix(contact_predictions, ...)
```

Arguments

contact_predictions	data frame with columns age_group_from, age_group_to, and contacts.
...	extra arguments

Value

Square matrix with the unique age groups from age_group_from/to in the rows and columns and contacts as the values.

Examples

```
fairfield <- abs_age_lga("Fairfield (C)")

# We can convert the predictions into a matrix

fairfield_school_contacts <- predict_contacts(
  model = polymod_setting_models$school,
  population = fairfield,
  age_breaks = c(0, 5, 10, 15, Inf)
)

fairfield_school_contacts

# convert them back to a matrix
predictions_to_matrix(fairfield_school_contacts)
```

---

predict_contacts *Predict contact rate between two age populations, given some model.*

Description

Predicts the expected contact rate over specified age breaks, given some model of contact rate and population age structure. This function is used internally in predict_setting_contacts(), which performs this prediction across all settings (home, work, school, other), and optionally performs an adjustment for per capita household size. You can use predict_contacts() by itself, just be aware you will need to separately apply a per capita household size adjustment if required. See details below on adjust_household_contact_matrix for more information.

Usage

```
predict_contacts(model, population, age_breaks = c(seq(0, 75, by = 5), Inf))
```

Arguments

model	A single fitted model of contact rate (e.g., fit_single_contact_model())
population	a dataframe of age population information, with columns indicating some lower age limit, and population, (e.g., get_polymod_population())
age_breaks	the ages to predict to. By default, the age breaks are 0-75 in 5 year groups.

Details

The population data is used to determine age range to predict contact rates, and removes ages with zero population, so we do not make predictions for ages with zero populations. Contact rates are predicted yearly between the age groups, using predict_contacts_1y(), then aggregates these predicted contacts using aggregate_predicted_contacts(), which aggregates the predictions back to the same resolution as the data, appropriately weighting the contact rate by the population.

Regarding the adjust_household_contact_matrix function, we use Per-capita household size instead of mean household size. Per-capita household size is different to mean household size, as the household size averaged over people in the **population**, not over households, so larger households get upweighted. It is calculated by taking a distribution of the number of households of each size in a population, multiplying the size by the household by the household count to get the number of people with that size of household, and computing the population-weighted average of household sizes. We use per-capita household size as it is a more accurate reflection of the average number of household members a person in the population can have contact with.

Value

A dataframe with three columns: age_group_from, age_group_to, and contacts. The age groups are factors, broken up into 5 year bins [0,5), [5,10). The contact column is the predicted number of contacts from the specified age group to the other one.

Examples

```
# If we have a model of contact rate at home, and age population structure
# for an LGA, say, Fairfield, in NSW:

polymod_setting_models$home

fairfield <- abs_age_lga("Fairfield (C)")

fairfield

# We can predict the contact rate for Fairfield from the existing contact
# data, say, between the age groups of 0-15 in 5 year bins for school:

fairfield_school_contacts <- predict_contacts(
  model = polymod_setting_models$school,
  population = fairfield,
  age_breaks = c(0, 5, 10, 15, Inf)
)

fairfield_school_contacts
```

`predict_contacts_1y` *Predict contact rate to a given population at full 1y resolution*

Description

Provides a predicted rate of contacts for contact ages. Take an already fitted model of contact rate and predict the estimated contact rate, and standard error, for all combinations of the provided ages in 1 year increments. So if the minimum age is 5, and the maximum age is 10, it will provide the estimated contact rate for all age combinations: 5 and 5, 5 and 6 ... 5 and 10, and so on. This function is used internally within `predict_contacts()`, and thus `predict_setting_contacts()` as well, although it can be used by itself. See examples for more details, and details for more information.

Usage

```
predict_contacts_1y(model, population, age_min = 0, age_max = 100)
```

Arguments

`model`	A single fitted model of contact rate (e.g., `fit_single_contact_model()`)
`population`	a dataframe of age population information, with columns indicating some lower age limit, and population, (e.g., `get_polymod_population()`)
`age_min`	Age range minimum value. Default: 0
`age_max`	Age range maximum value, Default: 100

Details

Prediction features are added using `add_modelling_features()`. These features include the population distribution of contact ages, fraction of population in each age group that attend school/work as well as the offset according to the settings on all combinations of the participant & contact ages.

Value

Data frame with four columns: `age_from`, `age_to`, `contacts`, and `se_contacts`. This contains the participant & contact ages from the minimum and maximum ages provided along with the predicted rate of contacts and standard error around the prediction.

Examples

```
fairfield <- abs_age_lga("Fairfield (C)")

fairfield

# predict the contact rates in 1 year blocks to Fairfield data

fairfield_contacts_1 <- predict_contacts_1y(
  model = polymod_setting_models$home,
```

```
  population = fairfield,
  age_min = 0,
  age_max = 2
)
```

`predict_setting_contacts` *Predict setting contacts*

Description

Predict contact rate for each setting. Note that this function is parallelisable with `future`, and will be impacted by any `future` plans provided.

Usage

```
predict_setting_contacts(
  population,
  contact_model,
  age_breaks,
  per_capita_household_size = NULL,
  model_per_capita_household_size = get_polymod_per_capita_household_size()
)
```

Arguments

`population`	population
`contact_model`	contact_model
`age_breaks`	age_breaks
`per_capita_household_size`	Optional (defaults to NULL). When set, it adjusts the household contact matrix by some per capita household size. To set it, provide a single number, the per capita household size. More information is provided below in Details. See `get_abs_per_capita_household_size()` function for a helper for Australian data with a workflow on how to get this number.
`model_per_capita_household_size`	modelled per capita household size. Default values for this are from `get_polymod_per_capita_household` which ends up being 3.248971

Details

We use Per-capita household size instead of mean household size. Per-capita household size is different to mean household size, as the household size averaged over **people** in the population, not over households, so larger households get upweighted. It is calculated by taking a distribution of the number of households of each size in a population, multiplying the size by the household by the household count to get the number of people with that size of household, and computing the

population-weighted average of household sizes. We use per-capita household size as it is a more accurate reflection of the average number of household members a person in the population can have contact with.

Value

List of setting matrices

Author(s)

Nicholas Tierney

Examples

```
# don't run as it takes too long to fit
## Not run:
fairfield <- abs_age_lga("Fairfield (C)")
fairfield

age_break_0_85_plus <- c(seq(0, 85, by = 5), Inf)

polymod_contact_data <- get_polymod_setting_data()
polymod_survey_data <- get_polymod_population()

setting_models <- fit_setting_contacts(
  contact_data_list = polymod_contact_data,
  population = polymod_survey_data
)

synthetic_settings_5y_fairfield <- predict_setting_contacts(
  population = fairfield,
  contact_model = setting_models,
  age_breaks = age_break_0_85_plus
)

fairfield_hh_size <- get_abs_per_capita_household_size(lga = "Fairfield (C)")
fairfield_hh_size

synthetic_settings_5y_fairfield_hh <- predict_setting_contacts(
  population = fairfield,
  contact_model = setting_models,
  age_breaks = age_break_0_85_plus,
  per_capita_household_size = fairfield_hh_size
)

## End(Not run)
```

prem_germany_contact_matrices *Contact matrices as calculated by Prem. et al.*

Description

Contact matrices as calculated by Prem. et al. PLoS Computational Biology. DOI: 10.1371/journal.pcbi.1005697

Usage

```
prem_germany_contact_matrices
```

Format

A list with 5 elements:

home A 16x16 matrix containing the number of home contacts, by 5 year age group

work A 16x16 matrix containing the number of workplace contacts, by 5 year age group

school A 16x16 matrix containing the number of school contacts, by 5 year age group

other A 16x16 matrix containing the number of other contacts, by 5 year age group

All age groups are 5 year age bands, from 0 to 80.

Source

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005697

raw_eigenvalue *Get raw eigvenvalue from NGM matrix*

Description

Get raw eigvenvalue from NGM matrix

Usage

```
raw_eigenvalue(list_matrix)
```

Arguments

list_matrix	object of class ngm_setting_matrix

Value

raw eigenvalue

Examples

```
# examples not run as they take a long time
## Not run:
perth <- abs_age_lga("Perth (C)")
perth_contact <- extrapolate_polymod(perth)
perth_ngm <- generate_ngm(
  perth_contact,
  age_breaks = c(seq(0, 85, by = 5), Inf)
)
raw_eigenvalue(perth_ngm)

## End(Not run)
```

`scaling` *Get the scaling from NGM matrix*

Description

This value is `scaling <- R_target / R_raw`, where `R_target` is the target R value provided to the NGM, and `R_raw` is the raw eigenvalue.

Usage

```
scaling(list_matrix)
```

Arguments

`list_matrix`	object of class `ngm_setting_matrix`

Value

scaling

Examples

```
# examples not run as they take a long time
## Not run:
perth <- abs_age_lga("Perth (C)")
perth_contact <- extrapolate_polymod(perth)
perth_ngm <- generate_ngm(
  perth_contact,
  age_breaks = c(seq(0, 85, by = 5), Inf)
)
raw_eigenvalue(perth_ngm)
scaling(perth_ngm)

## End(Not run)
```

`setting_prediction_matrix` *Create a setting prediction matrix*

Description

Helper function to create your own setting prediction matrix, which you may want to use in `generate_ngm`, or `autoplot`. This class is the output of functions like `extrapolate_polymod`, and `predict_setting_contacts`. We recommend using this function is only for advanced users, who are creating their own setting prediction matrix.

Usage

```
setting_prediction_matrix(..., age_breaks)
```

Arguments

`...`	list of matrices
`age_breaks`	age breaks - numeric

Value

setting prediction matrix

Examples

```
age_breaks_0_80_plus <- c(seq(0, 80, by = 10), Inf)
one_by_nine <- matrix(1, nrow = 9, ncol = 9)

x_example <- setting_prediction_matrix(
  home = one_by_nine,
  work = one_by_nine,
  age_breaks = age_breaks_0_80_plus
)

x_example <- setting_prediction_matrix(
  one_by_nine,
  one_by_nine,
  age_breaks = age_breaks_0_80_plus
)

x_example
```

setting_weights *Setting weights computed for transmission probabilities.*

Description

see ?get_setting_transmission_matrices for details of how to use these

Usage

```
setting_weights
```

Format

A named vector of weights relative to home, for home, work, school, and other

transmission_probability_matrix *Create a setting transmission matrix*

Description

Helper function to create your own setting transmission matrix, which you may want to use in ... or `autoplot`. This class is the output of functions like `...`, and ... . We recommend using this function is only for advanced users, who are creating their own transmission probability matrix.

Usage

```
transmission_probability_matrix(..., age_breaks)
```

Arguments

`...`	list of matrices
`age_breaks`	age breaks - numeric

Value

transmission probability matrix

Examples

```
age_breaks_0_80_plus <- c(seq(0, 80, by = 10), Inf)
one_05 <- matrix(0.05, nrow = 9, ncol = 9)

x_example <- transmission_probability_matrix(
  home = one_05,
  work = one_05,
  age_breaks = age_breaks_0_80_plus
)

x_example <- transmission_probability_matrix(
  one_05,
  one_05,
  age_breaks = age_breaks_0_80_plus
)

x_example
```

---

vaccination_effect_example_data

Example dataset with information on age based vaccination coverage, acquisition and transmission

---

Description

data frame with information on vaccine coverage, efficacy of acquisition/susceptibility and efficacy of transmission/infectiousness for the ordered age groups from lowest to highest of the next generation matrix.

Usage

```
vaccination_effect_example_data
```

Format

A data frame with 17 rows and 4 variables

age_band character. age bands: 0-4,5-11, 12-15, 16-19, 20-24, etc

coverage example vaccination coverage, between 0-1

acquisition example acquisition coverage, between 0-1

transmission example transmission coverage, between 0-1

Index