Package: popim (via r-universe)

August 18, 2024

Title POPulation IMmunity: Run a Demographic Model of Vaccine Exposure Over Time

Version 0.0.1

Description Tools for setting up an age-structured population, applying vaccination activities to it, and tracking vaccine-induced immunity through time.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, ggplot2, rlang, tidyr, tidyselect

VignetteBuilder knitr

Repository https://mrc-ide.r-universe.dev

RemoteUrl https://github.com/mrc-ide/popim

RemoteRef main

RemoteSha 86c6cdc06112d88fcadbd2a78e98dce48b9a1fb3

Contents

`apply_vacc` *Apply vaccination activities to a population object*

Description

`apply_vacc()` applies the vaccination activities listed in the input object `vacc` (a data.frame object of class `popim_vacc_activities`) to the `popim_population` object pop, then returns the updated population object pop.

Usage

```
apply_vacc(pop, vacc)
```

Arguments

`pop`	An object of class `popim_population` such as created by `popim_population()`.
`vacc`	An object of class `popim_vacc_activities` such as created by `popim_vacc_activities()`.

Details

Once objects holding data on the population and vaccination activities have been set up, the primary functionality provided by `popim` is to apply the vaccination activities to the population to evaluate the population immunity by age over time that results from the given vaccination activities, which is done with the function `apply_vacc()`.

Assumptions when applying vaccination activities to the population:

- 100% vaccine effectiveness
- no wastage of doses
- no waning immunity
- mortality is independent of immunity status

While the first two are clearly unrealistic assumptions, if there is a constant vaccine effectiveness (<100%), or a constant wastage factor, the results can easily be scaled up by these constant factors to obtain more realistic values for vaccine demand.

For a potentially deadly disease one would expect that mortality due to the disease would be strongly correlated with immunity status, however, even the most devastating outbreaks typically only affect a small part of the population, and therefore all-cause mortality will be much less impacted by mortality due to the specific disease. This means that this assumption is closer to reality than it might appear at first.

Waning immunity is a potentially important factor to consider for many vaccines, particularly when looking at the long term benefits of vaccination. However, allowing for this is currently not implemented.

Description of the algorithm:

The vaccination activities object has columns for `coverage`, the proportion of the target population that will be vaccinated, and `doses`, the number of vaccine doses to be administered in the activity. Given a target population size this is redundant (and potentially conflicting) information. The function `apply_vacc()` will always use `coverage` information in preference over `doses`, if this is non-missing. If `coverage` is missing, the target population size (based on the population size of the region and age groups targeted) will be calculated and the corresponding `coverage` calculated as `doses` / `target_population`.

While the function `apply_vacc()` will calculate the coverage if it is missing with respect to the target population size, without modifying the input `popim_vacc_activities` object, the function `complete_vacc_activities()` will fill in missing coverage and doses information from the other, and check for inconsistencies in activities that have both data on `coverage` and `doses`. This is done with respect to a particular `popim_population` object which contains the relevant population sizes.

Vaccinating a cohort with no previous immunity:

When vaccination is applied to a particular birth cohort that has no immunity, the result is simply that the proportion of the cohort receiving the vaccine will be immune from the time of the vaccination onwards, i.e., the immunity for this cohort will be set to equal the coverage of the vaccination activity in question. Due to the annual time step, vaccination is assumed to take effect at the end of the year in which it is implemented, so the immunity is updated from the next year onwards.

Vaccinating a cohort with previous immunity:

When a vaccination activity is applied to a cohort that has some previous immunity, we need to make some assumptions about who will be vaccinated in the new activity. There are 3 different options implemented which are governed by the parameter `targeting`, one of the input parameters to the function `apply_vacc()`. This parameter can take the values `"targeted"`, `"random"` or `"correlated"`.

`"targeted"` *targeting:*

The most effective way to implement a new vaccination activity is to use `"targeted"` targeting, meaning that the vaccine is targeted at previously non-immunised individuals. This means that the new immunity of after the vaccination activity is simply the sum of the previous immunity and the coverage of the current vaccination activity (capped by 1 which indicates complete immunity of the cohort in question):

immunity_new = min(immunity_prev + coverage, 1)

While this is not a realistic assumption in many settings, it can be useful for instance in multi-year campaigns that target the same region (but possibly proceed sub-region by sub-region) over several years. This multi-year campaign would be coded as separate vaccination activities for each year, using the option `targeting` = `"targeted"`.

`"random"` *targeting:*

A reasonable first assumption might be to use `targeting` = `"random"`, meaning that individuals will receive vaccination irrespective of their previous immunity status. The immunity after the new campaign is applied is calculated according to

immunity_new = coverage + immunity_prev - coverage * immunity_prev

`"correlated"` *targeting:*

On the opposite end of the spectrum is the most pessimistic assumption that might apply in situations where there is unequal access to vaccination: under `"correlated"` targeting, people who have previously been immunised get vaccinated first in the new vaccination activity, and those previously not immune will only receive any vaccine if the new coverage extends further:

immunity_new = max(immunity_prev, coverage)
Note that the order in which vaccination activities are applied to a population does not matter under any of the `targeting` assumptions.

Value

The input `popim_population` class object pop with updated column `immunity` to reflect the vaccination activities supplied in `vacc`.

Author(s)

Tini Garske

Examples

```
## set up population and vaccination activities:
pop <- popim_population(region = "UK", year_min = 2000, year_max = 2005,
                        age_min = 0, age_max = 10)
vacc <- popim_vacc_activities(region = "UK", year = c(2001, 2002),
                              age_first = 0, age_last = 0,
                              coverage = 0.8, doses = NA,
                              targeting = "random")

## update the population immunity based on the vaccination activities:
pop <- apply_vacc(pop, vacc)
```

`as_popim_pop` *Generate a* `popim_population` *object from a dataframe*

Description

Checks if the dataframe is suitable (i.e., contains appropriate columns and data ranges), and if so converts it to a `popim_population` object and returns this.

Usage

```
as_popim_pop(df)
```

Arguments

`df`	a dataframe with at least columns region, age, year and pop_size.

Details

The input dataframe has to have at least the columns `region`, `age`, and `year`. The output popim_population object is generated via expand.grid to have consecutive year and age ranges that are identical for all regions.

If the input dataframe contains a column `pop_size`, this must be numeric and non-negative. If it is missing, this column is generated and initialised to NA.

If the input dataframe contains a colum `immunity`, this must be numeric, with values between 0 and 1. If it is missing, this column is generated and initialised to 0.

Any further colunms are simply carried over into the popim_population object.

Value

an object of class `popim_population`

Author(s)

Tini Garske

Examples

```
## set up a minimal dataframe to convert to a popim_population object:
df <- expand.grid(region = "UK", age = 0:3, year = 2000:2004, stringsAsFactors = FALSE)
pop <- as_popim_pop(df)
```

`as_vacc_activities` *Generate a* `popim_vacc_activites` *object from a dataframe*

Description

Checks if the dataframe is suitable (i.e., contains appropriate columns and data ranges), and if so converts it to a `popim_vacc_activities` object and returns this.

Usage

```
as_vacc_activities(df)
```

Arguments

`df`	dataframe to be converted to class `popim_vacc_activities`

Details

The input dataframe has to have at least the columns `region`, `year`, `age_first`, `age_last`, `coverage`, `doses` and `targeting`. Any further columns are simply carried over into the `popim_vacc_activities` object.

Value

an object of class `popim_vacc_activities`

Author(s)

Tini Garske

See Also

`popim_vacc_activities()` for details of the S3 class.

Examples

```
df <- data.frame(region = "UK", year = c(2000, 2002),
                 age_first = 0, age_last = 0,
                 coverage = 0.8, doses = NA,
                 targeting = "random")
vacc <- as_vacc_activities(df)
```

`calc_pop_immunity` *Aggregate the population immunity over age*

Description

Calculate the overall population immunity (aggregating over age) from the supplied `popim_population` object.

Usage

```
calc_pop_immunity(pop)
```

Arguments

`pop`	A `popim_population` object for which the population size and immunity will be aggregated over age.

Value

A dataframe containing the `popim_population` aggregated by age.

Author(s)

Tini Garske

Examples

```
## set up population and vaccination activities:
pop <- popim_population(region = "UK", year_min = 2000, year_max = 2005,
                        age_min = 0, age_max = 10)
vacc <- popim_vacc_activities(region = "UK", year = c(2001, 2002),
                              age_first = 0, age_last = 0,
                              coverage = 0.8, doses = NA,
                              targeting = "random")

## update the population immunity based on the vaccination activities:
pop <- apply_vacc(pop, vacc)
```

```
## calculate the population immunity aggregated across ages:
pop_aggregated <- calc_pop_immunity(pop)
```

complete_vacc_activities *Add coverage or doses information to the input* `popim_vacc_activities` *object*

Description

For each line in the `popim_vacc_activities` object the given information of coverage is converted to doses, or vice versa, using the target population size implied by the `popim_population` object supplied. If both coverage and doses are given for any activity, the function checks if they are consistent with the population size, and fails if there are any inconsistencies.

Usage

```
complete_vacc_activities(vacc, pop)
```

Arguments

`vacc`	`popim_vacc_activities` object
`pop`	`popim_population` object

Value

The supplied object of class `popim_vacc_activities`, updated to have both `doses` and `coverage` information.

Author(s)

Tini Garske

Examples

```
## set up some vaccination activities:
vacc <- popim_vacc_activities(region = "UK", year = 2001:2003,
                              age_first = 0, age_last = 0,
                              coverage = c(0.8, 0.8, NA),
                              doses = c(NA, NA, 60),
                              targeting = "random")
## set up a population to which these activities shall apply:
pop <- popim_population(region = "UK", year_min = 2000, year_max = 2005,
                        age_min = 0, age_max = 10)
pop$pop_size <- 100 ## cohort size of 100 for all cohorts

## fill in missing coverage/doses information based on population size:
vacc <- complete_vacc_activities(vacc, pop)
```

`plotting` *Plot the immunity or population size of a* `popim_population` *object*

Description

Plot the immunity or population size of a `popim_population` object

Usage

```
plot_immunity(pop, cols = c("whitesmoke", "midnightblue"))

plot_pop_size(pop, rel = FALSE, cols = c("whitesmoke", "midnightblue"))
```

Arguments

`pop`	`popim_population` object such as created by `popim_population()`.
`cols`	vector of 2 colours to be used to generate the (continuous) colour palette for plotting. Defaults to `c("whitesmoke", "midnightblue")`.
`rel`	logical to indicate whether to use relative or absolute population size in `plot_pop_size()`. Defaults to FALSE (plotting absolute population size).

Details

The population is displayed in a grid showing the cohorts through time. Time is shown on the x-axis, age on the y-axis, such that a particular cohort tracks along diagonally from bottom left to top right. If there are several regions, these are shown as separate facets.

The colour in each cell corresponds to:

- for `plot_immunity()`: the proportion of each cohort that is immune, therefore varying between 0 and 1.
- for `plot_pop_size()`: the size of each cohort.

As the returned object is a regular `ggplot` object, it can be further modified with the ususal `ggplot2` syntax.

Value

A ggplot object.

Author(s)

Tini Garske

Examples

```
## set up population and vaccination activities:
pop <- popim_population(region = "UK", year_min = 2000, year_max = 2005,
                        age_min = 0, age_max = 10)
vacc <- popim_vacc_activities(region = "UK", year = c(2001, 2002),
                              age_first = 0, age_last = 0,
                              coverage = 0.8, doses = NA,
                              targeting = "random")

## update the population immunity based on the vaccination activities:
pop <- apply_vacc(pop, vacc)

## plot the population immunity by age and time:
plot_immunity(pop)

## adding some population size manually:
##  adding some population size manually:
pop$pop_size <- pop$cohort - 1990

## plot the population size by age and time:
plot_pop_size(pop)

##----------------------------------------------------------------------
## setting up a population with multiple regions:
pop <- popim_population(region = c("A", "B"),
                        year_min = 2000, year_max = 2005,
                        age_min = 0, age_max = 10)
pop$pop_size <- pop$cohort - 1990
pop$pop_size[pop$region == "A"] <- 5 * pop$pop_size[pop$region == "A"]

## adding some vaccination activities:
vacc <- popim_vacc_activities(region = c("A", "A", "B"),
                              year = c(2001, 2002, 2003),
                              age_first = c(0,0,0), age_last = c(0,0,10),
                              coverage = 0.8, doses = NA,
                              targeting = "random")
pop <- apply_vacc(pop, vacc)

plot_immunity(pop)

plot_pop_size(pop)
plot_pop_size(pop, rel = TRUE)
```

`popim_population` *Constructor of an object of the* `popim_population` *class*

Description

The `popim_population` object is a dataframe that models an age-structured population through time, tracking population size and vaccine-induced immunity in the population. The population may be spatially disaggregated into several regions.

Usage

```
popim_population(
  region = character(),
  year_min = integer(),
  year_max = integer(),
  age_min = 0,
  age_max = 100
)
```

Arguments

`region`	character vector, list of regions considered.
`year_min`	integer, first year to be considered.
`year_max`	integer, last year to be considered.
`age_min`	integer, youngest age to be considered, defaults to 0. Must be non-negative.
`age_max`	integer, oldest age to be considered, defaults to 100. Must be non-negative, and >= age_min.

Details

An object of S3 class `popim_population` is a dataframe that contains the columns

- `region`: character.
- `year`: integer.
- `age`: integer, non-negative.
- `cohort`: integer. This is redundant, equals `year` - `age`, and only included for ease of handling.
- `immunity`: numeric, between 0 and 1 (inclusive). Proportion of the cohort that is immune due to vaccination. Initialised to 0 by the constructor `popim_population()`.
- `pop_size`: numeric, non-negative. Size of the cohort. Initialised to NA_real by the constructor `popim_population()`.

This constructor sets up the population as fully susceptible (i.e., `immunity` = `0`), with missing population size (i.e., `pop_size` = `NA_real_`) throughout. The parameters passed to the constructor are retained as attributes to the dataframe object.

A population with non-missing population size can be read in from a suitable file using `read_popim_pop()`, while vaccine induced immunity can be generated through applying vaccination activities to the population with `apply_vacc()`.

Value

An object of class `popim_population`. This is a dataframe with columns `region`, `year`, `age`, `cohort`, `immunity` and `pop_size`. The first three cover the ranges given by the input parameters. `cohort` gives the year of birth. It is reduntant as it is calculated as `cohort` = `year` - `age`. It is included for ease of handling. Immunity and pop_size are initialised as 0 and NA, respectively, throughout the whole population.

Author(s)

Tini Garske

Examples

```
pop <- popim_population(region = "UK", year_min = 2000, year_max = 2010)

pop <- popim_population(region = c("FRA", "UK"),
                        year_min = 2000, year_max = 2010,
                        age_min = 0, age_max = 80)
```

---

popim_vacc_activities *Constructor of an object of the* popim_vacc_activites *class*

Description

The popim_vacc_activities object is a dataframe that holds information on vaccination activities that are typically meant to be applied to a popim_population object.

Usage

```
popim_vacc_activities(
  region = character(),
  year = integer(),
  age_first = integer(),
  age_last = integer(),
  coverage = double(),
  doses = double(),
  targeting = character()
)
```

Arguments

region	character vector: specifies the geographic region to which each vaccination activity is to be administered.
year	integer vector: specifies the year in which each vaccination activity takes place.
age_first, age_last	non-negative integer vectors: specify the age range targeted in the vaccination activity. age_first <= age_last for all entries.
coverage	numeric vector, 0 <= coverage <= 1 for all entries. Specifies the proportion of the target population to be immunised.
doses	numeric vector, non-negative. Specifies the number of doses to be used in each vaccination activity.

`targeting`	character vector, permissible entries are "random", "correlated", "targeted". Defines how vaccine is allocated if there is pre-existing immunity in the population: For "random" targeting individuals are vaccinated irrespective of immunity status, so if prior to the vaccination activity the proportion immune was x, then a proportion x of the vaccine will be administered to already vaccinated individuals and therefore be wasted. For "correlated" targeting vaccine is administered first to those already immune before any susceptible individuals receive vaccine. This option models the case of unequal access to vaccination. For "targeted" targeting, vaccine will be given first to as yet non-immune individuals. This is the most effective use of vaccine. It may be realistic in the case of multi-year campaigns targeting different areas within the geographical region specified.

Details

The input parameters are the columns of the `popim_vacc_activities` object to be returned, they should be vectors of the same length, or will be recycled as appropriate by `data.frame()`.

Value

S3 object of class `popim_vacc_activities`: a dataframe with columns `region`, `year`, `age_first`, `age_last`, `coverage`, `doses`, `targeting`.

Author(s)

Tini Garske

Examples

```
# setting up an empty class `popim_vacc_activities` object of the
# correct structure:
vacc <- popim_vacc_activities()

# setting up a two specific vaccination activities
vacc <- popim_vacc_activities(region = c("UK", "FRA"),
                              year = c(2010, 2005),
                              age_first = c(0, 0),
                              age_last = c(0, 10),
                              coverage = c(0.8, 0.5),
                              doses = NA,
                              targeting = "random")
```

`read_popim_pop` *Read popim_population data from a .csv file*

Description

Reads a population data from a .csv file, checks if the data fulfils the requirements for a `popim_population` object, and if so returns this object.

Usage

```
read_popim_pop(file)
```

Arguments

`file`	Name of the .csv file from which the population data are to be read. If it does not contain an absolute path, the file name is relative to the current working directory.

Details

The requirements are that the data contain the columns `region`, `age`, `year` which will be coerced to character, integer, integer. Columns `pop_size` and `immunity` are optional; they will be coerced to numeric, and if missing will be initialised to 0 and NA, respectively be set to NA and 0, respectively. Any additional columns will be retained in the `popim_population` object.

Limitations for values:

- `age` must be non-negative integer
- `immunity` must be between 0 and 1 (inclusive)
- `pop_size` must be non-negative

Value

An object of class `popim_population`, a dataframe with one row per birth cohort/year/region, with columns `region`, `year`, `age`, `cohort`, `immunity`, `pop_size`.

Author(s)

Tini Garske

See Also

`popim_population()` for details of the S3 class, and `utils::read.csv()` which handles the reading of the .csv file.

Examples

```
filename <- system.file("extdata", "pop_sample.csv", package = "popim")
pop <- read_popim_pop(file = filename)
```

`read_vacc_activities` *Read vaccination activities from a .csv file*

Description

Reads a list of vaccination activities from a .csv file, checks if the data fulfils the requirements for a `popim_vacc_activities` object, and if so returns this object.

Usage

```
read_vacc_activities(file)
```

Arguments

`file`	Name of the file from which the vaccination activities are to be read from. If it does not contain an absolute path, the file name is relative to the current working directory.

Details

The requirements are that the data contain the columns `region`, `year`, `age_first`, `age_last`, `coverage` and `targeting`, of types character, integer, integer, integer, double, character, respectively.

0 <= age_first <= age_last 0 <= coverage <= 1 targeting must be one of "random", "correlated", "targeted".

Column `region` details the (geographical) region to which each vaccination activity is to be administered.

Column `year` details the year in which each vaccination activity occurs.

Columns `age_first` and `age_last` give the age range targeted in each vaccination activity.

Column `coverage` gives the proportion of the target population that will be vaccinated, while column `doses` gives the absolute number of doses used in the vaccination activity. For a given population size (which is not recorded in the `popim_vacc_activities` object generated here), these can be converted into each other, when both are given, they may be inconsistent with each other once applied to a specific `popim_population` object. The consistency between these two colums cannot be confirmed in without reference to a popim_population object, but this function requires that at least one of these is non-missing in each row.

The column `targeting` defines how vaccine is allocated if there is pre-existing immunity in the population: For "random" targeting individuals are vaccinated irrespective of immunity status, so if prior to the vaccination activity the proportion immune was x, then a proportion x of the vaccine will be administered to already vaccinated individuals and therefore be wasted.

For "correlated" targeting vaccine is administered first to those already immune before any susceptible individuals receive vaccine. This option models the case of unequal access to vaccination.

For "targeted" targeting, vaccine will be given first to as yet non-immune individuals. This is the most effective use of vaccine. It may be realistic in the case of multi-year campaigns targeting different areas within the geographical region specified.

Value

object of class popim_vacc_activities, a dataframe with one row per vaccination activity, with columns year, age_first, age_last, coverage, targeting.

Author(s)

Tini Garske

See Also

popim_vacc_activities() for details of the S3 class, and utils::read.csv() which handles the reading of the .csv file.

Examples

```
filename <- system.file("extdata", "vacc_activities.csv", package = "popim")
vacc <- read_vacc_activities(filename)
```

---

vacc_from_immunity   *Infer vaccination activities from population*

Description

Given a popim_population object and an assumption of how vaccine is targeted in a partially immune population, this function infers the vaccination activities that have given rise to the specified population immunity. When there is ambiguity (e.g., due to subsequent campaigns achieving full coverage), the minimum coverage/doses needed will be returned.

Usage

```
vacc_from_immunity(pop, targeting = "random", n_digits = 10)
```

Arguments

pop	popim_population object for which vaccination activities are to be inferred.
targeting	character, determines the assumption of how doses are allocated. Valid options are "random" (the default), "correlated", "targeted", see apply_vacc() for details of these options.
n_digits	number of digits to which the coverage is to be rounded, defaults to 10.

Details

Default is the targeting option "random", which assumes that individuals receive vaccine independently of vaccination status, resulting in some double vaccination if there is pre-existing immunity in the population. The option "targeted" gives the most effective vaccine distribution, vaccinating unvaccinated people first, while the option "correlated" models a situation of inequalities in access to vaccination - in this extreme cases, all previously vaccinated individuals will be vaccinated first, before any remaining doses are given to unvaccinated individuals.

Note that this function will not return any potential vaccination activities that don't change the immunity, for instance if `targeting` is set to "correlated" and the coverage is too small to increase population immunity, or if the immunity prior to the activity was already at 1.

The vaccination of the oldest age group in the population will also never be picked up as this age group will have aged out of the population (i.e., died) before the immunity is updated in the next year.

Value

popim_vacc_activites object

Author(s)

Tini Garske

See Also

`apply_vacc()`

Examples

```
## set up population and vaccination activities:
pop <- popim_population(region = "UK", year_min = 2000, year_max = 2005,
                        age_min = 0, age_max = 10)
pop$pop_size <- 100
vacc <- popim_vacc_activities(region = "UK", year = c(2001, 2002),
                              age_first = 0, age_last = 0,
                              coverage = 0.8, doses = NA,
                              targeting = "random")

## update the population immunity based on the vaccination activities:
pop <- apply_vacc(pop, vacc)

## get the vaccination activities that have created the population
## immunity - this should match the input `vacc`
vacc_out <- vacc_from_immunity(pop, targeting = "random")
```

Index