Package: multigroup.vaccine (via r-universe)

May 15, 2026

Type Package

Title Analyze Outbreak Models of Multi-Group Populations with Vaccination

Version 0.2.0

Description Model infectious disease dynamics in populations with multiple subgroups having different vaccination rates, transmission characteristics, and contact patterns. Calculate final and intermediate outbreak sizes, form age-structured contact models with automatic fetching of U.S. census data, and explore vaccination scenarios with an interactive 'shiny' dashboard for a model with two subgroups, as described in Nguyen et al. (2024) <doi:10.1016/j.jval.2024.03.039> and Duong et al. (2026) <doi:10.1093/ofid/ofaf695.217>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LinkingTo Rcpp

Imports deSolve, graphics, shiny, stats, parallel, bslib (>= 0.9.0), htmltools, socialmixr, Rcpp

Suggests knitr, rmarkdown, testthat (>= 3.0.0),

Config/testthat/edition 3

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

URL https://epiforesite.github.io/multigroup-vaccine/

Depends R (>= 2.10)

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-05-15 21:48:19 UTC

RemoteUrl https://github.com/EpiForeSITE/multigroup-vaccine

RemoteRef HEAD

RemoteSha f2d81b13bd9e51088962498528ab981916522e6c

Contents

`aggregateByAgeGroups` *Aggregate population counts into age groups*

Description

Aggregate per-age population counts into coarser age groups defined by the (sorted) lower bounds in `age_groups`. When `length(age_groups) == 1`, all ages `>=` that value are aggregated into a single open-ended group ("Xplus"). When `length(age_groups) > 1`, groups are formed as `age_groups[i]` to `age_groups[i+1] - 1` for i = 1:(n-1) and `age_groups[n]` and above for the final group. Human-readable labels are produced: "under1" for the 0–0 group, "ageX" for single-year groups, "XtoY" for ranges, and "Xplus" for the final open group. When `verbose = TRUE`, the function prints aggregation summaries to the console for each group using `cat()`.

Usage

```
aggregateByAgeGroups(ages, pops, age_groups, verbose = FALSE)
```

Arguments

`ages`	Numeric vector of ages (typically integers) corresponding to the entries in `pops`.
`pops`	Numeric vector of population counts for each age; must be the same length as `ages`. Note: NA values in `pops` will propagate into group sums because `na.rm = TRUE` is not used; clean or impute missing values beforehand if required.

age_groups	Numeric vector of lower bounds for desired age groups. Must be sorted in ascending order. If length is 1, the single value defines an "Xplus" group (ages >= X). For length > 1, contiguous non-overlapping groups are created as described above.
verbose	Logical, if TRUE prints aggregation messages for each group. Default is FALSE.

Details

- Group boundaries are inclusive at both ends for finite ranges (i.e. ages satisfying lower <= age <= upper). For the last group the upper bound is infinite.
- If no ages fall into a group the aggregated count for that group is 0 (because `sum(numeric(0)) == 0`).
- When `verbose = TRUE`, the function writes progress messages to the console with `cat()` for each aggregated group (useful for debugging / logging). By default (`verbose = FALSE`), the function is silent.

Value

A named list with components:

pops Numeric vector of aggregated population counts, one element per group.

labels Character vector of labels for each group (e.g. "under1", "age5", "0to4", "65plus").

age_ranges List of numeric vectors of length 2 giving the inclusive lower and upper bounds for each group; the upper bound for the final group is `Inf`.

Examples

```
# Multiple groups example
ages <- 0:100
pops <- rep(100, length(ages))
aggregateByAgeGroups(ages, pops, c(0, 5, 18, 65))

# Single open-ended group (65plus)
aggregateByAgeGroups(ages, pops, 65)
```

---

contactMatrixAgeSchool

Calculate a contact matrix for age groups and schools

Description

Calculate a contact matrix for age groups and schools

Usage

```
contactMatrixAgeSchool(
  agelims,
  agepops,
  schoolagegroups,
  schoolpops,
  schportion
)
```

Arguments

agelims	minimum age in years for each age group
agepops	population size of each age group
schoolagegroups	index of the age group covered by each school
schoolpops	population size of each school
schportion	portion of within-age-group contacts that are exclusively within school

Value

a square matrix with the contact rate of each group (row) with members of each other group (column)

Examples

```
contactMatrixAgeSchool(agelims = c(0, 5, 18), agepops = c(500, 1300, 8200),
schoolagegroups = c(2, 2), schoolpops = c(600, 700), schportion = 0.7)
```

---

contactMatrixPolymod *Calculate a contact matrix for age groups based on Polymod contact survey data*

Description

Calculate a contact matrix for age groups based on Polymod contact survey data

Usage

```
contactMatrixPolymod(agelims, agepops = NULL)
```

Arguments

agelims	minimum age in years for each age group. The maximum valid age limit is 90, as the socialmixr contact_matrix function supports ages up to 90. Age limits greater than 90 will be replaced with 90 and a warning will be issued.
agepops	population size of each group, defaulting to demography of Polymod survey population. If provided, must match the length of the age groups defined by agelims (after any adjustments for exceeding the 90-year limit).

Details

The socialmixr contact_matrix function supports age limits up to 90. Any age limits above 90 will be adjusted to 90 with a warning, and the corresponding populations will be aggregated into a single "90+" group.

Value

A symmetric contact matrix with row and column names indicating the age groups.

Examples

```
#Default population distribution uses population data from POLYMOD survey locations:
contactMatrixPolymod(agelims = c(0, 5, 18))
#Specifying the age distribution will lead to an adjusted version:
contactMatrixPolymod(agelims = c(0, 5, 18), agepops = c(500, 1300, 8200))
```

---

contactMatrixPropPref *Calculate group contact matrix with proportional mixing and preferential mixing within group*

Description

Calculate group contact matrix with proportional mixing and preferential mixing within group

Usage

```
contactMatrixPropPref(popsize, contactrate, ingroup)
```

Arguments

popsize	population size of each group
contactrate	overall contact rate of each group
ingroup	fraction of each group's contacts that are exclusively in-group

Value

a square matrix with the contact rate of each group (row) with members of each other group (column)

Examples

```
contactMatrixPropPref(popsize = c(100, 150, 200), contactrate = c(1.1, 1, 0.9),
ingroup = c(0.2, 0.25, 0.22))
```

`finalsize` *Calculate final outbreak size or distribution of a multigroup transmission model for a given basic reproduction number, contact/transmission assumptions, and initial conditions*

Description

Calculate final outbreak size or distribution of a multigroup transmission model for a given basic reproduction number, contact/transmission assumptions, and initial conditions

Usage

```
finalsize(
  popsize,
  R0,
  contactmatrix,
  relsusc,
  reltransm,
  initR,
  initI,
  initV,
  method = "ODE",
  nsims = 1,
  nthreads = 1,
  cluster = NULL,
  seed = NULL
)
```

Arguments

`popsize`	the population size of each group
`R0`	the basic reproduction number
`contactmatrix`	matrix of group-to-group contact rates
`relsusc`	relative susceptibility to infection per contact of each group
`reltransm`	relative transmissibility per contact of each group
`initR`	initial number of each group already infected and removed (included in size result)
`initI`	initial number of each group infectious
`initV`	initial number of each group vaccinated
`method`	the method of final size calculation or simulation to use
`nsims`	the number of simulations to run for stochastic methods
`nthreads`	the number of threads (parallel workers) to use for stochastic simulations. Defaults to 1 (single-threaded). Set to 0 or -1 to automatically use all available cores. Ignored for `method = "hybrid"`.

cluster	an optional parallel cluster object to use for stochastic simulations. When supplied, nthreads is ignored, allowing callers to provide PSOCK, FORK, MPI, or scheduler-managed clusters. Ignored for method = "hybrid".
seed	optional integer base random seed for stochastic methods. When supplied, finalsize() is reproducible for that seed regardless of nthreads and independent of caller RNG state.

Value

a vector (nsims = 1) or matrix (nsims > 1) with the final number infected from each group (column) in each simulation (row)

Examples

```
popsize <- c(800, 200)
R0 <- 2
contactmatrix <- contactMatrixPropPref(popsize = popsize, contactrate = c(1, 1),
ingroup = c(0.2, 0.2))
relsusc <- c(1, 1)
reltransm <- c(1, 1)
initR <- c(0, 0)
initI <- c(1, 0)
initV <- 0.2 * popsize
# Default method "ODE" numerical solves ordinary differential equations until infectious count
# is close to 0
finalsize(popsize, R0, contactmatrix, relsusc, reltransm, initR, initI, initV)
finalsize(popsize, R0, contactmatrix, relsusc, reltransm, initR, initI, initV,
method = "analytic")
finalsize(popsize, R0, contactmatrix, relsusc, reltransm, initR, initI, initV,
method = "stochastic", nsims = 10)
# All "escaped" outbreaks set to deterministic final size:
finalsize(popsize, R0, contactmatrix, relsusc, reltransm, initR, initI, initV,
method = "hybrid", nsims = 10)
# Stochastic runs can be reproduced directly with seed:
finalsize(popsize, R0, contactmatrix, relsusc, reltransm, initR, initI, initV,
method = "stochastic", nsims = 10, nthreads = 2, seed = 2026)
# Parallel stochastic simulations are available for interactive use, but are
# omitted from examples to avoid spawning worker processes during R CMD check.
```

---

getCensusData *Get Census Population Data by Age and County*

Description

Downloads and processes U.S. Census Bureau population estimates for a specified state and county, organized by age groups. Supports single-year age data with optional sex disaggregation.

Usage

```
getCensusData(
  state_fips,
  county_name,
  year = 2024,
  age_groups = NULL,
  by_sex = FALSE,
  csv_path = NULL,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

state_fips	Two-digit FIPS code for the state (e.g., "49" for Utah)
county_name	Name of the county (e.g., "Salt Lake County")
year	Census estimate year: 2020-2024 for July 1 estimates, or 2020.1 for April 1, 2020 base
age_groups	Vector of age limits for grouping (e.g., c(0, 5, 18, 65)). Default NULL returns single-year ages 0-85+
by_sex	Logical, if TRUE returns separate male/female groups
csv_path	Optional path to a previously downloaded census CSV file. If provided, data will be read from this file instead of downloading. Use cache_dir for automatic caching.
cache_dir	Optional directory path for caching downloaded census files. If provided, the function will check for an existing cached file and use it, or download and save a new one. Default is NULL (no caching). Use "." for current directory or specify a custom path like "~/census_cache"
verbose	Logical, if TRUE prints messages about data loading and age aggregation. Default is FALSE.

Value

A list containing:

county	County name
state	State name
year	Census year
total_pop	Total population
age_pops	Vector of populations by age group
age_labels	Labels for each age group
sex_labels	If by_sex=TRUE, labels indicating sex
data	Full filtered data frame

Examples

```
# Use the included example data (recommended for package examples)
slc_data <- getCensusData(
  state_fips = "49",
  county_name = "Salt Lake County",
  year = 2024,
  csv_path = getCensusDataPath()
)

# Get age groups without sex disaggregation
slc_grouped <- getCensusData(
  state_fips = "49",
  county_name = "Salt Lake County",
  year = 2024,
  age_groups = c(0, 5, 18, 65),
  csv_path = getCensusDataPath()
)

# Get age groups by sex
slc_by_sex <- getCensusData(
  state_fips = "49",
  county_name = "Salt Lake County",
  year = 2024,
  age_groups = c(0, 5, 18, 65),
  by_sex = TRUE,
  csv_path = getCensusDataPath()
)


# Download from web (requires internet)
slc_web <- getCensusData(
  state_fips = "49",
  county_name = "Salt Lake County",
  year = 2024
)

# Use caching to avoid repeated downloads
slc_cached <- getCensusData(
  state_fips = "49",
  county_name = "Salt Lake County",
  year = 2024,
  cache_dir = "~/census_cache"
)
```

`getCensusDataPath` *Get path to example census data file*

Description

Returns the path to the example Utah census data CSV file included with the package. This is useful for examples, testing, and when internet access is not available.

Usage

```
getCensusDataPath()
```

Value

Character string with the path to the example census CSV file for Utah (FIPS 49)

Examples

```
# Get path to example Utah census file
utah_csv <- getCensusDataPath()

# Use it with getCensusData

slc_data <- getCensusData(
  state_fips = "49",
  county_name = "Salt Lake County",
  year = 2024,
  csv_path = getCensusDataPath()
)
```

---

getCityData *Get City Population Data by Age*

Description

Reads and processes population data for specific cities from ACS 5-year estimates, organized by age groups. The ACS data provides 5-year age groupings (0-4, 5-9, etc.) which can be disaggregated into single-year ages or aggregated into custom age groups.

Usage

```
getCityData(
  city_name,
  csv_path,
 age_groups = c(0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85),
  verbose = FALSE
)
```

Arguments

city_name	Name of the city (e.g., "Hildale city, Utah")
csv_path	Path to the city population CSV file
age_groups	Vector of age limits for grouping. If NULL, returns single-year ages (disaggregated from 5-year ACS groups). Default uses 5-year intervals: c(0,5,10,...,85)
verbose	Logical, if TRUE prints messages about age aggregation. Default is FALSE.

Value

A list containing:

city	City name
year	Data year
total_pop	Total population
age_pops	Vector of populations by age group
age_labels	Labels for each age group
data	Full data frame

Examples

```
# Load Hildale data with default 5-year age groups
hildale_data <- getCityData(
  city_name = "Hildale city, Utah",
 csv_path = system.file("extdata", "hildale_ut_2023.csv", package = "multigroup.vaccine")
)

# Load with single-year ages (disaggregated)
hildale_single <- getCityData(
  city_name = "Hildale city, Utah",
 csv_path = system.file("extdata", "hildale_ut_2023.csv", package = "multigroup.vaccine"),
  age_groups = NULL
)

# Load with custom age groups
hildale_custom <- getCityData(
  city_name = "Hildale city, Utah",
 csv_path = system.file("extdata", "hildale_ut_2023.csv", package = "multigroup.vaccine"),
  age_groups = c(0, 18, 65)
)
```

`getFinalSizeAnalytic` *Calculate final size of outbreak: the total number of infections in each group, by solving the analytic final size equation*

Description

Calculate final size of outbreak: the total number of infections in each group, by solving the analytic final size equation

Usage

```
getFinalSizeAnalytic(transmrates, recoveryrate, popsize, initR, initI, initV)
```

Arguments

`transmrates`	matrix of group-to-group (column-to-row) transmission rates
`recoveryrate`	inverse of mean infectious period
`popsize`	the population size of each group
`initR`	initial number of each group already infected and removed (included in final size)
`initI`	initial number of each group infectious
`initV`	initial number of each group vaccinated

Value

vector of final sizes (number of infected over whole outbreak) for each group

Examples

```
getFinalSizeAnalytic(transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10))
```

`getFinalSizeDist` *Estimate the distribution of final outbreak sizes by group using stochastic simulations of multi-group model*

Description

Estimate the distribution of final outbreak sizes by group using stochastic simulations of multi-group model

Usage

```
getFinalSizeDist(n, transmrates, recoveryrate, popsize, initR, initI, initV)
```

Arguments

n	the number of simulations to run
transmrates	matrix of group-to-group (column-to-row) transmission rates
recoveryrate	inverse of mean infectious period
popsize	the population size of each group
initR	initial number of each group already infected and removed (included in size result)
initI	initial number of each group infectious
initV	initial number of each group vaccinated

Value

a matrix with the final number infected from each group (column) in each simulation (row)

Examples

```
getFinalSizeDist(n = 10, transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10))
```

---

getFinalSizeDistEscape

Estimate the distribution of final outbreak sizes by group using a hybrid model: stochastic simulations for smaller-sized outbreaks and deterministic ordinary differential equation model for "escaped" outbreaks

---

Description

Estimate the distribution of final outbreak sizes by group using a hybrid model: stochastic simulations for smaller-sized outbreaks and deterministic ordinary differential equation model for "escaped" outbreaks

Usage

```
getFinalSizeDistEscape(
  n,
  transmrates,
  recoveryrate,
  popsize,
  initR,
  initI,
  initV
)
```

Arguments

n	the number of simulations to run
transmrates	matrix of group-to-group (column-to-row) transmission rates
recoveryrate	inverse of mean infectious period
popsize	the population size of each group
initR	initial number of each group already infected and removed (included in size result)
initI	initial number of each group infectious
initV	initial number of each group vaccinated

Value

a matrix with the final number infected from each group (column) in each simulation (row)

Examples

```
getFinalSizeDistEscape(n = 10, transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10))
```

---

getFinalSizeODE *Calculate outbreak final size, the total number of infections in each group, by numerically solving the multi-group ordinary differential equation*

Description

Calculate outbreak final size, the total number of infections in each group, by numerically solving the multi-group ordinary differential equation

Usage

```
getFinalSizeODE(transmrates, recoveryrate, popsize, initR, initI, initV)
```

Arguments

transmrates	matrix of group-to-group (column-to-row) transmission rates
recoveryrate	inverse of mean infectious period
popsize	the population size of each group
initR	initial number of each group already infected and removed (included in final size)
initI	initial number of each group infectious
initV	initial number of each group vaccinated

Value

vector of final sizes (number of infected over whole outbreak) for each group

Examples

```
getFinalSizeODE(transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10))
```

---

getSizeAtTime *Calculate outbreak size at a given time*

Description

Calculate outbreak size at a given time

Usage

```
getSizeAtTime(time, transmrates, recoveryrate, popsize, initR, initI, initV)
```

Arguments

time	the time at which to calculate the outbreak size
transmrates	matrix of group-to-group (column-to-row) transmission rates
recoveryrate	inverse of mean infectious period
popsize	the population size of each group
initR	initial number of each group already infected and removed (included in size result)
initI	initial number of each group infectious
initV	initial number of each group vaccinated

Value

a list with totalSize (total cumulative infections) and activeSize (total currently infected) in each group at the specified time

Examples

```
getSizeAtTime(time = 30, transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10))
```

getStateFIPS *Get state FIPS code by state name*

Description

Get state FIPS code by state name

Usage

```
getStateFIPS(state_name)
```

Arguments

state_name	State name (e.g., "Utah")

Value

Two-digit FIPS code as character

Examples

```
getStateFIPS("Utah")  # Returns "49"
```

listCounties *List available counties for a state*

Description

List available counties for a state

Usage

```
listCounties(
  state_fips,
  year = 2024,
  csv_path = NULL,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

state_fips	Two-digit FIPS code for the state
year	Census year (2020-2024), default 2024
csv_path	Optional path to a previously downloaded census CSV file
cache_dir	Optional directory path for caching downloaded census files
verbose	Logical, if TRUE prints messages about data loading. Default is FALSE.

Value

Character vector of county names

Examples

```
# Use the included example data
utah_counties <- listCounties(
  state_fips = "49",
  year = 2024,
  csv_path = getCensusDataPath()
)


# Download from web (requires internet)
utah_counties_web <- listCounties(state_fips = "49", year = 2024)

# With caching
utah_counties_cached <- listCounties(state_fips = "49", cache_dir = "~/census_cache")
```

---

odeSIR *Ordinary differential equation function for multi-group susceptible-infectious-removed (SIR) model used as "func" argument passed to the ode() function from deSolve package*

Description

Ordinary differential equation function for multi-group susceptible-infectious-removed (SIR) model used as "func" argument passed to the ode() function from deSolve package

Usage

```
odeSIR(time, state, par)
```

Arguments

time	vector of times at which the function will be evaluated
state	vector of number of individuals in each group at each state: S states followed by I states followed by R states
par	vector of parameter values: group-to-group transmission rate matrix elements (row-wise) followed by recovery rate

Value

a list of three vectors of derivatives dS, dI, and dR for each group, evaluated at the given state values

Examples

```
# Intended only for use as the func argument to the ode() function from the deSolve package:
y0 <- c(S1 = 79999, S2 = 20000, I1 = 1, I2 = 0, R1 = 0, R2 = 0)
parms <- c(beta11 = 1.6e-6, beta21 = 1.5e-6, beta12 = 1.4e-6, beta22 = 8.7e-6, recoveryrate = 1/7)
times <- seq(0, 350, len = 10)
deSolve::ode(y0, times, odeSIR, parms)
```

---

run_my_app *Runs the shiny app*

Description

Runs the shiny app

Usage

```
run_my_app(...)
```

Arguments

...	Further arguments passed to shiny::shinyAppDir().

Details

The app featured in this package is the one presented in the shiny demo: https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/index.html.

Value

Starts the execution of the app, printing the port on the console.

Examples

```
# To be executed interactively only
if (interactive()) {
  run_my_app()
}
```

simodeSIR *Calculate time trajectory of solution to a Susceptible-Infectious-Removed (SIR) system by numerically solving the multi-group ordinary differential equations*

Description

Calculate time trajectory of solution to a Susceptible-Infectious-Removed (SIR) system by numerically solving the multi-group ordinary differential equations

Usage

```
simodeSIR(
  transmrates,
  recoveryrate,
  popsize,
  initR,
  initI,
  initV,
  times = NULL
)
```

Arguments

transmrates	matrix of group-to-group (column-to-row) transmission rates
recoveryrate	inverse of mean infectious period
popsize	the population size of each group
initR	initial number of each group already infected and removed (included in final size)
initI	initial number of each group infectious
initV	initial number of each group vaccinated
times	optional argument for times at which the solution is to be calculated; if NULL (the default) the solution will be computed up to the time at which the total number of infectious individuals is sufficiently close to zero (outbreak extinction)

Value

vector of final sizes (number of infected over whole outbreak) for each group

Examples

```
simodeSIR(transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10))
simodeSIR(transmrates = matrix(0.2, 2 ,2), recoveryrate = 0.3,
popsize = c(100, 150), initR = c(0, 0), initI = c(0, 1), initV = c(10, 10), times = 0:10)
```

transmissionRates — *Calculate transmission rate matrix for multi-group model with specified R0*

Description

Calculate transmission rate matrix for multi-group model with specified R0

Usage

```
transmissionRates(R0, meaninf, reltransm)
```

Arguments

R0	overall basic reproduction number
meaninf	mean duration of infectious period
reltransm	matrix with relative transmission rates, from column-group to row-group

Value

a matrix of transmission rates to (row) and from (column) each group, in same time units as meaninf

Examples

```
transmissionRates(R0 = 15, meaninf = 7,
  reltransm = rbind(c(1, 0.5, 0.9), c(0.3, 1.9, 1), c(0.3, 0.6, 2.8)))
```

vaxrepnum — *Calculate reproduction number for a multigroup model with a given state of vaccination and immunity*

Description

Calculate reproduction number for a multigroup model with a given state of vaccination and immunity

Usage

```
vaxrepnum(meaninf, popsize, trmat, initR, initV, vaxeff)
```

Arguments

meaninf	mean infectious period with same time units as trmat
popsize	the population size of each group
trmat	matrix of group-to-group (column-to-row) transmission rates
initR	initial number of each group already infected and immune
initV	initial number of each group vaccinated
vaxeff	effectiveness (0 to 1) of vaccine in producing immunity to infection

Value

the reproduction number

Examples

```
meaninf <- 7
popsize <- c(200, 800)
initR <- c(0, 0)
initV <- c(0, 0)
vaxeff <- 1
trmat <- matrix(c(0.63, 0.31, 0.19, 1.2), 2, 2)
vaxrepnum(meaninf, popsize, trmat, initR, initV, vaxeff)
vaxrepnum(meaninf, popsize, trmat, initR, initV = c(160, 750), vaxeff)
```

Index