Package: sircovid (via r-universe)

June 17, 2026

Title SIR Model for COVID-19

Version 0.15.1

Description Mechanistic models of transmission of the SARS-Cov-2 (COVID-19) virus implemented as stochastic compartmental models in odin and dust. Uses mcstate to perfom Bayesian evidence synthesis from several surveillance data streams through the estimation of transmission parameters.

License MIT + file LICENSE

URL https://github.com/mrc-ide/sircovid, https://mrc-ide.github.io

BugReports https://github.com/mrc-ide/sircovid/issues

Encoding UTF-8

LazyData true

Language en-GB

Depends R (>= 3.1.0)

Imports dust (>= 0.15.1), eigen1 (>= 0.1.1), mcstate (>= 0.9.18), socialmixr

LinkingTo cpp11 (>= 0.4.4), dust (>= 0.15.1)

Suggests EpiEstim, distcrete, mockery, odin (>= 1.5.9), odin.dust (>= 0.3.9), testthat

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

SystemRequirements C++11

Remotes mrc-ide/dust, mrc-ide/eigen1, mrc-ide/mcstate, mrc-ide/odin, mrc-ide/odin.dust

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

Repository https://ncov-ic.r-universe.dev

Date/Publication 2024-06-19 08:41:01 UTC

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

RemoteRef master

RemoteSha 5492e5a47e9b973bbec57020fa79c4731acfb56d

Contents

`add_trajectory_incidence` *Add incidence to trajectories*

Description

Augment (or remove) trajectories from a `mcstate_trajectories` or `mcstate_pmcmc` object.

Usage

```
add_trajectory_incidence(obj, states, suffix = "_inc")

drop_trajectory_incidence(obj)
```

Arguments

`obj`	A `mcstate_trajectories` or `mcstate_pmcmc` object to update
`states`	A vector of cumulative states to compute incidence for
`suffix`	A string to append to the input states to create the incidence variables

---

`basic` *The basic sircovid model*

Description

The "basic" sircovid model. This is a dust model.

Examples

```
# Set up the basic model for England with default parameters and
# an initial seeding of early February:
p <- basic_parameters(sircovid_date("2020-02-07"), "england")
mod <- basic$new(p, 0, 10)

# Set the initial state and index as we would us for a run
# (without setting an initial state there is no seeding)
initial <- basic_initial(mod$info(), 10, p)
mod$update_state(state = initial)
mod$set_index(basic_index(mod$info())$run)

# Run the model up to the end of march
step_end <- sircovid_date("2020-03-31") / p$dt
```

```
# The filtered state is returned at the end of the run
mod$run(step_end)

# More state can be retrieved using the "state" method
mod$state(1:6)
```

`basic_compare` *Compare observed and modelled data for the basic model*

Description

Compare observed and modelled data from the basic model. This conforms to the mcstate interface.

Usage

```
basic_compare(state, observed, pars)
```

Arguments

`state`	State vector for the end of the current day. This is assumed to be filtered following `basic_index()` so contains rows corresponding to ICU and deaths.
`observed`	Observed data. This will be a list with elements `icu` (number of ICU beds occupied) and `deaths` (number of deaths over this day).
`pars`	A list of parameters, as created by `basic_parameters()`

Value

A vector of log likelihoods, the same length as the number of particles (the number of columns in the modelled state)

Examples

```
state <- rbind(icu = 10:15, deaths_inc = 1:6)
observed <- list(icu = 13, deaths = 3)
pars <- basic_parameters(sircovid_date("2020-02-07"), "england")
basic_compare(state, observed, pars)
basic_compare(state * 5, observed, pars)
```

basic_index *Index of basic model*

Description

Index of "interesting" elements for the basic model. This function conforms to the mcstate interface.

Usage

```
basic_index(info)
```

Arguments

info	The result of running the `$info()` method on an initialised basic model

Value

A list with element `run`, indicating the locations of the ICU and D compartments.

Examples

```
p <- basic_parameters(sircovid_date("2020-02-07"), "england")
mod <- basic$new(p, 0, 10)
basic_index(mod$info())
```

basic_initial *Initial conditions for the basic model*

Description

Create initial conditions for the basic model. This matches the interface required for mcstate

Usage

```
basic_initial(info, n_particles, pars)
```

Arguments

info	The result of running the `$info()` method on an initialised basic model
n_particles	The number of particles required. Currently only uniform initial seeding is implemented so this has no effect
pars	A parameter list created by `basic_parameters()`; from this list we will use the `population` element.

Value

A numeric vector of initial conditions

Examples

```
p <- basic_parameters(sircovid_date("2020-02-07"), "england")
mod <- basic$new(p, 0, 10)
basic_initial(mod$info(), 10, p)
```

`basic_parameters` *Parameters for the basic model*

Description

Parameters for the "basic" model.

Usage

```
basic_parameters(
  start_date,
  region,
  beta_date = NULL,
  beta_value = NULL,
  beta_type = "piecewise-linear",
  severity = NULL,
  exp_noise = 1e+06,
  initial_seed_size = 30,
  initial_seed_pattern = 1
)
```

Arguments

`start_date`	The start date, as a `sircovid_date()` (i.e., the number of days into 2020)
`region`	The region to run the model for. This will bee used to get population data, which is currently fixed within the package and is limited to "uk", the four constituent nations ("england", "wales", "scotland", "northern_ireland") and the 7 NHS regions (e.g., "midlands"). These names are case insensitive.
`beta_date`	A vector of dates (each as a `sircovid_date()`) for changes in beta (the contact rate parameter), or `NULL` if a single value is used for all times (see `sircovid_parameters_piecewise_li` or `sircovid_parameters_piecewise_constant()`, where this is passed as `date`). If `beta_type = "piecewise-constant"`, then the first date must be 0.
`beta_value`	A vector of values for beta (the contact rate parameter). If not given, and if `beta_date` is `NULL` then a value of 0.1 will be used through the whole simulation, otherwise if `beta_date` is `NULL` this must be a scalar. If `beta_date` is given then `beta_date` and `beta_value` must have the same length (see `sircovid_parameters_piecewise_li` or `sircovid_parameters_piecewise_constant()`, where this is passed as `value`).

beta_type — The type of form used for beta (the contact rate parameter), which currently can be "piecewise-linear" or "piecewise-constant"

severity — Severity data, via Bob Verity's markovid package. This needs to be NULL (use the default bundled data version in the package), a data.frame object (for raw severity data) or a list (for data that has already been processed by sircovid for use). New severity data comes from Bob Verity via the markovid package, and needs to be carefully calibrated with the progression parameters.

exp_noise — Rate of exponential noise used in the compare function - typically set to a large value so that noise is small but non-zero. If set to Inf then there is no noise in the observation process (not realistic but useful for testing).

initial_seed_size — Initial size of seeding from the S to E compartment; all seeding is in the 15-19 year old group from start_date according to initial_seed_pattern. The default is 30.

initial_seed_pattern — A vector of seeding weights for the initial seeding. The length represents the number of steps to seed over from the start_date, and the initial_seed_size is split over these steps according to those weights. If start_date is not a multiple of the step size (and thus falls between two steps) then we weight over an additional step and adjust the weights according to how far the start_date is from the previous full step.

Value

A list of inputs to the model, many of which are fixed and represent data. These correspond largely to user() calls within the odin code, though some are also used in processing just before the model is run.

Examples

```
basic_parameters(sircovid_date("2020-02-01"), "uk")
```

---

check_sircovid_model *Check a model type*

Description

Check that a model type is valid (i.e. in sircovid_models)

Usage

```
check_sircovid_model(sircovid_model)
```

Arguments

sircovid_model String

combine_rt *Combine Rt estimates*

Description

Combine Rt across multiple runs.

Usage

```
combine_rt(rt, samples, rank = TRUE, weight = "infections_inc")
```

Arguments

rt	A list of Rt calculations from lancelot_Rt_trajectories() (though any Rt calculation that confirms to this will work)
samples	A list of samples from mcstate::pmcmc
rank	A boolean deciding whether to rank trajectories by increasing incidence or not before combining Rt estimates
weight	A string specifying what trajectory to use for weighting, defaults to "infections_inc" (used for Rt calculation)

Value

A list of Rt output in the same structure as the first element of rt. All Rt estimates will be aggregated across regions (or whatever else you are aggregating on) based on the parameters in samples.

combine_rt_epiestim *Combine Rt estimates from EpiEstim*

Description

Combine Rt estimates from EpiEstim across multiple runs.

Usage

```
combine_rt_epiestim(rt, samples, q = NULL, rank = TRUE)
```

Arguments

rt	A list of Rt calculations from lancelot_rt_trajectories_epiestim() (though any Rt calculation that confirms to this will work)
samples	A list of samples from mcstate::pmcmc
q	A vector of quantiles to return values for
rank	A boolean deciding whether to rank trajectories by increasing incidence or not before combining Rt estimates

Value

A list of Rt output in the same structure as the first element of `rt`. Rt estimates will be aggregated across regions (or whatever else you are aggregating on) based on the parameters in `samples`.

`combine_trajectories` *Combine trajectories*

Description

Combine trajectories across multiple runs

Usage

```
combine_trajectories(samples, rank = TRUE)
```

Arguments

`samples`	A list of samples from mcstate::pmcmc
`rank`	Logical, indicating if trajectories should be ranked before combination.

Value

A set of combined trajectories (not a combined samples object).

`compile_gpu` *Create GPU model*

Description

Create a model for the GPU. This requires a working nvcc toolchain and GPU device to work properly. Note that this will cache the compilation within a session, so if you want to change GPU options you will need to restart R. However, if you change the *model* (e.g., by changing the result of options like `rewrite_dims`) the model will recompile. Compilation speed is slow enough that the cachine will be obvious.

Usage

```
compile_gpu(model = "lancelot", ..., real_type = "float", gpu = TRUE)
```

Arguments

`model`	The name of a sircovid model, either `lancelot` or `basic`
`...`	Additional arguments passed to odin.dust::odin_dust_ and from there into either odin's options or the gpu options.
`real_type`	The type to use for floating point numbers. The default is "float" which differs from the CPU model but is typically what you want on the GPU.
`gpu`	The argument passed to odin.dust::odin_dust_ as gpu. The default here is `TRUE`, but you may want to pass the results of dust::dust_cuda_options in order to control compilation.

Value

A model generator, like lancelot that can run on the GPU available to this computer.

`get_sample_rank` *Get the ranking of sample by a given variable, e.g. infections*

Description

Get the ranking of sample

Usage

```
get_sample_rank(sample, by = "infections")
```

Arguments

`sample`	An `mcstate_pmcmc` object
`by`	The name of the variable used for ranking. The ranking is computed based on the value of this variable at the last time step. Default is set to "infections", which will give the rank based on the cumulative number of infections.

Value

A vector of integers giving the ranks

inflate_state_strains *Inflate strains in model state*

Description

Inflate model state, so that additional strain compartments are created but zerod. Use this to run the model for a while, then set it up to work with new strains.

Usage

```
inflate_state_strains(state1, info1, info2)
```

Arguments

state1	The state matrix with one strain
info1	The model info with one strain (or a dust model object)
info2	The model info with two strains (or a dust model object)

Value

An expanded model state with two strains

Author(s)

Richard Fitzjohn

inflate_state_vacc_classes *Inflate vacc classes in model state*

Description

Inflate model state, so that additional vacc class compartments are created but zerod. Use this to run the model for a while, then set it up to work with new vacc classes (e.g. booster doses).

Usage

```
inflate_state_vacc_classes(state1, info1, info2)
```

Arguments

state1	The state matrix with X vacc classes
info1	The model info with X vacc classes
info2	The model info with Y > X vacc classes

Value

An expanded model state with Y > X vacc classes

`lancelot` *The lancelot sircovid model*

Description

##' Our "current" sircovid model. This is a dust model.

`lancelot_check_data` *Check data for particle filter*

Description

Check that data for the particle filter has required columns and constraints among columns. This function does not alter the data at present (though it does return it invisibly).

Usage

```
lancelot_check_data(data)
```

Arguments

`data`	A data.frame of data

Value

Invisibly, a data frame, identical to `data`

`lancelot_check_severity`
Check severity parameters

Description

Checks severity probabilities are valid

Usage

```
lancelot_check_severity(pars)
```

Arguments

pars	A list of probabilities from lancelot_parameters.

Value

Returns pars invisibly if all severity probabilities lie in

---

lancelot_compare *Compare observed and modelled data for the lancelot model*

Description

Compare observed and modelled data from the lancelot model. This conforms to the mcstate interface.

Usage

```
lancelot_compare(state, observed, pars)
```

Arguments

state	State vector for the end of the current day. This is assumed to be filtered following lancelot_index() so contains 10 rows corresponding to ICU, general beds, admissions, deaths and seroconversion compartments.
observed	Observed data. At the moment please see the tests for a full list as this changes frequently (and this function may be removed in future).
pars	A list of parameters, as created by lancelot_parameters()

Value

A vector of log likelihoods, the same length as the number of particles (the number of columns in the modelled state)

---

lancelot_ifr_excl_immunity *Compute "IFR" excluding immunity*

Description

Compute "IFR" excluding immunity.

Usage

```
lancelot_ifr_excl_immunity(time, pars)
```

Arguments

time	A vector of time steps to calculate IFR at
pars	An unnamed list of lancelot_parameters() objects

---

lancelot_index *Index of lancelot model*

Description

Index of "interesting" elements for the lancelot model. This function conforms to the mcstate interface.

Usage

```
lancelot_index(
  info,
  rt = TRUE,
  cum_admit = TRUE,
  diagnoses_admitted = TRUE,
  cum_infections_disag = TRUE,
  cum_n_vaccinated = TRUE,
  D_all = TRUE,
  D_hosp = TRUE,
  infections_inc_per_strain = TRUE,
  severity = FALSE,
  protected = FALSE
)
```

Arguments

info	The result of running the $info() method on an initialised basic model
rt	Logical, whether to output trajectories required for calculating Rt (default = TRUE)
cum_admit	Logical, whether to output cumulative admissions by age (default = TRUE)
diagnoses_admitted	Logical, whether to output cumulative combined confirmed admissions and inpatient diagnoses by age and vaccine class (default = TRUE)
cum_infections_disag	Logical, whether to output cumulative infections by age and vaccine class (default = TRUE)
cum_n_vaccinated	Logical, whether to output cumulative number vaccinated by age and vaccine class (default = TRUE)
D_all	Logical, whether to output all deaths by age and vaccine class (default = TRUE)

D_hosp	Logical, whether to output hospital deaths by age (default = TRUE)
infections_inc_per_strain	Logical, whether to output infections incidence per strain (default = TRUE)
severity	Logical, whether to output severity calculations (default = FALSE)
protected	Logical, whether to output protected levels

Value

A list with element run, indicating the locations of (in order) (1) ICU, (2) general, (3) deaths in community, (4) deaths in hospital, (5) total deaths, (6) cumulative confirmed admissions,(7) cumulative confirmed new admissions, (8) probability of a positive sero test, (9) probability of a positive pillar 2 test, and with element state containing the same values followed by 17 S compartments (one per age group, then one for carehome workers and carehome residents respectively) and 17 "cumulative admission" compartments.

Examples

```
p <- lancelot_parameters(sircovid_date("2020-02-07"), "england")
mod <- lancelot$new(p, 0, 10)
lancelot_index(mod$info())
```

---

lancelot_initial *Initial conditions for the lancelot model*

Description

Create initial conditions for the lancelot model. This matches the interface required for mcstate

Usage

```
lancelot_initial(info, n_particles, pars)
```

Arguments

info	The result of running the $info() method on an initialised basic model
n_particles	The number of particles required. Currently only uniform initial seeding is implemented so this has no effect
pars	A parameter list created by basic_parameters(); from this list we will use the population element.

Value

A numeric vector of initial conditions

Examples

```
p <- lancelot_parameters(sircovid_date("2020-02-07"), "england")
mod <- lancelot$new(p, 0, 10)
lancelot_initial(mod$info(), 10, p)
```

`lancelot_parameters` *Parameters for the lancelot model*

Description

Parameters for the "lancelot" model.

Usage

```
lancelot_parameters(
  start_date,
  region,
  beta_date = NULL,
  beta_value = NULL,
  beta_type = "piecewise-linear",
  population = NULL,
  has_carehomes = TRUE,
  carehome_beds = NULL,
  severity = NULL,
  progression = NULL,
  observation = NULL,
  sens_and_spec = NULL,
  initial_seed_size = 30,
  initial_seed_pattern = 1,
  eps = 0.1,
  m_CHW = 4e-06,
  m_CHR = 5e-05,
  strain_transmission = 1,
  strain_seed_date = NULL,
  strain_seed_size = NULL,
  strain_seed_pattern = NULL,
  strain_rel_gamma_E = 1,
  strain_rel_gamma_A = 1,
  strain_rel_gamma_P = 1,
  strain_rel_gamma_C_1 = 1,
  strain_rel_gamma_C_2 = 1,
  strain_rel_gamma_PCR_pre = 1,
  strain_rel_gamma_PCR_pos = 1,
  strain_rel_p_sympt = 1,
  strain_rel_p_hosp_if_sympt = 1,
  strain_rel_p_icu = 1,
  strain_rel_p_death = 1,
  strain_rel_p_G_D = 1,
  rel_susceptibility = 1,
  rel_p_sympt = 1,
  rel_p_hosp_if_sympt = 1,
  rel_p_death = 1,
```

```
  rel_infectivity = 1,
  vaccine_progression_rate = NULL,
  vaccine_schedule = NULL,
  vaccine_index_dose2 = NULL,
  vaccine_index_booster = NULL,
  vaccine_catchup_fraction = 1,
  n_doses = 2L,
  vacc_skip_progression_rate = NULL,
  vacc_skip_to = NULL,
  vacc_skip_weight = NULL,
  waning_rate = 0,
  exp_noise = 1e+06,
  cross_immunity = 1
)
```

Arguments

start_date	The start date, as a sircovid_date() (i.e., the number of days into 2020)
region	The region to run the model for. This will bee used to get population data, which is currently fixed within the package and is limited to "uk", the four constituent nations ("england", "wales", "scotland", "northern_ireland") and the 7 NHS regions (e.g., "midlands"). These names are case insensitive.
beta_date	A vector of dates (each as a sircovid_date()) for changes in beta (the contact rate parameter), or NULL if a single value is used for all times (see sircovid_parameters_piecewise_lin or sircovid_parameters_piecewise_constant(), where this is passed as date). If beta_type = "piecewise-constant", then the first date must be 0.
beta_value	A vector of values for beta (the contact rate parameter). If not given, and if beta_date is NULL then a value of 0.1 will be used through the whole simulation, otherwise if beta_date is NULL this must be a scalar. If beta_date is given then beta_date and beta_value must have the same length (see sircovid_parameters_piecewise_li or sircovid_parameters_piecewise_constant(), where this is passed as value).
beta_type	The type of form used for beta (the contact rate parameter), which currently can be "piecewise-linear" or "piecewise-constant"
population	Population data. A vector of length 17 for the population size for age groups 0-4, 5-9, ..., 75-79, 80+. If NULL, the population data will be sourced within the package for the specified region (only if available).
has_carehomes	Logical, whethere or not the model has carehomes.
carehome_beds	The number of care home beds in the region. If NULL, this will be sourced within the package for the specified region (only if available).
severity	Severity data, via Bob Verity's markovid package. This needs to be NULL (use the default bundled data version in the package), a data.frame object (for raw severity data) or a list (for data that has already been processed by sircovid for use). New severity data comes from Bob Verity via the markovid package, and needs to be carefully calibrated with the progression parameters.
progression	Progression data

observation
: Either NULL or a list of observation parameters. If NULL, then a list of observation parameters will be generated using lancelot_parameters_observation(exp_noise)

sens_and_spec
: Either NULL or a list of diagnostic test sensitivity and specificity parameters. If NULL, then a list of sensitivity and specificity parameters will be generated using lancelot_parameters_sens_and_spec()

initial_seed_size
: Initial size of seeding from the S to E compartment; all seeding is in the 15-19 year old group from start_date according to initial_seed_pattern. The default is 30.

initial_seed_pattern
: A vector of seeding weights for the initial seeding. The length represents the number of steps to seed over from the start_date, and the initial_seed_size is split over these steps according to those weights. If start_date is not a multiple of the step size (and thus falls between two steps) then we weight over an additional step and adjust the weights according to how far the start_date is from the previous full step.

eps
: Change in contact rate for carehome residents

m_CHW
: Contact rate between carehome workers and either residents or workers

m_CHR
: Contact rate between carehome residents

strain_transmission
: Vector of length two for relative transmissibility of each strain modelled. Length will define the number of strains used in the model, either 1 or 2.

strain_seed_date
: Either NULL (no seeding) or a sircovid_date corresponding to the date the seeding of strain 2 begins.

strain_seed_size
: Either NULL (no seeding) or the size of strain 2 seeding from the S to E compartment; all seeding is in the 15-19 year old group from strain_seed_date according to strain_seed_pattern.

strain_seed_pattern
: Either NULL (no seeding) or a vector of seeding weights for the initial seeding. The length represents the number of steps to seed over from the start_date, and the strain_seed_size is split over these steps according to those weights. If start_seed_date is not a multiple of the step size (and thus falls between two steps) then we weight over an additional step and adjust the weights according to how far the strain_seed_date is from the previous full step.

strain_rel_gamma_E
: Vector of relative rates of progression out of E (gamma_E) for each strain modelled. If 1 all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_gamma_A
: Vector of relative rates of progression out of I_A (gamma_A) for each strain modelled. If 1 all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_gamma_P
: Vector of relative rates of progression out of I_P (gamma_P) for each strain modelled. If 1 all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_gamma_C_1
: Vector of relative rates of progression out of I_C_1 (gamma_C_1) for each strain modelled. If 1 all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_gamma_C_2
: Vector of relative rates of progression out of I_C_2 (gamma_C_2) for each strain modelled. If 1 all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_gamma_PCR_pre
: Vector of relative rates of PCR pre-positive duration for each strain modelled. if '1' all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_gamma_PCR_pos
: Vector of relative rates of PCR positivity duration for each strain modelled. if '1' all strains have same rates. Otherwise vector of same length as strain_transmission, with entries that determines the relative scaling of the defaults for each strain.

strain_rel_p_sympt
: Vector of relative probabilities of symptoms for each strain modelled. If 1 all strains have same probabilities of symptoms. Otherwise vector of same length as strain_transmission, where the first value should be 1 (for the first strain) and subsequent values between 0 and 1. In this case parameters will be "mirrored" for pseudostrains i.e. the relative probability of symptoms will be assume the same irrespective of previous infection with another strain. Alternatively, a vector of twice the length of strain_transmission can be provided to allow specifying directly relative probability of symptoms for each pseudostrain (with strain 3 - 1.2 and strain 4 = 2.1). To ensure valid probabilities, p_sympt is upper-truncated at 1 after scaling.

strain_rel_p_hosp_if_sympt
: Vector of relative probabilities of hospitalisation given symptoms for each strain modelled. If 1 all strains have same probabilities of hospitalisation. Otherwise vector of same length as strain_transmission, where the first value should be 1 (for the first strain) and subsequent values between 0 and 1. In this case parameters will be "mirrored" for pseudostrains i.e. the relative probability of hospitalisation will be assume the same irrespective of previous infection with another strain. Alternatively, a vector of twice the length of strain_transmission can be provided to allow specifying directly relative probability of hospitalisation for each pseudostrain (with strain 3 - 1.2 and strain 4 = 2.1). To ensure valid probabilities, p_hosp_if_sympt is upper-truncated at 1 after scaling.

strain_rel_p_icu
: Vector of relative probabilities of icu given hospitalised for each strain modelled. If 1 all strains have same probabilities of icu admission. Otherwise vector

of same length as strain_transmission, where the first value should be 1 (for the first strain) and subsequent values between 0 and 1. In this case parameters will be "mirrored" for pseudostrains i.e. the relative probability of icu admission will be assume the same irrespective of previous infection with another strain. Alternatively, a vector of twice the length of strain_transmission can be provided to allow specifying directly relative probability of icu admission for each pseudostrain (with strain 3 - 1.2 and strain 4 = 2.1). To ensure valid probabilities, p_icu is upper-truncated at 1 after scaling.

strain_rel_p_death

Vector of relative probabilities of death for each strain modelled. If 1 all strains have same probabilities of death. Otherwise vector of same length as strain_transmission, where the first value should be 1 (for the first strain) and subsequent values between 0 and 1. In this case parameters will be "mirrored" for pseudostrains i.e. the relative probability of death will be assume the same irrespective of previous infection with another strain. Alternatively, a vector of twice the length of strain_transmission can be provided to allow specifying directly relative probability of death for each pseudostrain (with strain 3 - 1.2 and strain 4 = 2.1). To ensure valid probabilities, p_death is upper-truncated at 1 after scaling.

strain_rel_p_G_D

Vector of relative probabilities of death in the coummunity for each strain modelled. If 1 all strains have same probabilities of death. Otherwise vector of same length as strain_transmission, where the first value should be 1 (for the first strain) and subsequent values between 0 and 1. In this case parameters will be "mirrored" for pseudostrains i.e. the relative probability of death will be assume the same irrespective of previous infection with another strain. Alternatively, a vector of twice the length of strain_transmission can be provided to allow specifying directly relative probability of death for each pseudostrain (with strain 3 - 1.2 and strain 4 = 2.1). To ensure valid probabilities, p_G_D is upper-truncated at 1 after scaling.

rel_susceptibility

A vector or array of values representing the relative susceptibility of individuals in different vaccination groups. If a vector, the first value should be 1 (for the non-vaccinated group) and subsequent values be between 0 and 1. In that case relative susceptibility will be the same across all age groups within one vaccination category, and will be the same for all pathogen strains. Specifying an array instead of a vector allows different relative susceptibilities by age (first dimension of the array), pathogen strain (second dimension) and vaccination group (third dimension); in that case, the first layer (3rd dimension) of rel_susceptibility should be 1 (for the non-vaccinated group) for the first column (first infection with first strain) and other values between 0 and 1

rel_p_sympt

A vector or matrix of values of same dimension as rel_susceptibility representing the relative probability of symptomatic infection in different vaccination groups. If a vector, the first value should be 1 (for the non-vaccinated group) and subsequent values be between 0 and 1. In that case the relative reduction in probability of symptomatic infection will be the same across all age groups and all strains within one vaccination category. Specifying an array instead of a vector allows different relative reductions in probability of symptomatic infection by age (first dimension of the array), pathogen strain (second dimension) and

vaccination group (third dimension); in that case, the first layer of rel_p_sympt should be 1 (for the non-vaccinated group) for the first column (first infection with first strain) and other values between 0 and 1

`rel_p_hosp_if_sympt`

A vector or array of values of same dimension as rel_susceptibility representing the relative probability of hospitalisation for symptomatic cases in different vaccination groups. If a vector, the first value should be 1 (for the non-vaccinated group) and subsequent values be between 0 and 1. In that case the relative reduction in probability of hospitalisation for symptomatic cases will be the same across all age groups and all strains within one vaccination category. Specifying an array instead of a vector allows different relative reductions in probability of hospitalisation for symptomatic cases by age (first dimension of the array), pathogen strain (second dimension) and vaccination group (third dimension); in that case, the first layer of rel_p_hosp_if_sympt should be 1 (for the non-vaccinated group) for the first column (first infection with first strain) and other values between 0 and 1

`rel_p_death` A vector or array of values of same dimension as rel_susceptibility representing the relative probability of death for severe cases (either in hospital or the community) in different vaccination groups. If a vector, the first value should be 1 (for the non-vaccinated group) and subsequent values be between 0 and 1. In that case the relative reduction in probability of death for severe cases will be the same across all age groups and all strains within one vaccination category. Specifying an array instead of a vector allows different relative reductions in probability of death for severe cases by age (first dimension of the array), pathogen strain (second dimension) and vaccination group (third dimension); in that case, the first layer of rel_p_death should be 1 (for the non-vaccinated group) for the first column (first infection with first strain) and other values between 0 and 1

`rel_infectivity`

A vector or array of values representing the relative infectivity of individuals in different vaccination groups, if they are infected. If a vector, the first value should be 1 (for the non-vaccinated group) and subsequent values be between 0 and 1. In that case relative infectivity will be the same across all age groups and all strains within one vaccination category. Specifying an array instead of a vector allows different relative infectivities by age (first dimension of the array), pathogen strain (second dimension) and vaccination group (third dimension); in that case, the first layer of rel_infectivity should be 1 (for the non-vaccinated group) for the first column (first infection with first strain) and other values between 0 and 1

`vaccine_progression_rate`

A vector or matrix of values of same dimension as rel_susceptibility representing the rate of movement between different vaccination classes. If a vector, it should have as many values as vaccination classes, and the same rates of progression will be used for all age groups (the first rate is the vaccination rate and the last of the rates is the rate of returning to the initial vacination class); if a matrix, the element on row i and column j is the rate of progression from the jth vaccination class to the (j+1)th for age group i.

`vaccine_schedule`

A vaccine_schedule object indicating the people to be vaccinated by group over

time

vaccine_index_dose2
: The index to use for the second dose

vaccine_index_booster
: The indices to use for the booster doses

vaccine_catchup_fraction
: A value between 0 and 1 indicating the proportion of doses not distributed according to schedule (e.g. because too many people were in the I or H compartments and could not be vaccinated at the scheduled time) that we postpone to a later date. A value of 0 means we do not catch up at all on any missed doses; a value of 1 means we try to catch up for all missed doses. This is set to 1 by default

n_doses
: Number of doses given out, including boosters. Default is 2.

vacc_skip_progression_rate
: A vector of length equal to the number of vaccine strata. Represents the base progression rate for "vaccine skip moves" from each strata. The rate is assumed to be the same across all age groups. Must be 0 for all `j` for which `vacc_skip[j] = 0`. Default is a vector of 0s

vacc_skip_to
: A vector of integers of length equal to the number of vaccine strata. The value of `vacc_skip_to[j]` represents the vaccine stratum an individual in vaccine stratum `j` can move to in a "vaccine skip move". Values can be 0 (which mean there is no vaccine skip move from that stratum), or greater than or equal to `j + 2` (we already account for moves between successive vaccine strata). More than one vaccine skip move to the same vaccine stratum is not allowed, so all non-zero values in `vacc_skip_to` must be unique. Overlapping vaccine skip moves are also not allowed - if a there is a vaccine skip move from `i` to `j` there can be no vaccine skip moves from stratum `k` where `i < k < j`. For example, having vaccine skip moves from 2 to 4 and 3 to 5 would not be allowed, but moves from 2 to 4 and 4 to 6 would be allowed. Default is a vector of 0s

vacc_skip_weight
: A vector of weights of length equal to the number of vaccine strata. If movement into `vacc_skip_to[j]` from `vacc_skip_to[j] - 1` is controlled by doses, then the "vaccine skip move" is too, and the weight represents how much those in strata `j` are weighted in dose distribution relative to those in `vacc_skip_to - 1`. Must be between 0 and 1. Must be 0 for any `j` for which `vacc_skip_to = 0`. Default is a vector of 0s

waning_rate
: A single value or a vector of values representing the rates of waning of immunity after infection; if a single value the same rate is used for all age groups; if a vector of values if used it should have one value per age group.

exp_noise
: Rate of exponential noise used in the compare function - typically set to a large value so that noise is small but non-zero. If set to `Inf` then there is no noise in the observation process (not realistic but useful for testing).

cross_immunity
: A value or vector of same length as `strain_transmission` that controls the amount of immunity conferred by previous infection with one strain. If a scalar is given then same level of cross immunity is assumed between both strains. Otherwise a vector of length two should be provided where the first value is the relative protection against infection with strain 2 following infection with strain

1 (i.e. while in the R1 compartment), and vice versa for the second value. Values between 0 and 1 are allowed with values of 1 (default) indicating complete cross-immunity, and values of 0 mean no cross-immunity. Modelling 'superinfections' (being exposed to one strain after recovering from another) can be turned off by setting `cross_immunity = 1`.

Value

A list of inputs to the model, many of which are fixed and represent data. These correspond largely to `user()` calls within the odin code, though some are also used in processing just before the model is run.

Examples

```
region <- "london"
lancelot_parameters(sircovid_date("2020-02-01"), region)

# example set up of vaccination parameters independent of age
# 3 groups: 1) unvaccinated, 2) vaccinated with partial immunity
# 3) fully vaccinated (but with an imperfect vaccine). People return
# to group 1 after a period of time in group 3 to mirror waning immunity

# Assumption: immediately after vaccination susceptibility is reduced by
# 20%, and then by 50% when you reach full effect of the vaccination,
# then susceptibility returns to 100% upon waning of vaccine=induced
# immunity
# effect of vaccination similar across all age groups
rel_susceptibility <- c(1, 0.8, 0.5)

# The vaccine also reduces the risk of symptoms
rel_p_sympt <- c(1, 0.6, 0.3)
# and the risk of hospitalisation for those with symptoms
rel_p_hosp_if_sympt <- c(1, 0.95, 0.95)
# and the risk of death for those with severe disease
rel_p_death <- c(1, 0.9, 0.9)

# The vaccine also reduces infectivity of infected individuals by half
rel_infectivity <- c(1, 0.5, 0.5)

# On average 10000 first doses of vaccine are given every day
# Second doses are given on average 12 weeks after the first dose
# vaccine-induced immunity wanes after a period which is exponentially
# distributed and lasts on average 26 weeks (half a year);
# they are similar across all age groups
vaccine_progression_rate <- c(0, 0, 1/(26*7))

daily_doses <- rep(10000, 365)
mean_days_between_doses <- 12 * 7
n <- vaccine_priority_population(region, uptake = 1)
schedule <- vaccine_schedule_future(
  0, daily_doses, mean_days_between_doses, n)
```

```
# generate model parameters
p <- lancelot_parameters(
       sircovid_date("2020-02-01"), region,
       rel_susceptibility = rel_susceptibility,
       rel_p_sympt = rel_p_sympt,
       rel_p_hosp_if_sympt = rel_p_hosp_if_sympt,
       rel_p_death = rel_p_death,
       rel_infectivity = rel_infectivity,
       vaccine_progression_rate = vaccine_progression_rate,
       vaccine_schedule = schedule,
       vaccine_index_dose2 = 2)

# vaccination parameters are automatically copied across all age groups#
# (and across strains but here we only have 1 strain which is the 2nd
# dimension here)
p$rel_susceptibility
p$rel_p_sympt
p$rel_p_hosp_if_sympt
p$rel_p_death
p$rel_infectivity
# Note that this is only the "base" rate as we fill in the first
# column dynamically based on vaccine_daily_doses
p$vaccine_progression_rate

### same example as above BUT assume a different effect of vaccine in the
### first age group
n_groups <- 19
n_strains <- 1

# Assumption: vaccine is twice more effective at reducing susceptibility
# in the first age group
rel_susceptibility_agegp1 <- c(1, 0.4, 0.25)
rel_susceptibility_other_agegp <- c(1, 0.8, 0.5)
rel_susceptibility <- array(NA, dim = c(n_groups, n_strains, 3))
rel_susceptibility[1, , ] <- rel_susceptibility_agegp1
for (i in seq(2, n_groups)) {
  rel_susceptibility[i, , ] <- rel_susceptibility_other_agegp
}
rel_susceptibility

# But vaccine has the same impact on probability of symptoms and
# hospitalisation for the symptomatic across all age groups
rel_p_sympt <- array(rep(rel_p_sympt, each = n_groups),
  dim = c(n_groups, n_strains, 3))
rel_p_hosp_if_sympt <-
  array(rep(rel_p_hosp_if_sympt, each = n_groups),
  dim = c(n_groups, n_strains, 3))
rel_p_death <-
  array(rep(rel_p_hosp_if_sympt, each = n_groups),
  dim = c(n_groups, n_strains, 3))

# And vaccine has the same impact on onwards infectivity across age groups
rel_infectivity <- array(rep(rel_infectivity, each = n_groups),
```

```
  dim = c(n_groups, n_strains, 3))

# the period of build-up of immunity is the same for all age groups,
# lasting on average 2 weeks,
# but the first age group loses immunity more quickly
# (on average after 3 months) than the other age groups
# (on average after 6 months)
vaccine_progression_rate <- cbind(0, 0,
                                  c(1 / (13 * 7),
                                  rep( 1 / (26 * 7), n_groups - 1)))

# generate model parameters
p <- lancelot_parameters(
       sircovid_date("2020-02-01"), region,
       rel_susceptibility = rel_susceptibility,
       rel_p_sympt = rel_p_sympt,
       rel_p_hosp_if_sympt = rel_p_hosp_if_sympt,
       rel_p_death = rel_p_death,
       rel_infectivity = rel_infectivity,
       vaccine_progression_rate = vaccine_progression_rate,
       vaccine_schedule = schedule,
       vaccine_index_dose2 = 2)

# TODO: add an example of manually set up vaccine schedule
```

`lancelot_parameters_observation` *Lancelot observation parameters*

Description

Lancelot observation parameters

Usage

```
lancelot_parameters_observation(exp_noise = 1e+06)
```

Arguments

`exp_noise`	Rate parameter for the exponentially-distributed noise used in the likelihood calculation. Typically a large value so noise is small.

Value

A list of parameter values

`lancelot_parameters_progression` *Lancelot progression parameters*

Description

Lancelot progression parameters. The `s_` parameters are the scaling parameters for the Erlang distibution (a.k.a 'k'), while the `gamma_` parameters are the gamma parameters of that distribution. These need to be aligned with Bob's severity outputs, and we will come up with a better way of coordinating the two.

Usage

```
lancelot_parameters_progression(
  dt,
  gamma_E = NULL,
  gamma_A = NULL,
  gamma_P = NULL,
  gamma_C_1 = NULL,
  gamma_C_2 = NULL,
  gamma_ICU_pre = NULL,
  gamma_ICU_D = NULL,
  gamma_ICU_W_D = NULL,
  gamma_ICU_W_R = NULL,
  gamma_H_D = NULL,
  gamma_H_R = NULL,
  gamma_W_D = NULL,
  gamma_W_R = NULL,
  gamma_G_D = NULL,
  gamma_U = NULL,
  gamma_PCR_pre = NULL,
  gamma_PCR_pos = NULL
)
```

Arguments

`dt`	The step size
`gamma_E`	Time-varying parameters for the Erlang rate parameter of the duration in the E (exposed) compartment. See Details.
`gamma_A`	Time-varying parameters for the Erlang rate parameter of the duration in the I_A (asymptomatic) compartment. See Details.
`gamma_P`	Time-varying parameters for the Erlang rate parameter of the duration in the I_P (presymptomatic) compartment. See Details.
`gamma_C_1`	Time-varying parameters for the Erlang rate parameter of the duration in the I_C_1 (first stage symptomatic) compartment. See Details.

gamma_C_2 Time-varying parameters for the Erlang rate parameter of the duration in the I_C_2 (second stage symptomatic) compartment. See Details.

gamma_ICU_pre Time-varying parameters for the Erlang rate parameter of the duration in the ICU_pre (general beds stay before ICU) compartment. See Details.

gamma_ICU_D Time-varying parameters for the Erlang rate parameter of the duration in the ICU_D (ICU stay for individuals who die in ICU) compartment. See Details.

gamma_ICU_W_D Time-varying parameters for the Erlang rate parameter of the duration in the ICU_W_D (ICU stay for individuals who go on to die in stepdown) compartment. See Details.

gamma_ICU_W_R Time-varying parameters for the Erlang rate parameter of the duration in the ICU_W_R (ICU stay for individuals who go on to recover in stepdown) compartment. See Details.

gamma_H_D Time-varying parameters for the Erlang rate parameter of the duration in the H_D (general beds stay for individuals who die in general beds) compartment. See Details.

gamma_H_R Time-varying parameters for the Erlang rate parameter of the duration in the H_R (general beds stay for individuals who recover in general beds) compartment. See Details.

gamma_W_D Time-varying parameters for the Erlang rate parameter of the duration in the W_D (stepdown for individuals who die in stepdown) compartment. See Details.

gamma_W_R Time-varying parameters for the Erlang rate parameter of the duration in the W_R (stepdown for individuals who recover in stepdown) compartment. See Details.

gamma_G_D Time-varying parameters for the Erlang rate parameter of the duration in the G_D (delay to death in the community/care homes) compartment. See Details.

gamma_U Time-varying parameters for the Erlang rate parameter of the duration of the delay from hospital admission to diagnosis for those not confirmed on admission. Note this duration is a single-stage Erlang. See Details.

gamma_PCR_pre Time-varying parameters for the Erlang rate parameter the duration of PCR prepositivity affecting REACT and ONS infection prevalence. See Details.

gamma_PCR_pos Time-varying parameters for the Erlang rate parameter the duration of PCR positivity affecting REACT and ONS infection prevalence. See Details.

Value

A list of parameter values

Time-varying parameters

Every time varying parameter has the same format, which can be `NULL` (in which case a default single value is used) or a list with `date` and `value` for changes in the parameter. If `value` is scalar then `date` can be `NULL` or missing. If `value` is a vector then `date` must be a vector of sircovid dates of the same length as `value`.

`lancelot_parameters_sens_and_spec` *Lancelot sensitivity and specificity parameters*

Description

Lancelot observation parameters

Usage

```
lancelot_parameters_sens_and_spec(
  sero_specificity_1 = 0.9,
  sero_sensitivity_1 = 0.99,
  sero_specificity_2 = 0.9,
  sero_sensitivity_2 = 0.99,
  pillar2_specificity = 0.99,
  pillar2_sensitivity = 0.99,
  ons_specificity = 0.99,
  ons_sensitivity = 0.99,
  react_specificity = 0.99,
  react_sensitivity = 0.99
)
```

Arguments

`sero_specificity_1`
: Specificity of the first serology test assay

`sero_sensitivity_1`
: Sensitivity of the first serology test assay

`sero_specificity_2`
: Specificity of the second serology test assay

`sero_sensitivity_2`
: Sensitivity of the second serology test assay

`pillar2_specificity`
: Specificity of the Pillar 2 test

`pillar2_sensitivity`
: Sensitivity of the Pillar 2 test

`ons_specificity`
: Specificity of the ONS test

`ons_sensitivity`
: Sensitivity of the ONS test

`react_specificity`
: Specificity of the REACT test

`react_sensitivity`
: Sensitivity of the REACT test

Value

A list of parameter values

`lancelot_parameters_severity` *Lancelot severity parameters*

Description

Lancelot severity parameters

Usage

```
lancelot_parameters_severity(
  dt,
  severity = NULL,
  has_carehomes = TRUE,
  p_C = NULL,
  p_H = NULL,
  p_H_CHR = NULL,
  p_ICU = NULL,
  p_H_D = NULL,
  p_ICU_D = NULL,
  p_W_D = NULL,
  p_G_D = NULL,
  p_G_D_CHR = NULL,
  p_R = NULL,
  p_star = NULL
)
```

Arguments

`dt`	The step size
`severity`	Severity data, used to determine default severity parameter age-scalings and to provide default severity parameter values. Can be `NULL` (use the default bundled data version in the package), or a data.frame object (for raw severity data).
`has_carehomes`	Logical, whether or not the model has carehomes
`p_C`	Time-varying parameters for p_C (the probability of an infected individual becoming symptomatic). See Details.
`p_H`	Time-varying parameters for p_H (the probability of a symptomatic individual requiring hospitalisation). See Details.
`p_H_CHR`	Time-varying parameters for p_H (the probability of a symptomatic individual requiring hospitalisation) for care home residents. If `NULL` then the value for the oldest age group is used. See Details.

p_ICU	Time-varying parameters for p_ICU (the probability of a hospitalised individual going to ICU). See Details.
p_H_D	Time-varying parameters for p_H_D (the probability of death in general beds).See Details.
p_ICU_D	Time-varying parameters for p_ICU_D (the probability of death in ICU). See Details.
p_W_D	Time-varying parameters for p_W_D (the probability of death in stepdown). See Details.
p_G_D	Time-varying parameters for p_G_D (the probability of individuals requiring hospitalisation dying in the community or a care home). See Details.
p_G_D_CHR	Time-varying parameters for p_G_D (the probability of individuals requiring hospitalisation dying in a care home) for care home residents. If NULL then the value for the oldest age group is used. See Details.
p_R	Time-varying parameters for p_R (the probability of an non- fatally infected individual having immunity post-infection). See Details.
p_star	Time-varying parameters for p_star (the probability of patients being confirmed as covid on admission to hospital). See Details.

Value

A list of severity parameters

Time-varying parameters

Every time varying parameter has the same format, which can be NULL (in which case the value from severity is used) or a list with date and value for changes in the parameter. If value is scalar then date can be NULL or missing. If value is a vector then date must be a vector of sircovid dates of the same length as value.

---

lancelot_Rt *Compute "Rt"*

Description

Compute "Rt" for a single simulated trajectory and parameter set.

Usage

```
lancelot_Rt(
  time,
  S,
  p,
  prob_strain = NULL,
  type = NULL,
  interpolate_every = NULL,
```

```
  interpolate_critical_dates = NULL,
  interpolate_min = NULL,
  eigen_method = "power_iteration",
  R = NULL,
  weight_Rt = FALSE,
  keep_strains_Rt = FALSE
)
```

Arguments

time	A vector of time steps that the model was run over
S	A (n groups x n vaccine classes) x steps matrix of "S" compartment counts
p	A lancelot_parameters() object
prob_strain	A (n groups x n strains) x n time steps matrix of "prob_strain" outputs from the model. For a 2 strain model for example, prob_strain[1, j] and prob_strain[n_groups + 1, j] should give, for the j^th time step, the probabilities that new infections in group 1 are of strains 1 and 2 respectively. The default is NULL, but it must be specified if there is more than one strain
type	A character vector of possible Rt types to compute. Can be any or all of eff_Rt_all, eff_Rt_general, Rt_all and Rt_general
interpolate_every	Spacing (in days) to use between interpolated points
interpolate_critical_dates	Optional vector of critical sircovid dates to use when interpolating. Interpolation will be done in blocks between the first time step of these dates, with each block starting on the first time step of the date given. So if you give a interpolate_critical_dates of c(20, 50) then blocks *start* on first time step of days 20 and 50, i.e.: [1, 20), [20, 50), [50, end].
interpolate_min	The minimum number of steps to include within a block. If there are fewer points than this then all points are used (i.e., no interpolation is done) or interpolate_every is reduced until at least this many points were used. This can be used to specify a lower bound on the error of small regions. If Rt is small it won't matter that much. You do need to specify something though or interpolation will not happen, and do not use less than 3 as we use spline interpolation and that will not work with fewer than 3 points.
eigen_method	The eigenvalue method to use (passed to eigen1::eigen1 as method)
R	A (n groups x n strains x n vaccine classes) x time steps matrix of "R" compartment counts, required for multi-strain models.
weight_Rt	If TRUE then computes the weighted average of the Rt for all strains, otherwise all calculations are returned with an additional dimension to index each strain.
keep_strains_Rt	Additional argument for when weight_Rt is TRUE (has no impact otherwise). If TRUE, then the Rt for each strain is returned along with the weighted average, otherwise just the weighted average is returned. When TRUE, the dimension indexing these lists strains first, and then the weighted average.

Value

A list with elements `time`, `beta`, and any of the `type` values specified above.

`lancelot_Rt_trajectories` *Compute Rt for a set of trajectories*

Description

Compute "Rt" for a set of simulated trajectories (e.g., the result of the `$iterate()` method of lancelot, `mcstate::pmcmc()` or `mcstate::pmcmc_predict()`. The trajectories may or may not share parameters.

Usage

```
lancelot_Rt_trajectories(
  time,
  S,
  pars,
  prob_strain = NULL,
  initial_time_from_parameters = TRUE,
  shared_parameters = NULL,
  type = NULL,
  interpolate_every = NULL,
  interpolate_critical_dates = NULL,
  interpolate_min = NULL,
  eigen_method = "power_iteration",
  R = NULL,
  weight_Rt = FALSE,
  keep_strains_Rt = FALSE
)
```

Arguments

`time`	A vector of time steps
`S`	A 3d ((n groups x n vaccine classes) x n trajectories x n time steps) array of "S" compartment counts
`pars`	Either a single `lancelot_parameters()` object (shared parameters) or an unnamed list of `lancelot_parameters()` objects, the same length as `ncol(S)`.
`prob_strain`	A 3d ((n groups x n strains) x n trajectories x n time steps) array of "prob_strain" model outputs. Default is `NULL`, but it must be specified if there is more than one strain.
`initial_time_from_parameters`	If `TRUE`, then `time[[1]]` is replaced by the value of `initial_time` from the parameters. This is usually what you want. (From sircovid 0.12.13 this parameter means "initial time is zero" and will probably be updated in a future version).

`shared_parameters`
: Should `pars` be treated as a single shared list? Leave as `NULL` to detect automatically, set to `TRUE` or `FALSE` to force it to be interpreted one way or the other which may give more easily interpretable error messages.

`type`
: A character vector of possible Rt types to compute. Can be any or all of `eff_Rt_all`, `eff_Rt_general`, `Rt_all` and `Rt_general`

`interpolate_every`
: Spacing (in days) to use between interpolated points

`interpolate_critical_dates`
: Optional vector of critical sircovid dates to use when interpolating. Interpolation will be done in blocks between the first time step of these dates, with each block starting on the first time step of the date given. So if you give a `interpolate_critical_dates` of `c(20, 50)` then blocks *start* on first time step of days 20 and 50, i.e.: `[1, 20)`, `[20, 50)`, `[50, end]`.

`interpolate_min`
: The minimum number of steps to include within a block. If there are fewer points than this then all points are used (i.e., no interpolation is done) or `interpolate_every` is reduced until at least this many points were used. This can be used to specify a lower bound on the error of small regions. If Rt is small it won't matter that much. You do need to specify something though or interpolation will not happen, and do not use less than 3 as we use spline interpolation and that will not work with fewer than 3 points.

`eigen_method`
: The eigenvalue method to use (passed to eigen1::eigen1 as `method`)

`R`
: A 3d ((n groups x n strains x n vaccine classes) x n trajectories x n time steps) array of "R" compartment counts, required for multi-strain models.

`weight_Rt`
: If `TRUE` then computes the weighted average of the Rt for all strains, otherwise all calculations are returned with an additional dimension to index each strain.

`keep_strains_Rt`
: Additional argument for when `weight_Rt` is `TRUE` (has no impact otherwise). If `TRUE`, then the Rt for each strain is returned along with the weighted average, otherwise just the weighted average is returned. When `TRUE`, the dimension indexing these lists strains first, and then the weighted average.

Value

As for `lancelot_Rt()`, except that every element is a matrix, not a vector.

`lancelot_rt_trajectories_epiestim`
Compute Rt using EpiEstim for a set of trajectories

Description

Compute "Rt" using EpiEtim for a set of simulated trajectories (e.g., the result of the `$iterate()` method of lancelot, `mcstate::pmcmc()` or `mcstate::pmcmc_predict()`. The trajectories should share parameters.

Usage

```
lancelot_rt_trajectories_epiestim(
  step,
  incidence,
  p,
  gt_distr = NULL,
  sliding_window_ndays = 7,
  mean_prior = 1,
  sd_prior = 1,
  n_GT = 10000,
  n_R = 1000,
  save_all_Rt_sample = TRUE,
  q = NULL
)
```

Arguments

step	A vector of steps
incidence	A matrix (n trajectories x n steps) of incidence counts
p	A single lancelot_parameters() object.
gt_distr	A vector giving the discrete (daily) distribution of the generation time. The first value must be zero. If NULL, the generation time distribution will be determined automatically from p.
sliding_window_ndays	An integer giving the length of the sliding window on which Rt will be estimated
mean_prior	The mean prior for Rt
sd_prior	The standard deviation of the prior for Rt
n_GT	An integer giving the number of generation times to be drawn to construct the discrete distribution of the generation time
n_R	An integer giving the number of Rt values to sample from for each incidence trajectory. These will then be aggregated across all incidence trajectories.
save_all_Rt_sample	A boolean determining whether to save all samples of Rt estimated or only a summary
q	A vector of quantiles to return values for

Value

A list with elements t_start (vector of first days of the sliding windows over which Rt is estimated), t_end (vector of last days of the sliding windows over which Rt is estimated), Rt a matrix (only present if save_all = TRUE) containing for each sliding window (each col in the matrix) a sample of n_R * nrow(inc) values of Rt for that sliding window (rows of the matrix) Rt_summary a matrix containing for each sliding window (each col in the matrix) the quantiles (q) and the mean of Rt for that sliding window (rows of the matrix) gt_distr a vector giving the discrete (daily) distribution of the generation time

modify_severity *Modify severity and transmission of variants*

Description

Applies a modifier to severity and transmission parameters

Usage

```
modify_severity(efficacy, efficacy_strain_2, strain_severity_modifier)
```

Arguments

`efficacy`, `efficacy_strain_2`
: Vaccine efficacy parameters for strains 1 and 2 respectively. Expects a list with names rel_susceptibility, rel_p_sympt, rel_p_hosp_if_sympt, rel_infectivity, rel_p_death. Element columns correspond to vaccine strata and rows to age groups.

`strain_severity_modifier`
: List of modifiers to be applied to efficacy variables; length should correspond to number of strains and each element should be a list with same names as `efficacy`

Value

Returns a list with same length and names as `efficacy` and where each element has dimensions n_groups x n_strains x n_vacc_strata

regions *Regions*

Description

Generate lists of regions that we use for various tasks.

Usage

```
regions(type)
```

Arguments

`type`	The name of a region type; must be one of "all", "england", or "nations"

Value

A character vector

Examples

```
sircovid::regions("england")
```

`reorder_rt_ifr` *Reorder Rt or IFR trajectories*

Description

Reorder Rt or IFR trajectories

Usage

```
reorder_rt_ifr(x, rank)
```

Arguments

`x`	An `Rt_trajectories` object, as returned by `lancelot_Rt_trajectories()`
`rank`	A vector of ranks to reorder by

Value

An `Rt_trajectories` object with appropriately reordered elements

`reorder_sample` *Reorder samples*

Description

Reorder samples according to a predefined ranking

Usage

```
reorder_sample(sample, rank)
```

Arguments

`sample`	An `mcstate_pmcmc` object
`rank`	A vector of ranks to reorder by

Value

An `mcstate_pmcmc` object with appropriately reordered elements

rotate_strains *Rotate strains*

Description

Rotate strains, so that strain 1 becomes the sum of strains 1 and 2 and strain 2 is empty. Use this to allow sequential replacement of strains.

Usage

```
rotate_strains(state, info, ...)
```

Arguments

state	Model state
info	Model info
...	Additional arguments, ignored. This exists so that this function can be used as an argument to mcstate::multistage_epoch. Practically your two model informations in this case would be equivalent.

sircovid_date *Date handling for sircovid*

Description

We need to map "dates" onto dust::dust's concept of model "step" and we do this by mapping a date such as 2020-03-02 into the number of days into 2020 (62 here, with the 1st of January being day 1). We call this integer number a "sircovid date".

Usage

```
sircovid_date(date)

sircovid_date_as_date(date)

as_sircovid_date(date)

as_date(date)
```

Arguments

date	A Date object, or something that can be converted to one, or a "sircovid date"; see Details

Details

There are several related functions here

- `sircovid_date` converts its argument into an R `Date` object, then applies this tranformation. If the argument is not a `Date` object or a string representing one, an error will be thrown.
- `sircovid_date_to_date` does the reverse conversion to `sircovid_date`, converting an integer sircovid date into an R `Date`
- `as_sircovid_date` does the same conversion as `sircovid_date` but will assume that an integer *already* represents a sircovid date and will return it unmodified rather than erroring.
- `as_date` does a string to date conversion, using `as.Date()` but requiring the dates are in ISO 8601 (YYYY-MM-DD) format (it is a helper that avoids conversion to `NA`, instead throwing an error)

Value

An integer, being the number of days into 2020

Examples

```
# Convert dates into sircovid dates:
sircovid::sircovid_date("2020-01-01")
sircovid::sircovid_date(c("2020-03-01", "2020-10-01"))

# Reverse the conversion:
sircovid::sircovid_date_as_date(1)
sircovid::sircovid_date_as_date(c(61, 275))

# Double conversion not possible with sircovid_date...
try(sircovid::sircovid_date(61))
# ...but allowed with as_sircovid_date
sircovid::as_sircovid_date(61)

# Strict date conversion with as_date
sircovid::as_date("2020-03-01")
try(sircovid::as_date("03-01-2020"))
```

`sircovid_models` *Available sircovid Models*

Description

List the models available in this package

Usage

```
sircovid_models()
```

sircovid_parameters_expand_step
Expand beta steps

Description

Expand `value_step` based on a series of `steps`. Use this to convert between the values passed to `sircovid_parameters_piecewise_linear()` and the actual values for a given set of steps.

Usage

```
sircovid_parameters_expand_step(step, value_step)
```

Arguments

`step`	A vector of steps
`value_step`	A vector of values

Value

A numeric vector the same length as `step`

sircovid_parameters_piecewise_constant
Construct piecewise constant array

Description

Construct a piecewise constant quantity over time array for use within sircovid models.

Usage

```
sircovid_parameters_piecewise_constant(date, value, dt)
```

Arguments

`date`	Either `NULL`, if one value of the quantity will be used for all time steps, or a vector of times that will be used as change points. Must be provided as a `sircovid_date()`, i.e., days into 2020. The first date must be 0.
`value`	A vector of values to use for the quantity - either a scalar (if `date` is `NULL`) or a vector the same length as `date`.
`dt`	The timestep that will be used in the simulation. This must be of the form `1 / n` where `n` is an integer representing the number of steps per day. Ordinarily this is set by sircovid internally to be `0.25` but this will become tuneable in a future version.

Value

Returns a vector of piecewise constant values, one per timestep, until the values stabilise. After this point the quantity is assumed to be constant.

Examples

```
# If "date" is NULL, then the quantity is constant and this function is
# trivial:
sircovid::sircovid_parameters_piecewise_constant(NULL, 0.1, 0.25)

date <- sircovid::sircovid_date(
   c("2019-12-31", "2020-02-01", "2020-02-14", "2020-03-15"))
value <- c(0, 3, 1, 2)
y <- sircovid::sircovid_parameters_piecewise_constant(date, value, 1)

# The implied time series looks like this:
t <- seq(0, date[[4]])
plot(t, y, type = "o")
points(date, value, pch = 19, col = "red")

# After 2020-03-15, the quantity value will be fixed at 2, the value
# that it reached at that date.

# You can see this using sircovid_parameters_expand_step
# If a vector of dates is provided then, it's more complex. We'll
# use dt of 1 here as it's easier to visualise
t <- seq(0, 100, by = 1)
sircovid::sircovid_parameters_expand_step(t, y)
plot(t, sircovid::sircovid_parameters_expand_step(t, y), type = "o")
points(date, value, pch = 19, col = "red")

# If dt is less than 1, this is scaled, but the pattern of
# change is the same
y <- sircovid::sircovid_parameters_piecewise_constant(date, value, 0.5)
t <- seq(0, date[[4]], by = 0.5)
plot(t, y, type = "o", cex = 0.25)
points(date, value, pch = 19, col = "red")
```

---

sircovid_parameters_piecewise_linear

Construct piecewise linear array

Description

Construct a piecewise linear quantity over time array for use within sircovid models.

Usage

```
sircovid_parameters_piecewise_linear(date, value, dt)
```

Arguments

date	Either NULL, if one value of the quantity will be used for all time steps, or a vector of times that will be used as change points. Must be provided as a sircovid_date(), i.e., days into 2020. The first date must be 0.
value	A vector of values to use for the quantity - either a scalar (if date is NULL) or a vector the same length as date.
dt	The timestep that will be used in the simulation. This must be of the form 1 / n where n is an integer representing the number of steps per day. Ordinarily this is set by sircovid internally to be 0.25 but this will become tuneable in a future version.

Value

Returns a vector of piecewise linear values, one per timestep, until the values stabilise. After this point the quantity is assumed to be constant.

See Also

sircovid_parameters_expand_step() - see examples below

Examples

```
# If "date" is NULL, then the quantity is constant and this function is
# trivial:
sircovid::sircovid_parameters_piecewise_linear(NULL, 0.1, 0.25)

date <- sircovid::sircovid_date(
   c("2020-02-01", "2020-02-14", "2020-03-15"))
value <- c(3, 1, 2)
y <- sircovid::sircovid_parameters_piecewise_linear(date, value, 1)

# The implied time series looks like this:
t <- seq(0, date[[3]])
plot(t, y, type = "o")
points(date, value, pch = 19, col = "red")

# After 2020-03-15, the quantity value will be fixed at 2, the value
# that it reached at that date.

# You can see this using sircovid_parameters_expand_step
# If a vector of dates is provided then, it's more complex. We'll
# use dt of 1 here as it's easier to visualise
t <- seq(0, 100, by = 1)
sircovid::sircovid_parameters_expand_step(t, y)
plot(t, sircovid::sircovid_parameters_expand_step(t, y), type = "o")
points(date, value, pch = 19, col = "red")

# If dt is less than 1, this is scaled, but the pattern of
# change is the same
y <- sircovid::sircovid_parameters_piecewise_linear(date, value, 0.5)
```

```
t <- seq(0, date[[3]], by = 0.5)
plot(t, y, type = "o", cex = 0.25)
points(date, value, pch = 19, col = "red")
```

`sircovid_parameters_severity` *Process severity data*

Description

Process severity data

Usage

```
sircovid_parameters_severity(params)
```

Arguments

`params`	Severity data, via Bob Verity's `markovid` package. This needs to be `NULL` (use the default bundled data version in the package), a data.frame object (for raw severity data) or a list (for data that has already been processed by `sircovid` for use). New severity data comes from Bob Verity via the markovid package, and needs to be carefully calibrated with the progression parameters.

Value

A list of age-structured probabilities

`upgrade_state` *Upgrade model state after change*

Description

Upgrade model state

Usage

```
upgrade_state(state_orig, info_orig, info_new, allowed = NULL)
```

Arguments

`state_orig`	Old model info
`info_orig`	Old model info
`info_new`	New model info
`allowed`	Character vector of allowed states to add; new values will be initialised as zeros.

Value

A new copy of the state

---

vaccine_eligibility *Create vaccine eligibility vector*

Description

Create a vector of vaccine eligibility, based on a minimum age. The assumption is that everyone below that age is ineligible and everyone including and above is eligibility. This will practically be modified further by assumptions of uptake. Note that carehome residents and workers are **always** eligible for vaccination, regardless of the minimum age.

Usage

```
vaccine_eligibility(min_age)
```

Arguments

min_age	Minimum age to be eligible for vaccination (includes this age).

Value

A vector of length 19

---

vaccine_priority_proportion *Compute vaccination order*

Description

Compute vaccination priority following JCVI ordering.

Usage

```
vaccine_priority_proportion(
  uptake,
  prop_hcw = NULL,
  prop_very_vulnerable = NULL,
  prop_underlying_condition = NULL
)

vaccine_priority_population(
  region,
  uptake,
```

```
  prop_hcw = NULL,
  prop_very_vulnerable = NULL,
  prop_underlying_condition = NULL,
  carehomes = TRUE
)
```

Arguments

uptake	A vector of length 19 with fractional uptake per group. If a single number is given it is shared across all groups (note that this includes under-18s)
prop_hcw	Assumed fraction of healthcare workers in each group (length 19) - if NULL we use a default that is a guess with hopefully the right general shape.
prop_very_vulnerable	Assumed fraction "very vulnerable" in each group (length 19) - if NULL we use a default that is a guess with hopefully the right general shape.
prop_underlying_condition	Assumed fraction "underlying condition" in each group (length 19) - if NULL we use a default that is a guess with hopefully the right general shape.
region	Region to use to get total population numbers
carehomes	Logical parameter, whether or not we have carehomes in the model. Default is TRUE

Details

https://tinyurl.com/8uwtatvm

Assmuing independance between job (e.g. HCW) and clinical condition

But assuming one is either counted as "clinically extremely vulnerable" or "with underlying health conditions" but not both

The JCVI priority groups, in decending order are:

1. residents in a care home for older adults and their carers
2. all those 80 years of age and over and frontline health and social care workers
3. all those 75 years of age and over
4. all those 70 years of age and over and clinically extremely vulnerable individuals
5. all those 65 years of age and over
6. all individuals aged 16 years to 64 years with underlying health conditions which put them at higher risk of serious disease and mortality
7. all those 60 years of age and over
8. all those 55 years of age and over
9. all those 50 years of age and over
10. all those 40-49 years of age and over
11. all those 30-39 years of age and over
12. all those 18-29 years of age and over

Value

A matrix with n_groups rows (19) and columns representing priority groups, and element (i, j) is the proportion in group i who should be vaccinated as part of priority group j, accounting for uptake so that the sum over the rows corresponds to the total fractional uptake in that group. For `vaccine_priority_population`, the total number of individuals replaces the proportion (based on the demography used by sircovid).

`vaccine_remap_state` *Remap state to include vaccination*

Description

Remap state that was run with one stratum to support vaccination. This is useful where fitting was done pre-vaccination and the model state needs to be expanded to support vaccination for simulations.

Usage

```
vaccine_remap_state(state_orig, info_orig, info_vacc)
```

Arguments

`state_orig`	An original state matrix (rows representing state, columns representing particles or samples). This must be the complete model state.
`info_orig`	The results of `info()` from the model object without vaccination
`info_vacc`	The results of `info()` from the model object with vaccination

Value

A 3d array with more rows than `state`.

`vaccine_schedule` *Create vaccine schedule*

Description

Create a vaccine schedule for use with lancelot_parameters

Usage

```
vaccine_schedule(date, doses, n_doses = 2L)
```

Arguments

date	A single date, representing the first day that vaccines will be given
doses	A 3d array of doses representing (1) the model group (19 rows for the lancelot model), (2) the dose (must be length 2 at present) and (3) time (can be anything nonzero). The values represent the number of vaccine doses in that group for that dose for that day. So for `doses[i, j, k]` then it is for the ith group, the number of jth doses on day `(k - 1) + date`
n_doses	The number of doses in the schedule. Typically (and by default) this will be 2, but if using booster doses 3 (and in future we may extend further).

Value

A `vaccine_schedule` object

---

vaccine_schedule_from_data

Create historical vaccine schedule

Description

Create a historical vaccine schedule from data

Usage

```
vaccine_schedule_from_data(data, region, uptake, carehomes = TRUE)
```

Arguments

data	A data.frame with columns `date`, `age_band_min`, and numbered doses columns, e.g. if there are three doses these should be `dose1`, `dose2` and `dose3`. Values of `age_band_min` should be either multiples of 5 or NA - the latter means those doses are not age-specific and will be distributed across all ages according to priority after all age-specific doses have already been dealt with
region	Region to use to get total population numbers
uptake	A matrix of 19 rows, and number of columns equal to number of doses. The (i,j)th entry gives the fractional uptake of dose j for group i. Should be non-increasing across rows
carehomes	Logical parameter, whether or not we have carehomes in the model. Default is TRUE

Value

A vaccine_schedule object

`vaccine_schedule_future` *Create vaccination schedule*

Description

Create future vaccination schedule from a projected number of daily doses.

Usage

```
vaccine_schedule_future(
  start,
  daily_doses_value,
  mean_days_between_doses,
  priority_population,
  lag_groups = NULL,
  lag_days = NULL,
  booster_daily_doses_value = NULL,
  booster_proportion = rep(1L, 19)
)
```

Arguments

`start`	Either a sircovid_date object corresponding to the first date in daily_doses_value, or a vaccine_schedule object corresponding to previously carried out vaccination.
`daily_doses_value`	A vector of doses per day.
`mean_days_between_doses`	Assumed mean days between doses one and two
`priority_population`	Output from vaccine_priority_population, giving the number of people to vaccinate in each age (row) and priority group (column)
`lag_groups`	Row indices, corresponding to age groups in which a lag should be added to the start time of the dose schedule returned by vaccine_schedule, if NULL then no lag is added. Ignored if `lag_groups` is NULL.
`lag_days`	If `lag_groups` is not NULL then specifies the number of days to add the start of the dose schedule for the given groups. Ignored if `lag_groups` is NULL.
`booster_daily_doses_value`	A vector of booster doses per day.
`booster_proportion`	Proportion of the groups in `priority_population` to boost, default is all groups; ignored if `booster_daily_doses_value` is NULL.

`vaccine_schedule_scenario` *High-level vaccine scenario creation*

Description

Create a vaccination scenario

Usage

```
vaccine_schedule_scenario(
  schedule_past,
  doses_future,
  end_date,
  mean_days_between_doses,
  priority_population,
  lag_groups = NULL,
  lag_days = NULL,
  boosters_future = NULL,
  boosters_prepend_zero = TRUE,
  booster_proportion = rep(1L, 19)
)
```

Arguments

`schedule_past`	A vaccine_schedule object corresponding to previously carried out vaccination.
`doses_future`	A named vector of vaccine doses to give in the future. Names must be in ISO date format.
`end_date`	The final day in the future to create a schedule for. After this date the model will assume 0 vaccine doses given so an overestimate is probably better than an underestimate.
`mean_days_between_doses`	Assumed mean days between doses one and two
`priority_population`	Output from vaccine_priority_population, giving the number of people to vaccinate in each age (row) and priority group (column)
`lag_groups`	Row indices, corresponding to age groups in which a lag should be added to the start time of the dose schedule returned by vaccine_schedule, if NULL then no lag is added. Ignored if `lag_groups` is NULL.
`lag_days`	If `lag_groups` is not NULL then specifies the number of days to add the start of the dose schedule for the given groups. Ignored if `lag_groups` is NULL.
`boosters_future`	Optional named vector of booster doses to give in the future. Names must be in ISO date format.

`boosters_prepend_zero`
: If TRUE (default) and `boosters_future` is not NULL then sets booster doses to zero before the first date in `boosters_future`. This is in contrast to when it is FALSE and the previous value in `schedule_past` is replicated until the first date in boosters_future. Note that this should rarely be FALSE as this will likely lead to duplicating daily doses that are already replicated in `doses_future`.

`booster_proportion`
: Proportion of the groups in `priority_population` to boost, default is all groups; ignored if `booster_daily_doses_value` is NULL.

Value

A vaccine_schedule object

Index