Package 'imuGAP'

June 22, 2026

Title Immunity: Geographic and Age-Based Projection

Version 0.1.0

Description Fits Bayesian hierarchical models of vaccine coverage by location, birth cohort, and age. Observations of vaccination status (which may span cohorts, ages, and doses) are used to fit a latent survival-style process model that decomposes coverage into a lifetime propensity to vaccinate and a time-varying force of vaccination. Hierarchical spatial structure (e.g., state, county, school) supports partial pooling via random effects. Models are implemented in 'Stan' and fit via 'rstan'. Provides helpers to validate user-supplied input data, and to predict coverage from fitted models.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Biarch true

Depends R (>= 4.1.0)

Imports data.table, methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.5.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

Suggests knitr, rmarkdown, testthat (>= 3.0.0), roxygen2, roxyglobals, lintr, spelling, bayesplot, ggplot2

VignetteBuilder knitr

Config/testthat/edition 3

URL https://accidda.github.io/imuGAP/, https://github.com/ACCIDDA/imuGAP

BugReports https://github.com/ACCIDDA/imuGAP/issues

LazyData true

LazyDataCompression xz

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Carl Pearson [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0701-7860>),
Claire Perrin Smith [aut] (ORCID:
<https://orcid.org/0000-0003-1069-9460>),
Kelly Zhen [ctb],
Weston Voglesonger [ctb],
Joshua Chen [ctb],
Minjae Kung [ctb]

Maintainer Carl Pearson <carl.ab.pearson@gmail.com>

Repository CRAN

Date/Publication 2026-06-22 14:40:26 UTC

Contents

`imuGAP-package` *The 'imuGAP' package.*

Description

A package for estimating measles vaccine coverage

Author(s)

Maintainer: Carl Pearson <carl.ab.pearson@gmail.com> (ORCID)

Authors:

- Carl Pearson <carl.ab.pearson@gmail.com> (ORCID)
- Claire Perrin Smith (ORCID)

Other contributors:

- Kelly Zhen <zhen717605@gmail.com> [contributor]
- Weston Voglesonger <westonvogle@gmail.com> [contributor]
- Joshua Chen <joshuazchen1@gmail.com> [contributor]
- Minjae Kung <minjaekung@gmail.com> [contributor]

References

Stan Development Team (NA). RStan: the R interface to Stan. R package version NA. https://mc-stan.org

See Also

Useful links:

- https://accidda.github.io/imuGAP/
- https://github.com/ACCIDDA/imuGAP
- Report bugs at https://github.com/ACCIDDA/imuGAP/issues

as.data.frame.imugap_predict *Convert coverage predictions to a data.frame*

Description

Converts the 3D draws array of an `imugap_predict` object into a long-format `data.frame` containing `iteration`, `chain`, target metadata, and a `coverage` column.

Usage

```
## S3 method for class 'imugap_predict'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	an `imugap_predict` object returned by `[predict()]`.
`row.names`	`NULL` or a character vector giving the row names for the data frame.
`optional`	logical. If `TRUE`, setting row names and converting column names is optional.
`...`	additional arguments (currently ignored).

Value

A `data.table` with columns `iteration`, `chain`, the target metadata columns, and `coverage`.

Examples

```
# Load example prediction object
data("predict_sim", package = "imuGAP")

# Convert predictions to a data.frame/data.table
df <- as.data.frame(predict_sim)
head(df)
```

canonicalize *Canonicalize imuGAP Data Objects*

Description

These functions validate, clean, and convert raw user-supplied data structures (locations, observations, and populations) into the canonical forms required by the `[sampling()]` sampler and the underlying Stan models.

Usage

```
canonicalize_locations(locations)

canonicalize_observations(observations, drop_extra = TRUE)

canonicalize_populations(
  populations,
  observations,
  locations,
  max_cohort,
  max_age,
  max_dose = 2L
)
```

Arguments

locations	a [data.frame()], with columns loc_id and parent_id, of the same type. See Details for restrictions.
observations	a [data.frame()], the observed data, with at least three columns: • an obs_id column; any type, as long as unique, non-NA • a positive column; non-negative integers, the observed number of vaccinated individuals • a sample_n column; positive integers, the number of individuals sampled, must be greater than or equal to "positive" • optionally, a censored column; numeric, NA (uncensored) or 1 (right-censored); if not present, will be assumed NA
drop_extra	a logical scalar; drop extraneous columns? (default: yes)
populations	a [data.frame()], the observation meta data, with columns • obs_id, any type; the observation the row concerns (i.e. id shared with an observations data object) • loc_id, any type; the location the row concerns (i.e. id shared with a locations data object) • dose, a non-zero, positive integer (1, 2, ...); what dose row concerns • cohort, a positive integer; the cohort at the location row concerns • age, a positive integer; the age of that cohort row concerns • weight, a numeric, (0, 1); the relative contribution of this row to an observation. Optional if each population row has a unique obs_id.
max_cohort	if present, what is the maximum cohort that should be present?
max_age	if present, what is the maximum age that should be present?
max_dose	maximum dose number to allow (default: 2L)

Details

The imuGAP hierarchical modeling framework requires data structures to adhere to specific relational and format constraints. The three canonicalize functions process and validate these inputs as described below:

Locations (`canonicalize_locations`):

The `[sampling()]` sampler works on a hierarchical model of locations, and must be provided that structure. This method checks location structure validity, and returns a canonical version including the layer membership.

A valid structure has:

- a unique root,
- no cycles, and
- no duplicate `loc_id`s

Users may explicitly identify the root `loc_id` by providing a row with `parent_id` equal to `NA`. Otherwise, any `parent_id` that does not appear in `loc_id` is treated as the root.

If the input is valid, this method will create the canonicalized version. In that version, all ids run from 1:N, where N is the number of distinct ids. That order is determined by layer order, then position of parent within its layer, then "natural" order (i.e., whatever base R `sort()` yields).

Observations (`canonicalize_observations`):

The observations object documents observations used to fit the model. Conceptually, each row represents an observation of vaccination status within a population. That population need not be uniform (see `[canonicalize_populations()]`) or concerning a single cohort or time: each observation should generally be the best available resolution data. That resolution can vary across rows. The sampler uses information about the resolutions to automatically figure out how to compare the latent process model to those different observations.

For the optional `censored` column: the model supports vaccination status indicators which are vaccine specific as well as those which represent an individual having all of a set of vaccines (including the target vaccine). The specific coverage for the target vaccine is right-censored in the latter case: full-set-coverage is the minimum coverage for the target.

When at least some of the data are censored, you must supply the `censored` column to correctly estimate coverage. Mark any uncensored observations with `NA`, and any right-censored observations with `1`. Note that `0` is *not* a valid value at this time; we are preserving that for potential future support of left-censoring.

Populations (`canonicalize_populations`):

This method validates the meta-data associated with the observations, as well as converting that meta-data to use the canonical id formats.

Regarding "cohorts" and "ages": these are counted from 1, by 1 "unit". You can imagine the units are whatever resolution is appropriate for your data: months, quarters, years, etc. As long as these are used consistently, estimation will work, and take on the unit meaning you used for input.

Value

`canonicalize_locations` returns a `data.table`, with:

- `loc_id`, `parent_id` columns as originally supplied, possibly reordered
- `loc_c_id`, `loc_cp_id` columns, canonicalized id/parent_id columns, representing the order that will be used in the sampler
- `layer` column, an integer from 1 (root), 2 (root children), 3 (grandchildren), &c
- `layer_bound` column, an integer starting from 1 by layer. This provides index slice information used in the stan model.

canonicalize_observations returns a canonical observation object, a [data.table()] with:

- an obs_c_id column, an integer sequence from 1; the order observations will be passed to estimation
- the original obs_id column, possibly reordered
- positive and sample_n columns, possibly reordered
- a "censored" column; all NA, if not present in original observations argument

canonicalize_populations returns a canonical populations object, mirroring the input populations, with the following updates:

- obs_c_id, the observation id the row concerns, canonicalized to match the canonical observation ids
- loc_c_id, the location id the row concerns, canonicalized to match
- reordered to obs_c_id order

Examples

```
# --- canonicalize_locations ---
data("locations_sim")
locations_sim
canonicalize_locations(locations_sim)
# can also be provided in non-canonical order, and with an implicit root
weird_locations <- subset(locations_sim, !is.na(parent_id))[
  sample(nrow(locations_sim) - 1L)
]
canonicalize_locations(weird_locations)
# --- canonicalize_observations ---
data("observations_sim")
observations_sim
canonicalize_observations(observations_sim)
# --- canonicalize_populations ---
data("populations_sim"); data("locations_sim"); data("observations_sim")
populations_sim
canonicalize_populations(populations_sim, observations_sim, locations_sim)
```

---

create_target *Create target population for prediction*

Description

Creates a target object appropriate for use with [predict.imugap_fit()], which will be used with a specific imugap_fit object.

Usage

```
create_target(
  fit,
  location,
  age,
  cohort,
  dose,
  mode = c("error", "enumerate", "recycle", "snapshot")
)
```

Arguments

fit	an imugap_fit object returned by [sampling()]
location	either a vector of locations or a data.frame; if a vector of locations, treated as the target locations; if a data.frame, then validated as the target object.
age	vector of ages for which to predict coverage, consistent with [canonicalize_populations()]
cohort	vector of cohorts for which to predict coverage, consistent with [canonicalize_populations()]
dose	vector of doses for which to predict coverage, consistent with [canonicalize_observations()]
mode	how location, age, cohort, and dose are combined into the target grid: "error" (default) requires equal-length vectors and pairs them rowwise; "enumerate" takes all combinations; "recycle" recycles to the least-common-multiple length; "snapshot" takes a single reference cohort and derives per-age cohorts holding age + cohort constant. See Details.

Details

When location is a data.frame, this function validates that object against the fit argument. Non-missing values for the other arguments are an error for that approach.

Otherwise, location must correspond to a vector of location IDs and age, cohort, and dose must also be supplied. Depending on the mode argument, these arguments may have different lengths. If mode = "error" (default), then all of these arguments must have the same length. If mode = "enumerate", then the resulting target will be all combinations of the arguments. If mode = "recycle", then the resulting target will recycle all the arguments out to the least-common-multiple length. If mode = "snapshot", then the cohort argument must be a single reference value (which represents the oldest cohort). The resulting target will enumerate combinations of locations, ages, and doses, calculating cohorts for each age such that the sum of age and cohort is constant (i.e., cohort_i = cohort_ref + max_age - age_i). This corresponds to a snapshot in time of different birth date and age combinations.

Value

A data.table representing the canonicalized target population.

Examples

```
# Load example fit object
data("fit_sim")
```

```
# 1. Default "error" mode: All vector inputs must have the same length.
create_target(
  fit = fit_sim,
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2),
  cohort = c(2, 3),
  dose = c(1, 1),
  mode = "error"
)

# Providing mismatched length arguments in "error" mode throws an error:
try(create_target(
  fit = fit_sim,
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1), # length mismatch
  cohort = c(2, 3),
  dose = c(1, 1),
  mode = "error"
))

# 2. "enumerate" mode: Generates all combinations of the arguments.
create_target(
  fit = fit_sim,
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2),
  cohort = c(2, 3),
  dose = c(1),
  mode = "enumerate"
)

# 3. "recycle" mode: Recycles arguments to the least-common-multiple length.
create_target(
  fit = fit_sim,
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2, 3),
  cohort = c(2),
  dose = c(1),
  mode = "recycle"
)

# 4. "snapshot" mode: Cohort is a single reference value. Cohorts for each
# age are calculated so that cohort + age is constant (representing a snapshot in time).
create_target(
  fit = fit_sim,
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2, 3),
  cohort = 5,
  dose = c(1),
  mode = "snapshot"
)

# 5. Providing a data.frame for validation.
```

```
df_target <- data.frame(
  loc_id = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2),
  cohort = c(2, 3),
  dose = c(1, 1)
)
create_target(fit = fit_sim, location = df_target)
```

`extract_imugap` *Custom imuGAP fit extraction*

Description

Thin wrapper around `rstan::extract` to extract typical imuGAP parameters.

Usage

```
extract_imugap(fit, pars = c("beta_bs"), ...)
```

Arguments

`fit`	an `imugap_fit` object returned by `sampling()`
`pars`	character vector; parameters to extract. Defaults to `"beta_bs"`, the state-level B-spline parameter.
`...`	additional arguments passed to [`rstan::extract()`].

Value

a list, as returned by `rstan::extract()`

Examples

```
data("fit_sim")
extract_imugap(fit_sim)
extract_imugap(fit_sim, pars = "lambda_raw")
```

fit_sim *Example Stan Fit*

Description

A reference `stanfit` object produced by running `imuGAP()` on the bundled `locations_sim`, `populations_sim`, and `observations_sim` datasets. Intended as a lightweight fixture for examples, tests, and downstream tooling that needs a real fit without paying the cost of recompiling or re-running the Stan model.

Usage

```
fit_sim
```

Format

A `stanfit` object as returned by `rstan::sampling()`.

Details

Generated with the same minimal sampler settings as the smoke test:

- `iter` = `100`
- `chains` = `1`
- `seed` = `1L`
- `refresh` = `0`

These settings are not enough for convergence; `fit_sim` is a wiring fixture, not a scientifically meaningful posterior. The generating script lives at `data-raw/fit_sim.R` and can be re-run from the package root with `Rscript data-raw/fit_sim.R`.

Note that `stanfit` objects bundle references to the compiled Stan model and can be sensitive to major version changes in `rstan` and `StanHeaders`. If a future install fails to load `fit_sim`, regenerate it via `data-raw/fit_sim.R`.

imugap_options *imuGAP Model Options*

Description

This function encapsulates option passing for imuGAP settings.

Usage

```
imugap_options(df = 5L, dose_schedule = c(1, 4), object = c("default"))
```

Arguments

df	degrees of freedom to use in bspline
dose_schedule	an integer vector, the ages at which dose(s) n are scheduled, with vector indices and doses matching
object	which stan model object to use; currently only "default" is supported

Value

a list of imuGAP model options

Examples

```
imugap_options()
imugap_options(dose_schedule = c(1, 3))
```

---

latent_params_sim *Example Latent Parameter Values*

Description

A list containing the true/latent parameter values used to simulate the example datasets (locations_sim, populations_sim, observations_sim).

Usage

```
latent_params_sim
```

Format

A list with 7 components:

- phi_state, a numeric vector of length 30 representing the state-specific baseline vaccine uptake propensity over cohorts.
- lambda, a numeric vector of length 2 representing the rate parameters for vaccine doses 1 and 2 respectively.
- sigma_sch, a number, the standard deviation of school-level random effects.
- sigma_cnty, a number, the standard deviation of county-level random effects.
- off_sch, a numeric vector of length 24 containing school-level random offsets.
- off_cnty, a numeric vector of length 3 containing county-level random offsets.
- censor_reduction, a number representing the censoring offset multiplier applied to censored observations (0.95).

locations_sim *Example Location Data*

Description

A dataset providing example location input.

Usage

```
locations_sim
```

Format

A [data.table()] with 28 rows and 2 columns:

- loc_id, a string, the location
- parent_id, a string, the location parents

observations_sim *Example Observation Data*

Description

A dataset containing vaccine coverage observations.

Usage

```
observations_sim
```

Format

A [data.table()] with 698 rows and 4+ columns:

- obs_id, a number, the observation id (primary key)
- positive, a number, how many individuals had the vaccine
- sample_n, a number, the number of individuals in the observations
- censored, a number, 1 if positive is right censored and NA if uncensored
- ... assorted other columns that are irrelevant to use as observations arg

populations_sim *Example Population Data*

Description

A dataset containing the meta-data about vaccine coverage observations.

Usage

```
populations_sim
```

Format

A [data.table()] with 750 rows and 6 columns:

- obs_id, a number, the observation (foreign key to observations)
- loc_id, a string, the location (foreign key to locations)
- cohort, a number, the birth cohort
- age, a number, the age of cohort at time of observation
- dose, a number, which dose the observation concerns
- weight, a number (0-1), fraction of the observation this row represents

predict.imugap_fit *Predict coverage probabilities*

Description

Uses the output of [sampling()] and a target grid to generate predicted coverage probabilities.

Usage

```
## S3 method for class 'imugap_fit'
predict(object, target, posterior_size = NULL, ...)
```

Arguments

object	an imugap_fit object returned by sampling()
target	a [data.frame()] of target populations to predict for
posterior_size	optional single positive integer. When set, predict over only this many draws, taken from the end of each chain (the converged tail). Must be a multiple of the number of chains; a value that isn't is rounded up to the next multiple, with a warning. Must not exceed the number of draws in the fit. Defaults to NULL, which uses every draw.
...	additional arguments (currently ignored)

Details

The [predict()] method takes an imugap_fit object (typically the output of [sampling()]) and a target grid (typically output from [create_target()]), and generates predicted coverage probabilities for each entry in the target.

The [predict()] method can be used to generate estimated coverage for any location, cohort, or age considered within the bounds of the original sampling fit. Particularly, this includes enclosing locations without specific observation data, as long as those locations are *somewhere* in the locations hierarchy.

By default predict() uses every posterior draw in the fit. Supply posterior_size to predict over a sub-sample taken from the end of each chain; this is how the bundled predict_sim fixture is kept small. The returned draws keep the per-chain structure (iterations x chains x targets). When a sub-sample is taken predict() warns that it has not checked whether those draws are adequate (chain mixing, effective sample size).

Value

An object of class imugap_predict wrapping the 3D array of predicted draws and the canonical target dataset.

Examples

```
# Load example fit object and target population
data("fit_sim", package = "imuGAP")
data("target_sim", package = "imuGAP")

# Generate predictions over 100 posterior draws
preds <- predict(fit_sim, target = target_sim, posterior_size = 100)
```

---

predict_sim *Example Coverage Predictions*

Description

A dataset containing predicted vaccine coverage probabilities generated by calling predict() on fit_sim with target_sim as the target.

Usage

```
predict_sim
```

Format

An object of class imugap_predict wrapping:

- draws, a 3D array of predicted draws with dimensions [iteration, chain, variable]
- target, a [data.table()] containing target population parameters matching the variables

sampling *Immunity: Geographic & Age-based Projection,* imuGAP

Description

This a sampler interface to convert user-friendly data into the necessary format to feed the immunity estimation model.

Usage

```
sampling(
  observations,
  populations,
  locations,
  imugap_opts = imugap_options(),
  stan_opts = stan_options()
)
```

Arguments

observations	a [data.frame()], the observed data, with at least three columns: • an obs_id column; any type, as long as unique, non-NA • a positive column; non-negative integers, the observed number of vaccinated individuals • a sample_n column; positive integers, the number of individuals sampled, must be greater than or equal to "positive" • optionally, a censored column; numeric, NA (uncensored) or 1 (right-censored); if not present, will be assumed NA
populations	a [data.frame()], the observation meta data, with columns • obs_id, any type; the observation the row concerns (i.e. id shared with an observations data object) • loc_id, any type; the location the row concerns (i.e. id shared with a locations data object) • dose, a non-zero, positive integer (1, 2, ...); what dose row concerns • cohort, a positive integer; the cohort at the location row concerns • age, a positive integer; the age of that cohort row concerns • weight, a numeric, (0, 1); the relative contribution of this row to an observation. Optional if each population row has a unique obs_id.
locations	a [data.frame()], with columns loc_id and parent_id, of the same type. See Details for restrictions.
imugap_opts	options for the imuGAP model
stan_opts	passed to rstan::sampling (e.g. iter, chains).

Value

An object of class `imugap_fit` wrapping the raw `stanfit` object along with settings and dataset metadata.

Examples

```
data("locations_sim"); data("observations_sim"); data("populations_sim")
st_opts <- stan_options(chains = 2, iter = 500)
sampling(
  observations_sim, populations_sim, locations_sim,
  stan_opts = st_opts
)
```

`stan_options` *Stan Sampler Options*

Description

This function encapsulates option passing to the stan sampler, with the exception of the model object, which is passed in `imugap_options`.

Usage

```
stan_options(...)
```

Arguments

`...` Arguments passed on to `rstan::sampling`

`data` A named `list` or `environment` providing the data for the model or a character vector for all the names of objects used as data. See the **Passing data to Stan** section in `stan`.

`pars` A vector of character strings specifying parameters of interest. The default is `NA` indicating all parameters in the model. If `include = TRUE`, only samples for parameters named in `pars` are stored in the fitted results. Conversely, if `include = FALSE`, samples for all parameters *except* those named in `pars` are stored in the fitted results.

`chains` A positive integer specifying the number of Markov chains. The default is 4.

`iter` A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.

`warmup` A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter` and the default is `iter/2`.

`thin` A positive integer specifying the period for saving samples. The default is 1, which is usually the recommended value.

`seed` The seed for random number generation. The default is generated from 1 to the maximum integer supported by R on the machine. Even if multiple chains are used, only one seed is needed, with other chains having seeds derived from that of the first chain to avoid dependent samples. When a seed is specified by a number, `as.integer` will be applied to it. If `as.integer` produces `NA`, the seed is generated randomly. The seed can also be specified as a character string of digits, such as `"12345"`, which is converted to integer.

`init` Initial values specification. See the detailed documentation for the init argument in `stan`.

`check_data` Logical, defaulting to `TRUE`. If `TRUE` the data will be preprocessed; otherwise not. See the **Passing data to Stan** section in `stan`.

`sample_file` An optional character string providing the name of a file. If specified the draws for *all* parameters and other saved quantities will be written to the file. If not provided, files are not created. When the folder specified is not writable, `tempdir()` is used. When there are multiple chains, an underscore and chain number are appended to the file name prior to the `.csv` extension.

`diagnostic_file` An optional character string providing the name of a file. If specified the diagnostics data for *all* parameters will be written to the file. If not provided, files are not created. When the folder specified is not writable, `tempdir()` is used. When there are multiple chains, an underscore and chain number are appended to the file name prior to the `.csv` extension.

`verbose` `TRUE` or `FALSE`: flag indicating whether to print intermediate output from Stan on the console, which might be helpful for model debugging.

`algorithm` One of sampling algorithms that are implemented in Stan. Current options are `"NUTS"` (No-U-Turn sampler, Hoffman and Gelman 2011, Betancourt 2017), `"HMC"` (static HMC), or `"Fixed_param"`. The default and preferred algorithm is `"NUTS"`.

`control` A named `list` of parameters to control the sampler's behavior. See the details in the documentation for the `control` argument in `stan`.

`include` Logical scalar defaulting to `TRUE` indicating whether to include or exclude the parameters given by the `pars` argument. If `FALSE`, only entire multidimensional parameters can be excluded, rather than particular elements of them.

`cores` Number of cores to use when executing the chains in parallel, which defaults to 1 but we recommend setting the `mc.cores` option to be as many processors as the hardware and RAM allow (up to the number of chains).

`open_progress` Logical scalar that only takes effect if `cores > 1` but is recommended to be `TRUE` in interactive use so that the progress of the chains will be redirected to a file that is automatically opened for inspection. For very short runs, the user might prefer `FALSE`.

`show_messages` Either a logical scalar (defaulting to `TRUE`) indicating whether to print the summary of Informational Messages to the screen after a chain is finished or a character string naming a path where the summary is stored.

Setting to FALSE is not recommended unless you are very sure that the model is correct up to numerical error.

Value

a list of arguments matching rstan::sampling() inputs

Examples

```
stan_options()
stan_options(chains = 2, iter = 500)
```

---

subset.imugap_predict *Subset coverage predictions*

Description

Subsets predicted coverage draws by target metadata (variables), iterations, and chains.

Usage

```
## S3 method for class 'imugap_predict'
subset(x, subset, iteration, chain, ...)
```

Arguments

x	an imugap_predict object returned by [predict()].
subset	logical expression indicating which target variables to keep. Evaluated in the context of the target metadata data.table.
iteration	numeric/integer/logical vector of iterations to keep.
chain	numeric/integer/logical vector of chains to keep.
...	additional arguments (currently ignored).

Value

A subsetted imugap_predict object with corresponding subsetted draws and target metadata.

Examples

```
# Load example prediction object
data("predict_sim", package = "imuGAP")

# Subset predictions by target metadata
subset(predict_sim, dose == 2)

# Subset predictions by iteration and chain
subset(predict_sim, iteration = 1:10, chain = 1)
```

summary.imugap_predict *Summarize coverage predictions*

Description

Summarizes predicted coverage probabilities from an imugap_predict object by location, cohort, age, and dose for the requested quantiles.

Usage

```
## S3 method for class 'imugap_predict'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	an imugap_predict object returned by [predict()]
probs	numeric vector of probabilities/quantiles to compute. Defaults to c(0.025, 0.5, 0.975).
...	additional arguments (currently ignored)

Value

A data.table containing target population parameters, posterior mean coverage (mean), and the requested quantiles (e.g. q2.5, q50, q97.5).

Examples

```
# Load example prediction object
data("predict_sim", package = "imuGAP")

# Summarize coverage predictions
summary(predict_sim)

# Summarize with custom quantiles
summary(predict_sim, probs = c(0.1, 0.5, 0.9))
```

target_sim *Example Prediction Target Populations*

Description

A dataset specifying the target populations for coverage prediction, generated by calling `create_target()` on the simulated fit object `fit_sim` along with arguments for which locations, cohorts, and ages to target. Includes locations which were not present in the original simulated observations, namely the State and County levels.

Usage

```
target_sim
```

Format

A `[data.table()]` with 1008 rows and 7 columns:

- `obs_c_id`, a number, the target observation/combination ID (primary key)
- `loc_id`, a string, the location ID
- `age`, a number, the age
- `cohort`, a number, the birth cohort
- `dose`, a number, the vaccine dose (1 or 2)
- `weight`, a number, the weight of the prediction component (always 1)
- `loc_c_id`, a number, the compiled location ID

Index