Package 'imuGAP'

September 2, 2026

Title Immunity: Geographic and Age-Based Projection

Version 0.2.0

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, flexstanr (>= 0.2.0), methods, Rcpp (>= 1.1.2), RcppParallel (>= 5.1.11.2), rstan (>= 2.32), rstantools (>= 2.6.0)

LinkingTo BH (>= 1.90), Rcpp (>= 1.1.2), RcppEigen (>= 0.3.4), RcppParallel (>= 5.1.11), rstan (>= 2.32), StanHeaders (>= 2.32)

SystemRequirements GNU make

Config/Needs/check stan-dev/cmdstanr

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

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 8.1.0

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 2026-09-02 15:30:14 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 supports hierarchical models of arbitrary depth (e.g., 1-layer root only, 2-layer root + sub-locations, 3-layer like state -> county -> school, or deeper regional partitions). This method checks location structure validity and returns a canonical version including layer membership.

A valid structure has:

- a unique root,
- no cycles,
- no duplicate `loc_id`s, and
- **branching constraints**: every location node must have either **0 offspring** (leaf location) or **at least 2 offspring** (>= 2 children). Single-child chains (a parent location with exactly 1 child) are rejected to ensure variance components are statistically identifiable across adjacent layers.

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.

Weights must be non-negative, finite numbers with no `NA` values. If not explicitly supplied, weights default to 1 for observations with a single contributing row.

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)
```

canonicalize_target *Canonicalize a target grid against a fit*

Description

Normalizes a target grid and validates it against a specific imugap_fit.

Usage

```
canonicalize_target(target, fit)
```

Arguments

target	a target grid: the output of [create_target()], or a data.frame / data.table with loc_id, age, cohort, and dose columns.
fit	an imugap_fit object returned by [sampling()].

Details

Accepts either the output of [create_target()] or a plain data.frame / data.table with loc_id, age, cohort, and dose columns (optionally weight / obs_id). Fills obs_c_id and weight when absent, checks that every loc_id exists in the fit and that dose, age, and cohort are within the fit's ranges, and adds the canonical loc_c_id. Errors on any out-of-range value. [predict.imugap_fit()] calls this internally, so most users do not call it directly.

Value

the validated target (a data.table) with loc_c_id added.

See Also

[create_target()], [predict.imugap_fit()]

Examples

```
data("fit_sim")
target <- create_target(
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2, 3), cohort = 5, dose = c(1), mode = "snapshot"
)
canonicalize_target(target, fit_sim)
```

`create_observation_populations` *Create observation populations*

Description

`create_observation_populations` is a convenience function to construct a properly weighted `populations` object for typical modes of observation.

Usage

```
create_observation_populations(observations, mode = "snapshot", ...)
```

Arguments

`observations`	a pre- or post-canonicalization `observations` object. Optionally contains additional columns required for the specified `mode` that vary by row.
`mode`	character; the mode for populations creation (default: "snapshot").
`...`	additional arguments determined by the specified `mode` requirements for values which do not vary by row.

Details

This function uses a combination of varying information from `observations` and fixed information from `...` arguments to provide the necessary information for the modes to produce a `[canonicalize_populations()]` ready-object.

Supported modes and required information:

Value

A `data.table` representing the populations mapping.

"snapshot" mode

As with `[create_target()]`, a snapshot view is looking at a particular place, time, and dose target, but with varying birth cohorts. That means the sum of birth cohort and age is constant: if birth cohort 1 is age 10, then cohort 2 is 9, and so on.

Snapshots requires `obs_id`, `loc_id`, `dose`, `age_min`, and `cohort` the reference cohort corresponding to the oldest age. `age_max` may be provided, but if missing or `NA`, is assumed to be `age_min + 1`. `age_max` corresponds to the first *excluded* age - i.e. `[age_min, age_max)`

Taking this approach to `age_max` enables this method to naturally support partial cohorts. For example, if `age_max = 18.5` and `age_min = 17`, then age 17 population has 2/3rds the weight and the age 18 population has 1/3rd. `age_min` works the same way.

Note that "snapshot" mode assumes that all populations are uniformly sized with respect to weighting. This assumption may be inadequate when population age groups contributing to an observation are very differently sized.

`create_target` *Construct a target grid for prediction*

Description

Builds a target grid, for use with [`predict.imugap_fit()`], from vectors of locations, ages, cohorts, and doses. This is pure construction and does not reference a fitted model, so it can be called without a fit (e.g. to expand a request into rows before any fit exists). To validate a target against a specific fit – or to canonicalize a target you built yourself as a `data.frame` – use [`canonicalize_target()`]; [`predict.imugap_fit()`] does this for you.

Usage

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

Arguments

`location`	a vector of location IDs to target.
`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`	one of `"error"` (default), `"enumerate"`, `"recycle"`, or `"snapshot"`, controlling how the vector inputs combine: • `"error"`: all vector inputs must have the same length. • `"enumerate"`: all combinations of the inputs. • `"recycle"`: recycle the inputs out to the least-common-multiple length. • `"snapshot"`: `cohort` must be a single reference value (the oldest cohort); locations, ages, and doses are enumerated with a cohort for each age such that `age + cohort` is constant, using the **maximum** value of `age` to set that constant (`cohort_i = cohort_ref + max(age) - age_i`), i.e. a snapshot in time.

Value

a `data.table` target grid with columns `obs_c_id`, `loc_id`, `age`, `cohort`, `dose`, and `weight`.

See Also

[`canonicalize_target()`], [`predict.imugap_fit()`]

Examples

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

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

# "snapshot": cohort is a single reference; cohorts are set so age + cohort is
# constant, using max(age).
create_target(
  location = c("Blue Heron School", "Bluebird Learning Center"),
  age = c(1, 2, 3), cohort = 5, dose = c(1), mode = "snapshot"
)
```

---

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.

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. It is not tracked in git: it is regenerated on build by `data-raw/fit_data.R` (run `just data-fit` locally, or `just data` for the full pipeline).

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_data.R`.

fit_sim_1layer — *Example Stan Fit (1 Layer)*

Description

A reference `imugap_fit` object from fitting a 1-layer model (State only).

fit_sim_2layer — *Example Stan Fit (2 Layers)*

Description

A reference `imugap_fit` object from fitting a 2-layer model (State -> County).

imugap_options *imuGAP Model Options*

Description

Configures model-side options for imuGAP estimation.

Usage

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

Arguments

df	single positive integer; degrees of freedom to use for the cohort B-spline basis expansion (default: 5L).
dose_schedule	an ascending integer vector of ages at which each dose 1..n becomes eligible (default: c(1, 4) for 2-dose vaccines).
model	character string specifying the model formulation. Defaults to "default", with dispatch to optimized single versus multilayer versions within [sampling()]

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).

Format

A list with 8 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).
- `coverage`, a numeric vector of the true/background coverage for each row of `target_sim`, computed from the latent parameters above.

`locations_sim` *Example Location Data*

Description

A dataset providing example location input.

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.

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.

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.

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

predict_sim_1layer *Example Coverage Predictions (1 Layer)*

Description

Predicted coverage probabilities for 1-layer model.

predict_sim_2layer *Example Coverage Predictions (2 Layers)*

Description

Predicted coverage probabilities for 2-layer model.

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

Description

Fits the imuGAP Bayesian hierarchical vaccine coverage estimation model across arbitrary user-specified location partitions, birth cohorts, ages, and vaccine doses.

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, created by [imugap_options()].

`stan_opts` sampler configuration created by [stan_options()] (see [flexstanr::stan_options()] for details on supported sampler arguments, including iter, chains, cores, seed, and backend).

Details

sampling() automatically inspects the depth of the location hierarchy supplied in locations via [canonicalize_locations()] and [assemble_layer_data()]:

- **Single-layer (1 layer)**: When only a root location is supplied, sampling() automatically dispatches to the optimized single-location model.
- **Multi-layer (>= 2 layers)**: When hierarchical sub-locations are supplied (e.g., 2-layer state -> county, 3-layer state -> county -> school, or deeper trees), sampling() dispatches to the general hierarchical model with partial pooling across layer-specific variance components.

If the Stan sampler fails to initialize and produces no draws (for the rstan backend, a mode-2 stanfit with an empty @sim), sampling() raises an error of class imugap_no_draws rather than returning an empty fit, so the failure can be handled with tryCatch(). The check is backend-agnostic (see backend_has_draws()).

Value

An object of class imugap_fit wrapping the raw stanfit (or CmdStanMCMC) object along with model 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
```

```
)
```

`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 constructing a target grid with `create_target()` and validating it against the simulated fit `fit_sim` with `canonicalize_target()`, for the chosen locations, cohorts, and ages to target. Includes locations which were not present in the original simulated observations, namely the State and County levels.

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

target_sim_1layer *Example Prediction Target Populations (1 Layer)*

Description

Prediction target grid for 1-layer model.

target_sim_2layer *Example Prediction Target Populations (2 Layers)*

Description

Prediction target grid for 2-layer model.

Index