Package: bayescoveragemodel (via r-universe)

June 9, 2026

Title Bayesian Transition Models for Estimating and Forecasting Health Coverage Indicators

Version 0.12

Description Fits Bayesian hierarchical transition models to health coverage indicators (e.g., ANC4, institutional delivery) using survey and routine data. Implements models using Stan for Bayesian inference, with support for global and local (country-specific) model fitting, out-of-sample validation, and subnational estimation.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports brms, dplyr, ggplot2, ggpubr, ggnewscale, purrr, readr, rlang, splines, stringr, tibble, tidybayes, tidyr

Suggests cmdstanr, rstan, posterior, bayesplot, haven, here, knitr, rmarkdown, testthat (>= 3.0.0)

Remotes github::AlkemaLab/localhierarchy

Config/testthat/edition 3

VignetteBuilder knitr

URL https://alkemalab.github.io/bayescoveragemodel/

BugReports https://github.com/AlkemaLab/bayescoveragemodel/issues

Depends R (>= 3.5)

LazyData true

Config/pak/sysreqs cmake make libicu-dev libx11-dev

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

Date/Publication 2026-06-09 04:45:40 UTC

RemoteUrl https://github.com/AlkemaLab/bayescoveragemodel

RemoteRef HEAD

RemoteSha acc1ae94f34b1e498ab051e2c66f1d3328aa9ad3

Contents

`backtransform_residuals` *Back-transform Residuals to Proportion Scale*

Description

Transforms residuals from probit scale back to the original proportion scale. Computes the response on probit scale, then calculates residuals and standard errors on the proportion scale.

Usage

```
backtransform_residuals(df)
```

Arguments

`df`	Data frame containing columns: • `level`: Predicted value on probit scale • `residual`: Residual on probit scale • `scale`: Standard error on probit scale

Value

Data frame with added columns:

- `response`: Observed value on probit scale (level + residual)
- `residual_ori`: Residual on proportion scale
- `scale_ori`: Standard error on proportion scale

`compare_fits` *Compare multiple model fits in a single plot*

Description

This function takes run names, loads the corresponding fits, and plots them together using plot_estimates_local_all.

Usage

```
compare_fits(
  runnames,
  modelnames = NULL,
  iso_codes = NULL,
  dat_routine = NULL,
  indicator_name = NULL,
  save_plots = FALSE,
  output_folder = NULL,
  add_caption = FALSE,
  add_estimates = TRUE,
  use_for_facetting = FALSE
)
```

Arguments

`runnames`	Character vector of run names (2-4 names). Run names follow the pattern: indicator*runstep*folder_suffix, e.g., "anc4_step1b_run_alldata_jan16"
`modelnames`	Character vector of display names for the legend (same length as runnames). If NULL, uses the runnames.
`iso_codes`	Optional character vector of ISO codes to filter plots. If NULL, returns plots for all countries.
`dat_routine`	Optional routine data to overlay on plots
`indicator_name`	Custom title for y-axis (if NULL, pulled from fit$data)
`save_plots`	Boolean to save plots to PDF
`output_folder`	Override directory for saving
`add_caption`	Boolean to add explanatory caption
`add_estimates`	Boolean to show model estimates/ribbons
`use_for_facetting`	Boolean to adjust plot titles for facetted output

Value

List of ggplot objects, one per geographic unit (filtered by iso_codes if provided)

Examples

```
## Not run:
# Compare step1a and step1b for anc4
plots <- compare_fits(
  runnames = c("anc4_step1b_run_alldata_jan16", "anc4_step1a_run_alldata_jan16"),
  modelnames = c("Step 1b", "Step 1a")
)

# Compare validation and non-validation fits for specific countries
plots <- compare_fits(
  runnames = c("vdpt_step1b_run_alldata_jan16", "vdpt_step1b_val_alldata_jan16"),
  modelnames = c("Full model", "Validation"),
  iso_codes = c("ETH", "GHA", "KEN", "NGA", "UGA")
)

## End(Not run)
```

---

explore_data *Explore Coverage Indicators Over Time*

Description

This function performs exploratory data analysis on a dataset containing coverage indicators over time.

Usage

```
explore_data(
  data,
  indicator_col,
  indicator_se_col = NULL,
  data_types_col = NULL,
  group_col,
  region_col = NULL,
  indicator_name = NULL,
  routine_data = NULL
)
```

Arguments

data	A data frame containing the coverage indicators and year variable.
indicator_col	A string specifying the column name for the coverage indicator.

indicator_se_col
: An optional string specifying the column name for the standard error of the indicator.

data_types_col
: An optional string specifying the column name for data types in the data frame.

group_col
: A string specifying the column name for first-level grouping (e.g., country).

region_col
: An optional string specifying the column name for second-level grouping (e.g., region).

indicator_name
: An optional string to use as the title for plots instead of the default "Indicator".

routine_data
: An optional data frame containing routine data.

Value

A series of plots for exploratory data analysis.

---

fit_model *Fit the transition model to data.*

Description

Fit the transition model to data.

Usage

```
fit_model(
  survey_df,
  national_dat_df = NULL,
  popweights = NULL,
  y = "invprobit_indicator",
  se = "se_invprobit_indicator",
  year = "year",
  source = "data_series_type",
  area = "iso",
  iso_select = NULL,
  routine_data = NULL,
  fit_routine_obj = NULL,
  start_year = 2000,
  end_year = 2030,
  runstep,
  global_fit = NULL,
  t_star = 2010,
  spline_degree = 2,
  num_knots = 8,
  Betas_upper_bound = 0.5,
  Betas_lower_bound = 0.01,
  Ptilde_low = 0,
  add_dataoutliers = TRUE,
```

```
  extra_stan_data = list(),
  hierarchical_level = c("intercept", "cluster", "subcluster", "iso"),
  hierarchical_splines = c("intercept", "cluster", "subcluster", "iso"),
  hierarchical_asymptote = c("intercept", "iso"),
  add_subnational_hierarchy = "admin1",
  use_globalsubnat_fromnat = FALSE,
  model_name = "spline",
  held_out = FALSE,
  validation_cutoff_year = NULL,
  save_post_summ = TRUE,
  generate_quantities = TRUE,
  get_posteriors = TRUE,
  stan_file_path = NULL,
  stan_model = NULL,
  backend = c("cmdstanr", "rstan"),
  create_runname_and_outputdir = TRUE,
  runnumber = 1,
  rungroup = NULL,
  runname = NULL,
  chains = 4,
  iter_sampling = 200,
  iter_warmup = 150,
  add_sample = TRUE,
  compile_model = TRUE,
  force_recompile = FALSE,
  seed = 1234,
  refresh = 10,
  adapt_delta = 0.9,
  max_treedepth = 14,
  add_inits = TRUE
)
```

Arguments

survey_df	tibble with survey data
y	column name of outcome.
se	column name of outcome standard error.
year	column name of outcome year.
source	column name of data source.
area	column name of the area of each observation for subnational:
iso_select	ISO code to use for local national run
routine_data	data frame with routine data to use in the fit. If NULL (default), no routine data will be used.

`fit_routine_obj`
: optional fit_routine object (e.g., from brms) containing hyperparameters for routine data processing. If NULL (default), the function will load the internal package data object `fit_routine`.

`start_year`
: start year of estimates.

`end_year`
: end year of estimates.

Settings for global model fit 1a

`runstep`
: type of run, currently one of "step1a", "step1ab", "step1b", "local_national" (see Details).

`global_fit`
: optional object of class `"fpemplus"`, used to obtain fixed values to use for some parameters in the current fit (see Details).

`t_star`
: reference year used in model.

`spline_degree`
: spline degree. Degree 2 or 3 is supported.

`num_knots`
: number of spline knots.

`Betas_upper_bound`
: upper bound for the splines parameters

`Betas_lower_bound`
: lower bound for the splines parameters

`Ptilde_low`
: lower bound for the asymptote Ptilde

`add_dataoutliers`
: boolean indicator of whether to include data outliers in 1b

`extra_stan_data`
: list of additional data to pass to Stan model

`hierarchical_level`
: vector specifying hierarchical structure for the level in reference year (see Details).

`hierarchical_splines`
: vector specifying hierarchical structure for spline coefficients (see Details).

`hierarchical_asymptote`
: vector specifying hierarchical structure for asymptote (see Details).

`model_name`
: character string specifying the model to fit. Options are:

  - `"spline"` (default): Transition model, ARIMA(1,1,0) with spline-based drift.
  - `"rw2"`: Local rate of change model, ARIMA(1,2,0)

`held_out`
: binary vector indicating which observations are held out. Set to FALSE to hold out no observations. overwritten by validation_cutoff_year if that is not NULL

`validation_cutoff_year`
: year to use for out-of-sample validation, overwrites held_out

`save_post_summ`
: boolean indicator of whether to save summary object (does NOT work for rstan backend)

`generate_quantities`
: binary vector indicating whether to simulate data from the fitted model

`get_posteriors`
: boolean indicator of whether to return posterior samples

stan_file_path	stan file path (if NULL, uses internal stan file)
stan_model	precompiled Stan model object (if NULL, model will be compiled from stan_file_path). Used by bayescoveragedeploy to pass precompiled models for users without C++ compilers.
backend	character string specifying the Stan backend to use. Options are: • "cmdstanr" (default): Use cmdstanr interface (recommended, modern) • "rstan": Use rstan interface (for deployment of local model with precompiled stan models only)
	Setting for where to save things
create_runname_and_outputdir	boolean indicator of whether to create a runname and output directory
runnumber	number to add to runname
rungroup	group to add to runname
runname	name to use for run
	Settings for sampling
chains	number of chains to run
iter_sampling	number of posterior samples to draw
iter_warmup	number of warmup iterations
add_sample	boolean indicator of whether to return samples
compile_model	boolean indicator of whether to compile the Stan model
force_recompile	boolean indicator of whether to force recompilation of the Stan model
seed	random seed
refresh	number of iterations between progress updates
adapt_delta	target acceptance rate for the No-U-Turn Sampler
max_treedepth	maximum tree depth for the No-U-Turn Sampler
add_inits	boolean indicator of whether to add initial values to the Stan model
population_data	a data frame with yearly population counts for subnational regions. It should have columns matching the names specified for year and area. This data frame is only required if the primary data set contains a mix of observations at national and subnational levels.

Details

The fit_model function fits the transition model to data. The model is fitted using Stan, and the function returns an object of class "fpemplus". The argument runstep determines the type of run to perform. The following run steps are supported:

- "step1a": Fit the model without the smoothing term, to estimate longer term trends.
- "step1ab": Fit the model with smoothing terms, estimating all parameters (none fixed). Unlike step1b, this does not require a prior fit and estimates everything in one pass.

- "step1b": Fit the model with smoothing terms, using a fit from step1a, to estimate all smoothing and data model parameters.
- "local_national": Fit the model to data from a single country, using a 1b fit. This is also explained in the documentation folder.

Details on hierarchical set ups used Several area-specific parameters of the fpemplus model have hierarchical priors assigned to them so that information can be shared between areas. The package allows the structure of the hierarchical prior to be configured by the user through the `hierarchical_asymptote`, `hierarchical_level`, and `hierarchical_splines` arguments. These arguments expect a character vector that specifies a nesting hierarchical structure. Each element of the vector must be either "intercept" or a column name in the dataset, where "intercept" will add a global intercept for the parameter. The vector must be in descending order in terms of the hierarchy: that is, it starts with "intercept" and proceeds down the hierarchy.

For example, suppose we are fitting country-level data, where the dataset has columns "name_country", "name_sub_region", and "name_region" containing the name of the country, sub-region, and region that each observation belongs to. To specify that the spline coefficients should be fitted with a hierarchical model in which countries are nested within sub-regions within regions within world, we would use the argument `hierarchical_splines = c("intercept", "name_region", "name_sub_region", "name_country")`.

Optionally, model parameters can be fixed to values from a previous model fit provided via the `global_fit` argument. In a typical use case, the `global_fit` will have been fit to data from many geographic units (e.g., all countries), while the current fit uses data from a smaller number of locations. Note: remainder of details is currently determined by the runstep To use a global fit to fix parameter values, a number of settings must be the same in the global fit and the current call to `fpemplus`:

- For any of the hierarchical settings, e.g. `hierarchical_asymptote`, all of the hierarchical levels used for that parameter in the global fit must also be used in the current fit. For instance, if the `global_fit` used `hierarchical_asymptote = c("intercept", "name_region", "name_country")`, then in the current fit it is valid to use `hierarchical_asymptote = c("intercept", "name_region", "name_country")` again or `hierarchical_asymptote = c("intercept", "name_region", "name_country", "name_subnational")`, but it is not valid to use `hierarchical_asymptote = c("intercept", "name_region", "name_subnational")`.
- All hierarchical levels for terms and sigmas to fix in the current fit are contained in the levels in the hierarchy used for the corresponding parameters in the global fit. For example, if `hierarchical_asymptote_sigmas_fixed = c("intercept", "name_region", "name_country")` then the global fit must have included at least "intercept", "name_region", and "name_country" for `hierarchical_asymptote`.
- Any hierarchical levels to fix in the current fit are at the highest levels of the hierarchical structure. For example, if `hierarchical_asymptote = c("intercept", "name_region", "name_country")`, then it is valid to use `hierarchical_asymptote_sigmas_fixed = c("intercept", "name_region")`, but it is not valid to use `hierarchical_asymptote_sigmas_fixed = c("intercept", "name_country")`.
- It is not valid to fix terms at a given hierarchy level without also fixing the sigma estimate at that hierarchy level. For example, we cannot specify `hierarchical_asymptote_sigmas_fixed = c("intercept")` and `hierarchical_asymptote_terms_fixed = c("intercept", "name_region")`
- It is only valid to set `fix_smoothing = TRUE` if also `smoothing = TRUE`.

- All settings for the arguments `model`, `t_star`, `smoothing`, `tau_prior`, and `rho_prior` must be the same in the `global_fit` and the current fit.
- If `hierarchical_splines_sigmas_fixed` or `hierarchical_splines_terms_fixed` include any hierarchical levels (i.e., if either is different from the empty vector `c()`), all settings for `num_knots` and `spline_degree` must be the same in the `global_fit` and the current fit.
- All geographic units that appear in the `data` for the current fit at hierarchical levels for which any parameter is fixed must have also been included in the `data` used for the `global_fit`.
- If `fix_nonse = TRUE`, all data `sources` that appear in the `data` for the current fit must also have been included in the `data` used for the `global_fit`.

Value

fpemplus object.

`get_convergence_diagnostics` *Get convergence diagnostics for a fitted model*

Description

Computes and saves convergence diagnostics (Rhat, ESS) for hyperparameters and creates density overlay plots.

Usage

```
get_convergence_diagnostics(fit, model_name = "spline")
```

Arguments

`fit`	A fitted model object from `fit_model`
`model_name`	Character string specifying the model type. Either "spline" (checks Omega, Ptilde, Betas) or "rw2" (checks Omega, gamma). Default is "spline".

Value

NULL (invisibly). Saves diagnostics.csv and diagnostics.pdf to the output directory.

`get_eta_samples` *Extract posterior samples of eta (latent coverage) for a specific year*

Description

Extracts posterior draws of the latent coverage parameter eta from a fitted model object for a specified year. For validation runs, only includes countries that have both training and test data.

Usage

```
get_eta_samples(fit, year_select = 2023, countryyear_select = NULL)
```

Arguments

`fit`	A fitted model object containing `samples` (cmdstanr or rstan fit object), `geo_unit_index`, `time_index`, `stan_data`, and `data`.
`year_select`	The year for which to extract eta samples. Default is 2023. Ignored if `countryyear_select` is not NULL.
`countryyear_select`	A tibble with columns `iso` and `year` specifying country-year combinations to extract. If not NULL, takes precedence over `year_select`. Default is NULL.

Value

A tibble with columns:

iso Country ISO code

year Selected year

eta Posterior draw of eta (probit-scale coverage)

draw Draw number

cluster, subcluster, name_region Additional geographic identifiers if present

Examples

```
## Not run:
fit <- load_fit("my_model_run")
eta_2023 <- get_eta_samples(fit, year_select = 2023)

# Extract specific country-year combinations
cy_select <- tibble::tibble(iso = c("USA", "CAN"), year = c(2023, 2022))
eta_subset <- get_eta_samples(fit, countryyear_select = cy_select)

## End(Not run)
```

get_fit *Load a saved model fit*

Description

This helper function loads a previously saved model fit based on the indicator name, run step, and folder suffix.

Usage

```
get_fit(indicator, runstep, folder_suffix)
```

Arguments

indicator	Character string specifying the indicator name (e.g., "ideliv", "anc4").
runstep	Character string indicating the model step (e.g., "step1a", "step1b").
folder_suffix	Character or numeric. The suffix at the end of the folder name.

Value

A model fit object read from an RDS file.

get_inno_samples *Extract Innovation Samples from Fitted Model*

Description

Extracts epsilon_innovation (AR1 process innovations) and eta from posterior samples. These represent the temporal deviations from the expected coverage trajectory along with the underlying fitted values.

Usage

```
get_inno_samples(fit)
```

Arguments

fit	Fitted model object containing: \itemsamplescmdstan samples with epsilon_innovation CT and eta CT \itemdataData frame with iso, country, name_region, year \itemtime_indexTime index mapping with columns t and year \itemgeo_unit_indexGeographic unit index with cluster/subcluster info

Details

Only returns innovations for years within the observation range for each country (min_obs_yr to max_obs_yr).

Value

Nested tibble with one row per (iso, year) containing:

`iso`	ISO country code
`year`	Year
`cluster`	WHO region cluster (if available)
`subcluster`	Regional subcluster (if available)
`name_region`	Region name (if available)
`draws`	Nested tibble with draw-specific columns:

- draw: Posterior draw number
- eta: Eta parameter on probit scale
- sd_y: Scale parameter (always 1 for standardized innovations)
- residual: Innovation value (epsilon_innovation)
- level: Fitted coverage on inv-probit scale
- level_prop: Eta on proportion scale
- yhat: Fitted value on inv-probit scale (same as level)
- y: Reconstructed observation (residual + yhat)
- sd_y_prop: Standard deviation on proportion scale
- y_prop: Reconstructed observation on proportion scale

`get_relative_output_dir` *Get relative output directory path*

Description

Constructs a relative path to a subfolder inside `bayestransition_output`.

Usage

```
get_relative_output_dir(folder_name)
```

Arguments

`folder_name`	Character. The name of the subfolder inside `bayestransition_output`.

`get_residuals_samples` *Extract Residuals from Posterior Samples*

Description

Extracts residuals from a fitted model object by comparing observed data to posterior draws of eta_i (predicted coverage).

Usage

```
get_residuals_samples(fit)
```

Arguments

`fit`	Fitted model object containing: \itemsamplescmdstan samples object with eta_i and scale parameters \itemstan_dataList with y (observations) and held_out flag

Value

Nested tibble with one row per observation containing:

`obs_index`	Observation index
`y`	Observed value on modeled scale
`y_prop`	Observed value on proportion scale
`held_out`	Logical indicating if observation was held out
`iso`	Country ISO code
`year`	Year of observation
`cluster`	WHO region cluster (if available in fit$data)
`subcluster`	Regional subcluster (if available in fit$data)
`name_region`	Region name
`draws`	Nested tibble with draw-specific columns:

- draw: Posterior draw number
- yhat: Predicted mean on modeled scale (= level)
- sd_y: Standard deviation for observation on modeled scale
- level: Level on modeled (inv-probit) scale
- level_prop: Level on proportion scale
- sd_y_prop: Standard deviation on proportion scale
- y_sim: Simulated observation (validation runs only)

get_se_invprobitprop *Get Standard Error on invProbit Scale from Proportion*

Description

Transforms standard error from proportion scale to invprobit scale using the delta method. Uses the derivative of the inverse probit function: SE_probit = SE_prop / dnorm(qnorm(prop))

Usage

```
get_se_invprobitprop(prop, se_prop)
```

Arguments

prop	Proportion value (between 0 and 1)
se_prop	Standard error on proportion scale

Value

Standard error on probit scale

get_se_logitprop *Get Standard Error on Logit Scale*

Description

Transforms standard error from proportion scale to logit scale using the delta method.

Usage

```
get_se_logitprop(prop, se_prop)
```

Arguments

prop	Proportion value (between 0 and 1)
se_prop	Standard error on proportion scale

Value

Standard error on logit scale

get_se_probitofinvprobitprop *Get Standard Error on Proportion Scale from invProbit proportion*

Description

Transforms standard error from invprobit scale back to proportion scale using the delta method. Uses the derivative of the probit function: SE_prop = SE_invprobit * dnorm(invprobit_value)

Usage

```
get_se_probitofinvprobitprop(invprobitprop, se_invprobitprop)
```

Arguments

invprobitprop	Value on invprobit scale (i.e., qnorm of proportion)
se_invprobitprop	Standard error on invprobit scale

Value

Standard error on proportion scale

get_standata_routine *get_standata_routine*

Description

get_standata_routine

Usage

```
get_standata_routine(routine_data, fit_routine_obj, time_index, geo_unit_index)
```

Arguments

routine_data	tibble with iso, year, indicator_name, routine_value, countdownmean routine_value is the value of the routine data (eg coverage) and countdownmean is the data quality indicator for that iso-year-indicator combi needs to be filtered to iso of interest
fit_routine_obj	brm-fit object
time_index	from the model fit
geo_unit_index	from the model fit

Value

list with dat_routine (for plotting) and routine_list (to pass to stan)

`inv_probit` *Inverse Probit Transform (Proportion to InvProbit)*

Description

Transforms values from proportion scale to inv-probit scale using the inverse standard normal CDF.

Usage

```
inv_probit(x)
```

Arguments

x	Value on proportion scale (between 0 and 1)

Value

Value on inv-probit scale

`plot_estimates_local_all` *Plot model fits, with options to add routine data or additional fits for comparison.*

Description

Plot model fits, with options to add routine data or additional fits for comparison.

Usage

```
plot_estimates_local_all(
  results,
  dat_routine = NULL,
  indicator_name = NULL,
  results2 = NULL,
  results3 = NULL,
  results4 = NULL,
  modelnames = c("model1", "model2", "model3", "model4"),
  iso_codes = NULL,
  save_plots = FALSE,
  output_folder = NULL,
  add_caption = FALSE,
```

```
  add_estimates = TRUE,
  use_for_facetting = FALSE,
  plot_name = "fit"
)
```

Arguments

results	Model fit.
dat_routine	Optional routine data to add to model estimates for comparison.
indicator_name	Custom title for the y label of the plot. If NULL, name will be pulled from model fit.
results2	Optional second model fit.
results3	Optional third model fit.
results4	Optional fourth model fit.
modelnames	Optional names of the models to display in the plot legend.
iso_codes	Optional character vector of ISO codes to plot. If NULL, plots all countries.
save_plots	Boolean indicator, if set to TRUE plots will be saved in output directory of results fit.
output_folder	Folder to save plots in, if save_plots is TRUE, to overwrite where plots are saved.
plot_name	Plot name if saved as pdf. Defaults to "fit".

---

plot_hierchecks *Plot hierarchical model checks*

Description

Creates diagnostic plots for hierarchical model parameters including prior-posterior comparisons and mu_raw summaries.

Usage

```
plot_hierchecks(fit, model_name = "spline")
```

Arguments

fit	A fitted model object from fit_model
model_name	Character string specifying the model type. Either "spline" (checks Omega, Ptilde, Betas) or "rw2" (checks Omega, gamma). Default is "spline".

Value

NULL (invisibly). Saves hierchecks.pdf to the output directory.

probit *Probit Transform (Probit to Proportion)*

Description

Transforms values from probit scale to proportion scale using the standard normal CDF.

Usage

```
probit(x)
```

Arguments

x	Value on probit scale

Value

Value on proportion scale (between 0 and 1)

process_data *Process survey data for a selected indicator*

Description

Process survey data for a selected indicator

Usage

```
process_data(dat, regions_dat, indicator = "ideliv", verbose = TRUE)
```

Arguments

dat	data frame with iso, year, indic, r (indicator), se, source, final_year columns
regions_dat	data frame with iso and cluster columns
indicator	character, default "ideliv"
verbose	logical, whether to print messages about data processing

Value

processed data frame

rename_output_folder *Rename output folder and directory in fit object*

Description

Renames an existing folder in bayestransition_outputand updates the output_dir in the model fit object.

Usage

```
rename_output_folder(indicator, run_step, old_folder_name, new_folder_name)
```

Arguments

indicator	Character. Indicator name.
run_step	Character. Run step - either 1a, 1b, local_national.
old_folder_name	Character. The current folder name (inside bayestransition_output).
new_folder_name	Character. The new name to rename the folder to.

routinedata_add_roc_info *Calculate Rate of Change and DQ Indicators*

Description

This function calculates rate of change metrics and data quality (DQ) indicators for routine coverage data. It computes year-over-year changes and creates "start" and "worst" versions of DQ covariates for consecutive year pairs.

Usage

```
routinedata_add_roc_info(
  data,
  add_est_roc = FALSE,
  dq_covariates_min = c("countdownmean"),
  dq_covariates_max = c(),
  dq_covariates_max_abs = c()
)
```

Arguments

`data`	A data frame containing routine data with columns: iso (country code), year, indicator_name, routine_value, and optionally median_estimate. Must also contain the DQ covariate columns specified in the dq_covariates_* parameters.
`add_est_roc`	Logical. If TRUE, calculates rate of change for median_estimate in addition to routine_value. Default is FALSE.
`dq_covariates_min`	Character vector of DQ covariate names where "worst" means minimum value between current and start. Defaults to c("countdownmean").
`dq_covariates_max`	Character vector of DQ covariate names where "worst" means maximum value between current and start. Default is empty vector.
`dq_covariates_max_abs`	Character vector of DQ covariate names where "worst" means maximum absolute value between current and start. Default is c().

Value

A data frame with the original columns plus:

- routine_roc: Year-over-year change in routine_value (always added)
- est_roc: Year-over-year change in median_estimate (if add_est_roc = TRUE)
- posterior::var*start: Lagged value of each DQ covariate for consecutive years \item worst*posterior::var: Worst (min or max) value between current and start for each DQ covariate
- worst_abs_posterior::var: Worst (max absolute) value for covariates in dq_covariates_max_abs

Note: Rate of change and DQ values are only calculated for consecutive years (i.e., when year - lag(year) == 1), otherwise NA.

Examples

```
library(dplyr)

# Create toy data
toy_data <- data.frame(
  iso = rep(c("CD1", "CD2"), each = 3),
  year = rep(2020:2022, 2),
  indicator_name = "dtp3",
  routine_value = c(0.85, 0.87, 0.90, 0.80, 0.82, 0.85),
  median_estimate = c(88, 89, 91, 83, 84, 87),
  notoutliers = c(100, 95, 90, 100, 100, 95),
  notmissing = c(100, 100, 95, 100, 95, 90),
  rr = c(95, 90, 85, 90, 88, 85),
  diff_level = c(3, 2, 1, 3, 2, 2)
)

# Apply function
result <- routinedata_add_roc_info(
  toy_data,
```

```
  add_est_roc = TRUE,
  dq_covariates_min = c("notoutliers", "notmissing", "rr"),
  dq_covariates_max_abs = c("diff_level")
)

# View results for CD1
result %>% filter(iso == "CD1") %>% select(year, routine_value, routine_roc, rr, worst_rr)
```

`write_model` *Write Stan models for different model settings.*

Description

Write Stan models for different model settings.

Usage

```
write_model(add_aggregates = FALSE, add_routine = FALSE)
```

Arguments

`add_aggregates`	Add national aggregates? makes sense only for subnational all-region run for 1 country
`add_routine`	Add routine data?

Value

writes a stan file to the stan directory, naming ...

Index