Package 'MAPCtools'

July 21, 2025

Title Multivariate Age-Period-Cohort (MAPC) Modeling for Health Data

Version 0.1.0

Description Bayesian multivariate age-period-cohort (MAPC) models for analyzing health data, with support for model fitting, visualization, stratification, and model comparison. Inference focuses on identifiable cross-strata differences, as described by Riebler and Held (2010) <doi:10.1093/biostatistics/kxp037>. Methods for handling complex survey data via the 'survey' package are included, as described in Mercer et al. (2014) <doi:10.1016/j.spasta.2013.12.001>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Imports dplyr, tidyselect, fastDummies, stringr, rlang, tidyr, ggplot2, viridis, scales, purrr, grid, gridExtra, ggpubr, tibble, survey

URL https://github.com/LarsVatten/MAPCtools

BugReports https://github.com/LarsVatten/MAPCtools/issues

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

Additional_repositories https://inla.r-inla-download.org/R/stable/

Config/testthat/edition 3

Depends R (>= 3.5)

NeedsCompilation no

Author Lars Vatten [aut, cre]

Maintainer Lars Vatten <lavatt99@gmail.com>

Repository CRAN

Date/Publication 2025-06-25 15:40:02 UTC

Contents

`aggregate_df` *Aggregate data across an entire data frame using sufficient statistics*

Description

Aggregates specified columns of a data frame into summarizing statistics, preserving the potentially complex structure returned by aggregator functions (like data frames or inla.mdata objects) within list-columns. Aggregation is performed according to sufficient statistics for the specified distribution of the columns. Possible distributions: Gaussian, binomial. This function aggregates the entire data frame into a single row result.

Usage

```
aggregate_df(
  data,
  gaussian = NULL,
  gaussian.precision.scales = NULL,
  binomial = NULL
)
```

Arguments

`data`	A data frame.
`gaussian`	Gaussian columns in `data` to be aggregated. The Gaussian observations are collapsed into an `inla.mdata` object compatible with the `agaussian` family, see the documentation for the `agaussian` family in `INLA` for details. **Defaults to** `NULL` **(optional).**

gaussian.precision.scales
: Scales for the precision of Gaussian observations.
Must be one of:

  - NULL: Use default scales of 1 for all observations in all gaussian columns.
  - A single numeric vector: Applied only if *exactly one* column is specified in gaussian. Length must match nrow(data).
  - A *named list*: Where names(gaussian.precision.scales) are the names of the Gaussian columns (must match columns specified in gaussian). Each list element must be a numeric vector of scales for that column, with length matching nrow(data).
  Defaults to NULL (optional).

binomial
: Binomial columns in data to be aggregated. **Defaults to** NULL **(optional).**

Value

A single-row data frame (tibble) containing:

- A column n with the total number of rows in the input data.
- For each specified column in gaussian, binomial, a corresponding *list-column* (named e.g., colname_gaussian, colname_binomial. Each element of these list-columns can be accessed by using the $ operator twice, e.g. through data$colname_gaussian$Y1 for the first element of the Gaussian summary.

---

as.APC.df *Add 1-indexed APC columns to data frame, handling numeric or categorical age/period*

Description

Add 1-indexed APC columns to data frame, handling numeric or categorical age/period

Usage

```
as.APC.df(data, age, period, age_order = NULL, period_order = NULL, M = 1)
```

Arguments

data	Data frame with age and period columns.
age	Age column in data.
period	Period column in data.
age_order	(Optional) Character vector giving the desired order of age levels. If NULL and the age column is factor/character, uses unique(sort(data[[age]])).
period_order	(Optional) Vector (numeric or character) giving the desired order of periods. If NULL and period column is a factor/character, uses unique(sort(data[[period]])).
M	Grid factor, defined as the ratio of age interval width to period interval width. Defaults to 1 (i.e. assuming equal sized age and period increments).

Value

The data frame with new columns \code{age_index}, \code{period_index}, \code{cohort_index}, and sorted by \code{(age_index, period_index)}.

`as.APC.NA.df` *Create NA structure across age, period and cohort groups based on strata*

Description

Creates a data frame where age, period, and cohort values are placed into columns specific to their stratum (defined by `stratify_var`), with other strata combinations marked as NA. This structure is often useful for specific modeling approaches, like certain Age-Period-Cohort (APC) models. Optionally includes unique indices for random effects.

Usage

```
as.APC.NA.df(data, stratify_by, age, period, cohort, include.random = FALSE)
```

Arguments

`data`	Data frame with age, period, cohort, and stratification columns.
`stratify_by`	Stratification variable column. This column will be used to create the stratum-specific NA structure. It should ideally be a factor or character vector.
`age`	Age column in `data` (must be a numeric/integer column).
`period`	Name of the period column (must be a numeric/integer column).
`cohort`	Name of the cohort column (must be a numeric/integer column).
`include.random`	Logical. Whether to include a unique index ('random') for each combination of age, period, and stratum, potentially for use as random effect identifiers in models. **Defaults to FALSE.**

Value

A data frame containing the original `age`, `period`, `cohort`, and `stratify_by` columns, plus:

- Dummy indicator columns for each level of `stratify_by` (e.g., `Region_North`, `Region_South` if `Region` was a stratifying variable).
- Stratum-specific age, period, and cohort columns (e.g., `age_Region_North`, `period_Region_North`, `cohort_Region_North`), containing the respective value if the row belongs to that stratum, and `NA` otherwise.
- If `include.random = TRUE`, a column named `random` with unique integer indices. The rows are ordered primarily by the stratification variable levels. This is useful for defining random components in MAPC models.

`fit_all_MAPC` *Fit all configurations of MAPC models using INLA*

Description

Fits all configurations of shared vs. stratum-specific time effects:

APc Shared age and period effects, stratum-specific cohort effects.

ApC Shared age and cohort effects, stratum-specific period effects.

aPC Shared period and cohort effects, stratum-specific age effects.

Apc Shared age effects, stratum-specific period and cohort effects.

aPc Shared period effects, stratum-specific age and cohort effects.

apC Shared cohort effects, stratum-specific age and period effects.

Uses the `fit_MAPC` function. The multivariate APC model is based on Riebler and Held (2010) doi:10.1093/biostatistics/kxp037. For handling complex survey data, we follow Mercer et al. (2014) doi:10.1016/j.spasta.2013.12.001, implemented using the **survey** package.

Usage

```
fit_all_MAPC(
  data,
  response,
  family,
  stratify_by,
  reference_strata = NULL,
  age = "age",
  period = "period",
  grid.factor = 1,
  all_models = c("apC", "aPc", "Apc", "aPC", "ApC", "APc"),
  extra.fixed = NULL,
  extra.random = NULL,
  extra.models = NULL,
  extra.hyper = NULL,
  apc_prior = "rw1",
  include.random = FALSE,
  binomial.n = NULL,
  poisson.offset = NULL,
  apc_hyperprior = NULL,
  survey.design = NULL,
  control.compute = list(dic = TRUE, waic = TRUE, cpo = TRUE),
  track.progress = FALSE,
  verbose = FALSE
)
```

Arguments

data	A data frame containing the age, period, response, and stratification variables. Age and period are assumed to be on the raw scale, not transformed to 1-indexed index columns. Factor/character columns are handled, as long as they are properly sorted by sort(unique(data$age/period)) (e.g. values of the form "20-25" for age groups are handled).
response	A string naming the response (outcome) variable in data.
family	A string indicating the likelihood family. The default is "gaussian" with identity link.
stratify_by	The column in data to use for stratification.
reference_strata	Level of stratify_by to set as the reference level.
age	The age column in data.
period	The period column in data.
grid.factor	(Optional) Grid factor, defined as the ratio of age interval width to period interval width; defaults to 1.
all_models	(Optional) Character vectors of valid APC-formats (e.g. c("ApC", "apC", "APc")), specifying the MAPC models to be estimated. Requirements for a valid APC-format (lowercase letter means stratum-specific, uppercase means shared): - Only one time effect: shared/stratum-specific both fine. - Two time effects: shared/stratum-specific both fine. - Three time effects: either one or two must be stratum-specific. Defaults to c("apC", "aPc", "Apc", "aPC", "ApC", "APc").
extra.fixed	(Optional) If desired, the user can specify additional fixed effects to be added. This is passed as a character argument, specifying the name of the variable to be added. Multiple variables can be added by passing a character vector of names. Defaults to NULL.
extra.random	(Optional) If desired, the user can specify additional random effects to be added. This is passed as a character argument, specifying the name of the variable to be added. Multiple variables can be added by passing a character vector of names. Defaults to NULL.
extra.models	(Optional) If the user specifies one or more additional random effects to be added in extra.random, this argument can be used to specify the model to be used for the additional random effects. Either passed as a single string, in which case all extra random effects are assigned the same model, or a character vector matching the length of extra.ranom, mapping unique models to each variable in extra.random. If NULL and extra.random is non-empty, all extra random effects are assigned the "iid" model in inla(). Defaults to NULL.
extra.hyper	(Optional) If the user specifies one or more additional random effects to be added in extra.random, this argument can be used to specify the priors of the hyperparameters of the models used for the random effects. The hyperpriors are specified as strings that can be passed directly to the hyper=... argument in the formula passed to the inla()-function. See the argument apc_prior below for a concrete example. Defaults to NULL, in which case the default INLA priors are used.

`apc_prior` (Optional) A string specifying the prior for the age, period, and cohort effects (e.g. `"rw1"`, `"rw2"`). Defaults to `"rw1"`.

`include.random` (Optional) Logical; if `TRUE`, include an overall random effect in the APC model. Defaults to `FALSE`.

`binomial.n` (Optional) For the `family=binomial` likelihood. Either an integer giving the number of trials for the binomial response, or the name of the column containing the number of trials for each observation.

`poisson.offset` (Optional) For the `family=poisson` likelihood. Either an integer giving the denominator for the Poisson count response, or the name of the column containing the denominator for each observation.

`apc_hyperprior` (Optional) If the user wants non-default hyperpriors for the time effects, this can be achieved by passing the entire prior specification as a string. If e.g. `hyper = list(theta = list(prior="pc.prec", param=c(0.5,0.01)))` is desired, pass the string `"list(theta = list(prior="pc.prec", param=c(0.5,0.01)))"` to this argument.

`survey.design` (Optional) In the case of complex survey data, explicit handling of unequal sampling probabilities can be required. The user can pass a `survey.design` object created with the `svydesign` function from the **survey** package. In this case, a Gaussian model is fit for the survey adjusted estimates, based on the asymptotic normality of Hájek estimator. The argument `family` should still indicate the underlying distribution of the response, and based on this, an appropriate transformation is applied to the adjusted mean estimates.

`control.compute` (Optional) A list of control variables passed to the `inla()`-function, that specifies what to be computed during model fitting. See options for `control.compute` in the `INLA` docs. Defaults to `list(dic=TRUE, waic=TRUE, cpo=TRUE)`. If posterior sampling is desired, `config=TRUE` must be passed as a control option inside `control.compute`.

`track.progress` (Optional) Whether to report progress of the estimation of models in the console; defaults to `FALSE`.

`verbose` (Optional) This is argument is passed along to the `inla()` function that estimates the MAPC model. If `verbose=TRUE`, the `inla`-program runs in verbose mode, which can provide more informative error messages.

Details

The returned object is of class `all_mapc`, which is a container for multiple `mapc` model fits (each typically fitted with a different APC formats). It also contains a `model_selection` element, which holds plots summarizing comparative fit metrics (DIC, WAIC and log-scores).

The following S3 methods are available:

- `print()`: Prints a compact summary for each individual model fit.
- `summary()`: Calls `summary()` on each contained `mapc` object, providing detailed posterior summaries.
- `plot()`: Displays model comparison plots (DIC/WAIC/log-score comparisons).

These methods are intended to streamline multi-model workflows and allow quick comparison of results across model specifications.

Value

A named list of mapc objects, one for each configuration of shared vs. stratum-specific time effects: APc, ApC, aPC, Apc, aPc, apC.

References

Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using Integrated Nested Laplace Approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2), 319-392. doi:10.1111/j.14679868.2008.00700.x See also https://www.r-inla.org for more information about the INLA method and software.

See Also

fit_MAPC for fitting a single model (more flexible; can pass your own formula and lincombs), and the function inla() from the INLA package for the estimation machinery. For complex survey data, see svydesign for the creation of a survey design object which can be passed to survey.design.

Examples

```
data("toy_data")
fits <- fit_all_MAPC(
  data               = toy_data,
  response           = count,
  family             = "poisson",
  stratify_by        = education,
  reference_strata   = 1,
  age                = age,
  period             = period,
  apc_prior          = "rw2",
  include.random     = TRUE
)

# Print concise summary of the models and estimation procedure
print(fits)

# Plot comparison plots, based on comparative fit metrics
plot(fits)

# Optional: view full summary of all models (can be long)
# summary(fits)
```

---

fit_MAPC *Fit a multivariable age-period-cohort model*

Description

Fit a Bayesian multivariate age-period-cohort model, and obtain posteriors for identifiable cross-strata contrasts. The method is based on Riebler and Held (2010) doi:10.1093/biostatistics/kxp037. For handling complex survey data, we follow Mercer et al. (2014) doi:10.1016/j.spasta.2013.12.001, implemented using the **survey** package.

Usage

```
fit_MAPC(
  data,
  response,
  family,
  apc_format,
  stratify_by,
  reference_strata = NULL,
  age,
  period,
  grid.factor = 1,
  apc_prior = "rw1",
  extra.fixed = NULL,
  extra.random = NULL,
  extra.models = NULL,
  extra.hyper = NULL,
  include.random = FALSE,
  binomial.n = NULL,
  poisson.offset = NULL,
  inla_formula = NULL,
  lincombs = NULL,
  survey.design = NULL,
  apc_hyperprior = NULL,
  control.compute = list(dic = TRUE, waic = TRUE, cpo = TRUE),
  verbose = FALSE
)
```

Arguments

`data`	A data frame containing the age, period, response, and stratification variables. Age and period are assumed to be on the raw scale, not transformed to 1-indexed index columns. Factor/character columns are handled, as long as they are properly sorted by `sort(unique(data$age/period))` (e.g. values of the form "20-25" for age groups are handled).
`response`	A string naming the response (outcome) variable in `data`.
`family`	A string indicating the likelihood family. The default is `"gaussian"` with identity link. See `names(inla.models()$likelihood)` for a list of possible alternatives and use `inla.doc()` for detailed docs for individual families.
`apc_format`	A specification of the APC structure, with options: **APc** Shared age and period effects, stratum-specific cohort effects.

ApC Shared age and cohort effects, stratum-specific period effects.

aPC Shared period and cohort effects, stratum-specific age effects.

Apc Shared age effects, stratum-specific period and cohort effects.

aPc Shared period effects, stratum-specific age and cohort effects.

apC Shared cohort effects, stratum-specific age and period effects.

Note: It is also possible to specify models with only one or two time effects, by omitting the letters corresponding to the time effects to be excluded.

`stratify_by` A string naming the column in `data` to use for stratification (e.g. region or sex).

`reference_strata`
Level of `stratify_by` to set as the reference level.

`age` Name of the age variable in `data`.

`period` Name of the period variable in `data`.

`grid.factor` (Optional) Grid factor, defined as the ratio of age interval width to period interval width; defaults to 1.

`apc_prior` (Optional) A string specifying the prior for the age, period, and cohort effects (e.g. `"rw1"`, `"rw2"`). Defaults to `"rw1"`.

`extra.fixed` (Optional) If desired, the user can specify additional fixed effects to be added. This is passed as a character argument, specifying the name of the variable to be added. Multiple variables can be added by passing a character vector of names. Defaults to `NULL`.

`extra.random` (Optional) If desired, the user can specify additional random effects to be added. This is passed as a character argument, specifying the name of the variable to be added. Multiple variables can be added by passing a character vector of names. Defaults to `NULL`.

`extra.models` (Optional) If the user specifies one or more additional random effects to be added in `extra.random`, this argument can be used to specify the model to be used for the additional random effects. Either passed as a single string, in which case all extra random effects are assigned the same model, or a character vector matching the length of `extra.ranom`, mapping unique models to each variable in `extra.random`. If `NULL` and `extra.random` is non-empty, all extra random effects are assigned the `"iid"` model in `inla()`. Defaults to `NULL`.

`extra.hyper` (Optional) If the user specifies one or more additional random effects to be added in `extra.random`, this argument can be used to specify the priors of the hyperparameters of the models used for the random effects. The hyperpriors are specified as strings that can be passed directly to the `hyper=...` argument in the formula passed to the `inla()`-function. See the argument `apc_prior` below for a concrete example. Defaults to `NULL`, in which case the default `INLA` priors are used.

`include.random` (Optional) Logical; if `TRUE`, include an overall random effect in the APC model, to capture unobserved heterogeneity. Defaults to `FALSE`.

`binomial.n` (Optional) For the `family=binomial` likelihood. Either an integer giving the number of trials for the binomial response, or the variable in `data` containing the number of trials for each observation.

`poisson.offset` (Optional) For the `family=poisson` likelihood. Either an integer giving the denominator for the Poisson count response, or the variable in `data` containing the denominator for each observation.

`inla_formula` (Optional) If desired, the user can pass its own INLA-compatible formula to define the model. If not, a formula is generated automatically, with the models and priors defined.

`lincombs` (Optional) If desired, the user can pass its own INLA-compatible linear combinations to be computed by the `inla` program. See the `inla()`-function or `f()`-function documentations in `INLA` for details.

`survey.design` (Optional) In the case of complex survey data, explicit handling of unequal sampling probabilities can be required. The user can pass a `survey.design` object created with the `svydesign` function from the **survey** package. In this case, a Gaussian model is fit for the survey adjusted estimates, based on the asymptotic normality of Hájek estimator. The argument `family` should still indicate the underlying distribution of the response, and based on this, an appropriate transformation is applied to the adjusted mean estimates.

`apc_hyperprior` (Optional) If the user wants non-default hyperpriors for the time effects, this can be achieved by passing the entire prior specification as a string. If e.g. `hyper = list(theta = list(prior="pc.prec", param=c(0.5,0.01)))` is desired, pass the string `"list(theta = list(prior="pc.prec", param=c(0.5,0.01)))"` to this argument.

`control.compute` (Optional) A list of control variables passed to the `inla()`-function, that specifies what to be computed during model fitting. See options for `control.compute` in the `INLA` docs. Defaults to `list(dic=TRUE, waic=TRUE, cpo=TRUE)`. If posterior sampling is desired, `config=TRUE` must be passed as a control option inside `control.compute`.

`verbose` (Optional) This is argument is passed along to the `inla()` function that estimates the MAPC model. If `verbose=TRUE`, the `inla`-program runs in verbose mode, which can provide more informative error messages.

Details

This function works as a wrapper around the `inla()`-function from the `INLA` package, which executes the model fitting procedures using Integrated Neste Laplace Approximations.

The returned object is of class `mapc`. S3 methods are available for:

- `print()`: Displays a concise summary of the model, including the APC format used, CPU time, number of estimated parameters (fixed, random, hyperparameters, linear combinations), and model fit scores (DIC, WAIC, log-score).
- `summary()`: Prints detailed posterior summaries of all estimated components, including fixed effects, random effects, hyperparameters, and linear combinations, as estimated by the `inla()`-function.
- `plot()`: Visualizes model estimates of cross-stata contrast trends, using precomputed plots stored in the object. The available plots depends on the APC-format that was used. You can control which effects to plot using the `which` argument (e.g. `which="age"` or `which=c("age", "period")`).

Value

An named list, containing the following arguments:

`model_fit` An object of class `"inla"`, containing posterior densities, posterior summaries, measures of model fit etc. See documentation for the `inla()`-function for details.

`plots` A named list of plots for each time effect. Extract them as `plots\$age/plots\$periodplots\$cohort`.

References

Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using Integrated Nested Laplace Approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2), 319-392. doi:10.1111/j.14679868.2008.00700.x See also https://www.r-inla.org for more information about the INLA method and software.

See Also

`fit_all_MAPC` for fitting multiple models at once, and the function `inla()` from the `INLA` package for the estimation machinery. For complex survey data, see `svydesign` for the creation of a survey design object which can be passed to `survey.design`.

Examples

```
data("toy_data")
fit <- fit_MAPC(
  data               = toy_data,
  response           = count,
  family             = "poisson",
  apc_format         = "ApC",
  stratify_by        = education,
  reference_strata   = 1,
  age                = age,
  period             = period
)

# Print concise summary of the MAPC fit and the estimation procedure
print(fit)

# Plot estimated cross-strata contrast trends
plot(fit)

# Optional: view full summary of the model (can be long)
# summary(fit)
```

`generate_apc_lincombs` *Generate Age-Period-Cohort Linear Combinations for INLA*

Description

Constructs a set of linear combinations (contrasts) for age, period, and/or cohort effects across different strata, relative to a specified reference strata, suitable for use with `inla.make.lincomb` from the `INLA` package.

Usage

```
generate_apc_lincombs(
  apc_format,
  data,
  strata,
  reference_strata,
  age = "age",
  period = "period",
  cohort = "cohort"
)
```

Arguments

`apc_format`	Character string containing any combination of `"a"`, `"p"`, `"c"`: `"a"` include age contrasts `"p"` include period contrasts `"c"` include cohort contrasts e.g. `"ap"` to generate age and period contrasts only.
`data`	A `data.frame` containing the variables specified by `age`, `period`, `cohort`, and `strata`. The age, period, and cohort variables must be integer-valued (or coercible to integer).
`strata`	String giving the name of the factor column in `data` that defines strata.
`reference_strata`	String indicating which level of `strata` should be used as the reference.
`age`	String name of the column in `data` containing age indices (default `"age"`).
`period`	String name of the column in `data` containing period indices (default `"period"`).
`cohort`	String name of the column in `data` containing cohort indices (default `"cohort"`).

Details

For each specified dimension (`a`, `p`, `c`), the function loops over all unique values of age, period, or cohort in the data, and over all strata levels except the reference. It then constructs a contrast that subtracts the effect in the reference stratum from the effect in the other strata at each index.

Value

A named `list` of linear combination objects as returned by `inla.make.lincomb()` (`INLA` function). Each element corresponds to one contrast, with names of the form "Age = x, Strata = y vs ref", "Period = x, Strata = y vs ref", or "Cohort = x, Strata = y vs ref", depending on `apc_format`.

`generate_MAPC_formula` *Generate MAPC formula for INLA*

Description

Based on APC-format, generate the proper formula to pass to INLA for fitting MAPC models.

Usage

```
generate_MAPC_formula(
  df,
  APC_format,
  response,
  stratify_var,
  age = "age",
  period = "period",
  cohort = "cohort",
  intercept = FALSE,
  apc_prior = "rw1",
  apc_hyper = NULL,
  random_term = TRUE,
  extra.fixed = NULL,
  extra.random = NULL,
  extra.models = NULL,
  extra.hyper = NULL
)
```

Arguments

`df`	Data frame for which MAPC models should be fit
`APC_format`	A string where lower-case letters indicate stratum-specific time effects and upper-case letters indicate shared time effects.
`response`	A string, name of the column in `df` that represents the response variable.
`stratify_var`	Stratification variable. At least one time effect should be stratum-specific, and at least one should be shared.
`age`	Name of age column
`period`	Name of period column
`cohort`	Name of cohort column
`intercept`	Boolean, indicating if an overall intercept should be included in the formula. **Defaults to TRUE (optional).**

apc_prior	Which prior model to use for the time effects. **Defaults to "rw1" (optional).**
apc_hyper	If the user wants non-default hyperpriors for the random time effects, this can be achieved by passing the entire prior specification as a string. If e.g. `hyper = list(theta = list(prior="pc.prec", param=c(0.5,0.01)))` is desired, pass the string `"list(theta = list(prior="pc.prec", param=c(0.5,0.01)))"` to this argument.
random_term	Indicator, indicating if a random term should be included in the model. **Defaults to TRUE (optional).**
extra.fixed	Name of additional fixed effects. **Defaults to NULL (optional).**
extra.random	Name of additional random effects. **Defaults to NULL (optional).**
extra.models	Models for additional random effects. Supported `INLA` models include `'iid'`, `'rw1'` and `'rw2'`. **Defaults to NULL (optional).**
extra.hyper	If the user wants non-default hyperpriors for the additional random effects, this can be achieved by passing the entire prior specification as a string. If e.g. `hyper = list(theta = list(prior="pc.prec", param=c(0.5,0.01)))` is desired, pass the string `"list(theta = list(prior="pc.prec", param=c(0.5,0.01)))"` to this argument.

Value

A formula object that can be passed to INLA to fit the desired MAPC model.

`plot_binned_counts` *Plot counts of observations across bins of a numeric variable, optionally stratified*

Description

Bins a specified numeric variable into intervals, counts observations per value of a specified variable and bin groups, and plots lines for each bin group using **ggplot2**. If a stratification variable is provided, counts are calculated per strata and plotted as separate colored lines. If an additional stratification variable is provided, separate plot windows are created for each level.

Usage

```
plot_binned_counts(
  data,
  x,
  bin_by,
  stratify_by = NULL,
  for_each = NULL,
```

```
  n_bins = 8,
  bin_width = NULL,
  title = "Observation counts",
  subtitle = NULL,
  legend_title = NULL,
  x_lab = NULL,
  y_lab = NULL,
  viridis_color_option = "D"
)
```

Arguments

data	Data frame containing all input variables.
x	Variable in data whose values define the x-axis for counts.
bin_by	Numeric variable in data for which to bin observations.
stratify_by	(Optional) Stratification variable. If supplied, counts are computed for each combination of x, bin of bin_by and stratify_by, and separate panels are made for each level of stratify_by.
for_each	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of for_each.
n_bins	(Optional) Number of bins to create across bin_by; defaults to 8.
bin_width	(Optional) Width of the bins created across bin_by; defaults to NULL. Overrides n_bins if both are supplied.
title	(Optional) Plot title; defaults to "Observation counts".
subtitle	(Optional) Plot subtitle; defaults to NULL if for for_each is NULL, defaults to <name of for_each>: <level of for_each> for each plot window if for_each is supplied.
legend_title	(Optional) Legend title; defaults to name of bin_by + "bin".
x_lab	(Optional) Label for the x-axis; defaults to the name of x.
y_lab	(Optional) Label for the y-axis; defaults to the name of y.
viridis_color_option	(Optional) Option for color gradient; defaults to "D". Options are "A", "B", "C", "D", E", "F", "G", "H". See **viridis** for information, or experiment yourself.

Value

If for_each is not supplied, a ggplot object showing counts per x and bin groups, optionally faceted by stratify_by. If for_each is supplied, a named list of such plots.

See Also

plot_counts_1D, plot_counts_2D, plot_counts_with_mean, ggplot

Examples

```
data("toy_data")
# Counts by period, binned by age
plot_binned_counts(toy_data, x = period,
                   bin_by = age, n_bins = 4)
# Counts by period, binned by age, stratified by education levels
plot_binned_counts(toy_data, period,
                   bin_by = age, n_bins = 4,
                   stratify_by = education)
# Counts by period, binned by age, stratified by education levels, for each sex
plot_binned_counts(toy_data, period,
                   bin_by = age, n_bins = 4,
                   stratify_by = education, for_each = sex)
```

---

plot_counts_1D *Plot counts of observations across a single variable, optionally stratified*

Description

Computes the number of observations at each value of a specified variable and creates a line plot of these counts using **ggplot2**. If a stratification variable is provided, counts are calculated per strata and plotted as separate colored lines. If an additional stratification variable is provided, separate plot windows are created for each level.

Usage

```
plot_counts_1D(
  data,
  x,
  stratify_by = NULL,
  for_each = NULL,
  title = "Observation counts",
  subtitle = NULL,
  legend_title = NULL,
  x_lab = NULL,
  y_lab = NULL,
  viridis_color_option = "D"
)
```

Arguments

`data`	Data frame containing all input variables.
`x`	Variable in `data` whose values define the x-axis for counts.
`stratify_by`	(Optional) Stratification variable. If supplied, counts are computed for each combination of `x` and `stratify_by`, and separate lines are drawn per level of `stratify_by`.

for_each	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of for_each.
title	(Optional) Plot title; defaults to "Observation counts".
subtitle	(Optional) Plot subtitle; defaults to NULL if for for_each is NULL, defaults to <name of for_each>: <level of for_each> for each plot window if for_each is supplied.
legend_title	(Optional) Legend title; defaults to name of stratify_var if it is supplied.
x_lab	(Optional) Label for the x-axis; defaults to the name of x.
y_lab	(Optional) Label for the y-axis; defaults to the name of y.
viridis_color_option	(Optional) Option for color gradient; defaults to "D". Options are "A", "B", "C", "D", E", "F", "G", "H". See **viridis** for information, or experiment yourself.

Value

A ggplot object displaying counts across the variable supplied in x, optionally stratified by stratify_by. If for_each is supplied, separate plots are created in separate windows for each level. Visuals can be modified with **ggplot2**.

See Also

plot_counts_2D, plot_binned_counts, plot_counts_with_mean, ggplot

Examples

```
data("toy_data")
# Counts by age
plot_counts_1D(toy_data, x = age)
# Counts by age, stratified by education level
plot_counts_1D(toy_data, x = age,
               stratify_by = education)
# Count by age, stratified by education level, for each sex
plot_counts_1D(toy_data, x = age,
               stratify_by = education, for_each = sex)
```

---

plot_counts_2D *Plot counts of observations across two dimensions, optionally stratified*

Description

Computes the number of observations for each combination of two specified variables, and displays the result as a heatmap using **ggplot2**. If a stratification variable is provided, counts are calculated per strata and strata-specific heatmaps are displayed in individual panels. If an additional stratification variable is provided, separate plot windows are created for each level.

Usage

```
plot_counts_2D(
  data,
  x,
  y,
  stratify_by = NULL,
  for_each = NULL,
  color_gradient = c("blue", "beige", "red"),
  title = "Observation counts",
  legend_title = NULL,
  subtitle = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

data	Data frame containing all input variables.
x	Variable in data whose values define the x-axis for counts.
y	Variable in data whose values define the y-axis for counts.
stratify_by	(Optional) Stratification variable. If supplied, counts are computed for each combination of x, y and stratify_by, and separate heatmaps are generated per level of stratify_by.
for_each	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of for_each.
color_gradient	(Optional) Color gradient for the heatmap. Specified as a character vector of three colors, representing: c(<low_counts>, <middle_counts>, <high_counts>). Defaults to c("blue", "beige", "red"). Colors must be recognized by ggplot.
title	(Optional) Plot title; defaults to "Observation counts".
legend_title	(Optional) Legend title for color gradient; defaults to "Count".
subtitle	(Optional) Plot subtitle; defaults to NULL if for_each is NULL, defaults to <name of for_each>: <level of for_each> for each plot window if for_each is supplied.
x_lab	(Optional) Label for the x-axis; defaults to the name of x.
y_lab	(Optional) Label for the y-axis; defaults to the name of y.

Value

If for_each is not supplied, a ggplot object showing a heatmap of counts for each x-y combination, optionally faceted by stratify_by. If for_each is supplied, a named list of such ggplot objects, one per unique value of for_each.

See Also

plot_counts_1D, plot_binned_counts, plot_counts_with_mean, ggplot

Examples

```
data("toy_data")
# Heatmap of counts by age and period
plot_counts_2D(toy_data, x = age, y = period)
# Heatmap of counts by age and period, stratified by education
plot_counts_2D(toy_data, x = period, y = age,
               stratify_by = education)
# Heatmap of counts by age and period, stratified by education, for each sex
plot_counts_2D(toy_data, x = period, y = age,
               stratify_by = education, for_each = sex)
```

---

plot_counts_with_mean *Plot heatmap of observation counts with mean overlay, optionally stratified*

Description

Computes counts of observations for each combination of two variables and displays them as a heatmap, with an overlaid line showing the mean of the second variable across the first. If a stratification variable is provided, observations are counted per strata and strata-specific heatmaps are displayed in individual panels. If an additional stratification variable is provided, separate plot windows are created for each level.

Usage

```
plot_counts_with_mean(
  data,
  x,
  y,
  stratify_by = NULL,
  for_each = NULL,
  title = NULL,
  subtitle = NULL,
  heatmap_legend = "Count",
  mean_legend = "Mean",
  viridis_color_option = "D",
  mean_color = "coral"
)
```

Arguments

data	Data frame containing all input variables.
x	Variable in data whose values define the x-axis for counts.
y	Variable in data whose values define the y-axis for counts, and for which the mean is computed for each value for x.

stratify_by	(Optional) Stratification variable. If supplied, counts and means are computed for each level of stratify_by, and separate heatmaps are generated per level of stratify_by.
for_each	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of for_each.
title	(Optional) Plot title; defaults to NULL, in which case a title of the form "\<Y\> distribution across \<X\>" is used.
subtitle	(Optional) Plot subtitle; defaults to NULL, or to "\<for_each\>: \<level\>" when for_each is supplied.
heatmap_legend	(Optional) Label for the heatmap legend; defaults to "Count".
mean_legend	(Optional) Label for the overlay mean line legend; defaults to "Mean".
viridis_color_option	(Optional) Option for color gradient; defaults to "D". Options are "A", "B", "C", "D", E", "F", "G", "H". See **viridis** for information, or experiment yourself.
mean_color	(Optional) Color for the overlay mean line; defaults to "coral".

Value

If for_each is not supplied, a ggplot object showing a heatmap of counts with a mean overlay line, optionally faceted by stratify_by. If for_each is supplied, a named list of such plots.

See Also

plot_counts_1D, plot_counts_2D, plot_binned_counts, ggplot

Examples

```
data("toy_data")
# Heatmap of counts by age vs. period with mean age overlay
plot_counts_with_mean(toy_data, x = period, y = age)
# Heatmap of counts by age vs. period with mean age overlay, stratified by education
plot_counts_with_mean(toy_data, x = period, y = age,
                      stratify_by = education)
# Heatmap of counts by age vs. period with mean age overlay, stratified by education, for each sex
plot_counts_with_mean(toy_data, x = period, y = age,
                      stratify_by = education, for_each = sex)
```

---

plot_lincombs *Plot Linear Combinations of Age-Period-Cohort Effects by Strata*

Description

Generates ggplot2 line plots of estimated linear combinations for age, period, and/or cohort effects from an INLA fit, stratified by a factor. Returns a named list of ggplot objects for each requested effect.

Usage

```
plot_lincombs(
  inla_fit,
  apc_model,
  data,
  strata_col,
  reference_level,
  family = NULL,
  age_ind = "age",
  period_ind = "period",
  cohort_ind = "cohort",
  age_title = NULL,
  period_title = NULL,
  cohort_title = NULL,
  y_lab = NULL,
  age_vals = NULL,
  period_vals = NULL,
  cohort_vals = NULL,
  age_breaks = NULL,
  age_limits = NULL,
  period_breaks = NULL,
  period_limits = NULL,
  cohort_breaks = NULL,
  cohort_limits = NULL,
  PDF_export = FALSE
)
```

Arguments

inla_fit	An object returned by the inla()-function, containing the data frame summary.lincomb.derived, which holds the posterior summaries of the cross strata contrasts from the MAPC model. This function assumes that the rownames of the linear combinations are of the specific format produced by generate_apc_lincombs.
apc_model	Character string indicating the configuration of shared vs. stratum-specific time effects in the model.
data	The data frame used to fit inla_fit, containing columns for age, period, cohort, and the stratification variable.
strata_col	Character name of the factor column in data defining strata.
reference_level	Character value of strata_col to use as the reference.
family	Optional character; if NULL, y_lab defaults to "Mean differences". If "gaussian", same; if "poisson", "Log mean ratio"; if "binomial", "Log odds ratio".
age_ind	Character name of the age variable in data (default "age").
period_ind	Character name of the period variable in data (default "period").
cohort_ind	Character name of the cohort variable in data (default "cohort").
age_title	Optional plot title for the age effect.

period_title	Optional plot title for the period effect.
cohort_title	Optional plot title for the cohort effect.
y_lab	Optional y-axis label; if NULL, set according to family.
age_vals	Optional numeric vector of x-values for age; defaults to min(data\$age):max(data\$age).
period_vals	Optional numeric vector of x-values for period; defaults to min(data\$period):max(data\$period).
cohort_vals	Optional numeric vector of x-values for cohort; defaults to min(data\$cohort):max(data\$cohort).
age_breaks	Optional vector of breaks for the age plot x-axis.
age_limits	Optional numeric vector of length 2 giving x-axis limits for age.
period_breaks	Optional vector of breaks for the period plot x-axis.
period_limits	Optional numeric vector of length 2 giving x-axis limits for period.
cohort_breaks	Optional vector of breaks for the cohort plot x-axis.
cohort_limits	Optional numeric vector of length 2 giving x-axis limits for cohort.
PDF_export	Logical; if TRUE, use larger font sizes/layout for PDF output.

Value

A named list of ggplot objects. Elements are "age", "period", and/or "cohort" depending on apc_model.

Examples

```
if (requireNamespace("INLA", quietly = TRUE)) {
  # Load toy dataset
  data("toy_data")

  # Filter away unobserved cohorts (see plot_missing_data() function):
  require(dplyr)
  toy_data.f <- toy_data %>% filter(sex == "female") %>% subset(cohort > 1931)

  # Load precomputed 'mapc' object
 apC_fit.f <- readRDS(system.file("extdata", "quickstart-apC_fit_f.rds", package = "MAPCtools"))

  # Extract INLA object:
  apC_fit.inla <- apC_fit.f$model_fit
  apC_plots <- plot_lincombs(
    inla_fit    = apC_fit.inla,
    apc_model   = "apC",
    data        = toy_data.f,
    strata_col  = "education",
    reference_level = "1",
    family      = "poisson",

  )
  # Display the age effect plot
  print(apC_plots$age)
  # Display the period effect plot
  print(apC_plots$period)
}
```

`plot_mean_response_1D` *Plot mean of a response variable across a single variable, optionally stratified*

Description

Computes the mean of a specified response variable at each value of a specified x variable and displays a line plot using **ggplot2**. If a stratification variable is provided, means are calculated per strata and plotted as separate colored lines. If an additional stratification variable is provided, separate plot windows are created for each level.

Usage

```
plot_mean_response_1D(
  data,
  response,
  x,
  stratify_by = NULL,
  for_each = NULL,
  title = NULL,
  subtitle = NULL,
  legend_title = NULL,
  x_lab = NULL,
  y_lab = NULL,
  viridis_color_option = "D"
)
```

Arguments

`data`	A `data.frame` or tibble containing the dataset.
`response`	A numeric variable in `data` whose mean will be plotted.
`x`	A variable in `data` defining the x-axis for computing means.
`stratify_by`	(Optional) Stratification variable. If supplied, counts are computed for each combination of `x` and `stratify_by`, and separate lines are drawn per level of `stratify_by`.
`for_each`	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of `for_each`.
`title`	(Optional) Plot title; defaults to `"Observation counts"`.
`subtitle`	(Optional) Plot subtitle; defaults to `NULL` if for `for_each` is `NULL`, defaults to `<name of for_each>: <level of for_each>` for each plot window if `for_each` is supplied.
`legend_title`	(Optional) Legend title; defaults to name of `stratify_var` if it is supplied.
`x_lab`	(Optional) Label for the x-axis; defaults to the name of `x`.
`y_lab`	(Optional) Label for the y-axis; defaults to the name of `paste("Mean", <response_name>)`.

viridis_color_option
: (Optional) Option for color gradient; defaults to "D". Options are "A", "B", "C", "D", E", "F", "G", "H". See **viridis** for information, or experiment yourself.

Value

A ggplot object displaying the mean of the response across the variable supplied in x, optionally stratified by stratify_by. If for_each is supplied, separate plots are created in separate windows for each level. Visuals can be modified with **ggplot2**.

See Also

plot_mean_response_2D, ggplot

Examples

```
data("toy_data")
# Mean by age
plot_mean_response_1D(toy_data, response = count, x = age)
# Mean count by age, stratified by education
plot_mean_response_1D(toy_data, response = count, x = age,
                      stratify_by = education)
# Mean count by age, stratified by education, for each sex
plot_mean_response_1D(toy_data, response = count, x = age,
                      stratify_by = education, for_each = sex)
```

---

plot_mean_response_2D *Plot mean of a response variable across two dimensions, optionally stratified*

Description

Computes the mean of a specified response variable for each combination of two variables and displays it as a heatmap using **ggplot2**. If a stratification variable is provided, means are calculated per strata and strata-specific heatmaps are displayed in individual panels. If an additional stratification variable is provided, separate plot windows are created for each level.

Usage

```
plot_mean_response_2D(
  data,
  response,
  x,
  y,
  stratify_by = NULL,
  for_each = NULL,
  color_gradient = c("blue", "beige", "red"),
  title = NULL,
```

```
  subtitle = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

data	Data frame containing all input variables.
response	Numeric variable in data whose mean to compute.
x	Variable in data for the horizontal axis.
y	Variable in data for the vertical axis.
stratify_by	(Optional) Stratification variable. If supplied, means are computed for each combination of x, y and stratify_by, and separate heatmaps are generated per level of stratify_by.
for_each	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of for_each.
color_gradient	(Optional) Color gradient for the heatmap. Specified as a character vector of three colors, representing: c(<low_counts>, <middle_counts>, <high_counts>). Defaults to c("blue", "beige", "red"). Colors must be recognized by ggplot.
title	Plot title; defaults to NULL, in which case a title of the form "Mean <response>" is used.
subtitle	(Optional) Plot subtitle; defaults to NULL if for_each is NULL, defaults to <name of for_each>: <level of for_each> for each plot window if for_each is supplied.
x_lab	(Optional) Label for the x-axis; defaults to the name of x.
y_lab	(Optional) Label for the y-axis; defaults to the name of y.

Value

A ggplot object showing the mean of response across x and y, optionally faceted by facet_row and/or facet_col.

See Also

plot_mean_response_1D, ggplot

Examples

```
data("toy_data")
# Mean count by age and period
plot_mean_response_2D(toy_data, response = count, x = period, y = age)
# Mean count by age and period, stratified by education level
plot_mean_response_2D(toy_data, response = count, x = period, y = age,
                      stratify_by = education)
# Mean count by age and period, stratified by education level, for each sex
plot_mean_response_2D(toy_data, response = count, x = period, y = age,
                      stratify_by = education, for_each = sex)
```

`plot_missing_data` *Plot Missing Group Combinations*

Description

Creates a tile plot highlighting combinations of grouping variables that are expected but missing from the data. Allows for faceting.

Usage

```
plot_missing_data(
  data,
  x,
  y,
  stratify_by = NULL,
  for_each = NULL,
  facet_labeller = NULL,
  title = "Missing data",
  subtitle = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`data`	Data frame.
`x`	Variable in `data` whose values define the x-axis.
`y`	Variable in `data` whose values define the y-axis.
`stratify_by`	(Optional) Stratification variable. If supplied, missing data is examined separately for each leves of `stratify_by`, and each level gets its own panel.
`for_each`	(Optional) Additional stratification variable. If supplied, separate plot windows are created per level of `for_each`.
`facet_labeller`	A `labeller` function (e.g. `labeller`), or a named list where names match facet variables and values are named vectors/lists mapping levels to labels (optional).
`title`	Character string for the plot title. Defaults to "Missing data".
`subtitle`	Character string for the plot subtitle. Defaults to NULL.
`x_lab`	Character string for the x-axis label. Defaults to the name of `x_var`.
`y_lab`	Character string for the y-axis label. Defaults to the name of `y_var`.

Value

A ggplot object, or NULL if no missing combinations found.

See Also

ggplot

Examples

```
data("toy_data")

# Plot missing data across age and period, stratified by education, for each sex
plot_missing_data (data       = toy_data,
                   x          = period,
                   y          = age,
                   stratify_by = education,
                   for_each    = sex)
```

---

toy_data *Synthetic Age-Period-Cohort Dataset*

Description

A toy dataset generated to illustrate modeling of age, period, and cohort effects, including interactions with education and sex. This data simulates count outcomes (e.g., disease incidence or event counts) as a function of demographic variables using a Poisson process.

Usage

```
data(toy_data)
```

Format

A data frame with 10000 rows and 7 variables:

age Age of individuals, sampled uniformly from 20 to 59.

period Calendar year of observation, sampled uniformly from 1990 to 2019.

education Factor for education level, with levels 1, 2 and 3.

sex Factor indicating biological sex, with levels: "male", "female".

count Simulated event count, generated from a Poisson distribution.

known_rate The true Poisson rate used to generate `count`, computed from the log-linear model.

cohort Derived variable indicating year of birth (period - age).

Details

The underlying event rate is modeled on the log scale as a linear combination of age, period, sex, education, and an age-education interaction. The count outcome is drawn from a Poisson distribution with this rate. This dataset is handy for testing APC models.

The true log-rate is computed (for observation n) as:

$$\log(\lambda_n) = \beta_0 + \beta_{\text{period}} \left(2020 - \text{period}_n\right) + \beta_{\text{sex}} \, I(\text{sex}_n = \text{female}) \quad + \beta_{\text{edu}} \, (\text{edu level}_n) + \beta_{\text{edu-age}} \, (\text{age}_n - 20) \, (\text{edu level}_n - 1) \, I($$

where the rate decreases over time (periods), increases with age up to age 40, and decreases after. The coefficients used are:

- `intercept` = `1.0`
- `b_period` = `0.02`
- `b_sex` = `0.5` (female effect)
- `b_education_base` = `0.5`
- `b_education_age_interaction` = `0.015`

Source

Simulated data, created using base R and **tibble**.

`validate_lincombs_against_formula`
Validate lincomb terms against an INLA formula

Description

Checks that all variable names used in `inla.make.lincomb()` expressions (inside a list of lincombs) are present in the provided INLA model formula.

Usage

```
validate_lincombs_against_formula(lincombs, formula)
```

Arguments

`lincombs`	A list of linear combinations (as generated by `generate_apc_lincombs()`).
`formula`	The INLA model formula object (e.g., from `generate_MAPC_formula()`).

Value

Invisible TRUE if all terms match. Otherwise, stops with an informative error.

Index