Package: bage (via r-universe)

September 18, 2024

Type Package

Title Bayesian Estimation and Forecasting of Age-Specific Rates

Version 0.7.6

Description Fast Bayesian estimation and forecasting of age-specific rates, probabilities, and means, based on 'Template Model Builder'.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.3.0)

Imports cli, generics, Matrix, matrixStats, methods, parallel, poputils (>= 0.3.3), rvec (>= 0.0.7), stats, tibble, TMB, utils, vctrs

Suggests bookdown, dplyr, ggplot2, knitr, lifecycle, rmarkdown, testthat (>= 3.0.0), tidyr

Config/testthat/edition 3

RoxygenNote 7.3.2

LinkingTo TMB, RcppEigen

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

URL https://bayesiandemography.github.io/bage/, https://github.com/bayesiandemography/bage

BugReports https://github.com/bayesiandemography/bage/issues

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

RemoteUrl https://github.com/bayesiandemography/bage

RemoteRef HEAD

RemoteSha ab09632503d0b7443fd361c12d001763b0684322

Contents

`AR` *Autoregressive Prior*

Description

Use an autoregressive process to model a main effect, or use multiple autoregressive processes to model an interaction. Typically used with time effects or with interactions that involve time.

Usage

```
AR(n_coef = 2, s = 1, along = NULL)
```

Arguments

`n_coef`	Number of lagged terms in the model, ie the order of the model. Default is `2`.
`s`	Scale for the prior for the innovations. Default is `1`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `AR()` is used with an interaction, separate AR processes are constructed along the "along" variable, within each combination of the "by" variables.

By default, the autoregressive processes have order 2. Alternative choices can be specified through the `n_coef` argument.

Argument `s` controls the size of innovations. Smaller values for `s` tend to give smoother estimates.

Value

An object of class `"bage_prior_ar"`.

Mathematical details

When `AR()` is used with a main effect,

$$\beta_j = \phi_1 \beta_{j-1} + \cdots + \phi_n \beta_{j-n} + \epsilon_j$$
$$\epsilon_j \sim \text{N}(0, \omega^2),$$

and when it is used with an interaction,

$$\beta_{u,v} = \phi_1 \beta_{u,v-1} + \cdots + \phi_n \beta_{u,v-n} + \epsilon_{u,v}$$
$$\epsilon_{u,v} \sim \text{N}(0, \omega^2),$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction;
- u denotes position within the "by" variable(s) of the interaction; and
- n is `n_coef`.

Internally, `AR()` derives a value for ω that gives every element of β a marginal variance of τ^2. Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

The autocorrelation coefficients $\phi_1, \cdots, \phi_n$ are restricted to values between -1 and 1 that jointly lead to a stationary model. The quantity $r = \sqrt{\phi_1^2 + \cdots + \phi_n^2}$ has the boundary-avoiding prior

$$r \sim \text{Beta}(2, 2).$$

References

- `AR()` is based on the TMB function ARk

See Also

- `AR1()` Special case of `AR()`
- `Lin_AR()`, `Lin_AR1()` Straight line with AR errors
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
AR(n_coef = 3)
AR(n_coef = 3, s = 2.4)
AR(along = "cohort")
```

`AR1` *Autoregressive Prior of Order 1*

Description

Use an autoregressive process of order 1 to model a main effect, or use multiple AR1 processes to model an interaction. Typically used with time effects or with interactions that involve time.

Usage

```
AR1(min = 0.8, max = 0.98, s = 1, along = NULL)
```

Arguments

`min`, `max`	Minimum and maximum values for autocorrelation coefficient. Defaults are `0.8` and `0.98`.
`s`	Scale for the prior for the innovations. Default is `1`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `AR()` is used with an interaction, separate AR processes are constructed along the "along" variable, within each combination of the "by" variables.

Arguments `min` and `max` can be used to specify the permissible range for autocorrelation.

Argument `s` controls the size of innovations. Smaller values for `s` tend to give smoother estimates.

Value

An object of class `"bage_prior_ar"`.

Mathematical details

When `AR1()` is used with a main effect,

$$\beta_j = \phi \beta_{j-1} + \epsilon_j$$
$$\epsilon_j \sim \text{N}(0, \omega^2),$$

and when it is used with an interaction,

$$\beta_{u,v} = \phi \beta_{u,v-1} + \epsilon_{u,v}$$
$$\epsilon_{u,v} \sim \text{N}(0, \omega^2),$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction; and
- u denotes position within the "by" variable(s) of the interaction.

Internally, `AR1()` derives a value for ω that gives every element of β a marginal variance of τ^2. Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

Coefficient ϕ is constrained to lie between `min` and `max`. Its prior distribution is

$$\phi = (\text{max} - \text{min})\phi' - \text{min}$$

where

$$\phi' \sim \text{Beta}(2, 2).$$

References

- `AR1()` is based on the TMB function AR1
- The defaults for `min` and `max` are based on the defaults for `forecast::ets()`.

See Also

- `AR()` Generalization of `AR1()`
- `Lin_AR()`, `Lin_AR1()` Line with AR errors
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
AR1()
AR1(min = 0, max = 1, s = 2.4)
AR1(along = "cohort")
```

`augment.bage_mod` *Extract Data and Modelled Values*

Description

Extract data and rates, probabilities, or means from a model object. The return value consists of the original data and one or more columns of modelled values.

Usage

```
## S3 method for class 'bage_mod'
augment(x, quiet = FALSE, ...)
```

Arguments

`x`	Object of class `"bage_mod"`, typically created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`quiet`	Whether to suppress messages. Default is `FALSE`.
`...`	Unused. Included for generic consistency only.

Value

A tibble, with the original data plus one or more of the following columns:

- `.<outcome>` Corrected or extended version of the outcome variable, in applications where the outcome variable has missing values, or a data model is being used.
- `.observed` 'Direct' estimates of rates or probabilities, ie counts divided by exposure or size (in Poisson and binomial models.)
- `.fitted` Draws of rates, probabilities, or means.

- `.expected` Draws of expected values for rates or probabilities (in Poisson that include exposure, or in binomial models.)

Uncertain quantities are represented using rvecs.

Fitted vs unfitted models

`augment()` is typically called on a fitted model. In this case, the modelled values are draws from the joint posterior distribution for rates, probabilities, or means.

`augment()` can, however, be called on an unfitted model. In this case, the modelled values are draws from the joint prior distribution. In other words, the modelled values are informed by model priors, and by values for `exposure`, `size`, or `weights`, but not by observed outcomes.

Imputed values for outcome variable

`augment()` automatically imputes any missing values for the outcome variable. If outcome variable `var` has one or more `NA`s, then `augment` creates a variable `.var` holding original and imputed values.

Data model for outcome variable

If the overall model includes a data model for the outcome variable `var`, then `augment()` creates a new variable `.var` containing estimates of the true value for the outcome.

See Also

- `components()` Extract values for hyper-parameters from a model
- `tidy()` Short summary of a model
- `mod_pois()` Specify a Poisson model
- `mod_binom()` Specify a binomial model
- `mod_norm()` Specify a normal model
- `fit()` Fit a model
- `is_fitted()` See if a model has been fitted
- `unfit()` Reset a model
- datamods Overview of data models implemented in **bage**

Examples

```
## specify model
mod <- mod_pois(divorces ~ age + sex + time,
                data = divorces,
                exposure = population) |>
  set_n_draw(n_draw = 100) ## smaller sample, so 'augment' faster

## draw from the prior distribution
mod |> augment()

## fit model
mod <- mod |>
```

```
  fit()

## draw from the posterior distribution
mod |> augment()

## insert a missing value into outcome variable
divorces_missing <- divorces
divorces_missing$divorces[1] <- NA

## fitting model and calling 'augument'
## creates a new variable called '.divorces'
## holding observed and imputed values
mod_pois(divorces ~ age + sex + time,
         data = divorces_missing,
         exposure = population) |>
  fit() |>
  augment()

## specifying a data model for the
## original data also leads to a new
## variable called '.divorces'
mod_pois(divorces ~ age + sex + time,
         data = divorces,
         exposure = population) |>
  set_datamod_outcome_rr3() |>
  fit() |>
  augment()
```

`components.bage_mod` *Extract Values for Hyper-Parameters*

Description

Extract values for hyper-parameters from a model object. Hyper-parameters include main effects and interactions, dispersion and variance terms, and SVD or spline coefficients.

Usage

```
## S3 method for class 'bage_mod'
components(
  object,
  standardize = c("terms", "anova", "none"),
  quiet = FALSE,
  ...
)
```

Arguments

`object`	Object of class `"bage_mod"`, typically created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.

`standardize`	Standardization method: one of `"terms"`, `"anova"`, or "none". The default is `"terms"`. See below for details.
`quiet`	Whether to suppress messages. Default is `FALSE`.
`...`	Unused. Included for generic consistency only.

Value

A tibble with four columns columns:

The return value contains the following columns:

- `term` Model term that the hyper-parameter belongs to.
- `component` Component within term.
- `level` Element within component .
- `.fitted` An rvec containing draws from the posterior distribution.

Fitted vs unfitted models

`components()` is typically called on a fitted model. In this case, the modelled values are draws from the joint posterior distribution for the hyper-parameters in the model.

`components()` can, however, be called on an unfitted model. In this case, the modelled values are draws from the joint prior distribution. In other words, the modelled values are informed by model priors, and by any `exposure`, `size`, or `weights` argument in the model, but not by the observed outcomes.

Standardizing estimates

Often the sum of the intercept, main effect, and interaction terms is well-identified by the data, but the values for individual terms is not. This indeterminancy does not affect the ultimate estimation of rates, probabilities, and means, but does complicate the interpretation of the higher-level terms.

One way of dealing with poorly- identified terms is to post-process the estimates, imposing some sort of standardization. There are three options, specified through the `standardize` argument:

- `"terms"` Each main effect or interaction, and any component terms such as trend or seasonal effects, are independently scaled so that they sum to 0.
- `"anova"` An ANOVA-style decomposition is carried out so that all variation associated with age is attributed to the age term, all variation associated with the interaction between age and sex is attributed to the age-sex term, and so on. Components terms such as trend and seasonal effects are left untouched.
- `"none"` No standardization is done.

`"terms"` standardization is helpful for understanding model dynamics. `"anova"` standardization is helpful for understanding the contribution of each variable to overall patterns.

For a description of the standardization algorithms used by **bage**, see `vignette("vig2_math")`.

See Also

- `augment()` Extract data and values for rates, means, or probabilities
- `tidy()` Extract a one-line summary of a model
- `mod_pois()` Specify a Poisson model
- `mod_binom()` Specify a binomial model
- `mod_norm()` Specify a normal model
- `fit()` Fit a model
- `is_fitted()` See if a model has been fitted
- `unfit()` Reset a model

Examples

```
## specify model
mod <- mod_pois(injuries ~ age + sex + year,
                data = injuries,
                exposure = popn)

## extract prior distribution
## of hyper-parameters
mod |>
  components()

## fit model
mod <- mod |>
  fit()

## extract posterior distribution
## of hyper-parameters
mod |>
  components()
```

`components.bage_ssvd` *Extract Components used by SVD Summary*

Description

Extract the matrix and offset used by a scaled SVD summary of a demographic database.

Usage

```
## S3 method for class 'bage_ssvd'
components(object, n_comp = NULL, indep = NULL, age_labels = NULL, ...)
```

Arguments

object	An object of class "bage_ssvd".
n_comp	The number of components. The default is half the total number of components of object.
indep	Whether to use independent or joint SVDs for each sex/gender. If no value is supplied, an SVD with no sex/gender dimension is used. Note that the default is different from SVD().
age_labels	Age labels for the desired age or age-sex profile. If no labels are supplied, the most detailed profile available is used.
...	Not currently used.

Value

A tibble with the offset and components.

Scaled SVDs of demographic databases in bage

- HMD Mortality rates from the Human Mortality Database.
- LFP Labor forcce participation rates from the OECD.

See Also

- generate() Randomly generate age-profiles, or age-sex profiles, based on a scaled SVD summary.
- SVD() SVD prior for terms involving age.
- SVD_AR1(), SVD_AR(), SVD_RW(), SVD_RW2() Dynamic SVD priors for terms involving age and time.
- poputils::age_labels() Generate age labels.

Examples

```
## females and males combined
components(LFP, n_comp = 3)

## females and males modelled independently
components(LFP, indep = TRUE, n_comp = 3)

## joint model for females and males
components(LFP, indep = FALSE, n_comp = 3)

## specify age groups
labels <- poputils::age_labels(type = "five", min = 15, max = 60)
components(LFP, age_labels = labels)
```

`datamods` *Data Models*

Description

The models for rates, probabilities, or means created with functions `mod_pois()`, `mod_binom()`, and `mod_norm()` can be extended by adding data models, also referred to as measurement error models. Data models can be applied to the outcome variable, the exposure variable (in Poisson models), or the size variable (in binomial models).

Details

Data models for outcome variable

Function	Assumptions about data	pois	binom	norm
`set_datamod_outcome_rr3()`	Outcome randomly rounded to base 3	Yes	Yes	No

Data models for exposure or size variable

None implemented yet

`deaths` *Deaths in Iceland*

Description

Deaths and mid-year populations in Iceland, by age, sex, and calendar year.

Usage

```
deaths
```

Format

A tibble with 5,300 rows and the following columns:

- `age` Single year of age, up to `"105+"`
- `sex` `"Female"` and `"Male"`
- `time` Calendar year, 1998-2022
- `deaths` Counts of deaths
- `popn` Mid-year population

Source

Tables "Deaths by municipalities, sex and age 1981-2022", and "Average annual population by municipality, age and sex 1998-2022 - Current municipalities", on the Statistics Iceland website. Data downloaded on 12 July 2023.

`divorces` *Divorces in New Zealand*

Description

Counts of divorces and population, by age, sex, and calendar year, in New Zealand, 2011-2021.

Usage

```
divorces
```

Format

A tibble with 242 rows and the following columns:

- `age`: 5-year age groups, `"15-19"` to `"65+"`
- `sex`: `"Female"` or `"Male"`
- `time`: Calendar year
- `divorces`: Numbers of divorces during year
- `population`: Person-years lived during year

Source

Divorce counts from data in table "Age at divorces by sex (marriages and civil unions) (Annual-Dec)" in the online database Infoshare on the Statistics New Zealand website. Data downloaded on 22 March 2023. Population estimates derived from data in table "Estimated Resident Population by Age and Sex (1991+) (Annual-Dec)" in the online database Infoshare on the Statistics New Zealand website. Data downloaded on 26 March 2023.

`expenditure` *Per Capita Health Expenditure in the Netherlands, 2003-2011*

Description

Per capita health expenditure, in Euros, by diagnostic group, age group, and year, in the Netherlands.

Usage

```
expenditure
```

Format

A tibble with 1,296 rows and the following columns:

- `diag` Diagnostic group
- `age` 5-year age groups, with open age group of 85+
- `year` 2003, 2005, 2007, and 2011

Source

Calculated from data in table "Expenditure by disease, age and gender under the System of Health Accounts (SHA) Framework : Current health spending by age" from OECD database 'OECD.Stat' (downloaded on 25 May 2016) and in table "Historical population data and projections (1950-2050)" from OECD database 'OECD.Stat' (downloaded 5 June 2016).

`fit.bage_mod` *Fit a Model*

Description

Calculate the posterior distribution for a model.

Usage

```
## S3 method for class 'bage_mod'
fit(object, method = c("standard", "inner-outer"), vars_inner = NULL, ...)
```

Arguments

`object`	A `bage_mod` object, created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`method`	Estimation method. Current choices are `"standard"` (the default) and `"inner-outer"`. See below for details.
`vars_inner`	Names of variables to use for inner model when `method` is `"inner-outer". If NULL(the default)var` is the age, sex/gender, and time variables.
`...`	Not currently used.

Value

A `bage_mod` object

Estimation methods

- `"standard"` All parameters, other than the lowest-level rates, probabilities, or means are jointly estimated within TMB. The default.
- `"inner-outer"`. Multiple-stage estimation, which can be faster than `"standard"` for models with many parameters. In Step 1, the data is aggregated across all dimensions other than those specified in `var_inner`, and a model for the `inner` variables is fitted to the data. In Step 2, the data is aggregated across the remaining variables, and a model for the `outer` variables is fitted to the data. Parameter estimtes from steps 1 and 2 are then combined.

See Also

- `mod_pois()`, `mod_binom()`, `mod_norm()` Specify a model
- `augment()`, `components()`, `tidy()` Examine output from a model
- `forecast()` Forecast, based on a model
- `report_sim()` Simulation study of a model
- `unfit()` Reset a model
- `is_fitted()` Check if a model has been fitted

Examples

```
## specify model
mod <- mod_pois(injuries ~ age + sex + year,
                data = injuries,
                exposure = popn)

## examine unfitted model
mod

## fit model
mod <- fit(mod)

## examine fitted model
mod

## extract rates
aug <- augment(mod)
aug

## extract hyper-parameters
comp <- components(mod)
comp
```

`forecast.bage_mod` *Use a Model to Make a Forecast*

Description

Forecast rates, probabilities, means, and other model parameters.

Usage

```
## S3 method for class 'bage_mod'
forecast(
  object,
  newdata = NULL,
  output = c("augment", "components"),
  include_estimates = FALSE,
  standardize = c("terms", "anova", "none"),
  labels = NULL,
  ...
)
```

Arguments

`object`	A bage_mod object, typically created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`newdata`	Data frame with data for future periods.
`output`	Type of output returned
`include_estimates`	Whether to include historical estimates along with the forecasts. Default is `FALSE`.
`standardize`	Standardization method: one of `"terms"`, `"anova"`, or "none". The default is `"terms"`. See below for details.
`labels`	Labels for future values.
`...`	Not currently used.

Value

A tibble.

How the forecasts are constructed

Internally, the steps involved in a forecast are:

1. Forecast time-varying main effects and interactions, e.g. a time main effect, or an age-time interaction.
2. Combine forecasts for the time-varying main effects and interactions with non-time-varying parameters, e.g. age effects or dispersion.
3. Use the combined parameters to generate values for rates, probabilities or means.
4. If a `newdata` argument has been provided, and `output` is `"augment"`, draw values for outcome.

`vignette("vig2_math")` has the technical details.

Output

When `output` is `"augment"` (the default), the return value from `forecast()` looks like output from function `augment()`. When `output` is `"components"`, the return value looks like output from `components()`.

When `include_estimates` is `FALSE` (the default), the output of `forecast()` excludes values for time-varying parameters for the period covered by the data. When `include_estimates` is `TRUE`, the output includes these values. Setting `include_estimates` to `TRUE` can be helpful when creating graphs that combine estimates and forecasts.

Standardization

The standardization used by `forecast()` is equivalent to the standardization applied by components(), except that values for forecasted terms are are shifted so that they line up with the values for estimates.

Fitted and unfitted models

`forecast()` is typically used with a fitted model, i.e. a model in which parameter values have been estimated from the data. The resulting forecasts reflect data and priors.

`forecast()` can, however, be used with an unfitted model. In this case, the forecasts are based entirely on the priors. See below for an example. Experimenting with forecasts based entirely on the priors can be helpful for choosing an appropriate model.

Warning

The interface for `forecast()` has not been finalised.

See Also

- `mod_pois()`, `mod_binom()`, `mod_norm()` to specify a model
- `fit()` to fit a model

Examples

```
## specify and fit model
mod <- mod_pois(injuries ~ age * sex + ethnicity + year,
                data = injuries,
                exposure = popn) |>
  fit()
mod

## forecasts
mod |>
  forecast(labels = 2019:2024)

## combined estimates and forecasts
mod |>
  forecast(labels = 2019:2024,
           include_estimates = TRUE)

## hyper-parameters
mod |>
  forecast(labels = 2019:2024,
           output = "components")

## hold back some data and forecast
library(dplyr, warn.conflicts = FALSE)
data_historical <- injuries |>
  filter(year <= 2015)
data_forecast <- injuries |>
```

```
  filter(year > 2015) |>
  mutate(injuries = NA)
mod_pois(injuries ~ age * sex + ethnicity + year,
         data = data_historical,
         exposure = popn) |>
  fit() |>
  forecast(newdata = data_forecast)

## forecast based on priors only
mod_unfitted <- mod_pois(injuries ~ age * sex + ethnicity + year,
                         data = injuries,
                         exposure = popn)
mod_unfitted |>
  forecast(labels = 2019:2024)
```

`generate.bage_ssvd` *Generate Random Age or Age-Sex Profiles*

Description

Generate random age or age-sex profiles from an object of class `"bage_ssvd"`. An object of class `"bage_ssvd"` holds results from an SVD decomposition of demographic data.

Usage

```
## S3 method for class 'bage_ssvd'
generate(x, n_draw = 20, n_comp = NULL, indep = NULL, age_labels = NULL, ...)
```

Arguments

`x`	An object of class `"bage_ssvd"`.
`n_draw`	Number of random draws to generate.
`n_comp`	The number of components. The default is half the total number of components of `object`.
`indep`	Whether to use independent or joint SVDs for each sex/gender. If no value is supplied, an SVD with no sex/gender dimension is used. Note that the default is different from `SVD()`.
`age_labels`	Age labels for the desired age or age-sex profile. If no labels are supplied, the most detailed profile available is used.
`...`	Not currently used.

Value

A tibble

Scaled SVDs of demographic databases in bage

- HMD Mortality rates from the Human Mortality Database.
- LFP Labor forcce participation rates from the OECD.

See Also

- components() Components used by SVD prior.
- SVD() SVD prior for term involving age.
- SVD_AR1(), SVD_AR(), SVD_RW(), SVD_RW2() Dynamic SVD priors for terms involving age and time.
- poputils::age_labels() Generate age labels.

Examples

```
## SVD for females and males combined
generate(HMD)

## separate SVDs for females and males
generate(HMD, indep = TRUE)

## specify age groups
labels <- poputils::age_labels(type = "lt", max = 60)
generate(HMD, age_labels = labels)
```

HMD *Components from Human Mortality Database*

Description

An object of class "bage_ssvd" holding components extracted from mortality data from the Human Mortality Database. The object holds 5 components.

Usage

```
HMD
```

Format

Object of class "bage_ssvd".

Source

Derived from data at Human Mortality Database. Max Planck Institute for Demographic Research (Germany), University of California, Berkeley (USA), and French Institute for Demographic Studies (France). Available at www.mortality.org. Code to create HMD is in folder 'data-raw/ssvd_hmd' in the source code for **bage** package.

`households` *People in One-Person households in New Zealand*

Description

Counts of people in one-person households, and counts of people living in any household, by age, region, and year.

Usage

```
households
```

Format

A tibble with 528 rows and the following columns:

- `age`: 5-year age groups, with open age group of 65+
- `region`: Region within New Zealand
- `year`: Calendar year
- `oneperson`: Count of people living in one-person households
- `total`: Count of people living in all types of household

Source

Derived from data in table "Household composition by age group, for people in households in occupied private dwellings, 2006, 2013, and 2018 Censuses (RC, TA, DHB, SA2)" in the online database NZ.Stat, on the Statistics New Zealand website. Data downloaded on 3 January 2023.

`injuries` *Fatal Injuries in New Zealand*

Description

Counts of fatal injuries in New Zealand, by age, sex, ethnicity, and year, plus estimates of the population at risk.

Usage

```
injuries
```

Format

A tibble with 912 rows and the following columns:

- `age`: 5-year age groups, up to age 55-59
- `sex`: `"Female"` or `"Male"`
- `ethnicity`: `"Maori"` or `"Non Maori"`
- `year`: Calendar year
- `injuries`: Count of injuries, randomly rounded to base 3
- `popn`: Population on 30 June

Source

Derived from data in tables "Estimated Resident Population by Age and Sex (1991+) (Annual-Jun)" and "Maori Ethnic Group Estimated Resident Population by Age and Sex (1991+) (Annual-Jun)" in the online database Infoshare, and table "Count of fatal and serious non-fatal injuries by sex, age group, ethnicity, cause, and severity of injury, 2000-2021" in the online database NZ.Stat, on the Statistics New Zealand website. Data downloaded on 1 January 2023.

`is_fitted` *Test Whether a Model has Been Fitted*

Description

Test whether fit() has been called on a model object.

Usage

```
is_fitted(x)
```

Arguments

`x`	An object of class `"bage_mod"`.

Value

`TRUE` or `FALSE`

See Also

- `mod_pois()`, `mod_binom()`, `mod_norm()` to specify a model
- `fit()` to fit a model

Examples

```
mod <- mod_pois(injuries ~ age + sex + year,
                data = injuries,
                exposure = popn)
is_fitted(mod)
mod <- fit(mod)
is_fitted(mod)
```

Known *Known Prior*

Description

Treat an intercept, a main effect, or an interaction as fixed and known.

Usage

```
Known(values)
```

Arguments

`values`	A numeric vector

Value

An object of class `"bage_prior_known"`.

See Also

- `NFix()` Prior where level unknown, but variability known.
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
Known(-2.3)
Known(c(0.1, 2, -0.11))
```

`LFP` *Components from OECD Labor Force Participation Data*

Description

An object of class `"bage_ssvd"` holding components extracted from labor force participation data from the OECD Data Explorer.

Usage

```
LFP
```

Format

Object of class `"bage_ssvd"`.

Source

Derived from data in the "Labor Force Indicators" table of the OECD Data Explorer. Code to create `LFS` is in folder 'data-raw/ssvd_lfp' in the source code for the **bage** package.

`Lin` *Linear Prior with Independent Normal Errors*

Description

Use a line or lines with independent normal errors to model a main effect or interaction. Typically used with time.

Usage

```
Lin(s = 1, sd = 1, along = NULL)
```

Arguments

`s`	Scale for the prior for the errors. Default is `1`.
`sd`	Standard deviation in prior for slope of line. Default is 1.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `Lin()` is used with an interaction, then separate lines are constructed along the "along" variable, within each combination of the "by" variables.

Argument `s` controls the size of the errors. Smaller values tend to give smoother estimates.

Argument `sd` controls the size of the slopes of the lines. Larger values can give more steeply sloped lines.

Value

An object of class "bage_prior_lin".

Mathematical details

When `Lin()` is used with a main effect,

$$\begin{aligned}\beta_j &= \alpha + j\eta + \epsilon_j \\ \alpha &\sim \text{N}(0, 1) \\ \epsilon_j &\sim \text{N}(0, \tau^2),\end{aligned}$$

and when it is used with an interaction,

$$\begin{aligned}\beta_{u,v} &\sim \alpha_u + v\eta_u + \epsilon_{u,v} \\ \alpha_u &\sim \text{N}(0, 1) \\ \epsilon_{u,v} &\sim \text{N}(0, \tau^2),\end{aligned}$$

where

- $\pmb{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction; and
- u denotes position within the "by" variable(s) of the interaction.

The slopes have priors

$$\eta \sim \text{N}(0, \text{sd}^2)$$

and

$$\eta_u \sim \text{N}(0, \text{sd}^2).$$

Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2).$$

See Also

- `Lin_AR()` Linear with AR errors
- `Lin_AR1()` Linear with AR1 errors
- `RW2()` Second-order random walk
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
Lin()
Lin(s = 0.5, sd = 2)
Lin(along = "cohort")
```

`Lin_AR` *Linear Prior with Autoregressive Errors*

Description

Use a line or lines with autoregressive errors to model a main effect or interaction. Typically used with time.

Usage

```
Lin_AR(n_coef = 2, s = 1, sd = 1, along = NULL)
```

Arguments

`n_coef`	Number of lagged terms in the model, ie the order of the model. Default is `2`.
`s`	Scale for the innovations in the AR process. Default is `1`.
`sd`	Standard deviation in the prior for the slope of the line. Larger values imply steeper slopes. Default is 1.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `Lin_AR()` is used with an interaction, separate lines are constructed along the "along" variable, within each combination of the "by" variables.

The order of the autoregressive errors is controlled by the `n_coef` argument. The default is 2.

Argument `s` controls the size of the innovations. Smaller values tend to give smoother estimates.

Argument `sd` controls the size of the slopes of the lines. Larger values can give more steeply sloped lines.

Value

An object of class `"bage_prior_linar"`.

Mathematical details

When `Lin_AR()` is used with a main effect,

$$\beta_1 = \alpha + \epsilon_1$$
$$\beta_j = \alpha + (j-1)\eta + \epsilon_j, \quad j > 1$$
$$\alpha \sim \text{N}(0, 1)$$
$$\epsilon_j = \phi_1 \epsilon_{j-1} + \cdots + \phi_n \epsilon_{j-n} + \varepsilon_j$$
$$\varepsilon_j \sim \text{N}(0, \omega^2),$$

and when it is used with an interaction,

$$
\begin{aligned}
\beta_{u,1} &= \alpha_u + \epsilon_{u,1} \\
\beta_{u,v} &= \eta(v-1) + \epsilon_{u,v}, \quad v = 2, \cdots, V \\
\alpha_u &\sim \text{N}(0,1) \\
\epsilon_{u,v} &= \phi_1 \epsilon_{u,v-1} + \cdots + \phi_n \epsilon_{u,v-n} + \varepsilon_{u,v}, \\
\varepsilon_{u,v} &\sim \text{N}(0, \omega^2).
\end{aligned}
$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- u denotes position within the "along" variable of the interaction;
- u denotes position within the "by" variable(s) of the interaction; and
- n is `n_coef`.

The slopes have priors

$$\eta \sim \text{N}(0, \text{sd}^2)$$

and

$$\eta_u \sim \text{N}(0, \text{sd}^2).$$

Internally, `Lin_AR()` derives a value for ω that gives ϵ_j or $\epsilon_{u,v}$ a marginal variance of τ^2. Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where a value for `s` is provided by the user.

The $\phi_1, \cdots, \phi_k$ are restricted to values between -1 and 1 that jointly lead to a stationary model. The quantity $r = \sqrt{\phi_1^2 + \cdots + \phi_k^2}$ has boundary-avoiding prior

$$r \sim \text{Beta}(2, 2).$$

See Also

- `Lin_AR1()` Special case of `Lin_AR()`
- `Lin()` Line with independent normal errors
- `AR()` AR process with no line
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
Lin_AR()
Lin_AR(n_coef = 3, s = 0.5, sd = 2)
```

Lin_AR1 *Linear Prior with Autoregressive Errors of Order 1*

Description

Use a line or lines with AR1 errors to model a main effect or interaction. Typically used with time.

Usage

```
Lin_AR1(min = 0.8, max = 0.98, s = 1, sd = 1, along = NULL)
```

Arguments

`min, max`	Minimum and maximum values for autocorrelation coefficient. Defaults are `0.8` and `0.98`.
`s`	Scale for the innovations in the AR process. Default is `1`.
`sd`	Standard deviation in the prior for the slope of the line. Larger values imply steeper slopes. Default is 1.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `Lin_AR()` is used with an interaction, separate lines are constructed along the "along" variable, within each combination of the "by" variables.

Arguments `min` and `max` can be used to specify the permissible range for autocorrelation.

Argument `s` controls the size of the innovations. Smaller values tend to give smoother estimates.

Argument `sd` controls the size of the slopes of the lines. Larger values can give more steeply sloped lines.

Value

An object of class `"bage_prior_linar"`.

Mathematical details

When `Lin_AR1()` is being used with a main effect,

$$\beta_j = \eta q_j + \epsilon_j$$
$$\epsilon_j = \phi \epsilon_{j-1} + \varepsilon_j,$$
$$\varepsilon_j \sim \text{N}(0, \omega^2).$$

and when it is being used with an interaction,

$$\beta_{u,v} = \eta_j q_{u,v} + \epsilon_{u,v}$$
$$\epsilon_{u,v} = \phi + \varepsilon_{u,v},$$
$$\varepsilon_{u,v} \sim \text{N}(0, \omega^2).$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- u denotes position within the "along" variable of the interaction;
- u denotes position within the "by" variable(s) of the interaction;
- $q = -(J+1)/(J-1) + 2j/(J-1)$; and
- $q_v = -(V+1)/(V-1) + 2v/(V-1)$.

The slopes have priors
$$\eta \sim \text{N}(0, \text{sd}^2)$$
and
$$\eta_u \sim \text{N}(0, \text{sd}^2).$$
Larger values for `sd` permit steeper slopes.

Internally, `Lin_AR1()` derives a value for ω that gives ϵ_j or $\epsilon_{u,v}$ a marginal variance of τ^2. Parameter τ has a half-normal prior
$$\tau \sim \text{N}^+(0, \text{s}^2),$$
where a value for `s` is provided by the user.

Coefficient ϕ is constrained to lie between `min` and `max`. Its prior distribution is
$$\phi = (\text{max} - \text{min})\phi' - \text{min}$$
where
$$\phi' \sim \text{Beta}(2, 2).$$

References

- The defaults for `min` and `max` are based on the defaults for `forecast::ets()`.

See Also

- `Lin_AR()` Generalization of `Lin_AR1()`
- `Lin()` Line with independent normal errors
- `AR1()` AR1 process with no line
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
Lin_AR1()
Lin_AR1(min = 0, s = 0.5, sd = 2)
```

mod_binom *Specify a Binomial Model*

Description

Specify a model where the outcome is drawn from a binomial distribution.

Usage

```
mod_binom(formula, data, size)
```

Arguments

`formula`	An R formula, specifying the outcome and predictors.
`data`	A data frame containing the outcome and predictor variables, and the number of trials.
`size`	Name of the variable giving the number of trials, or a formula.

Details

The model is hierarchical. The probabilities in the binomial distribution are described by a prior model formed from dimensions such as age, sex, and time. The terms for these dimension themselves have models, as described in priors. These priors all have defaults, which depend on the type of term (eg an intercept, an age main effect, or an age-time interaction.)

Value

An object of class `bage_mod`.

Mathematical details

The likelihood is

$$y_i \sim \text{binomial}(\gamma_i; w_i)$$

where

- y_i is a count, such of number of births, for some combination i of classifying variables, such as age, sex, and region;
- γ_i is a probability of 'success'; and
- w_i is the number of trials.

The probabilities γ_i are assumed to be drawn a beta distribution

$$y_i \sim \text{Beta}(\xi^{-1}\mu_i, \xi^{-1}(1 - \mu_i))$$

where

- μ_i is the expected value for γ_i; and
- ξ governs dispersion (ie variance.)

Expected value μ_i equals, on a logit scale, the sum of terms formed from classifying variables,

$$\text{logit}\mu_i = \sum_{m=0}^{M} \beta_{j_i^m}^{(m)}$$

where

- β^0 is an intercept;
- $\beta^{(m)}$, $m = 1, \dots, M$, is a main effect or interaction; and
- j_i^m is the element of $\beta^{(m)}$ associated with cell i.

The $\beta^{(m)}$ are given priors, as described in priors.

The prior for ξ is described in `set_disp()`.

Specifying size

The `size` argument can take two forms:

- the name of a variable in `data`, with or without quote marks, eg `"population"` or `population`; or
- a formula, which is evaluated with `data` as its environment (see below for example).

See Also

- `mod_pois()` Specify Poisson model
- `mod_norm()` Specify normal model
- `set_prior()` Specify non-default prior for term
- `set_disp()` Specify non-default prior for dispersion
- `fit()` Fit a model
- `forecast()` Forecast a model
- `report_sim()` Do a simulation study on a model

Examples

```
mod <- mod_binom(oneperson ~ age:region + age:year,
                 data = households,
                 size = total)

## use formula to specify size
mod <- mod_binom(ncases ~ agegp + tobgp + alcgp,
                 data = esoph,
                 size = ~ ncases + ncontrols)
```

`mod_norm` *Specify a Normal Model*

Description

Specify a model where the outcome is drawn from a normal distribution.

Usage

```
mod_norm(formula, data, weights)
```

Arguments

`formula`	An R formula, specifying the outcome and predictors.
`data`	A data frame containing outcome, predictor, and, optionally, weights variables.
`weights`	Name of the weights variable, a `1`, or a formula. See below for details.

Details

The model is hierarchical. The means in the normal distribution are described by a prior model formed from dimensions such as age, sex, and time. The terms for these dimension themselves have models, as described in priors. These priors all have defaults, which depend on the type of term (eg an intercept, an age main effect, or an age-time interaction.)

Internally, the outcome variable scaled to have mean 0 and sd 1.

Value

An object of class `bage_mod_norm`.

Mathematical details

The likelihood is

$$y_i \sim \mathrm{N}(\mu_i, \xi^2/w_i)$$

where

- y_i is a scaled value for an, such of the log of income, for some combination i of classifying variables, such as age, sex, and region;
- μ_i is a mean;
- ξ is a standard deviation parameter; and
- w_i is a weight.

The scaling of the outcome variable is done internally. If y_i^* is the original, then $y_i = (y_i^* - m)/s$ where m and s are the sample mean and standard deviation of y_i^*.

In some applications, w_i is set to 1 for all i.

The means μ_i equal the sum of terms formed from classifying variables,

$$\mu_i = \sum_{m=0}^{M} \beta_{j_i^m}^{(m)}$$

where

- β^0 is an intercept;
- $\beta^{(m)}$, $m = 1, \dots, M$, is a main effect or interaction; and
- j_i^m is the element of $\beta^{(m)}$ associated with cell i.

The $\beta^{(m)}$ are given priors, as described in priors.

The prior for ξ is described in `set_disp()`.

Specifying weights

The `weights` argument can take three forms:

- the name of a variable in `data`, with or without quote marks, eg `"wt"` or `wt`;
- the number `1`, in which no weights are used; or
- a formula, which is evaluated with `data` as its environment (see below for example).

See Also

- `mod_pois()` Specify Poisson model
- `mod_binom()` Specify binomial model
- `set_prior()` Specify non-default prior for term
- `set_disp()` Specify non-default prior for standard deviation
- `fit()` Fit a model
- `forecast()` Forecast a model
- `report_sim()` Do a simulation study on a model

Examples

```
mod <- mod_norm(value ~ diag:age + year,
                data = expenditure,
                weights = 1)

## use formula to specify weights
mod <- mod_norm(value ~ diag:age + year,
                data = expenditure,
                weights = ~sqrt(value))
```

`mod_pois` *Specify a Poisson Model*

Description

Specify a model where the outcome is drawn from a Poisson distribution.

Usage

```
mod_pois(formula, data, exposure)
```

Arguments

`formula`	An R formula, specifying the outcome and predictors.
`data`	A data frame containing outcome, predictor, and, optionally, exposure variables.
`exposure`	Name of the exposure variable, or a `1`, or a formula. See below for details.

Details

The model is hierarchical. The rates in the Poisson distribution are described by a prior model formed from dimensions such as age, sex, and time. The terms for these dimension themselves have models, as described in priors. These priors all have defaults, which depend on the type of term (eg an intercept, an age main effect, or an age-time interaction.)

Value

An object of class `bage_mod_pois`.

Mathematical details

The likelihood is

$$y_i \sim \text{Poisson}(\gamma_i w_i)$$

where

- subscript i identifies some combination of classifying variables, such as age, sex, and time;
- y_i is an outcome, such as deaths;
- γ_i is rates; and
- w_i is exposure.

In some applications, there is no obvious population at risk. In these cases, exposure w_i can be set to 1 for all i.

The rates γ_i are assumed to be drawn a gamma distribution

$$y_i \sim \text{Gamma}(\xi^{-1}, (\xi \mu_i)^{-1})$$

where

- μ_i is the expected value for γ_i; and
- ξ governs dispersion (ie variance.)

Expected value μ_i equals, on the log scale, the sum of terms formed from classifying variables,

$$\log \mu_i = \sum_{m=0}^{M} \beta_{j_i^m}^{(m)}$$

where

- β^0 is an intercept;
- $\beta^{(m)}$, $m = 1, \dots, M$, is a main effect or interaction; and
- j_i^m is the element of $\beta^{(m)}$ associated with cell i.

The $\beta^{(m)}$ are given priors, as described in priors.

The prior for ξ is described in `set_disp()`.

Specifying exposure

The `exposure` argument can take three forms:

- the name of a variable in `data`, with or without quote marks, eg `"population"` or `population`;
- the number `1`, in which case a pure "counts" model with no exposure, is produced; or
- a formula, which is evaluated with `data` as its environment (see below for example).

See Also

- `mod_binom()` Specify binomial model
- `mod_norm()` Specify normal model
- `set_prior()` Specify non-default prior for term
- `set_disp()` Specify non-default prior for dispersion
- `fit()` Fit a model
- `forecast()` Forecast a model
- `report_sim()` Do a simulation study on a model

Examples

```
## specify a model with exposure
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = popn)

## specify a model without exposure
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = 1)
```

```
## use a formula to specify exposure
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = ~ pmax(popn, 1))
```

`N` *Normal Prior*

Description

Use independent draws from a normal distribution to model a main effect or interaction. Typically used with variables other than age or time, such as region or ethnicity, where there is no natural ordering.

Usage

```
N(s = 1)
```

Arguments

`s`	Scale for the standard deviation. Default is `1`.

Details

Argument `s` controls the size of errors. Smaller values for `s` tend to give more tightly clustered estimates.

Value

An object of class `"bage_prior_norm"`.

Mathematical details

$$\beta_j \sim \text{N}(0, \tau^2)$$

where β is the main effect or interaction.

Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

See Also

- `NFix()` Similar to `N()` but standard deviation parameter is supplied rather than estimated from data
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
N()
N(s = 0.5)
```

`NFix` *Normal Prior with Fixed Variance*

Description

Normal prior where, in contrast to `N()`, the variance is treated as fixed and known. Typically used for main effects or interactions where there are too few elements to reliably estimate variance from the available data.

Usage

```
NFix(sd = 1)
```

Arguments

`sd`	Standard deviation. Default is `1`.

Details

`NFix()` is the default prior for the intercept.

Value

An object of class `"bage_prior_normfixed"`.

Mathematical details

$$\beta_j \sim \text{N}(0, \tau^2)$$

where β is the main effect or interaction, and a value for `sd` is supplied by the user.

See Also

- `N()` Similar to `NFix()`, but standard deviation parameter is estimated from the data rather than being fixed in advance
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
NFix()
NFix(sd = 10)
```

`print.bage_mod` *Printing a Model*

Description

After calling a function such as `mod_pois()` or `set_prior()` it is good practice to print the model object at the console, to check the model's structure. The output from `print()` has the following components:

- A header giving the class of the model and noting whether the model has been fitted.
- A formula giving the outcome variable and terms for the model.
- Priors for each model term.
- A table giving the number of parameters, and (fitted models only) the standard deviation across those parameters, a measure of the term's importance.
- Values for other model settings.

Usage

```
## S3 method for class 'bage_mod'
print(x, ...)
```

Arguments

`x`	Object of class `"bage_mod"`, typically created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`...`	Unused. Included for generic consistency only.

Value

`x`, invisibly.

See Also

- `mod_pois()`, `mod_binom()`, `mod_norm()` Model specification and class
- fit.bage_mod() and `is_fitted()` Model fitting
- priors Overview of priors for model terms
- tidy.bage_mod() Number of parameters, and standard deviations
- `set_disp()` Dispersion
- `set_var_age()`, `set_var_sexgender()`, `set_var_time()` Age, sex/gender and time variables
- `set_n_draw()` Model draws

Examples

```
mod <- mod_pois(injuries ~ age + sex + year,
                data = injuries,
                exposure = popn)

## print unfitted model
mod

mod <- fit(mod)

## print fitted model
mod
```

---

priors *Priors for Intercept, Main Effects, Interactions*

Description

The models created with functions `mod_pois()`, `mod_binom()`, and `mod_norm()` always include an intercept, and typically include main effects and interactions formed from variables in input data. Most models, for instance include an age effect, and many include an interaction between age and sex/gender, or age and time.

The intercept, main effects, and interactions all have prior models that capture the expected behavior of the term. Current choices of prior models are summarised here.

Details

Prior	**Description**	**Uses**	**Forecast**
`N()`	Elements drawn from normal distribution	Term with no natural order	Yes
`NFix()`	As for `N()`, but standard deviation fixed	Term with few elements	Yes
`RW()`	Random walk	Smoothing	Yes
`RW2()`	Second-order random walk	Like `RW()`, but smoother	Yes
`RW_Seas()`	Random walk, with seasonal effect	Terms involving time	Yes
`RW2_Seas()`	Second-order random walk, with seasonal effect	Term involving time	Yes
`AR()`	Auto-regressive prior of order *k*	Mean reversion	Yes
`AR1()`	Auto-regressive prior of order 1 Special case of `AR()`	Mean reversion	Yes
`Known()`	Values treated as known	Simulations, prior knowledge	No
`Lin()`	Linear trend, with independent normal	Parsimonious model for time	Yes
`Lin_AR()`	Linear trend, with autoregressive errors	Term involving time	Yes
`Lin_AR1()`	Linear trend, with AR1 errors	Terms involving time	Yes
`Sp()`	P-Spline (penalised spline)	Smoothing, eg over age	No
`SVD()`	Age or age-sex profile based on SVD of database	Age or age-sex	No
`SVD_AR()`	`SVD()`, but coefficients follow `AR()`	Age or age-sex and time	Yes
`SVD_AR1()`	`SVD()`, but coefficients follow `AR1()`	Age or age-sex and time	Yes
`SVD_RW()`	`SVD()`, but coefficients follow `RW()`	Age or age-sex and time	Yes
`SVD_RW2()`	`SVD()`, but coefficients follow `RW2()`	Age or age-sex and time	Yes

`replicate_data` *Create Replicate Data*

Description

Use a fitted model to create replicate datasets, typically as a way of checking a model.

Usage

```
replicate_data(x, condition_on = NULL, n = 19)
```

Arguments

`x`	A fitted model, typically created by calling `mod_pois()`, `mod_binom()`, or `mod_norm()`, and then `fit()`.
`condition_on`	Parameters to condition on. Either `"expected"` or `"fitted"`. See details.
`n`	Number of replicate datasets to create. Default is 19.

Details

Use `n` draws from the posterior distribution for model parameters to generate `n` simulated datasets. If the model is working well, these simulated datasets should look similar to the actual dataset.

Value

A tibble with the following structure:

`.replicate`	data
`"Original"`	Original data supplied to `mod_pois()`, `mod_binom()`, `mod_norm()`
`"Replicate 1"`	Simulated data.
`"Replicate 2"`	Simulated data.
...	...
`"Replicate <n>"`	Simulated data.

The `condition_on` argument

With Poisson and binomial models that include dispersion terms (which is the default), there are two options for constructing replicate data.

- When `condition_on` is `"fitted"`, the replicate data is created by (i) drawing values from the posterior distribution for rates or probabilities (the γ_i defined in `mod_pois()` and `mod_binom()`), and (ii) conditional on these rates or probabilities, drawing values for the outcome variable.

- When condition_on is "expected", the replicate data is created by (i) drawing values from hyper-parameters governing the rates or probabilities (the μ_i and ξ defined in mod_pois() and mod_binom()), then (ii) conditional on these hyper-parameters, drawing values for the rates or probabilities, and finally (iii) conditional on these rates or probabilities, drawing values for the outcome variable.

The default for condition_on is "expected". The "expected" option provides a more severe test for a model than the "fitted" option, since "fitted" values are weighted averages of the "expected" values and the original data.

As described in mod_norm(), normal models have a different structure from Poisson and binomial models, and the distinction between "fitted" and "expected" does not apply.

Data models for outcomes

If a data model has been provided for the outcome variable, then creation of replicate data will include a step where errors are added to outcomes. For instance, the a rr3 data model is used, then replicate_data() rounds the outcomes to base 3.

See Also

- mod_pois(), mod_binom(), mod_norm() Create model.
- fit() Fit model.
- report_sim() Simulation study of model.

Examples

```
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = 1) |>
  fit()

rep_data <- mod |>
  replicate_data()

library(dplyr)
rep_data |>
  group_by(.replicate) |>
  count(wt = injuries)

## when the overall model includes an rr3 data model,
## replicate data are rounded to base 3
mod_pois(injuries ~ age:sex + ethnicity + year,
         data = injuries,
         exposure = popn) |>
  set_datamod_outcome_rr3() |>
  fit() |>
  replicate_data()
```

`report_sim` *Simulation Study of a Model*

Description

Use simulated data to assess the performance of an estimation model.

Usage

```
report_sim(
  mod_est,
  mod_sim = NULL,
  method = c("standard", "inner-outer"),
  vars_inner = NULL,
  n_sim = 100,
  point_est_fun = c("median", "mean"),
  widths = c(0.5, 0.95),
  report_type = c("short", "long", "full"),
  n_core = 1
)
```

Arguments

`mod_est`	The model whose performance is being assessed. An object of class `bage_mod`.
`mod_sim`	The model used to generate the simulated data. If no value is supplied, `mod_est` is used.
`method`	Estimation method used for `mod_est`. See `fit()`.
`vars_inner`	Variables used in inner model with `"inner-outer"`estimation method. See `fit()`.
`n_sim`	Number of sets of simulated data to use. Default is 100.
`point_est_fun`	Name of the function to use to calculate point estimates. The options are `"mean"` and `"median"`. The default is `"mean"`.
`widths`	Widths of credible intervals. A vector of values in the interval `(0, 1]`. Default is `c(0.5, 0.95)`.
`report_type`	Amount of detail in return value. Options are `"short"` and `"long"`. Default is `"short"`.
`n_core`	Number of cores to use for parallel processing. If `n_core` is `1` (the default), no parallel processing is done.

Value

When `report_type` is `"short"`, a tibble with the following columns:

- `component`. Part of model. See Details. `"par"` is the rate, probability, or mean parameter from the likelihood.

- vals_sim. Simulated value for parameter, averaged across all simulations and cells.
- error_point_est. Point estimate minus simulation-true value, averaged across all simulations and cells.
- cover. Actual proportion of simulation-true values that fall within each type of interval, averaged across all simulations and cells.

When report_type is "long", a tibble with the following columns:

- component. Part of model. See components(). "par" is the rate, probability, or mean parameter from the likelihood.
- term. Category within component.
- level. Category within term.
- vals_sim. Simulated values for parameter, stored in an rvec.
- error_point_est. Point estimates minus simulation-true values, stored in an rvec.
- cover. Actual proportions of simulation-true values falling within each type of interval, stored in an rvec.

See Also

- mod_pois(), mod_binom(), mod_norm() Specify a model
- components(), augment() Draw from joint prior or posterior distribution of model
- replicate_data() Generate replicate data for a model

Examples

```
## results random, so set seed
set.seed(0)

## make data - outcome variable (deaths here)
## needs to be present, but is not used
data <- data.frame(region = c("A", "B", "C", "D", "E"),
                   population = c(100, 200, 300, 400, 500),
                   deaths = NA)

## simulation with estimation model same as
## data-generating model
mod_est <- mod_pois(deaths ~ region,
                    data = data,
                    exposure = population) |>
  set_prior(`(Intercept)` ~ Known(0))
report_sim(mod_est = mod_est,
           n_sim = 10) ## in practice should use larger value

## simulation with estimation model different
## from data-generating model
mod_sim <- mod_est |>
  set_prior(region ~ N(s = 2))
report_sim(mod_est = mod_est,
           mod_sim = mod_sim,
           n_sim = 10)
```

`RW` *Random Walk Prior*

Description

Use a random walk to model a main effect, or use multiple random walks to model an interaction. Typically used with age or time effects or with interactions that involve age or time.

Usage

```
RW(s = 1, along = NULL)
```

Arguments

`s`	Scale for the prior for the innovations. Default is `1`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `RW()` is used with an interaction, separate random walks are constructed along the "along" variable, within each combination of the "by" variables.

Argument `s` controls the size of innovations. Smaller values for `s` tend to give smoother series.

Value

An object of class `"bage_prior_rw"`.

Mathematical details

When `RW()` is used with a main effect,

$$\beta_j = \beta_{j-1} + \epsilon_j$$
$$\epsilon_j \sim \text{N}(0, \tau^2),$$

and when it is used with an interaction,

$$\beta_{u,v} = \beta_{u,v-1} + \epsilon_{u,v}$$
$$\epsilon_{u,v} \sim \text{N}(0, \tau^2),$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction; and

- u denotes position within the "by" variable(s) of the interaction.

Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

See Also

- `RW2()` Second-order random walk
- `AR()` Autoregressive with order k
- `AR1()` Autoregressive with order 1
- `Sp()` Smoothing via splines
- `SVD()` Smoothing of age via singular value decomposition
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
RW()
RW(s = 0.5)
RW(along = "cohort")
```

RW2 *Second-Order Random Walk Prior*

Description

Use a second-oder random walk to model a main effect, or use multiple second-order random walks to model an interaction. A second-order random walk is effectively a random walk with drift where the drift term varies. It is typically used with main effects or interactions that involve time, where there are sustained trends upward or downward.

Usage

```
RW2(s = 1, along = NULL)
```

Arguments

`s`	Scale for the prior for the innovations. Default is `1`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `RW2()` is used with an interaction, separate series are used for the "along" variable within each combination of the "by" variables.

Argument `s` controls the size of innovations in the random walk. Smaller values for `s` tend to give smoother series.

Argument `n_seas` controls the number of `seasons`. When using quarterly data, for instance, `n_seas` should be `4`, and when using monthly data, `n_seas` should be `12`.

By default, the magnitude of seasonal effects can change over time. However, setting `s_seas` to `0` produces seasonal effects that are fixed, eg where "January" effect is the same every year, the "Feburary" effect is the same every year, and so on.

Value

An object of class `"bage_prior_rw2"`.

Mathematical details

When `RW()` is used with a main effect,

$$\beta_j = 2\beta_{j-1} - \beta_{j-2} + \epsilon_j$$
$$\epsilon_j \sim \text{N}(0, \tau^2),$$

and when it is used with an interaction,

$$\beta_{u,v} = 2\beta_{u,v-1} - \beta_{u,v-2} + \epsilon_{u,v}$$
$$\epsilon_{u,v} \sim \text{N}(0, \tau^2),$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction; and
- u denotes position within the "by" variable(s) of the interaction.

Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

See Also

- `RW()` Random walk
- `AR()` Autoregressive with order k
- `AR1()` Autoregressive with order 1
- `Sp()` Smoothing via splines
- `SVD()` Smoothing of age via singular value decomposition
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
RW2()
RW2(s = 0.5)
```

RW2_Seas *Second-Order Random Walk Prior with Seasonal Effect*

Description

Use a second-oder random walk with seasonal effects to model a main effect, or use multiple second-order random walks, each with their own seasonal effects, to model an interaction. A second-order random walk is effectively a random walk with drift where the drift term varies. It is typically used with main effects or interactions that involve time, where there are sustained trends upward or downward.

Usage

```
RW2_Seas(n_seas, s = 1, s_seas = 1, along = NULL)
```

Arguments

`n_seas`	Number of seasons
`s`	Scale for prior for innovations in the random walk. Default is `1`.
`s_seas`	Scale for prior for innovations in the seasonal effect. Default is `1`. Can be `0`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `RW2_Seas()` is used with an interaction, separate series are used for the "along" variable within each combination of the "by" variables.

Argument `s` controls the size of innovations in the random walk. Smaller values for `s` tend to give smoother series.

Argument `n_seas` controls the number of `seasons`. When using quarterly data, for instance, `n_seas` should be `4`, and when using monthly data, `n_seas` should be `12`.

By default, the magnitude of seasonal effects can change over time. However, setting `s_seas` to `0` produces seasonal effects that are fixed, eg where "January" effect is the same every year, the "Feburary" effect is the same every year, and so on.

Value

Object of class `"bage_prior_rw2seasvary"` or `"bage_prior_rw2seasfix"`.

Mathematical details

When `RW2_Seas()` is used with a main effect,

$$\begin{aligned} \beta_j &= \alpha_j + \lambda_j \\ \alpha_j &\sim \text{N}(2\alpha_{j-1} - \alpha_{j-2}, \tau^2) \\ \lambda_j &\sim \text{N}(\lambda_{j-n}, \omega^2), \end{aligned}$$

and when it is used with an interaction,

$$\begin{aligned} \beta_{u,v} &= \alpha_{u,v} + \lambda_{u,v} \\ \alpha_{u,v} &\sim \text{N}(2\alpha_{u,v-1} - \alpha_{u,v-2}, \tau^2), \\ \lambda_{u,v} &\sim \text{N}(\lambda_{u,v-n}, \omega^2) \end{aligned}$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- α_j or $\alpha_{u,v}$ is an element of the random walk;
- λ_j or $\lambda_{u,v}$ is an element of the seasonal effect;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction;
- u denotes position within the "by" variable(s) of the interaction; and
- n is `n_seas`.

Parameter ω has a half-normal prior

$$\omega \sim \text{N}^+(0, \text{s\_seas}^2),$$

where `s_seas` is provided by the user. If `s_seas` is set to 0, then ω is 0, and the seasonal effects are fixed over time.

Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

See Also

- `RW2()` Second-order random walk, without seasonal effect
- `RW_Seas()` Random walk, with seasonal effect
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
RW2_Seas(n_seas = 4)             ## seasonal effects evolve
RW2_Seas(n_seas = 4, s_seas = 0) ## seasonal effects fixed
```

RW_Seas *Random Walk Prior with Seasonal Effect*

Description

Use a random walk with seasonal effects to model a main effect, or use multiple random walks, each with their own seasonal effects, to model an interaction. Typically used with main effects or interactions that involve time.

Usage

```
RW_Seas(n_seas, s = 1, s_seas = 1, along = NULL)
```

Arguments

`n_seas`	Number of seasons
`s`	Scale for prior for innovations in the random walk. Default is `1`.
`s_seas`	Scale for prior for innovations in the seasonal effect. Default is `1`. Can be `0`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `RW_Seas()` is used with an interaction, separate series are used for the "along" variable within each combination of the "by" variables.

Argument `s` controls the size of innovations in the random walk. Smaller values for `s` tend to give smoother series.

Argument `n_seas` controls the number of `seasons`. When using quarterly data, for instance, `n_seas` should be `4`, and when using monthly data, `n_seas` should be `12`.

By default, the magnitude of seasonal effects can change over time. However, setting `s_seas` to `0` produces seasonal effects that are fixed, eg where "January" effect is the same every year, the "Feburary" effect is the same every year, and so on.

Value

Object of class `"bage_prior_rwseasvary"` or `"bage_prior_rwseasfix"`.

Mathematical details

When `RW_Seas()` is used with a main effect,

$$\beta_j = \alpha_j + \lambda_j$$
$$\alpha_j \sim \text{N}(\alpha_{j-1}, \tau^2)$$
$$\lambda_j \sim \text{N}(\lambda_{j-n}, \omega^2),$$

and when it is used with an interaction,

$$\beta_{u,v} = \alpha_{u,v} + \lambda_{u,v}$$

$$\alpha_{u,v} \sim \text{N}(\alpha_{u,v-1}, \tau^2),$$

$$\lambda_{u,v} \sim \text{N}(\lambda_{u,v-n}, \omega^2)$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction;
- α_j or $\alpha_{u,v}$ is an element of the random walk;
- λ_j or $\lambda_{u,v}$ is an element of the seasonal effect;
- j denotes position within the main effect;
- v denotes position within the "along" variable of the interaction;
- u denotes position within the "by" variable(s) of the interaction; and
- n is `n_seas`.

Parameter ω has a half-normal prior

$$\omega \sim \text{N}^+(0, \text{s\_seas}^2),$$

where `s_seas` is provided by the user. If `s_seas` is set to 0, then ω is 0, and the seasonal effects are fixed over time.

Parameter τ has a half-normal prior

$$\tau \sim \text{N}^+(0, \text{s}^2),$$

where `s` is provided by the user.

See Also

- `RW()` Random walk without seasonal effect
- `RW2_Seas()` Second-order random walk, with seasonal effect
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction

Examples

```
RW_Seas(n_seas = 4)             ## seasonal effects evolve
RW_Seas(n_seas = 4, s_seas = 0) ## seasonal effects fixed
```

`set_datamod_outcome_rr3` *Specify RR3 Data Model*

Description

Specify a data model where the outcome variable has been randomly rounded to base 3.

Usage

```
set_datamod_outcome_rr3(mod)
```

Arguments

`mod`	An object of class `"bage_mod"`, created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.

Details

`set_datamod_outcome_rr3()` can only be used with Poisson and binomial models (created with `mod_pois()` and `mod_binom()`.)

Random rounding to base 3 (RR3) is a confidentialization technique that is sometimes applied by statistical agencies. RR3 is applied to integer data. The procedure for rounding value n is as follows:

- If n is divisible by 3, leave it unchanged
- If dividing n by 3 leaves a remainder of 1, then round down (subtract 1) with probability 2/3, and round up (add 2) with probability 1/3.
- If dividing n by 3 leaves a remainder of 1, then round down (subtract 2) with probability 1/3, and round up (add 1) with probability 2/3.

If `set_datamod_outcome_rr3()` is applied to a fitted model, it ’unfits’ the model, deleting existing estimates.

Value

A modified version of `mod`.

See Also

- datamods Overview of data models implemented in **bage**
- `mod_pois()`, `mod_binom()`, `mod_norm()` Specify a model for rates, probabilities, or means

Examples

```
## 'injuries' variable in 'injuries' dataset
## has been randomly rounded to base 3
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = popn) |>
  set_datamod_outcome_rr3() |>
  fit()
```

`set_disp` *Specify Prior for Dispersion or Standard Deviation*

Description

Specify the mean of prior for the dispersion parameter (in Poisson and binomial models) or the standard deviation parameter (in normal models.)

Usage

```
set_disp(mod, mean)
```

Arguments

`mod`	An object of class "bage_mod", created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`mean`	Mean value for the exponential prior. In Poisson and binomial models, can be set to 0.

Details

The dispersion or mean parameter has an exponential distribution with mean μ,

$$p(\xi) = \frac{1}{\mu} \exp\left(\frac{-\xi}{\mu}\right).$$

In Poisson and binomial models, `mean` can be set to `0`, implying that the dispersion term is also `0`. In normal models, `mean` must be non-negative.

If `set_disp()` is applied to a fitted model, it 'unfits' the model, deleting existing estimates.

Value

A bage_mod object

See Also

- `mod_pois()`, `mod_binom()`, `mod_norm()` Specify a model for rates, probabilities, or means
- `set_prior()` Specify prior for a term
- `set_n_draw()` Specify the number of draws
- `is_fitted()` Test whether a model is fitted

Examples

```
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = popn)
mod
mod |> set_disp(mean = 0.1)
mod |> set_disp(mean = 0)
```

set_n_draw *Specify Number of Draws from Prior or Posterior Distribution*

Description

Specify the number of draws from the posterior distribution to be used in model output. A newly-created bage_mod object has an n_draw value of 1000. Higher values may be appropriate for characterizing the tails of distributions, or for publication-quality graphics and summaries.

Usage

```
set_n_draw(mod, n_draw = 1000L)
```

Arguments

mod	An object of class "bage_mod", created with mod_pois(), mod_binom(), or mod_norm().
n_draw	Number of draws.

Details

If the new value for n_draw is greater than the old value, and the model has already been fitted, then the model is unfitted, and function fit() may need to be called again.

Value

A bage_mod object

See Also

- augment(), components() functions for drawing from prior or posterior distribution - the output of which is affected by the value of n_draw.
- mod_pois(), mod_binom(), mod_norm() Specify a model
- set_prior() Specify prior for a term
- set_disp() Specify prior for dispersion
- fit() Fit a model
- unfit() Reset a model

Examples

```
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = injuries,
                exposure = popn)
mod

mod |>
  set_n_draw(n_draw = 5000)
```

`set_prior` *Specify Prior for Model Term*

Description

Specify a prior distribution for an intercept, a main effect, or an interaction.

Usage

```
set_prior(mod, formula)
```

Arguments

`mod`	A bage_mod object, created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`formula`	A formula giving the term and a function for creating a prior.

Details

If `set_prior()` is applied to a fitted model, it 'unfits' the model, deleting existing estimates.

Value

A modified bage_mod object.

See Also

- priors Current choices for prior distributions
- `is_fitted()` Test whether a model is fitted
- `set_disp()` Specify prior for dispersion

Examples

```
mod <- mod_pois(injuries ~ age + year,
                data = injuries,
                exposure = popn)
mod
mod |> set_prior(age ~ RW2())
```

set_var_age *Specify Age Variable*

Description

Specify which variable (if any) represents age. Functions `mod_pois()`, `mod_binom()`, and `mod_norm()` try to infer the age variable from variable names, but do not always get it right.

Usage

```
set_var_age(mod, name)
```

Arguments

`mod`	An object of class `"bage_mod"`, created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`name`	The name of the age variable.

Details

In an R `formula`, a 'variable' is different from a 'term'. For instance,

`~ age + region + age:region`

contains variables `age` and `region`, and terms `age`, `region`, and `age:region`.

By default, **bage** gives a term involving age a (`RW()`) prior. Changing the age variable via `set_var_age()` can change priors: see below for an example.

If `set_var_age()` is applied to a fitted model, it 'unfits' the model, deleting existing estimates.

Value

A `bage_mod` object

See Also

- `set_var_sexgender()` Set sex or gender variable
- `set_var_time()` Set time variable
- `is_fitted()` Test whether a model is fitted
- internally, **bage** uses `poputils::find_var_age()` to locate age variables

Examples

```
## rename 'age' variable to something unusual
injuries2 <- injuries
injuries2$age_last_birthday <- injuries2$age

## mod_pois does not recognize age variable
mod <- mod_pois(injuries ~ age_last_birthday * ethnicity + year,
                data = injuries2,
                exposure = popn)
mod

## so we set the age variable explicitly
## (which, as a side effect, changes the prior on
## the age main effect)
mod |>
  set_var_age(name = "age_last_birthday")
```

set_var_sexgender *Specify Sex or Gender Variable*

Description

Specify which variable (if any) represents sex or gender. Functions mod_pois(), mod_binom(), and mod_norm() try to infer the sex/gender variable from variable names, but do not always get it right.

Usage

```
set_var_sexgender(mod, name)
```

Arguments

mod	An object of class "bage_mod", created with mod_pois(), mod_binom(), or mod_norm().
name	The name of the sex or gender variable.

Details

In an R formula, a 'variable' is different from a 'term'. For instance,

~ gender + region + gender:region

contains variables gender and region, and terms gender, region, and gender:region.

If set_var_sexgender() is applied to a fitted model, it 'unfits' the model, deleting existing estimates.

Value

A "bage_mod" object

See Also

- `set_var_age()` Set age variable
- `set_var_time()` Set time variable
- `is_fitted()` Test whether model is fitted
- internally, **bage** uses `poputils::find_var_sexgender()` to locate sex or gender variables
- internally, **bage** uses `poputils::find_label_female()` to locate female categories within a sex or gender variable
- internally, **bage** uses `poputils::find_label_male()` to locate male categories within a sex or gender variable

Examples

```
## rename 'sex' variable to something unexpected
injuries2 <- injuries
injuries2$biological_sex <- injuries2$sex

## mod_pois does not recognize sex variable
mod <- mod_pois(injuries ~ age * biological_sex + year,
                data = injuries2,
                exposure = popn)
mod

## so we set the sex variable explicitly
mod |>
  set_var_sexgender(name = "biological_sex")
```

---

set_var_time *Specify Time Variable*

Description

Specify which variable (if any) represents time. Functions `mod_pois()`, `mod_binom()`, and `mod_norm()` try to infer the time variable from variable names, but do not always get it right.

Usage

```
set_var_time(mod, name)
```

Arguments

`mod`	An object of class "bage_mod", created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`name`	The name of the time variable.

Details

In an R formula, a 'variable' is different from a 'term'. For instance,

~ time + region + time:region

contains variables time and region, and terms time, region, and time:region.

By default, **bage** gives a term involving time a (RW()) prior. Changing the time variable via set_var_time() can change priors: see below for an example.

If set_var_time() is applied to a fitted model, it 'unfits' the model, deleting existing estimates.

Value

A bage_mod object

See Also

- set_var_age() Set age variable
- set_var_sexgender() Sex sex or gender variable
- is_fitted() Test if model has been fitted
- internally, **bage** uses poputils::find_var_time() to locate time variables

Examples

```
## rename time variable to something unusual
injuries2 <- injuries
injuries2$calendar_year <- injuries2$year

## mod_pois does not recognize time variable
mod <- mod_pois(injuries ~ age * ethnicity + calendar_year,
                data = injuries2,
                exposure = popn)
mod

## so we set the time variable explicitly
## (which, as a side effect, changes the prior on
## the time main effect)
mod |>
  set_var_time(name = "calendar_year")
```

Sp *P-Spline Prior*

Description

Use a p-spline (penalised spine) to model main effects or interactions. Typically used with age, but can be used with any variable where outcomes are expected to vary smoothly from one element to the next.

Usage

```
Sp(n_comp = NULL, s = 1, along = NULL)
```

Arguments

`n_comp`	Number of spline basis functions (components) to use.
`s`	Scale for the prior for the innovations. Default is `1`.
`along`	Name of the variable to be used as the "along" variable. Only used with interactions.

Details

If `Sp()` is used with an interaction, separate splines are used for the "along" variable within each combination of the "by" variables.

Value

An object of class `"bage_prior_spline"`.

Mathematical details

When `Sp()` is used with a main effect,

$$\boldsymbol{\beta} = \boldsymbol{X}\boldsymbol{\alpha}$$

and when it is used with an interaction,

$$\boldsymbol{\beta}_u = \boldsymbol{X}\boldsymbol{\alpha}_u$$

where

- $\boldsymbol{\beta}$ is the main effect or interaction, with J elements;
- $\boldsymbol{beta}_u$ is a subvector of $\boldsymbol{\beta}$ holding values for the uth combination of the "by" variables;
- J is the number of elements of $\boldsymbol{\beta}$;
- U is the number of elements of $\boldsymbol{\beta}_u$;
- X is a $J \times n$ or $V \times n$ matrix of spline basis functions; and
- n is `n_comp`.

The elements of $\boldsymbol{\alpha}$ or $\boldsymbol{alpha}_u$ are assumed to follow a second-order random walk.

References

- Eilers, P.H.C. and Marx B. (1996). "Flexible smoothing with B-splines and penalties". Statistical Science. 11 (2): 89–121.

See Also

- RW() Smoothing via random walk
- RW2() Smoothing via second-order random walk
- SVD() Smoothing of age via singular value decomposition
- priors Overview of priors implemented in **bage**
- set_prior() Specify prior for intercept, main effect, or interaction
- **bage** uses function splines::bs() to construct spline basis functions

Examples

```
Sp()
Sp(n_comp = 10)
```

SVD — *SVD-Based Prior for Age or Age-Sex Profiles*

Description

Use components from a Singular Value Decomposition (SVD) to model a main effect or interaction involving age.

Usage

```
SVD(ssvd, n_comp = NULL, indep = TRUE)
```

Arguments

ssvd	Object of class "bage_ssvd" holding a scaled SVD. See below for scaled SVDs of databases currently available in **bage**.
n_comp	Number of components from scaled SVD to use in modelling. The default is half the number of components of ssvd.
indep	Whether to use separate or combined SVDs in terms involving sex or gender. Default is TRUE. See below for details.

Details

A SVD() prior assumes that the age, age-sex, or age-gender profiles for the quantity being modelled looks like they were drawn at random from an external demographic database. For instance, the prior obtained via

```
SVD(HMD)
```

assumes that age or age-sex profiles look like they were drawn from the Human Mortality Database.

If `SVD()` is used with an interaction involving variables other than age and sex/gender, separate profiles are constructed within each combination of other variables.

bage chooses the appropriate age-specific or age-sex-specific SVD values internally. The choice depends on the model term that the `SVD()` prior is applied to, and on the age labels used in `data` argument to `mod_pois()`, `mod_binom()` or `mod_norm()`. **bage** makes its choice when `set_prior()` is called.

Value

An object of class `"bage_prior_svd"`.

Joint or independent SVDs

Two possible ways of extracting patterns from age-sex-specific data are

1. carry out separate SVDs on separate datasets for each sex/gender; or
2. carry out a single SVD on dataset that has separate entries for each sex/gender.

Option 1 is more flexible. Option 2 is more robust to sampling or measurement errors. Option 1 is obtained by setting the `joint` argument to `FALSE`. Option 1 is obtained by setting the `indep` argument to `TRUE`. The default is `TRUE`.

Mathematical details

Case 1: Term involving age and no other variables

When `SVD()` is used with an age main effect,

$$\boldsymbol{\beta} = \boldsymbol{F}\boldsymbol{\alpha} + \boldsymbol{g},$$

where

- $\boldsymbol{\beta}$ is a main effect or interaction involving age;
- J is the number of elements of $\boldsymbol{\beta}$;
- n is the number of components from the SVD;
- $\boldsymbol{F}$ is a known matrix with dimension $J \times n$; and
- $\boldsymbol{g}$ is a known vector with J elements.

$\boldsymbol{F}$ and $\boldsymbol{g}$ are constructed from a large database of age-specific demographic estimates by performing an SVD and standardizing.

The elements of $\boldsymbol{\alpha}$ have prior

$$\alpha_k \sim \text{N}(0, 1), \quad k = 1, \cdots, K.$$

Case 2: Term involving age and non-sex, non-gender variable(s)

When `SVD()` is used with an interaction that involves age but that does not involve sex or gender,

$$\boldsymbol{\beta}_u = \boldsymbol{F}\boldsymbol{\alpha}_u + \boldsymbol{g},$$

where

- $\boldsymbol{\beta}_u$ is a subvector of $\boldsymbol{\beta}$ holding values for the uth combination of the non-age variables;
- V is the number of elements of $\boldsymbol{\beta}_u$;
- n is the number of components from the SVD;
- $\boldsymbol{F}$ is a known matrix with dimension $V \times n$; and
- $\boldsymbol{g}$ is a known vector with V elements.

Case 3: Term involving age, sex/gender, and no other variables

When `SVD()` is used with an interaction that involves age and sex or gender, there are two sub-cases, depending on the value of `indep`.

When `indep` is `TRUE`,

$$\boldsymbol{\beta}_s = \boldsymbol{F}_s \boldsymbol{\alpha}_s + \boldsymbol{g}_s,$$

and when `indep` is `FALSE`,

$$\boldsymbol{\beta} = \boldsymbol{F} \boldsymbol{\alpha} + \boldsymbol{g},$$

where

- $\boldsymbol{\beta}$ is an interaction involving age and sex/gender;
- $\boldsymbol{\beta}_s$ is a subvector of $\boldsymbol{\beta}$, holding values for sex/gender s;
- J is the number of elements in $\boldsymbol{\beta}$;
- S is the number of sexes/genders;
- n is the number of components from the SVD;
- $\boldsymbol{F}_s$ is a known $(J/S) \times n$ matrix, specific to sex/gender s;
- $\boldsymbol{g}_s$ is a known vector with J/S elements, specific to sex/gender s;
- $\boldsymbol{F}$ is a known $J \times n$ matrix, with values for all sexes/genders; and
- $\boldsymbol{g}$ is a known vector with J elements, with values for all sexes/genders.

The elements of $\boldsymbol{\alpha}_s$ and $\boldsymbol{\alpha}$ have prior

$$\alpha_k \sim \text{N}(0, 1).$$

Case 4: Term involving age, sex/gender, and other variable(s)

When `SVD()` is used with an interaction that involves age, sex or gender, and other variables, there are two sub-cases, depending on the value of `indep`.

When `indep` is `TRUE`,

$$\boldsymbol{\beta}_{u,s} = \boldsymbol{F}_s \boldsymbol{\alpha}_{u,s} + \boldsymbol{g}_s,$$

and when `indep` is `FALSE`,

$$\boldsymbol{\beta}_u = \boldsymbol{F} \boldsymbol{\alpha}_u + \boldsymbol{g},$$

where

- $\boldsymbol{\beta}$ is an interaction involving sex/gender;
- $\boldsymbol{\beta}_{u,s}$ is a subvector of $\boldsymbol{\beta}$, holding values for sex/gender s for the uth combination of the other variables;
- V is the number of elements in $\boldsymbol{\beta}_u$;
- S is the number of sexes/genders;
- n is the number of components from the SVD;
- $\boldsymbol{F}_s$ is a known $(V/S) \times n$ matrix, specific to sex/gender s;
- $\boldsymbol{g}_s$ is a known vector with V/S elements, specific to sex/gender s;
- $\boldsymbol{F}$ is a known $V \times n$ matrix, with values for all sexes/genders; and
- $\boldsymbol{g}$ is a known vector with V elements, with values for all sexes/genders.

Scaled SVDs of demographic databases in bage

- `HMD` Mortality rates from the Human Mortality Database.
- `LFP` Labor forcce participation rates from the OECD.

References

- For details of the construction of scaled SVDS see the vignette here.

See Also

- `SVD_AR()`, `SVD_AR1()`, `SVD_RW()`, `SVD_RW2()` SVD priors for for time-varying age profiles;
- `RW()` Smoothing via random walk
- `RW2()` Smoothing via second-order random walk
- `Sp()` Smoothing via splines
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- `set_var_sexgender()` Identify sex or gender variable in data

Examples

```
SVD(HMD)
SVD(HMD, n_comp = 3)
```

SVD_AR *Dynamic SVD-Based Priors for Age Profiles or Age-Sex Profiles*

Description

Use components from a Singular Value Decomposition (SVD) to model an interaction involving age and time, or age, sex/gender and time, where the coefficients evolve over time.

Usage

```
SVD_AR(ssvd, n_comp = NULL, indep = TRUE, n_coef = 2, s = 1)

SVD_AR1(ssvd, n_comp = NULL, indep = TRUE, min = 0.8, max = 0.98, s = 1)

SVD_RW(ssvd, n_comp = NULL, indep = TRUE, s = 1)

SVD_RW2(ssvd, n_comp = NULL, indep = TRUE, s = 1)
```

Arguments

`ssvd`	Object of class `"bage_ssvd"` holding a scaled SVD. See below for scaled SVDs of databases currently available in **bage**.
`n_comp`	Number of components from scaled SVD to use in modelling. The default is half the number of components of `ssvd`.
`indep`	Whether to use separate or combined SVDs in terms involving sex or gender. Default is `TRUE`. See below for details.
`n_coef`	Number of AR coefficients in `SVD_RW()`.
`s`	Scale for standard deviations terms.
`min, max`	Minimum and maximum values for autocorrelation coefficient in `SVD_AR()`. Defaults are `0.8` and `0.98`.

Details

`SVD_AR()`, `SVD_AR1()`, `SVD_RW()`, and `SVD_RW2()` priors assume that, in any given period, the age profiles or age-sex profiles for the quantity being modelled looks like they were drawn at random from an external demographic database. For instance, the `SVD_AR()` prior obtained via

```
SVD_AR(HMD)
```

assumes that profiles look like they were obtained from the Human Mortality Database.

Value

An object of class `"bage_prior_svd_ar"`, `"bage_prior_svd_rw"`, or `"bage_prior_svd_rw2"`.

Mathematical details

When the interaction being modelled only involves age and time, or age, sex/gender, and time

$$\boldsymbol{\beta}_t = \boldsymbol{F}\boldsymbol{\alpha}_t + \boldsymbol{g},$$

and when it involves other variables besides age, sex/gender, and time,

$$\boldsymbol{\beta}_{u,t} = \boldsymbol{F}\boldsymbol{\alpha}_{u,t} + \boldsymbol{g},$$

where

- $\boldsymbol{\beta}$ is an interaction involving age, time, possibly sex/gender, and possibly other variables;
- $\boldsymbol{\beta}_t$ is a subvector of $\boldsymbol{\beta}$ holding values for period t;
- $\boldsymbol{\beta}_{u,t}$ is a subvector of $\boldsymbol{\beta}_t$ holding values for the uth combination of the non-age, non-time, non-sex/gender variables for period t;
- J is the number of elements of $\boldsymbol{\beta}_t$;
- V is the number of elements of $\boldsymbol{\beta}_{u,t}$;
- n is `n_coef`;
- $\boldsymbol{F}$ is a known matrix with dimension $J \times n$ or $V \times n$;
- $\boldsymbol{g}$ is a known vector with J or V elements.

$\boldsymbol{F}$ and $\boldsymbol{g}$ are constructed from a large database of age-specific demographic estimates by performing an SVD and standardizing.

With `SVD_AR()`, the prior for the kth element of $\boldsymbol{\alpha}_t$ or $\boldsymbol{\alpha}_{u,t}$ is

$$\alpha_{k,t} = \phi_1 \alpha_{k,t-1} + \cdots + \phi_n \beta_{k,t-n} + \epsilon_{k,t}$$

or

$$\alpha_{k,u,t} = \phi_1 \alpha_{k,u,t-1} + \cdots + \phi_n \beta_{k,u,t-n} + \epsilon_{k,u,t};$$

with `SVD_AR1()`, it is

$$\alpha_{k,t} = \phi \alpha_{k,t-1} + \epsilon_{k,t}$$

or

$$\alpha_{k,u,t} = \phi \alpha_{k,u,t-1} + \epsilon_{k,u,t};$$

with `SVD_RW()`, it is

$$\alpha_{k,t} = \alpha_{k,t-1} + \epsilon_{k,t}$$

or

$$\alpha_{k,u,t} = \alpha_{k,u,t-1} + \epsilon_{k,u,t};$$

and with `SVD_RW2()`, it is

$$\alpha_{k,t} = 2\alpha_{k,t-1} - \alpha_{k,t-2} + \epsilon_{k,t}$$

or

$$\alpha_{k,u,t} = 2\alpha_{k,u,t-1} - \alpha_{k,u,t-2} + \epsilon_{k,u,t}.$$

For more on the ϕ and ϵ, see `AR()`, `AR1()`, `RW()`, and `RW2()`.

Scaled SVDs of demographic databases in bage

- `HMD` Mortality rates from the Human Mortality Database.
- `LFP` Labor forcce participation rates from the OECD.

References

- For details of the construction of scaled SVDS see the vignette here.

See Also

- `SVD()` SVD prior for non-time-varying terms
- `RW()` Smoothing via random walk
- `RW2()` Smoothing via second-order random walk
- `Sp()` Smoothing via splines
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- `set_var_sexgender()` Identify sex or gender variable in data

Examples

```
SVD_AR1(HMD)
SVD_RW(HMD, n_comp = 3)
SVD_RW2(HMD, indep = FALSE)
```

tidy.bage_mod *Summarize Terms from a Fitted Model*

Description

Summarize the intercept, main effects, and interactions from a fitted model.

Usage

```
## S3 method for class 'bage_mod'
tidy(x, ...)
```

Arguments

x	Object of class "bage_mod", typically created with mod_pois(), mod_binom(), or mod_norm().
...	Unused. Included for generic consistency only.

Details

The tibble returned by `tidy()` contains the following columns:

- `term` Name of the intercept, main effect, or interaction
- `spec` Specification for prior
- `n_par` Number of parameters
- `n_par_free` Number of free parameters
- `std_dev` Standard deviation for point estimates.

With some priors, the number of free parameters is less than the number of parameters for that term. For instance, an `SVD()` prior might use three vectors to represent 101 age groups so that the number of parameters is 101, but the number of free parameters is 3.

`std_dev` is the standard deviation across elements of a term, based on point estimates of those elements. For instance, if the point estimates for a term with three elements are 0.3, 0.5, and 0.1, then the value for `std_dev` is

```
sd(c(0.3, 0.5, 0.1))
```

`std_dev` is a measure of the contribution of a term to variation in the outcome variable.

Value

A tibble

References

`std_dev` is modified from Gelman et al. (2014) *Bayesian Data Analysis. Third Edition*. pp396–397.

See Also

- `augment()` Extract data, and values for rates, probabilities, or means
- `components()` Extract values for hyper-parameters

Examples

```
mod <- mod_pois(injuries ~ age + sex + year,
                data = injuries,
                exposure = popn)
mod <- fit(mod)
tidy(mod)
```

---

unfit *Unfit a Model*

Description

Reset a model, deleting all estimates.

Usage

```
unfit(mod)
```

Arguments

mod	A fitted object of class `"bage_mod"`, object, created through a call to `mod_pois()`, `mod_binom()`, or `mod_norm()`.

Value

An unfitted version of `mod`.

See Also

- `fit()` Fit a model
- `mod_pois()`, `mod_binom()`, `mod_norm()` Specify a model
- Functions such as `set_prior()`, `set_disp()` and `set_var_age()` unfit models as side effects.

Examples

```
## create a model, which starts out unfitted
mod <- mod_pois(injuries ~ age + sex + year,
                data = injuries,
                exposure = popn)
is_fitted(mod)

## calling 'fit' produces a fitted version
```

```
mod <- fit(mod)
is_fitted(mod)

## calling 'unfit' resets the model
mod <- unfit(mod)
is_fitted(mod)
```

`us_acc_deaths` *Accidental Deaths in the USA*

Description

Counts of accidental deaths in the USA, by month, for 1973-1978.

Usage

```
us_acc_deaths
```

Format

A tibble with 72 rows and the following columns:

- `month`: Year and month.
- `deaths`: Count of deaths.

Source

Reformatted version of `datasets::USAccDeaths`.

Index