Package 'bage'

May 20, 2026

Type Package

Title Bayesian Estimation and Forecasting of Age-Specific Rates

Version 0.10.9

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 (>= 4.2.0), rvec (>= 1.0.0)

Imports cli, generics, lifecycle, Matrix, methods, parallel, poputils (>= 0.3.4), sparseMVN, stats, tibble, TMB (>= 1.9.1), utils, vctrs

Suggests bookdown, dplyr, ggplot2, knitr, mockery, patchwork, rmarkdown, rlang, testthat (>= 3.0.0), tidyr

Config/testthat/edition 3

Config/Needs/website bookdown

LinkingTo TMB (>= 1.9.1), RcppEigen

VignetteBuilder knitr

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

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

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author John Bryant [aut, cre], Junni Zhang [aut], Bayesian Demography Limited [cph]

Maintainer John Bryant <john@bayesiandemography.com>

Repository CRAN

Date/Publication 2026-05-20 06:21:17 UTC

Contents

`bage-package` *bage: Bayesian Estimation and Forecasting of Age-Specific Rates*

Description

Modeling of rates, probabilities, and other values, typically disaggregated by age. Estimation is done using TMB, which makes it fast and scalable.

Example workflow

1. Specify model using `mod_pois()`
2. Fit model using `fit()`
3. Extract results using `augment()`
4. Check model using `replicate_data()`

Functions

Specify model

- `mod_pois()` Specify a Poisson model
- `mod_binom()` Specify a binomial model
- `mod_norm()` Specify a normal model
- `set_prior()` Specify prior for main effect or interaction
- priors Overview of priors for main effects or interactions
- `set_disp()` Specify prior for dispersion/variance
- `set_covariates()` Add covariates to model
- datamods Overview of data models (measurement error models)
- confidential Overview of confidentialization models

Fit model

- `fit()` Derive posterior distribution

Extract results

- `augment()` Original data, plus observation-level estimates
- components() Hyper-parameters
- `dispersion()` Dispersion parameter (a type of hyper-parameter)
- `tidy()` One-line summary
- `set_n_draw()` Specify number of prior or posterior draws

Forecast

- `forecast()` Use model to obtain estimates for future periods

Check model

- `replicate_data()` Compare real and simulated data
- `report_sim()` Simulation study of model

Data

- datasets Overview of datasets
- svds Overview of scaled SVDs

Author(s)

Maintainer: John Bryant <john@bayesiandemography.com>

Authors:

- John Bryant <john@bayesiandemography.com>
- Junni Zhang <junnizhang@163.com>

Other contributors:

- Bayesian Demography Limited [copyright holder]

See Also

Useful links:

- https://bayesiandemography.github.io/bage/
- https://github.com/bayesiandemography/bage
- Report bugs at https://github.com/bayesiandemography/bage/issues

---

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,
  shape1 = 5,
  shape2 = 5,
  along = NULL,
  con = c("none", "by")
)
```

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.
shape1, shape2	Parameters for beta-distribution prior for coefficients. Defaults are 5 and 5.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If AR() is used with an interaction, then 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,

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

and when it is used with an interaction,

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

where

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

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

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

The correlation coefficients $\phi_1, \cdots, \phi_{\texttt{n\_coef}}$ each have prior

$$\phi_k \sim \mathrm{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

References

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

See Also

- AR1() Special case of AR(). Can be more numerically stable than higher-order models.
- Lin_AR(), Lin_AR1() Line with AR errors
- RW2_AR(), RW2_AR1() RW2 with AR errors
- priors Overview of priors implemented in **bage**
- set_prior() Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

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(
  s = 1,
  shape1 = 5,
  shape2 = 5,
  min = 0.8,
  max = 0.98,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

s	Scale for the prior for the innovations. Default is 1.
shape1, shape2	Parameters for beta-distribution prior for coefficients. Defaults are 5 and 5.
min, max	Minimum and maximum values for autocorrelation coefficient. Defaults are 0.8 and 0.98.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

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,

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

and when it is used with an interaction,

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

where

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

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

where `s` is provided by the user.

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

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

where

$$\phi' \sim \mathrm{Beta}(\mathtt{shape1}, \mathtt{shape2}).$$

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- `RW2_AR()`, `RW2_AR1()` RW2 with AR errors
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

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

---

augment.bage_mod *Extract Data and Modeled 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 modeled values.

Usage

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

Arguments

x	Object of class "bage_mod", typically created with mod_pois(), mod_binom(), or mod_norm().
rows	Results are returned only for these rows of data. A logical vector, an index vector, or a logical expression evaluated within data. Optional
quiet	Whether to suppress messages. Default is FALSE.
...	Unused. Included for generic consistency only.

Details

The rows argument can be used to obtain results for a subset within data. This is faster, and uses less memory, than generating results for the whole dataset and then subsetting.

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 modeled 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 modeled values are draws from the joint prior distribution. In other words, the modeled 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 NAs, 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
- dispersion() Extract values for dispersion
- 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

```
set.seed(0)

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

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

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

## results for females only
mod |>
  augment(rows = sex == "Female")

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

## fitting model and calling 'augument'A
## 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 = nzl_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,
- trends, seasonal effects, errors,
- SVD, spline, and covariate coefficients,
- standard deviations, correlation coefficients.

Usage

```
## S3 method for class 'bage_mod'
components(object, quiet = FALSE, original_scale = FALSE, ...)
```

Arguments

`object`	Object of class `"bage_mod"`, typically created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`quiet`	Whether to suppress messages. Default is `FALSE`.
`original_scale`	Whether values for `"effect"`, `"trend"`, `"season"`, `"error"` and `"disp"` components from a normal model are on the original scale or the transformed scale. 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 values returned 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 values returned are draws from the joint *prior* distribution. In other words, the values incorporate model priors, and any `exposure`, `size`, or `weights` argument, but not observed outcomes.

Scaling and Normal models

Internally, models created with `mod_norm()` are fitted using transformed versions of the outcome and weights variables. By default, when `components()` is used with these models, it returns values for `.fitted` that are based on the transformed versions. To instead obtain values for `"effect"`, `"trend"`, `"season"`, `"error"` and `"disp"` that are based on the untransformed versions, set `original_scale` to `TRUE`.

See Also

- `augment()` Extract values for rates, means, or probabilities, together with original data
- `dispersion()` Extract values for dispersion
- `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

```
set.seed(0)

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

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

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

## extract posterior distribution
## of hyper-parameters
```

```
mod |>
  components()

## fit normal model
mod <- mod_norm(value ~ age * diag + year,
                data = nld_expenditure,
                weights = 1) |>
  fit()

## dispersion (= standard deviation in normal model)
## on the transformed scale
mod |>
  components() |>
  subset(component == "disp")

## disperson on the original scale
mod |>
  components(original_scale = TRUE) |>
  subset(component == "disp")
```

`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,
  v = NULL,
  n_comp = NULL,
  indep = NULL,
  age_labels = NULL,
  ...
)
```

Arguments

`object`	An object of class `"bage_ssvd"`.
`v`	Version of scaled SVD components to use. If no value is suppled, the most recent version is used.
`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 the data contains a sex/gender variable. The default is to use independent SVDs. To obtain results for the total population when the data contains a sex/gender variable, set indep to NA.
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.
- HFD Fertility rates from the Human Fertility 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 modeled independently
components(LFP, n_comp = 3)

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

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

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

`computations` *Information on Computations Performed During Model Fitting*

Description

Get information on computations performed by function `fit()`. The information includes the total time used for fitting, and the time used for two particular tasks that can be slow: running the optimizer `stats::nlminb()`, and drawing from the multivariate normal returned by the TMB. It also includes values returned by the optimizer: the number of iterations needed, and messages about convergence.

Usage

```
computations(object)
```

Arguments

`object`	A fitted object of class `"bage_mod"`.

Value

A tibble with the following variables:

- `time_total` Seconds used for whole fitting process.
- `time_max` Seconds used for optimisiation.
- `time_draw` Seconds used by function `TMB::sdreport()`.
- `iter` Number of iterations required for optimization.
- `message` Message about convergence returned by optimizer.

See Also

- `mod_pois()`,`mod_binom()`,`mod_norm()` Specify a model
- `fit()` Fit a model
- `tidy()` Summarise a model
- `set_n_draw()` Specify number of posterior draws

Examples

```
mod <- mod_pois(divorces ~ age + sex + time,
                data = nzl_divorces,
                exposure = population) |>
  fit()

computations(mod)
```

confidential *Confidentialization*

Description

The models for rates, probabilities, or means created with functions `mod_pois()`, `mod_binom()`, and `mod_norm()` can be extended by adding descriptions of confidentalization procedures applied to the outcome variable.

Details

Data models for outcome variable

Function	Confidentialization procedure	pois	binom	norm
`set_confidential_rr3()`	Outcome randomly rounded to multiple of 3	Yes	Yes	No

CSA *Scaled SVD Components from Census School Attendance Data*

Description

An object of class `"bage_ssvd"` holding scaled SVD components derived from census data on school attendance. The attendance data is assembed by the United Nations Statistics Division. `CSA` holds 5 components.

Usage

```
CSA
```

Format

Object of class `"bage_ssvd"`.

Versions:

- `"v2025"` (default). Data downloaded on 2025-11-05

Warning

Compared other age-sex patterns for other demographic processes such as mortality, age-sex patterns for school attendance show substantial variation across populations. More components may be needed to obtain satisfactory models of age-sex patterns for school attendance than for other processes.

Source

Derived from data in the "Population 5 to 24 years of age by school attendance, sex and urban/rural residence" table from the Population Censuses' Datasets database assembled by the United Nations Statistics Division. Code to create `CSA` is in folder '`data-raw/ssvd_csa`' in the source code for the **bage** package.

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- `SVD()` A prior based on a scaled SVD

`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.

Details

Function	Assumptions about measurement error	Poisson
`set_datamod_miscount()`	Reported outcome has undercount and overcount	Yes
`set_datamod_undercount()`	Reported outcome has undercount	Yes
`set_datamod_overcount()`	Reported outcome has overcount	Yes
`set_datamod_noise()`	Reported outcome unbiased, but with positive and negative measurement errors	Yes*
`set_datamod_exposure()`	Reported exposure unbiased, but with positive and negative measurement errors	Yes*

*Models with no dispersion term for rates.

`datasets` *Datasets*

Description

Datasets in **bage**

Details

dataset	Outcome	Variables	Country
isl_deaths	Deaths	age, sex, time, deaths, popn	Iceland
kor_births	Births	age, region, time, popn, gdp_pc_2023, dens2020	South Korea
nld_expenditure	Health expenditure	diag, age, year, value	Netherlands
nzl_divorces	Divorces	age, sex, time, divorces, population	New Zealand
nzl_households	One-person households	age, region, year, oneperson, total	New Zealand
nzl_injuries	Accidental deaths	age, sex, ethnicity, year, injuries, popn	New Zealand
prt_deaths	Deaths	age, time, deaths, exposure	Portugal
swe_infant	Infant deaths	county, time, births, deaths	Sweden
usa_deaths	Accidental deaths	month, deaths	United States

`data_wmd` *Data to Create Scaled SVD Object Based on World Marriage Database*

Description

A subset of the data needed to produce a scaled SVD object, derived from data from the World Marriage Database. The data is formatted using function `data_ssvd_wmd()` in package **bssvd**.

Usage

```
data_wmd
```

Format

A tibble with 6 rows and with columns `version`, `type`, `labels_age`, `labels_sexgender`, `matrix`, and `offset`.

Source

Derived from data from the *World Marriage Data 2019* database available on the United Nations Population Division website, and assembled by the UNPD from national census and survey data.

See Also

- `ssvd()` Function to create scaled SVD objects
- WMD_C Scaled SVD object based on a full set of World Marriage Database data.
- Scaled SVDs Overview of scaled SVDs implemented in **bage**

`dispersion` *Extract Values for Dispersion*

Description

Extract values for the 'dispersion' parameter from a model object.

Usage

```
dispersion(object, quiet = FALSE, original_scale = FALSE)
```

Arguments

`object`	Object of class `"bage_mod"`, typically created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`quiet`	Whether to suppress messages. Default is `FALSE`.
`original_scale`	Whether values for disperson are on the original scale or the transformed scale. Default is `FALSE`.

Value

An rvec (or `NULL` if the model does not include a dispersion parameter.)

Fitted vs unfitted models

`dispersion()` is typically called on a fitted model. In this case, the values for dispersion are draws from the posterior distribution. `dispersion()` can, however, be called on an unfitted model. In this case, the values are drawn from the prior distribution.

Scaling and Normal models

Internally, models created with `mod_norm()` are fitted using transformed versions of the outcome and weights variables. By default, when `dispersion()` is used with these models, it returns values on the transformed scale. To instead obtain values on the untransformed scale, set `original_scale` to `TRUE`.

See Also

- `components()` Extract values for hyper-parameters, including dispersion
- `set_disp()` Specify a prior for dispersion

Examples

```
set.seed(0)

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

## prior distribution
mod |>
  dispersion()

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

## posterior distribution
mod |>
  dispersion()

## fit normal model
mod <- mod_norm(value ~ age * diag + year,
                data = nld_expenditure,
                weights = 1) |>
  fit()

## values on the transformed scale
mod |>
  dispersion()

## values on the original scale
mod |>
  dispersion(original_scale = TRUE)
```

DRW *Damped Random Walk Prior*

Description

Use a damped random walk as a model for a main effect, or use multiple damped random walks as a model for an interaction. Typically used with terms that involve time, particularly in forecasts. Damping often improves forecast accuracy.

Usage

```
DRW(
  s = 1,
  sd = 1,
  shape1 = 5,
```

```
  shape2 = 5,
  min = 0.8,
  max = 0.98,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

s	Scale for the prior for the innovations. Default is 1.
sd	Standard deviation of initial value. Default is 1. Can be 0.
shape1, shape2	Parameters for beta-distribution prior for damping coefficient. Defaults are 5 and 5.
min, max	Minimum and maximum values for damping coefficient. Defaults are 0.8 and 0.98.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If DRW() is used with an interaction, a separate damped random walk is constructed within each combination of the 'by' variables.

Arguments min and max can be used to control the amount of damping that occurs.

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

Argument sd controls variance in initial values. Setting sd to 0 fixes initial values at 0.

Value

An object of class "bage_prior_drwrandom" or "bage_prior_drwzero".

Mathematical details

When DRW() is used with a main effect,

$$\beta_1 \sim \mathrm{N}(0, \mathtt{sd}^2)$$
$$\beta_j \sim \mathrm{N}(\phi\beta_{j-1}, \tau^2), \quad j > 1$$

and when it is used with an interaction,

$$\beta_{u,1} \sim \mathrm{N}(0, \mathtt{sd}^2)$$
$$\beta_{u,v} \sim \mathrm{N}(\phi\beta_{u,v-1}, \tau^2), \quad v > 1$$

where

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

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

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

where

$$\phi' \sim \text{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Standard deviation τ has a half-normal prior

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

where `s` is provided by the user.

`DRW()` has the same basic structure as `AR1()`. However, in `DRW()`, τ controls the variance of the innovations, but in `AR1()` τ controls the marginal variance of each β_j or $\beta_{u,v}$.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

- `DRW2()` Damped second-order random walk
- `RW()` Random walk, without damping
- `RW2()` Second-order random walk, without damping
- `RW_Seas()` Random walk with seasonal effect
- `AR()` Autoregressive with order k
- `AR1()` Autoregressive with order 1
- `Sp()` Smoothing via splines
- `SVD()` Smoothing over age using singular value decomposition
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

```
DRW()
DRW(min = 0, max = 1)
DRW(sd = 0)
```

DRW2 *Damped Second-Order Random Walk Prior*

Description

Use a damped second-order random walk as a model for a main effect, or use multiple second-order random walks as a model for an interaction. A damped second-order random walk is a random walk with drift where the drift term varies, with a tendency to converge on zero. It is typically used with terms that involve time, where there are sustained trends upward or downward. Damping often improves forecast accuracy.

Usage

```
DRW2(
  s = 1,
  sd = 1,
  sd_slope = 1,
  shape1 = 5,
  shape2 = 5,
  min = 0.8,
  max = 0.98,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

s	Scale for the prior for the innovations. Default is 1.
sd	Standard deviation of initial value. Default is 1. Can be 0.
sd_slope	Standard deviation of initial slope. Default is 1.
shape1, shape2	Parameters for beta-distribution prior for damping coefficient. Defaults are 5 and 5.
min, max	Minimum and maximum values for damping coefficient. Defaults are 0.8 and 0.98.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If `DRW2()` is used with an interaction, a separate damped random walk is constructed within each combination of the 'by' variables.

Arguments `min` and `max` can be used to control the amount of damping that occurs.

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

Argument `sd` controls variance in initial values. Setting `sd` to `0` fixes initial values at `0`.

Argument `sd_slope` controls variance in the initial slope.

Value

An object of class `"bage_prior_drw2random"` or `"bage_prior_drw2zero"`.

Mathematical details

When `DRW2()` is used with a main effect,

$$\begin{aligned}\beta_1 &\sim \mathrm{N}(0, \texttt{sd}^2)\\ \beta_2 &\sim \mathrm{N}(\beta_1, \texttt{sd\_slope}^2)\\ \beta_j &\sim \mathrm{N}(\beta_{j-1} + \phi(\beta_{j-1} - \beta_{j-2}), \tau^2), \quad j = 2, \cdots, J\end{aligned}$$

and when it is used with an interaction,

$$\begin{aligned}\beta_{u,1} &\sim \mathrm{N}(0, \texttt{sd}^2)\\ \beta_{u,2} &\sim \mathrm{N}(\beta_{u,1}, \texttt{sd\_slope}^2)\\ \beta_{u,v} &\sim \mathrm{N}(\beta_{u,v-1} + \phi(\beta_{u,v-1} - \beta_{u,v-2}), \tau^2), \quad v = 3, \cdots, V\end{aligned}$$

where

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

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

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

where

$$\phi' \sim \mathrm{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Standard deviation τ has a half-normal prior

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

where `s` is provided by the user.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

- `DRW()` Damped first-order random walk
- `RW2()` Second-order random walk, without damping
- `RW2_Seas()` Second order random walk with seasonal effect
- `AR()` Autoregressive with order k
- `AR1()` Autoregressive with order 1
- `Sp()` Smoothing via splines
- `SVD()` Smoothing over age via singular value decomposition
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

```
DRW2()
DRW2(s = 0.5)
DRW2(min = 0, max = 1)
```

`fit.bage_mod` *Fit a Model*

Description

Derive the posterior distribution for a model.

Usage

```
## S3 method for class 'bage_mod'
fit(
  object,
  method = c("standard", "inner-outer"),
  vars_inner = NULL,
  optimizer = c("multi", "nlminb", "BFGS", "CG"),
  quiet = TRUE,
  max_jitter = 1e-04,
  start_oldpar = FALSE,
  ...
)
```

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.
optimizer	Which optimizer to use. Current choices are "multi", "nlminb", "BFGS", and "CG". Default is "multi". See below for details.
quiet	Whether to suppress messages from optimizer. Default is TRUE.
max_jitter	Maximum quantity to add to diagonal of precision matrix, if Cholesky factorization is failing. Default is 0.0001.
start_oldpar	Whether the optimizer should start at previous estimates. Used only when fit() is being called on a fitted model. Default is FALSE.
...	Not currently used.

Value

A bage_mod object

Estimation methods

When method is "standard" (the default), all parameters, other than the lowest-level rates, probabilities, or means are jointly estimated within TMB.

When method is "inner-outer", estimation is carried out in multiple steps, which, in large models, can sometimes reduce computation times. In Step 1, a model only using the inner variables is fitted to the data. In Step 2, a model only using the outer variables is fitted to the data. In Step 3, values for dispersion are calculated. Parameter estimates from steps 1, 2, and 3 are then combined.

Optimizer

The choices for the optimizer argument are:

- "multi" Try "nlminb", and if that fails, restart from the parameter values where "nlminb" stopped, using "BFGS". The default.
- "nlminb" stats::nlminb()
- "BFGS" stats::optim() using method "BFGS".
- "GC" stats::optim() using method "CG" (conjugate gradient).

Cholesky factorization and max_jitter

Sampling from the posterior distribution requires performing a Cholesky factorization of the precision matrix returned by TMB. This factorization sometimes fails because of numerical problems. Adding a small quantity to the diagonal of the precision matrix can alleviate numerical problems, while potentially reducing accuracy. If the Cholesky factorization initially fails, bage will try again with progressively larger quantities added to the diagonal, up to the maximum set by max_jitter. Increasing the value of max_jitter can help suppress numerical problems. A safer strategy, however, is to simplify the model, or to use more informative priors.

Aggregation

Up to version 0.9.8 of bage, fit() always aggregated across cells with identical values of the predictor variables in formula (ie the variables to the right of ~) before fitting. For instance, if a dataset contained deaths and population disaggregated by age and sex, but the model formula was deaths ~ age, then fit() would aggregate deaths and population within each age category before fitting the model. From version 0.9.9, fit() only aggregates across cells with identical values if no data model is used, and if the model is Poisson with dispersion set to 0 or is normal. Note that this change in behavior has no effect on most models, since most models include all variables used to classify outcomes.

See Also

- mod_pois() Specify a Poisson model
- mod_binom() Specify a binomial model
- mod_norm() Specify a normal model
- augment() Extract values for rates, probabilities, or means, together with original data
- components() Extract values for hyper-parameters
- dispersion() Extract values for dispersion
- 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
- Mathematical Details vignette

Examples

```
## specify model
mod <- mod_pois(injuries ~ age + sex + year,
                data = nzl_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,
  labels = NULL,
  output = c("augment", "components"),
  include_estimates = FALSE,
  rows = NULL,
  quiet = FALSE,
  ...
)
```

Arguments

object	A bage_mod object, typically created with mod_pois(), mod_binom(), or mod_norm().
newdata	Data frame with data for future periods.

labels	Labels for future time periods.
output	Type of output returned. "augment" (the default) or "components".
include_estimates	Whether to include historical estimates along with the forecasts. Default is FALSE.
rows	A logical vector, an index vector, or a logical expression specifying which rows to return from an "augment" forecast. Can only be used if include_estimates is FALSE.
quiet	Whether to suppress messages. Default is FALSE.
...	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. Optionally, generate values for the outcome variable.

forecast() generates values for the outcome variable when,

- output is "augment",
- a value has been supplied for newdata,
- newdata included a value for the exposure, size, or weights variable (except if exposure = 1 or weights = 1 in the original call to mod_pois() or mod_norm()).

Mathematical Details gives more details on the internal calculations in forecasting.

Output format

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.

Forecasting with covariates

Models that contain covariates can be used in forecasts, provided that

- all coefficients (the ζ_p) are estimated from historical data via `fit()`, and
- if any covariates (the columns of $\boldsymbol{Z}$) are time-varying, then future values for these covariates are supplied via the `newdata` argument.

Forecasting with data models

Models that contain data models can be used in forecasts, provided that

- the data models have no time-varying parameters, or
- future values for time-varying parameters are supplied when the data model is first specified.

For examples, see the Data Models vignette.

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()` Specify a Poisson model
- `mod_binom()` Specify a binomial model
- `mod_norm()` Specify a normal model
- `fit()` Fit a model
- `augment()` Extract values for rates, probabilities, or means, together with original data
- `components()` Extract values for hyper-parameters
- Mathematical Details vignette

Examples

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

## forecasts
```

```
mod |>
  forecast(labels = 2019:2024)

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

## only selected rows
mod |>
  forecast(labels = 2019:2024,
           rows = age == "40-44")

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

## hold back some data and forecast
library(dplyr, warn.conflicts = FALSE)
data_historical <- nzl_injuries |>
  filter(year <= 2015)
data_forecast <- nzl_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 using GDP per capita in 2023 as a covariate
mod_births <- mod_pois(births ~ age * region + time,
                       data = kor_births,
                       exposure = popn) |>
  set_covariates(~ gdp_pc_2023) |>
  fit()
mod_births |>
  forecast(labels = 2024:2025)
```

`generate.bage_prior_ar` *Generate Values from Priors*

Description

Generate draws from priors for model terms.

Usage

```
## S3 method for class 'bage_prior_ar'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_drwrandom'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_drwzero'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_drw2random'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_drw2zero'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_known'
generate(x, n_element = 20, n_draw = 25, ...)

## S3 method for class 'bage_prior_lin'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_linar'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_linex'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_norm'
generate(x, n_element = 20, n_draw = 25, ...)

## S3 method for class 'bage_prior_normfixed'
generate(x, n_element = 20, n_draw = 25, ...)

## S3 method for class 'bage_prior_rwrandom'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rwrandomseasfix'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rwrandomseasvary'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rwzero'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rwzeroseasfix'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)
```

```
## S3 method for class 'bage_prior_rwzeroseasvary'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2random'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2randomar'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2randomseasfix'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2randomseasvary'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2zero'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2zeroar'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2zeroseasfix'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_rw2zeroseasvary'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_spline'
generate(x, n_along = 20, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd'
generate(x, n_element = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_ar'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_drwrandom'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_drwzero'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_drw2random'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_drw2zero'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)
```

```
## S3 method for class 'bage_prior_svd_lin'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_linex'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_rwrandom'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_rwzero'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_rw2random'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)

## S3 method for class 'bage_prior_svd_rw2zero'
generate(x, n_along = 5, n_by = 1, n_draw = 25, ...)
```

Arguments

x	Object of class "bage_prior"
n_along	Number of elements of 'along' dimension. Default is 20.
n_by	Number of combinations of 'by' variables. Default is 1.
n_draw	Number of draws. Default is 25.
...	Unused. Included for generic consistency only.
n_element	Number of elements in term, in priors that do not distinguish 'along' and 'by' dimensions. Default is 20.

Details

Some priors distinguish between 'along' and 'by' dimensions, while others do not: see priors for a complete list. Arguments n_along and n_by are used with priors that make the distinction, and argument n_element is used with priors that do not.

Value

A tibble

See Also

- priors Overview of priors implemented in **bage**

Examples

```
## prior that distinguishes 'along' and 'by'
x <- RW()
generate(x, n_along = 10, n_by = 2)
```

```
## prior that does not distinguish
x <- N()
generate(x, n_element = 20)

## SVD_AR(), SVD_RW(), and SVD_RW2()
## distinguish 'along' and 'by'
x <- SVD_AR(HFD)
generate(x, n_along = 5, n_by = 2)

## SVD() does not
x <- SVD(HFD)
generate(x, n_element = 10)
```

`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,
  v = NULL,
  n_draw = 20,
  n_comp = NULL,
  indep = NULL,
  age_labels = NULL,
  ...
)
```

Arguments

`x`	An object of class `"bage_ssvd"`.
`v`	Version of data to use.
`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 the data contains a sex/gender variable. The default is to use independent SVDs. To obtain results for the total population when the data contains a sex/gender variable, set `indep` to `NA`.

age_labels	Age labels for the desired age or age-sex profile. If no labels are supplied, the most detailed profile available is used.
...	Unused. Included for generic consistency only.

Value

A tibble

Scaled SVDs of demographic databases in bage

- HMD Mortality rates from the Human Mortality Database.
- HFD Fertility rates from the Human Fertility 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

```
## females and males modeled independently
generate(HMD)

## joint model for females and males
generate(HMD, indep = FALSE)

## SVD for females and males combined
generate(HMD, indep = NA)

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

HFD *Scaled SVD Components from Human Fertility Database*

Description

An object of class "bage_ssvd" holding scaled SVD components derived from data from the Human Fertility Database. HFD holds 5 components.

Usage

```
HFD
```

Format

Object of class "bage_ssvd".

Versions:

- "v2025" (default) Data published on 2025-07-24
- "v2024" Data published on October 2024-10-23

Source

Derived from data from the Human Fertility Database. Max Planck Institute for Demographic Research (Germany) and Vienna Institute of Demography (Austria). Code to create HFD is in folder 'data-raw/ssvd_hfd' in the source code for the **bage** package.

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- SVD() A prior based on a scaled SVD

HIMD_R *Scaled SVD Components from Human Internal Migration Database*

Description

Objects of class "bage_ssvd" holding scaled SVD components derived from data from the Human Internal Migration Database. HIMD_P1, HIMD_P5, and HIMD_R each hold 5 components

Usage

```
HIMD_R

HIMD_P1

HIMD_P5
```

Format

Object of class "bage_ssvd".

Versions:

- "v2024" (default) Data published on 2024-10-23

Details

- HIMD_P1 is derived from data on 1-year migration probabilities, ie the probability that a person will migrate during a time interval of 1 year.
- HIMD_P5 is derived from data on 5-year migration probabilities, ie the probability that a person will migrate during a time interval of 5 years.
- HIMD_R is derived from data on 1-year migration probabilities, using the formula $r = -\log(1-p)$.

Source

Dyrting, S. (2024, October 23). Data from: Estimating Complete Migration Probabilities from Grouped Data. Retrieved from osf.io/vmrfk on 1 September 2025. Code to create `HIMD_R`, `HIMD_P1` and `HIMD_P5` is in folder '`data-raw/ssvd_himd`' in the source code for the **bage** package.

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- `SVD()` A prior based on a scaled SVD

`HMD` *Scaled SVD Components from Human Mortality Database*

Description

An object of class `"bage_ssvd"` holding scaled SVD components derived from data from the Human Mortality Database. `HMD` holds 5 components.

Usage

```
HMD
```

Format

Object of class `"bage_ssvd"`.

Versions:

- `"v2025"` (default) Data published on 2025-09-25, all years
- `"v2025-50"` Data published on 2025-09-25, 1950 and later
- `"v2024"` Data published on 2024-02-26, all years

Source

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

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- `SVD()` A prior based on a scaled SVD

isl_deaths *Deaths in Iceland*

Description

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

Usage

```
isl_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.

See Also

- datasets Overview of datasets in **bage**

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 = nzl_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
- Mathematical Details vignette

Examples

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

`kor_births` *Births in South Korea*

Description

Births and mid-year population by age of mother, region, and calendar year, 2011-2023, plus regional data on GDP per capita and population density.

Usage

```
kor_births
```

Format

A tibble with 1,872 rows and the following columns:

- `age` Five-year age groups from `"10-14" to "50-54"`‘
- `region` Administrative region
- `time` Calendar year, 2011-2023
- `births` Counts of births
- `popn` Mid-year population
- `gdp_pc_2023` Regional GDP per capita in 2023
- `dens_2020` Regional population density (people per km-squared) in 2020

Source

Tables "Live Births by Age Group of Mother, Sex and Birth Order for Provinces", and "Resident Population in Five-Year Age Groups", on the Korean Statistical Information Service website. Data downloaded on 24 September 2024. Data on GDP per capita and population density from Wikipedia https://w.wiki/DMFA, data downloaded on 8 March 2025, and https://w.wiki/DMF9, data downloaded on 8 March 2025.

See Also

- datasets Overview of datasets in **bage**

LFP *Scaled SVD Components from OECD Labor Force Participation Data*

Description

An object of class "bage_ssvd" holding scaled SVD components derived from labor force participation data assembled by the OECD. LFP holds 5 components.

Usage

```
LFP
```

Format

Object of class "bage_ssvd".

Versions:

- "v2025" Data downloaded on 2025-10-17

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.

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- SVD() A prior based on a scaled SVD

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, mean_slope = 0, sd_slope = 1, along = NULL, con = c("none", "by"))
```

Arguments

s	Scale for the prior for the errors. Default is 1. Can be 0.
mean_slope	Mean in prior for slope of line. Default is 0.
sd_slope	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.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

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 give smoother estimates. s can be zero, in which case errors are zero, and all values lie exactly on straight lines. This is clearly a simplification, but it allows the prior to be used with very large interactions.

Argument sd_slope 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,

$$\beta_j = (j - (J+1)/2)\eta + \epsilon_j$$
$$\eta \sim \mathrm{N}(\texttt{mean\_slope}, \texttt{sd\_slope}^2)$$
$$\epsilon_j \sim \mathrm{N}(0, \tau^2),$$

and when it is used with an interaction,

$$\beta_{u,v} = (v - (V+1)/2)\eta_u + \epsilon_{u,v}$$
$$\eta_u \sim \mathrm{N}(\texttt{mean\_slope}, \texttt{sd\_slope}^2)$$
$$\epsilon_{u,v} \sim \mathrm{N}(0, \tau^2),$$

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

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

When $\mathtt{s} = 0$, the model reduces to

$$\beta_j = (j - (J+1)/2)\eta$$
$$\eta \sim \mathrm{N}(\mathtt{mean\_slope}, \mathtt{sd\_slope}^2)$$

or

$$\beta_{u,v} = (v = (V+1)/2)\eta_u$$
$$\eta_u \sim \mathrm{N}(\mathtt{mean\_slope}, \mathtt{sd\_slope}^2)$$

.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- Mathematical Details vignette

Examples

```
Lin()
Lin(s = 0.5, sd_slope = 2)
Lin(s = 0)
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,
  shape1 = 5,
  shape2 = 5,
  mean_slope = 0,
  sd_slope = 1,
  along = NULL,
  con = c("none", "by")
)
```

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.
shape1, shape2	Parameters for beta-distribution prior for coefficients in the AR process. Defaults are 5 and 5.
mean_slope	Mean in prior for slope of line. Default is 0.
sd_slope	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.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

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_slope controls 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,

$$\beta_j = \alpha_j + \epsilon_j$$

$$\alpha_j = (j - (J+1)/2)\eta$$

$$\epsilon_j = \phi_1 \epsilon_{j-1} + \cdots + \phi_{\texttt{n\_coef}} \epsilon_{j-\texttt{n\_coef}} + \varepsilon_j$$

$$\varepsilon_j \sim \text{N}(0, \omega^2),$$

and when it is used with an interaction,

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

$$\alpha_{u,v} = (v - (V+1)/2)\eta_u$$

$$\epsilon_{u,v} = \phi_1 \epsilon_{u,v-1} + \cdots + \phi_{\texttt{n\_coef}} \epsilon_{u,v-\texttt{n\_coef}} + \varepsilon_{u,v},$$

$$\varepsilon_{u,v} \sim \text{N}(0, \omega^2).$$

where

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

The slopes have priors

$$\eta \sim \text{N}(\texttt{mean\_slope}, \texttt{sd\_slope}^2)$$

and

$$\eta_u \sim \text{N}(\texttt{mean\_slope}, \texttt{sd\_slope}^2).$$

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

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

The correlation coefficients $\phi_1, \cdots, \phi_{\texttt{n\_coef}}$ each have prior

$$0.5\phi_k - 0.5 \sim \text{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- Mathematical Details vignette

Examples

```
Lin_AR()
Lin_AR(n_coef = 3, s = 0.5, sd_slope = 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(
  s = 1,
  shape1 = 5,
  shape2 = 5,
  min = 0.8,
  max = 0.98,
  mean_slope = 0,
  sd_slope = 1,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

`s`	Scale for the innovations in the AR process. Default is `1`.
`shape1`, `shape2`	Parameters for beta-distribution prior for coefficients in the AR process. Defaults are `5` and `5`.
`min`, `max`	Minimum and maximum values for autocorrelation coefficient in the AR process. Defaults are `0.8` and `0.98`.
`mean_slope`	Mean in prior for slope of line. Default is 0.
`sd_slope`	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.
`con`	Constraints on parameters. Current choices are `"none"` and `"by"`. Default is `"none"`. See below for details.

Details

If `Lin_AR1()` 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_slope` controls 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,

$$\begin{aligned}\beta_j &= \alpha_j + \epsilon_j \\ \alpha_j &= (j - (J+1)/2)\eta \\ \epsilon_j &= \phi\epsilon_{j-1} + \varepsilon_j \\ \varepsilon &\sim \mathrm{N}(0, \omega^2),\end{aligned}$$

and when it is used with an interaction,

$$\begin{aligned}\beta_{u,v} &= \alpha_{u,v} + \epsilon_{u,v} \\ \alpha_{u,v} &= (v - (V+1)/2)\eta_u \\ \epsilon_{u,v} &= \phi\epsilon_{u,v-1} + \varepsilon_{u,v} \\ \varepsilon_{u,v} &\sim \mathrm{N}(0, \omega^2),\end{aligned}$$

where

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

The slopes have priors

$$\eta \sim \mathrm{N}(\texttt{mean\_slope}, \texttt{sd\_slope}^2)$$

and

$$\eta_u \sim \mathrm{N}(\texttt{mean\_slope}, \texttt{sd\_slope}^2).$$

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

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

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

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

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

where

$$\phi' \sim \mathrm{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- Mathematical Details vignette

Examples

```
Lin_AR1()
Lin_AR1(min = 0, s = 0.5, sd_slope = 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 (with or without quote marks.)

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

- subscript i identifies some combination of the the classifying variables, such as age, sex, and time;
- y_i is a count, such of number of births, 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;
- $\beta^{(m)}$, $m = 1, \ldots, M$, 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.

ξ has an exponential prior with mean 1. Non-default values for the mean can be specified with `set_disp()`.

The model for μ_i can also include covariates, as described in `set_covariates()`.

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
- `augment()` Extract values for probabilities, together with original data
- `components()` Extract values for hyper-parameters
- `forecast()` Forecast parameters and outcomes
- `report_sim()` Check model using simulation study
- `replicate_data()` Check model using replicate data
- Mathematical Details Detailed descriptions of models

Examples

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

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 = NULL)
```

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.)

Value

An object of class bage_mod_norm.

Scaling of outcome and weights

Internally, mod_norm() scales the outcome variable to have mean 0 and standard deviation 1, and scales the weights to have mean 1. This scaling allows mod_norm() to use the same menu of priors as mod_pois() and mod_binom().

augment() always returns values on the original scale, rather than the transformed scale.

components() by default returns values on the transformed scale. But if original_scale is TRUE, it returns some types of values on the original scale. See components() for details.

Specifying weights

There are three options for creating an unweighted model:

- do not supply a value for the weights variable;
- set weights equal to NULL; or
- **[Deprecated]** set weights equal to 1, though this option is deprecated, and will eventually be removed.

To create a weighted model, supply the name of the weighting variable in data, quoted or unquoted.

Mathematical details

The likelihood is

$$y_i \sim \mathrm{N}(\gamma_i, w_i^{-1}\sigma^2)$$

where

- subscript i identifies some combination of the classifying variables, such as age, sex, and time,
- y_i is the value of the outcome variable,
- w_i is a weight.

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

Internally, **bage** works with standardized versions of γ_i and σ^2:

$$\mu_i = (\gamma_i - \bar{y})/s$$
$$\xi^2 = \sigma^2/(\bar{w}s^2)$$

where

$$\bar{y} = \sum_{i=1}^{n} y_i/n$$

$$s = \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2/(n-1)}$$

$$\bar{w} = \sum_{i=1}^{n} w_i/n$$

Mean parameter μ_i is modelled as the sum of terms formed from classifying variables and covariates,

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

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.

ξ has an exponential prior with mean 1. Non-default values for the mean can be specified with `set_disp()`.

The model for μ_i can also include covariates, as described in `set_covariates()`.

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
- augment() Extract values for means, together with original data
- components() Extract values for hyper-parameters
- forecast() Forecast parameters and outcomes
- report_sim() Check model using a simulation study
- replicate_data() Check model using replicate data data for a model
- Mathematical Details Detailed description of models

Examples

```
## model without weights
mod <- mod_norm(value ~ diag:age + year,
                data = nld_expenditure)

## model with weights
nld_expenditure$wt <- sqrt(nld_expenditure$value)
mod <- mod_norm(value ~ diag:age + year,
                data = nld_expenditure,
                weights = wt)
```

---

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 NULL. 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`.

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`;
- `NULL`, in which case a pure "counts" model with no exposure, is produced; or
- **[Deprecated]** the number `1`, in which case a pure "counts" model is also produced (though this option is deprecated, and will eventually be removed).

Mathematical details

The likelihood is

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

where

- subscript i identifies some combination of the 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 (i.e. variation), with lower values implying less dispersion.

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;
- $\beta^{(m)}$, $m = 1, \ldots, M$, 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.

ξ has an exponential prior with mean 1. Non-default values for the mean can be specified with `set_disp()`.

The model for μ_i can also include covariates, as described in `set_covariates()`.

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
- `augment()` Extract values for rates, together with original data
- `components()` Extract values for hyper-parameters
- `forecast()` Forecast parameters and outcomes
- `report_sim()` Check model using a simulation study
- `replicate_data()` Check model using replicate data
- Mathematical Details Detailed description of models

Examples

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

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

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

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

where β is the main effect or interaction.

Parameter τ has a half-normal prior

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

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
- Mathematical Details vignette

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

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

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
- Mathematical Details vignette

Examples

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

nld_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

```
nld_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
- `value` Expenditures, in Euros

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).

See Also

- datasets Overview of datasets in **bage**

nzl_divorces *Divorces in New Zealand*

Description

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

Usage

```
nzl_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.

See Also

- datasets Overview of datasets in **bage**

`nzl_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

```
nzl_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.

See Also

- datasets Overview of datasets in **bage**

---

nzl_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

```
nzl_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.

See Also

- datasets Overview of datasets in **bage**

n_draw.bage_mod *Get the Number of Draws for a Model Object*

Description

Get the value of n_draw for a model object. n_draw controls the number of posterior draws that are generated by functions such as augment() and components().

Usage

```
## S3 method for class 'bage_mod'
n_draw(x)
```

Arguments

x	An object of class "bage_mod", created using mod_pois(), mod_binom(), or mod_norm().

Value

An integer

See Also

- set_n_draw() Modify the value of n_draw
- mod_pois(),mod_binom(),mod_norm() Create a model object

Examples

```
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = nzl_injuries,
                exposure = popn)
n_draw(mod)
mod <- mod |>
  set_n_draw(n_draw = 5000)
n_draw(mod)
```

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.
- A table giving the number of parameters, and (fitted models only) the standard deviation across those parameters, a measure of the term's importance. See priors() and tidy().
- Values for other model settings. See set_disp(), set_var_age(), set_var_sexgender(), set_var_time(), set_n_draw()
- Details on computations (fitted models only). See computations().

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() Specify a Poisson model
- mod_binom() Specify a binomial model
- mod_norm() Specify a normal model
- fit.bage_mod() and is_fitted() Model fitting
- augment() Extract values for rates, probabilities, or means, together with original data
- components() Extract values for hyper-parameters
- dispersion() Extract values for dispersion
- 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 = nzl_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 mod_pois(), mod_binom(), and mod_norm() include terms such as age effects and region-time interactions. Each of these terms requires a prior distribution. Current options for these priors are summarised in the table below.

Details

Prior	Description	Uses	Forecast	Along/By
N()	Elements drawn from normal distribution	Term with no natural order	Yes	No
NFix()	N() with standard deviation fixed	Term with few elements	Yes	No
Known()	Values treated as known	Simulations, prior knowledge	No	No
RW()	Random walk	Smoothing	Yes	Yes
RW2()	Second-order random walk	Like RW(), but with trends	Yes	Yes
DRW()	Damped random walk	Smoothing, forecasting	Yes	Yes
DRW2()	Damped second-order random walk	Like DRW(), but with trends	Yes	Yes
RW2_AR()	RW2() with AR errors	Terms involving time, forecasting	Yes	Yes
RW2_AR1()	RW2() with AR1 errors	Terms involving time, forecasting	Yes	Yes
RW2_Infant()	RW2() with infant indicator	Mortality age profiles	No	Yes
RW_Seas()	RW(), with seasonal effect	Terms involving time	Yes	Yes
RW2_Seas()	RW2(), with seasonal effect	Term involving time	Yes	Yes
AR()	Auto-regressive prior of order k	Mean reversion, forecasting	Yes	Yes
AR1()	Special case of AR()	Mean reversion, forecasting	Yes	Yes
Lin()	Linear trend, with independent errors	Parsimonious model for time	Yes	Yes
Lin_AR()	Linear trend, with AR errors	Term involving time, forecasting	Yes	Yes

Lin_AR1()	Linear trend, with AR1 errors	Terms involving time, forecasting	Yes	Yes
Sp()	P-Spline (penalised spline)	Smoothing, eg over age	No	Yes
SVD()	Age-sex profile based on SVD	Age or age-sex	No	No
SVD_AR()	SVD(), but coefficients follow AR()	Age or age-sex and time	Yes	Yes
SVD_AR1()	SVD(), but coefficients follow AR1()	Age or age-sex and time	Yes	Yes
SVD_Lin()	SVD(), but coefficients follow Lin()	Age or age-sex and time	Yes	Yes
SVD_RW()	SVD(), but coefficients follow RW()	Age or age-sex and time	Yes	Yes
SVD_RW2()	SVD(), but coefficients follow RW2()	Age or age-sex and time	Yes	Yes
SVD_DRW()	SVD(), but coefficients follow DRW()	Age or age-sex and time	Yes	Yes
SVD_DRW2()	SVD(), but coefficients follow DRW2()	Age or age-sex and time	Yes	Yes

'Along' and 'by' dimensions

Priors for interaction terms often consist of a time-series-style model along one dimension, with a separate series for each combination of the remaining dimensions. For instance, a prior for an age-sex-time interaction might consist of a separate random walk along time for each combination of age-group and sex. In **bage** the dimension with the time-series-type model is referred to as the 'along' dimension, and the remaining dimensions are referred to as the 'by' dimensions.

Default prior

If no prior is specified for a term, then **bage** assigns the term a default prior using the following algorithm:

- if the term has one or two elements, use NFix();
- otherwise, if the term involves time, use RW(), with time as the 'along' dimension;
- otherwise, if the term involves age, use RW(), with age as the 'along' dimension;
- otherwise, use N().

Forecasting

A model can only be used for forecasting if

- the model includes a time dimension, and
- the prior for the time dimension supports forecasting.

If necessary, the time dimension can be identified using set_var_time(). The table above lists the priors that support forecasting.

`prt_deaths` *Deaths in Portugal*

Description

Deaths and exposure in Portugal, by age, sex, and year.

Usage

```
prt_deaths
```

Format

A tibble with 3,168 rows and the following columns:

- `age`: Age groups `"0"`, `"1-4"`, "5-9", ..., `"95-99"`, `"100+"`
- `sex`: `"Female"` or `"Male"`
- `time`: Calendar year
- `deaths`: Count of deaths
- `exposure`: Person-years lived by population

Details

The data are from the Human Mortality Database. Deaths are rounded to the nearest integer. More recent versions, and a comprehensive description of the data, are available at the HMD website.

Source

Human Mortality Database. University of California, Berkeley (USA), and Max Planck Institute for Demographic Research (Germany). Available at `https://www.mortality.org`. (data downloaded on 17 July 2018).

See Also

- datasets Overview of datasets in **bage**

`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 ‘"expected" option is only possible in Poisson and binomial models, and only when dispersion is non-zero.

The default for `condition_on` is `"expected"`, in cases where it is feasible. 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.

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()` Specify a Poisson model
- `mod_binom()` Specify a binomial model
- `mod_norm()` Specify a normal model
- `fit()` Fit model.
- `augment()` Extract values for rates, probabilities, or means, together with original data
- `components()` Extract values for hyper-parameters
- `dispersion()` Extract values for dispersion
- `forecast()` Forecast, based on a model
- `report_sim()` Simulation study of model.
- Mathematical Details vignette

Examples

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

A named list with a tibble called "components" and a tibble called "augment".

Warning

The interface for report_sim() is still under development and may change in future.

See Also

- mod_pois() Specify binomial model
- 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
- 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 as a model for a main effect, or use multiple random walks as a model for an interaction. Typically used with terms that involve age or time.

Usage

```
RW(s = 1, sd = 1, along = NULL, con = c("none", "by"))
```

Arguments

s	Scale for the prior for the innovations. Default is 1.
sd	Standard deviation of initial value. Default is 1. Can be 0.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If RW() is used with an interaction, a separate random walk is constructed within each combination of the 'by' variables.

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

Argument sd controls variance in initial values. Setting sd to 0 fixes initial values at 0.

Value

An object of class "bage_prior_rwrandom" or "bage_prior_rwzero".

Mathematical details

When RW() is used with a main effect,

$$\beta_1 \sim \mathrm{N}(0, \mathtt{sd}^2)$$

$$\beta_j \sim \mathrm{N}(\beta_{j-1}, \tau^2), \quad j > 1$$

and when it is used with an interaction,

$$\beta_{u,1} \sim \mathrm{N}(0, \mathtt{sd}^2)$$

$$\beta_{u,v} \sim \mathrm{N}(\beta_{u,v-1}, \tau^2), \quad v > 1$$

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

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

where s is provided by the user.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

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

Examples

```
RW()
RW(s = 0.5)
RW(sd = 0)
RW(along = "cohort")
```

RW2 *Second-Order Random Walk Prior*

Description

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

Usage

```
RW2(s = 1, sd = 1, sd_slope = 1, along = NULL, con = c("none", "by"))
```

Arguments

s	Scale for the prior for the innovations. Default is 1.
sd	Standard deviation of initial value. Default is 1. Can be 0.
sd_slope	Standard deviation of initial slope. Default is 1.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If RW2() is used with an interaction, a separate random walk is constructed within each combination of the 'by' variables.

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

Argument sd controls variance in initial values. Setting sd to 0 fixes initial values at 0.

Argument sd_slope controls variance in the initial slope.

Value

An object of class "bage_prior_rw2random" or "bage_prior_rw2zero".

Mathematical details

When RW2() is used with a main effect,

$$\beta_1 \sim \mathrm{N}(0, \mathtt{sd}^2)$$
$$\beta_2 \sim \mathrm{N}(\beta_1, \mathtt{sd\_slope}^2)$$
$$\beta_j \sim \mathrm{N}(2\beta_{j-1} - \beta_{j-2}, \tau^2), \quad j = 2, \cdots, J$$

and when it is used with an interaction,

$$\beta_{u,1} \sim \mathrm{N}(0, \mathtt{sd}^2)$$
$$\beta_{u,2} \sim \mathrm{N}(\beta_{u,1}, \mathtt{sd\_slope}^2)$$
$$\beta_{u,v} \sim \mathrm{N}(2\beta_{u,v-1} - \beta_{u,v-2}, \tau^2), \quad v = 3, \cdots, V$$

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

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

.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

- `RW()` Random walk
- `RW2_Seas()` Second order random walk with seasonal effect
- `AR()` Autoregressive with order k
- `AR1()` Autoregressive with order 1
- `Sp()` Smoothing via splines
- `SVD()` Smoothing over age via singular value decomposition
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

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

RW2_AR *Second-Order Random Walk Prior with Autoregressive Errors*

Description

Use one or more second-order random walks, combined with an autoregressive error term, to model a main effect or an interaction. Typically used with time.

Usage

```
RW2_AR(
  s_rw = 1,
  sd = 1,
  sd_slope = 1,
  n_coef = 2,
  s_ar = 1,
  shape1 = 5,
  shape2 = 5,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

s_rw	Scale for the innovations in the RW2 process. Default is 1.
sd	Standard deviation for initial term in RW2 process. Default is 1. Can be 0.
sd_slope	Standard deviation in the prior for the initial slope of the RW2 process. Larger values imply steeper slopes. Default is 1.
n_coef	Number of lagged terms in the model, ie the order of the model. Default is 2.
s_ar	Scale for the innovations in the AR process. Default is 1.
shape1, shape2	Parameters for beta-distribution prior for coefficients. Defaults are 5 and 5.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If RW2_AR() is used with an interaction, separate random walks are constructed along the 'along' variable, within each combination of the 'by' variables.

Parameters controlling the RW2 process:

- s_rw
- sd
- sd_slope

Parameters controlling the AR process:

- n_coef
- s_ar
- shape1
- shape2

Value

An object of class "bage_prior_rw2randomar" or "bage_prior_rw2zeroar".

Mathematical details

When `RW2_AR()` is used with a main effect,

$$\beta_j = \alpha_j + \epsilon_j$$

$$\alpha_1 \sim \mathrm{N}(0, \texttt{sd}^2)$$

$$\alpha_2 \sim \mathrm{N}(\alpha_1, \texttt{sd\_slope}^2)$$

$$\alpha_j \sim \mathrm{N}(2\alpha_{j-1} - \alpha_{j-2}, \tau^2), \quad j = 3, \cdots, J$$

$$\epsilon_j = \phi_1 \epsilon_{j-1} + \cdots + \phi_{\texttt{n\_coef}} \epsilon_{j-\texttt{n\_coef}} + \varepsilon_j$$

$$\varepsilon_j \sim \mathrm{N}(0, \omega^2),$$

and when it is used with an interaction,

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

$$\alpha_{u,1} \sim \mathrm{N}(0, \texttt{sd}^2)$$

$$\alpha_{u,2} \sim \mathrm{N}(\alpha_{u,1}, \texttt{sd\_slope}^2)$$

$$\alpha_{u,v} \sim \mathrm{N}(2\alpha_{u,v-1} - \alpha_{u,v-2}, \tau^2), \quad v = 3, \cdots, V$$

$$\epsilon_{u,v} = \phi_1 \epsilon_{u,v-1} + \cdots + \phi_{\texttt{n\_coef}} \epsilon_{u,v-\texttt{n\_coef}} + \varepsilon_{u,v}$$

$$\varepsilon_{u,v} \sim \mathrm{N}(0, \omega^2),$$

where

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

The τ parameter in the random walk has prior

$$\tau \sim \mathrm{N}^+(0, \texttt{s\_rw}^2)$$

Internally, `RW2_AR()` derives a value for ω that gives ϵ_j or $\epsilon_{u,v}$ a marginal variance of ν^2. Parameter ν has a half-normal prior

$$\nu \sim \mathrm{N}^+(0, \texttt{s\_ar}^2).$$

The correlation coefficients $\phi_1, \cdots, \phi_{\texttt{n\_coef}}$ each have prior

$$0.5\phi_k - 0.5 \sim \mathrm{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

- `RW2_AR1()` Special case of `RW2_AR()`
- `RW2()` Second-order random walk
- `AR()` AR process
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

```
RW2_AR()
RW2_AR(sd_slope = 2, n_coef = 3, s_ar = 0.5)
```

RW2_AR1 *Second-Order Random Walk Prior with First Order Autoregressive Errors*

Description

Use one or more second-order random walks, combined with an AR1 error term, to model a main effect or an interaction. Typically used with time.

Usage

```
RW2_AR1(
  s_rw = 1,
  sd = 1,
  sd_slope = 1,
  s_ar = 1,
  shape1 = 5,
  shape2 = 5,
  min = 0.8,
```

```
  max = 0.98,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

s_rw	Scale for the innovations in the RW2 process. Default is 1.
sd	Standard deviation for initial term in RW2 process. Default is 1. Can be 0.
sd_slope	Standard deviation in the prior for the initial slope of RW2 process. Larger values imply steeper slopes. Default is 1.
s_ar	Scale for the innovations in the AR1 process. Default is 1.
shape1, shape2	Parameters for beta-distribution prior for coefficients. Defaults are 5 and 5.
min, max	Minimum and maximum values for autocorrelation coefficient in AR1 process. Defaults are 0.8 and 0.98.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If RW2_AR1() is used with an interaction, separate random walks are constructed along the 'along' variable, within each combination of the 'by' variables.

Parameters controlling the RW2 process:

- s_rw
- sd
- sd_slope

Parameters controlling the AR1 process:

- s_ar
- shape1
- shape2
- min
- max

Value

An object of class "bage_prior_rw2randomar" or "bage_prior_rw2zeroar".

Mathematical details

When `RW2_AR1()` is used with a main effect,

$$\begin{gathered}
\beta_j = \alpha_j + \epsilon_j \\
\alpha_1 \sim \mathrm{N}(0, \texttt{sd}^2) \\
\alpha_2 \sim \mathrm{N}(\alpha_1, \texttt{sd_slope}^2) \\
\alpha_j \sim \mathrm{N}(2\alpha_{j-1} - \alpha_{j-2}, \tau^2), \quad j = 3, \cdots, J \\
\epsilon_j = \phi\epsilon_{j-1} + \varepsilon_j \\
\varepsilon_j \sim \mathrm{N}(0, \omega^2),
\end{gathered}$$

and when it is used with an interaction,

$$\begin{gathered}
\beta_{u,v} = \alpha_{u,v} + \epsilon_{u,v} \\
\alpha_{u,1} \sim \mathrm{N}(0, \texttt{sd}^2) \\
\alpha_{u,2} \sim \mathrm{N}(\alpha_{u,1}, \texttt{sd_slope}^2) \\
\alpha_{u,v} \sim \mathrm{N}(2\alpha_{u,v-1} - \alpha_{u,v-2}, \tau^2), \quad v = 3, \cdots, V \\
\epsilon_{u,v} = \phi\epsilon_{u,v-1} + \varepsilon_{u,v} \\
\varepsilon_{u,v} \sim \mathrm{N}(0, \omega^2),
\end{gathered}$$

where

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

The τ parameter in the random walk has prior

$$\tau \sim \mathrm{N}^+(0, \texttt{s\_rw}^2)$$

Internally, `RW2_AR()` derives a value for ω that gives ϵ_j or $\epsilon_{u,v}$ a marginal variance of ν^2. Parameter ν has a half-normal prior

$$\nu \sim \mathrm{N}^+(0, \texttt{s\_ar}^2).$$

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

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

where

$$\phi' \sim \mathrm{Beta}(\texttt{shape1}, \texttt{shape2}).$$

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

- `RW2_AR()` Generalization of `RW2_AR1()`
- `Lin_AR1()` Sepcial case of `RW2_AR1()`
- `RW2()` Second-order random walk
- `AR1()` AR1 process
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

```
RW2_AR1()
RW2_AR1(sd_slope = 2, s_ar = 0.5)
```

RW2_Infant *Second-Order Random Walk Prior with 'Infant' Indicator*

Description

Use a second-order random walk to model variation over age, with an indicator variable for the first age group. Designed for use in models of mortality rates.

Usage

```
RW2_Infant(s = 1, sd_slope = 1, con = c("none", "by"))
```

Arguments

`s`	Scale for the prior for the innovations. Default is `1`.
`sd_slope`	Standard deviation for initial slope of random walk. Default is `1`.
`con`	Constraints on parameters. Current choices are `"none"` and `"by"`. Default is `"none"`. See below for details.

Details

A second-order random walk prior RW2() works well for smoothing mortality rates over age, except at age 0, where there is a sudden jump in rates, reflecting the special risks of infancy. The RW2_Infant() extends the RW2() prior by adding an indicator variable for the first age group.

If RW2_Infant() is used in an interaction, the 'along' dimension is always age, implying that there is a separate random walk along age 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 sd controls the sl size of innovations in the random walk. Smaller values for s tend to give smoother series.

Value

Object of class "bage_prior_rw2infant".

Mathematical details

When RW2_Infant() is used with a main effect,

$$\beta_1 \sim \mathrm{N}(0, 1)$$
$$\beta_2 \sim \mathrm{N}(0, \texttt{sd\_slope}^2)$$
$$\beta_3 \sim \mathrm{N}(2\beta_2, \tau^2)$$
$$\beta_j \sim \mathrm{N}(2\beta_{j-1} - \beta_{j-2}, \tau^2), \quad j = 3, \cdots, J$$

and when it is used with an interaction,

$$\beta_{u,1} \sim \mathrm{N}(0, 1)$$
$$\beta_{u,2} \sim \mathrm{N}(0, \texttt{sd\_slope}^2)$$
$$\beta_{u,3} \sim \mathrm{N}(2\beta_{u,2}, \tau^2)$$
$$\beta_{u,v} \sim \mathrm{N}(2\beta_{u,v-1} - \beta_{u,v-2}, \tau^2), \quad v = 3, \cdots, V$$

where

- $\boldsymbol{\beta}$ is a 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

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

.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

See Also

- `RW2()` Second-order random walk, without infant indicator
- `Sp()` Smoothing via splines
- `SVD()` Smoothing over age via singular value decomposition
- priors Overview of priors implemented in **bage**
- `set_prior()` Specify prior for intercept, main effect, or interaction
- Mathematical Details vignette

Examples

```
RW2_Infant()
RW2_Infant(s = 0.1)
```

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

Description

Use a second-oder random walk with seasonal effects as a model for a main effect, or use multiple second-order random walks, each with their own seasonal effects, as a model for an interaction. Typically used with temrs that involve time.

Usage

```
RW2_Seas(
  n_seas,
  s_seas,
  s = 1,
  sd = 1,
  sd_slope = 1,
  sd_seas = 1,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

n_seas	Number of seasons
s_seas	Scale for innovations in seasonal effects. Can be 0. When greater than 0, seasonal effects vary from year to year.
s	Scale for prior for innovations in random walk. Default is 1.
sd	Standard deviation of initial value. Default is 1. Can be 0.
sd_slope	Standard deviation for initial slope of random walk. Default is 1.
sd_seas	Standard deviation for initial values of seasonal effects. Default is 1.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If RW2_Seas() is used with an interaction, a separate series is constructed within each combination of the 'by' variables.

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

Argument n_seas controls the number of seasons. When using quarterly data, for instance, n_seas should be 4.

Setting s_seas to 0 produces seasonal effects that are the same each year. Setting s_seas to a value greater than 0 produces seasonal effects that evolve over time.

Value

Object of class "bage_prior_rw2randomseasvary", "bage_prior_rw2randomseasfix", "bage_prior_rw2zeroseasvary or "bage_prior_rw2zeroseasfix".

Mathematical details

When RW2_Seas() is used with a main effect,

$$\beta_j = \alpha_j + \lambda_j, \quad j = 1, \cdots, J$$

$$\alpha_1 \sim \mathrm{N}(0, \mathtt{sd}^2)$$

$$\alpha_2 \sim \mathrm{N}(0, \mathtt{sd\_slope}^2)$$

$$\alpha_j \sim \mathrm{N}(2\alpha_{j-1} - \alpha_{j-2}, \tau^2), \quad j = 3, \cdots, J$$

$$\lambda_j \sim \mathrm{N}(0, \mathtt{sd\_seas}^2), \quad j = 1, \cdots, \mathtt{n\_seas} - 1$$

$$\lambda_j = -\sum_{s=1}^{\mathtt{n\_seas}-1} \lambda_{j-s}, \quad j = \mathtt{n\_seas}, 2\mathtt{n\_seas}, \cdots$$

$$\lambda_j \sim \mathrm{N}(\lambda_{j-\mathtt{n\_seas}}, \omega^2), \quad \text{otherwise},$$

and when it is used with an interaction,

$$\beta_{u,v} = \alpha_{u,v} + \lambda_{u,v}, \quad v = 1, \cdots, V$$

$$\alpha_{u,1} \sim \mathrm{N}(0, \texttt{sd}^2)$$

$$\alpha_{u,2} \sim \mathrm{N}(0, \texttt{sd\_slope}^2)$$

$$\alpha_{u,v} \sim \mathrm{N}(2\alpha_{u,v-1} - \alpha_{u,v-2}, \tau^2), \quad v = 3, \cdots, V$$

$$\lambda_{u,v} \sim \mathrm{N}(0, \texttt{sd\_seas}^2), \quad v = 1, \cdots, \texttt{n\_seas} - 1$$

$$\lambda_{u,v} = -\sum_{s=1}^{\texttt{n\_seas}-1} \lambda_{u,v-s}, \quad v = \texttt{n\_seas}, 2\texttt{n\_seas}, \cdots$$

$$\lambda_{u,v} \sim \mathrm{N}(\lambda_{u,v-\texttt{n\_seas}}, \omega^2), \quad \text{otherwise,}$$

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; and
- u denotes position within the 'by' variable(s) of the interaction.

Parameter ω has a half-normal prior

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

. 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

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

.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- Mathematical Details vignette

Examples

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

## seasonal effects evolve
RW2_Seas(n_seas = 4, s_seas = 1)

## first term in random walk fixed at 0
RW2_Seas(n_seas = 4, s_seas = 1, sd = 0)
```

RW_Seas *Random Walk Prior with Seasonal Effect*

Description

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

Usage

```
RW_Seas(
  n_seas,
  s_seas,
  s = 1,
  sd = 1,
  sd_seas = 1,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

n_seas	Number of seasons
s_seas	Scale for innovations in seasonal effects. Can be 0. When greater than 0, seasonal effects vary from year to year.
s	Scale for prior for innovations in random walk. Default is 1.

sd	Standard deviation of initial value. Default is 1. Can be 0.
sd_seas	Standard deviation for initial values of seasonal effects. Default is 1.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

Details

If RW_Seas() is used with an interaction, a separate series is constructed within each combination of the 'by' variables.

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

Argument sd controls variance in initial values of the random walk. sd can be 0.

Argument n_seas controls the number of seasons. When using quarterly data, for instance, n_seas should be 4.

Setting s_seas to 0 produces seasonal effects that are the same each year. Setting s_seas to a value greater than 0 produces seasonal effects that evolve over time.

Value

Object of class "bage_prior_rwrandomseasvary", "bage_prior_rwrandomseasfix", "bage_prior_rwzeroseasvary", or "bage_prior_rwzeroseasfix".

Mathematical details

When RW_Seas() is used with a main effect,

$$\beta_j = \alpha_j + \lambda_j, \quad j = 1, \cdots, J$$
$$\alpha_1 \sim \mathrm{N}(0, \mathtt{sd}^2)$$
$$\alpha_j \sim \mathrm{N}(\alpha_{j-1}, \tau^2), \quad j = 2, \cdots, J$$
$$\lambda_j \sim \mathrm{N}(0, \mathtt{sd\_seas}^2), \quad j = 1, \cdots, \mathtt{n\_seas} - 1$$
$$\lambda_j = -\sum_{s=1}^{\mathtt{n\_seas}-1} \lambda_{j-s}, \quad j = \mathtt{n\_seas}, 2\mathtt{n\_seas}, \cdots$$
$$\lambda_j \sim \mathrm{N}(\lambda_{j-\mathtt{n\_seas}}, \omega^2), \quad \text{otherwise},$$

and when it is used with an interaction,

$$\beta_{u,v} = \alpha_{u,v} + \lambda_{u,v}, \quad v = 1, \cdots, V$$
$$\alpha_{u,1} \sim \mathrm{N}(0, \mathtt{sd}^2)$$
$$\alpha_{u,v} \sim \mathrm{N}(\alpha_{u,v-1}, \tau^2), \quad v = 2, \cdots, V$$
$$\lambda_{u,v} \sim \mathrm{N}(0, \mathtt{sd\_seas}^2), \quad v = 1, \cdots, \mathtt{n\_seas} - 1$$

$$\lambda_{u,v} = -\sum_{s=1}^{\texttt{n\_seas}-1} \lambda_{u,v-s}, \quad v = \texttt{n\_seas}, 2\texttt{n\_seas}, \cdots$$

$$\lambda_{u,v} \sim \mathrm{N}(\lambda_{u,v-\texttt{n\_seas}}, \omega^2), \quad \text{otherwise},$$

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; and
- u denotes position within the 'by' variable(s) of the interaction.

Parameter ω has a half-normal prior

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

If `s_seas` is set to 0, then ω is 0, and seasonal effects are time-invariant.

Parameter τ has a half-normal prior

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

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- Mathematical Details vignette

Examples

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

## seasonal effects evolve
RW_Seas(n_seas = 4, s_seas = 1)

## first term in random walk fixed at 0
RW_Seas(n_seas = 4, s_seas = 1, sd = 0)
```

---

set_confidential_rr3 *Specify RR3 Confidentialization*

Description

Specify a confidentialization procedure where the outcome variable is randomly rounded to a multiple of 3.

Usage

```
set_confidential_rr3(mod)
```

Arguments

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

Details

set_confidential_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. The procedure for randomly-rounding an integer 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_confidential_rr3() is applied to a fitted model, set_confidential_rr3() unfits the model, deleting existing estimates.

Value

A revised version of mod.

See Also

- confidential Overview of confidentialization procedures currently modeled in **bage**
- `mod_pois()`, `mod_binom()`, `mod_norm()` Specify a model for rates, probabilities, or means

Examples

```
## 'injuries' variable in 'nzl_injuries' dataset
## has been randomly rounded to base 3
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = nzl_injuries,
                exposure = popn) |>
  set_confidential_rr3() |>
  fit()
```

`set_covariates` *Specify Covariates*

Description

Add covariates to a model.

Usage

```
set_covariates(mod, formula)
```

Arguments

`mod`	An object of class `"bage_mod"`, created with `mod_pois()`, `mod_binom()`, or `mod_norm()`.
`formula`	A one-sided R formula, specifying the covariates.

Details

If `set_covariates()` is applied to a model that already has covariates, `set_covariates()` deletes the existing covariates.

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

Value

A modified version of `mod`

Covariate data

All variables contained in the `formula` argument to `set_covariates()` should be in the dataset supplied in the original call to `mod_pois()`, `mod_binom()`, or `mod_norm()`.

`set_covariates()` processes the covariate data before adding it to the model:

- All numeric variables are standardized, using `x <- scale(x)`.
- Categorical variables are converted to sets of indicator variables, using treatment contrasts. For instance, variable `x` with categories `"high"`, `"medium"`, and `"low"`, is converted into two indicator variables, one called `xmedium` and one called `xlow`.

Mathematical details

When a model includes covariates, the quantity

$$\boldsymbol{Z}\boldsymbol{\zeta}$$

is added to the linear predictor, where $\boldsymbol{Z}$ is a matrix of standardized covariates, and $\boldsymbol{\zeta}$ is a vector of coefficients. The elements of $\boldsymbol{\zeta}$ have prior

$$\zeta_p \sim \mathrm{N}(0, 1)$$

.

See Also

- `mod_pois()`, `mod_binom()`, `mod_norm()` Specify a model for rates, probabilities, or means

Examples

```
## create a COVID covariate
library(dplyr, warn.conflicts = FALSE)
births <- kor_births |>
  mutate(is_covid = time %in% 2020:2022)
mod <- mod_pois(births ~ age * region + time,
                data = births,
                exposure = popn) |>
  set_covariates(~ is_covid)
mod
```

`set_datamod_exposure` *Specify Exposure Data Model*

Description

Specify a data model for the exposure variable in a Poisson model. The data model assumes that, within each cell, observed exposure is drawn from an Inverse-Gamma distribution. In this model,

E[expected exposure | true exposure] = true exposure

and

sd[expected exposure | true exposure] = cv $\times$ true exposure

where cv is a coefficient of variation parameter.

Usage

```
set_datamod_exposure(mod, cv)
```

Arguments

mod	An object of class "bage_mod_pois", created with `mod_pois()`.
cv	Coefficient of variation for measurement errors in exposure. A single number, or a data frame with a variable called "cv" and one or more 'by' variables.

Details

In the exposure data model, cv, the coefficient of variation, does not depend on true exposure. This implies that errors do not fall, in relative terms, as population rises. Unlike sampling errors, measurement errors do not get averaged away in large populations.

The exposure data model assumes that the exposure variable is unbiased. If there is in fact evidence of biases, then this evidence should be used to create a de-biased version of the variable (eg one where estimated biases have been subtracted) to supply to `mod_pois()`.

`set_datamod_exposure()` can only be used with a Poisson model for rates in which the dispersion in the rates has been set to zero. The dispersion in the rates can be set explicitly to zero using `set_disp()`, though `set_datamod_exposure()` will also do so.

Value

A revised version of mod.

The cv argument

cv can be a single number, in which case the same value is used for all cells. cv can also be a data frame with a with a variable called "cv" and one or more columns with 'by' variables. For instance, a cv of

```
data.frame(sex = c("Female", "Male"),
           cv = c(0.01, 0.012))
```

implies that the coefficient of variation is 0.01 for females and 0.012 for males.

See below for an example where the coefficient of variation is based on aggregated age groups.

Mathematical details

The model for observed exposure is

$$w_i^{\text{obs}} \sim \text{InvGamma}(2 + d_{g[i]}^{-1}, (1 + d_{g[i]}^{-1}) w_i^{\text{true}})$$

where

- w_i^{obs} is observed exposure for cell i (the `exposure` argument to `mod_pois()`);
- w_i^{true} is true exposure for cell i; and
- $d_{g[i]}$ is the value for dispersion that is applied to cell i.

`cv` is $\sqrt{d_g}$.

See Also

- `mod_pois()` Specify a Poisson model
- `set_disp()` Specify dispersion of rates
- `augment()` Original data plus estimated values, including estimates of true value for exposure
- datamods Data models implemented in bage
- confidential Confidentialization procedures modeled in bage
- Mathematical Details vignette

Examples

```
## specify model
mod <- mod_pois(injuries ~ age * sex + year,
                data = nzl_injuries,
                exposure = popn) |>
  set_disp(mean = 0) |>
  set_datamod_exposure(cv = 0.025)

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

## examine results - note the new variable
## '.popn' with estimates of the true
## population
aug <- mod |>
  augment()

## allow different cv's for each sex
cv_sex <- data.frame(sex = c("Female", "Male"),
                     cv = c(0.03, 0.02))
```

```
mod <- mod |>
  set_datamod_exposure(cv = cv_sex)
mod

## our outcome variable is confidentialized,
## so we recognize that in the model too
mod <- mod |>
  set_confidential_rr3()
mod

## now a model where everyone aged 0-49
## receives one value for cv, and
## everyone aged 50+ receives another
library(poputils) ## for 'age_upper()'
library(dplyr, warn.conflicts = FALSE)
nzl_injuries_age <- nzl_injuries |>
  mutate(age_group = if_else(age_upper(age) < 50,
                             "0-49",
                             "50+"))
cv_age <- data.frame(age_group = c("0-49", "50+"),
                     cv = c(0.05, 0.01))
mod <- mod_pois(injuries ~ age * sex + year,
                data = nzl_injuries_age,
                exposure = popn) |>
  set_disp(mean = 0) |>
  set_datamod_exposure(cv = cv_age)
```

`set_datamod_miscount` *Specify Miscount Data Model*

Description

Specify a data model for the outcome in a Poisson model, where the outcome is subject to undercount and overcount.

Usage

```
set_datamod_miscount(mod, prob, rate)
```

Arguments

`mod`	An object of class `"bage_mod_pois"`, created with `mod_pois()`.
`prob`	The prior for the probability that a person or event in the target population will correctly enumerated. A data frame with a variable called `"mean"`, a variable called `"disp"`, and, optionally, one or more 'by' variables.
`rate`	The prior for the overcoverage rate. A data frame with a variable called `"mean"`, a variable called `"disp"`, and, optionally, one or more 'by' variables.

Details

The miscount data model is essentially a combination of the undercount and overcount data models. It assumes that reported outcome is the sum of two quantities:

1. *Units from target population, undercounted* People or events belonging to the target population, in which each unit's inclusion probability is less than 1.
2. *Overcount* People or events that do not belong to target population, or that are counted more than once.

If, for instance, a census enumerates 91 people from a true population of 100, but also mistakenly enumerates a further 6 people, then

- the true value for the outcome variable is 100
- the value for the undercounted target population is 91,
- the value for the overcount is 6, and
- the observed value for the outcome variable is 91 + 6 = 97.

Value

A revised version of `mod`.

The `prob` argument

The `prob` argument specifies a prior distribution for the probability that a person or event in the target population is included in the reported outcome. `prob` is a data frame with a variable called `"mean"`, a variable called `"disp"`, and, optionally, one or more 'by' variables. For instance, a `prob` of

```
data.frame(sex = c("Female", "Male"),
           mean = c(0.95, 0.92),
           disp = c(0.02, 0.015))
```

implies that the expected value for the inclusion probability is 0.95 for females and 0.92 for males, with slightly more uncertainty for females than for males.

The `rate` argument

The `rate` argument specifies a prior distribution for the overcoverage rate. `rate` is a data frame with a variable called `"mean"`, a variable called `"disp"`, and, optionally, one or more 'by' variables. For instance, a `rate` of

```
data.frame(mean = 0.03, disp = 0.1)
```

implies that the expected value for the overcoverage rate is 0.03, with a dispersion of 0.1. Since no 'by' variables are included, the same mean and dispersion values are applied to all cells.

Mathematical details

The model for the observed outcome is

$$y_i^{\text{obs}} = u_i + v_i$$

$$u_i \sim \text{Binomial}(y_i^{\text{true}}, \pi_{g[i]})$$

$$v_i \sim \text{Poisson}(\kappa_{h[i]}\gamma_i w_i)$$

$$\pi_g \sim \text{Beta}(m_g^{(\pi)}/d_g^{(\pi)}, (1 - m_g^{(\pi)})/d_g^{(\pi)})$$

$$\kappa_h \sim \text{Gamma}(1/d_h^{(\kappa)}, 1/(d_h^{(\kappa)} m_h^{(\kappa)}))$$

where

- y_i^{obs} is the observed outcome for cell i;
- y_i^{true} is the true outcome for cell i;
- γ_i is the rate for cell i;
- w_i is exposure for cell i;
- $\pi_{g[i]}$ is the probability that a member of the target population in cell i is correctly enumerated in that cell;
- $\kappa_{h[i]}$ is the overcoverage rate for cell i;
- $m_g^{(\pi)}$ is the expected value for π_g (specified via `prob`);
- $d_g^{(\pi)}$ is disperson for π_g (specified via `prob`);
- $m_h^{(\kappa)}$ is the expected value for κ_h (specified via `rate`); and
- $d_h^{(\kappa)}$ is disperson for κ_h (specified via `rate`).

See Also

- `mod_pois()` Specify a Poisson model
- `augment()` Original data plus estimated values, including estimates of true value for the outcome variable
- `components()` Estimated values for model parameters, including inclusion probabilities and overcount rates
- `set_datamod_undercount()` An undercount-only data model
- `set_datamod_overcount()` An overcount-only data model
- datamods All data models implemented in bage
- confidential Confidentialization procedures modeled in bage
- Mathematical Details vignette

Examples

```
## specify 'prob' and 'rate'
prob <- data.frame(sex = c("Female", "Male"),
                   mean = c(0.95, 0.97),
                   disp = c(0.05, 0.05))
rate <- data.frame(mean = 0.03, disp = 0.15)

## specify model
mod <- mod_pois(divorces ~ age * sex + time,
                data = nzl_divorces,
                exposure = population) |>
  set_datamod_miscount(prob = prob, rate = rate)
mod

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

## original data, plus imputed values for outcome
mod |>
  augment()

## parameter estimates
library(dplyr)
mod |>
  components() |>
  filter(term == "datamod")

## the data have in fact been confidentialized,
## so we account for that, in addition
## to accounting for undercoverage and
## overcoverage
mod <- mod |>
 set_confidential_rr3() |>
 fit()
mod
```

`set_datamod_noise` *Specify Noise Data Model*

Description

Specify a data model in which
`observed outcome = true outcome + error`,

where the error has a symmetric distribution with mean 0.

If the true outcome has a normal distribution, then the error has a normal distribution. If the true outcome has a Poisson distribution, then the error has a symmetric Skellam distribution.

Usage

```
set_datamod_noise(mod, sd)
```

Arguments

mod	An object of class "bage_mod", created with mod_norm() or mod_pois().
sd	Standard deviation of measurement errors. A single number, or a data frame with 'by' variables.

Details

The model assumes that the outcome variable is unbiased. If there is in fact evidence of biases, then this evidence should be used to create a de-biased version of the outcome variable in data, and this de-biased version should be used by mod_norm() or mod_pois().

If set_datamod_noise() is used with a Poisson model, then the dispersion term for the Poisson rates must be set to zero. This can be done using set_disp(), though set_datamod_noise() will also do so.

Value

A revised version of mod.

The Skellam distribution

The Skellam distribution is restricted to integers, but can take positive and negative values.

If

$$X_1 \sim \text{Poisson}(\mu_1)$$

$$X_2 \sim \text{Poisson}(\mu_2)$$

then

$$Y = X_1 - X_2$$

has a Skellam(μ_1, μ_2) distribution. If $\mu_1 = \mu_2$, then the distribution is symmetric.

The sd argument

sd can be a single number, in which case the same standard deviation is used for all cells. sd can also be a data frame with a with a variable called "sd" and one or more columns with 'by' variables. For instance, a sd of

```
data.frame(sex = c("Female", "Male"),
           sd = c(330, 240))
```

implies that measurement errors have standard deviation 330 for females and 240 for males.

Mathematical details

The model for the observed outcome is

$$y_i^{\text{obs}} = y_i^{\text{true}} + \epsilon_i$$

with

$$\epsilon_i \sim \text{N}(0, s^2_{g[i]})$$

if y_i^{true} has a normal distribution, and

$$\epsilon_i \sim \text{Skellam}(0.5s^2_{g[i]}, 0.5s^2_{g[i]})$$

if y_i^{true} has a Poisson distribution, where

- y_i^{obs} is the observed outcome for cell i;
- y_i^{true} is the true outcome for cell i;
- ϵ_i is the measurement error for cell i; and
- $s_{g[i]}$ is the standard deviation of the measurement error for cell i.

See Also

- `mod_norm()` Specify a normal model
- `mod_pois()` Specify a Poisson model
- `augment()` Original data plus estimated values, including estimates of true value for outcome
- datamods Data models implemented in bage
- Mathematical Details vignette

Examples

```
## Normal model -------------------------------

## prepare outcome variable
library(dplyr, warn.conflicts = FALSE)
spend <- nld_expenditure |>
  mutate(log_spend = log(value + 1))

## specify model
mod <- mod_norm(log_spend ~ age * diag + year,
                data = spend,
                weights = 1) |>
  set_datamod_noise(sd = 0.1)

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

```
## create new aggregated diagnositic
## group variable
library(dplyr, warn.conflicts = FALSE)
spend <- spend |>
  mutate(diag_ag = case_when(
    diag == "Neoplasms" ~ diag,
    diag == "Not allocated" ~ diag,
    TRUE ~ "Other"
  ))

## assume size of measurement errors
## varies across these aggregated groups
sd_diag <- data.frame(diag_ag = c("Neoplasms",
                                  "Not allocated",
                                  "Other"),
                      sd = c(0.05, 0.2, 0.1))

## fit model that uses diagnostic-specific
## standard deviations
mod <- mod_norm(log_spend ~ age * diag + year,
                data = spend,
                weights = 1) |>
  set_datamod_noise(sd = sd_diag)


## Poisson model ------------------------------

mod <- mod_pois(deaths ~ month,
                data = usa_deaths,
                exposure = 1) |>
  set_datamod_noise(sd = 200)
```

`set_datamod_outcome_rr3` *Specify RR3 Data Model*

Description

#' 'r lifecycle::badge('deprecated')

Usage

```
set_datamod_outcome_rr3(mod)
```

Arguments

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

Details

This function has been deprecated, and will be removed from future versions of bage. Please used function set_confidential_rr3() instead.

Value

A revised version of mod.

Examples

```
## 'injuries' variable in 'nzl_injuries' dataset
## has been randomly rounded to base 3
mod <- mod_pois(injuries ~ age:sex + ethnicity + year,
                data = nzl_injuries,
                exposure = popn) |>
  set_confidential_rr3() |> ## rather than set_datamod_outcome_rr3
  fit()
```

---

set_datamod_overcount *Specify Overcount Data Model*

Description

Specify a data model for the outcome in a Poisson model, where the outcome is subject to overcount

Usage

```
set_datamod_overcount(mod, rate)
```

Arguments

mod	An object of class "bage_mod_pois", created with mod_pois().
rate	The prior for the overcoverage rate. A data frame with a variable called "mean", a variable called "disp", and, optionally, one or more 'by' variables.

Details

The overcount data model assumes that reported values for the outcome overstate the actual values. The reported values might be affected by double-counting, for instance, or might include some people or events that are not in the target population.

Value

A revised version of mod.

The `rate` argument

The `rate` argument specifies a prior distribution for the overcoverage rate. `rate` is a data frame with a variable called `"mean"`, a variable called `"disp"`, and, optionally, one or more 'by' variables. For instance, a `rate` of

```
data.frame(sex = c("Female", "Male"),
           mean = c(0.05, 0.03),
           disp = c(0.1, 0.15))
```

implies that the reported value for the outcome is expected to overstate the true value by about 5% for females, and about 3% for females, with greater unceratinty for males than females.

Mathematical details

The model for the observed outcome is

$$y_i^{\text{obs}} = y_i^{\text{true}} + \epsilon_i$$
$$\epsilon_i \sim \text{Poisson}(\kappa_{g[i]}\gamma_i w_i)$$
$$\kappa_g \sim \text{Gamma}(1/d_g, 1/(d_g m_g))$$

where

- y_i^{obs} is the observed outcome for cell i;
- y_i^{true} is the true outcome for cell i;
- ϵ_i overcount in cell i;
- γ_i is the rate for cell i;
- w_i is exposure for cell i;
- $\kappa_{g[i]}$ is the overcoverage rate for cell i;
- m_g is the expected value for κ_g (specified via `rate`); and
- d_g is disperson for κ_g (specified via `rate`).

See Also

- `mod_pois()` Specify a Poisson model
- `augment()` Original data plus estimated values, including estimates of true value for the outcome variable
- `components()` Estimated values for model parameters, including inclusion probabilities and overcount rates
- `set_datamod_undercount()` An undercount-only data model
- `set_datamod_miscount()` An undercount-and-overcount data model
- datamods All data models implemented in bage
- confidential Confidentialization procedures modeled in bage
- Mathematical Details vignette

Examples

```
## specify 'rate'
rate <- data.frame(sex = c("Female", "Male"),
                   mean = c(0.1, 0.13),
                   disp = c(0.2, 0.2))

## specify model
mod <- mod_pois(divorces ~ age * sex + time,
                data = nzl_divorces,
                exposure = population) |>
  set_datamod_overcount(rate)
mod

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

## original data, plus imputed values for outcome
mod |>
  augment()

## parameter estimates
library(dplyr)
mod |>
  components() |>
  filter(term == "datamod")

## the data have in fact been confidentialized,
## so we account for that, in addition
## to accounting for overcoverage
mod <- mod |>
 set_confidential_rr3() |>
 fit()
mod
```

`set_datamod_undercount` *Specify Undercount Data Model*

Description

Specify a data model for the outcome in a Poisson or binomial model, where the outcome is subject to undercount.

Usage

```
set_datamod_undercount(mod, prob)
```

Arguments

mod	An object of class "bage_mod", created with mod_pois() or mod_binom().
prob	The prior for the probability that a person or event in the target population will correctly enumerated. A data frame with a variable called "mean", a variable called "disp", and, optionally, one or more 'by' variables.

Details

The undercount data model assumes that reported values for the outcome variable understate the true values, because the reported values miss some people or events in the target population. In other words, the probability that any given unit in the target population will be included in the reported outcome is less than 1.

Value

A revised version of mod.

The prob argument

The prob argument specifies a prior distribution for the probability that a person or event in the target population is included in the reported outcome. prob is a data frame with a variable called "mean", a variable called "disp", and, optionally, one or more 'by' variables. For instance, a prob of

```
data.frame(sex = c("Female", "Male"),
           mean = c(0.95, 0.92),
           disp = c(0.02, 0.015))
```

implies that the expected value for the inclusion probability is 0.95 for females and 0.92 for males, with slightly more uncertainty for females than for males.

Mathematical details

The model for the observed outcome is

$$y_i^{\text{obs}} \sim \text{Binomial}(y_i^{\text{true}}, \pi_{g[i]})$$
$$\pi_g \sim \text{Beta}(m_g^{(\pi)}/d_g^{(\pi)}, (1 - m_g^{(\pi)})/d_g^{(\pi)})$$

where

- y_i^{obs} is the observed outcome for cell i;
- y_i^{true} is the true outcome for cell i;
- $\pi_{g[i]}$ is the probability that a member of the target population in cell i is correctly enumerated in that cell;
- m_g is the expected value for π_g (specified via prob); and
- d_g is disperson for π_g (specified via prob).

See Also

- `mod_pois()` Specify a Poisson model
- `mod_binom()` Specify a binomial model
- `augment()` Original data plus estimated values, including estimates of true value for the outcome variable
- `components()` Estimated values for model parameters, including inclusion probabilities and overcount rates
- `set_datamod_overcount()` An overcount-only data model
- `set_datamod_miscount()` An undercount-and-overcount data model
- datamods All data models implemented in bage
- confidential Confidentialization procedures modeled in bage
- Mathematical Details vignette

Examples

```
## specify 'prob'
prob <- data.frame(sex = c("Female", "Male"),
                   mean = c(0.95, 0.97),
                   disp = c(0.05, 0.05))

## specify model
mod <- mod_pois(divorces ~ age * sex + time,
                data = nzl_divorces,
                exposure = population) |>
  set_datamod_undercount(prob)
mod

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

## original data, plus imputed values for outcome
mod |>
  augment()

## parameter estimates
library(dplyr)
mod |>
  components() |>
  filter(term == "datamod")

## the data have in fact been confidentialized,
## so we account for that, in addition
## to accounting for undercoverage
mod <- mod |>
 set_confidential_rr3() |>
 fit()
mod
```

`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 = 1)
```

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. Default is 1.

Details

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

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

By default μ equals 1.

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, `set_disp()` 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 = nzl_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

- `n_draw.bage_mod()` query the value of `n_draw`
- `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 = nzl_injuries,
                exposure = popn)
mod # value for 'n_draw' displayed in object
n_draw(mod) # or use 'n_draw()' to query

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

`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, `set_prior()` 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 = nzl_injuries,
                exposure = popn)
mod
mod |> set_prior(age ~ RW2())
```

set_seeds *Reset Random Seeds in Model Object*

Description

Reset random seeds stored in a model object. When new_seeds is NULL (the default), the new seeds are generated randomly; otherwise they are taken from new_seeds.

Usage

```
set_seeds(mod, new_seeds = NULL)
```

Arguments

mod	An object of class "bage_mod", created with mod_pois(), mod_binom(), or mod_norm().
new_seeds	NULL (the default) or a list of integers with names "seed_components" "seed_augment", "seed_forecast_components", and "seed_forecast_augment".

Details

When an object of class "bage_mod" is first created, values are generated four four random seeds:

- seed_components
- seed_augment
- seed_forecast_components
- seed_forecast_augment

When fit(), components(), augment(), and forecast() are called on the model object, the seeds are used internally to ensure that he same inputs generate the same outputs, even when the outputs involve random draws.

End users are unlikely to call set_seeds() in a data analysis, though it may occasionally by useful when building a simulation from scratch.

Value

A revised version of mod.

See Also

- report_sim() Do a simulation study. (report_sim() calls set_seeds() internally.)
- mod_pois(), mod_binom(), mod_norm() Specify a model
- fit() Fit a model
- unfit() Reset model, deleting estimates

Examples

```
## fit model
mod <- mod_pois(injuries ~ age,
                data = nzl_injuries,
                exposure = popn) |>
  fit()

## call 'components()'
components(mod)

## call 'components()' again - same results
components(mod)

## reset seeds
mod <- set_seeds(mod)

## calling 'set_seeds' unfits the model
is_fitted(mod)

## so we fit it again
mod <- fit(mod)

## when we call components, we get
## different results from earlier
components(mod)
```

---

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, set_var_age() 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 <- nzl_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, set_var_sexgender() 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 <- nzl_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, `set_var_time()` 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 <- nzl_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 spline) 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,
  sd = 1,
  sd_slope = 1,
  along = NULL,
  con = c("none", "by")
)
```

Arguments

n_comp	Number of spline basis functions (components) to use.
s	Scale for the prior for the innovations. Default is 1.
sd	Standard deviation in prior for first element of random walk.
sd_slope	Standard deviation in prior for initial slope of random walk. Default is 1.
along	Name of the variable to be used as the 'along' variable. Only used with interactions.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

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.

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the `con` argument.

Current options for `con` are:

- `"none"` No constraints. The default.
- `"by"` Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

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
- splines::bs() Function used by **bage** to construct spline basis functions
- Mathematical Details vignette

Examples

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

---

ssvd *Create Object to Hold Data from a Scaled SVD*

Description

Create an object of class "bage_ssvd" to hold results from a scaled Singular Value Decomposition (SVD) with n_comp components.

Usage

```
ssvd(data)
```

Arguments

data	A data frame. See Details for description.

Details

data has the following columns:

- version Vintage of data
- type Type of decomposition. Choices are "total", "joint", and "indep".
- labels_age Age labels for individual rows of matrices within matrix and individual elements of vectors within offset.
- labels_sexgender Sex/gender labels for individual rows of matrices within matrix and individual elements of vectors within offset, or NULL. NULL when sexgender is "total", since in this case results average across sexes/genders.
- matrix List column of sparse matrices. Must have rownames. Must not have NAs. When type is "total" or "joint", each matrix has n_comp columns. When "type" is "indep", each matrix has 2 * n_comp columns.
- offset List column of vectors. Must have names, which are identical to the rownames of the corresponding element of matrix.

data would normally be constructed using functions in package bssvd.

Value

An object of class "bage_ssvd".

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- SVD() Prior based on scaled SVD

Examples

```
ssvd(data_wmd)
```

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, v = NULL, n_comp = 3, indep = TRUE)
```

Arguments

ssvd	Object of class "bage_ssvd" holding a scaled SVD. See below for scaled SVDs of databases currently available in **bage**.
v	Version of scaled SVD components to use. If no value is suppled, the most recent version is used.
n_comp	Number of components from scaled SVD to use in modelling. The default is 3.
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

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

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.
- `HFD` Fertility rates from the Human Fertility Database.
- `LFP` Labor forcce participation rates from the OECD.

References

- For details of the construction of scaled SVDS see the Mathematical Details vignette

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
- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- 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 = 2)
```

svds *Scaled SVDs*

Description

Scaled SVDs contain information on typical age-patterns or age-sex patterns for demographic processes, extracted from international databases. The information is extracted using a singular value decomposition (SVD), and then scaled to make it easier to formulate priors.

Scaled SVDs can have multiple versions, based on data released at different dates, or on subsets of the available data.

Some datasets, and hence some scaled SVDs, include information on age but not on sex or gender.

Details

Scaled SVD	Process	Source	Versions
CSA	School attendance	Census data assembled by UN	`"v2025"`, `"v2024"`
HFD	Fertility	Human Fertility Database	`"v2025"`, `"v2024"`
HIMD_R	Internal migration: annual rates	Human Internal Migration Database	`"v2024"`
HIMD_P1	Internal migration: 1-year probabilities	Human Internal Migration Database	`"v2024"`
HIMD_P5	Internal migration: 5-year probabilities	Human Internal Migration Database	`"v2024"`
HMD	Mortality	Human Mortality Database	`"v2025"`, `"v2025_50"`, `"v202`
LFP	Labour Force Participation	OECD	`"v2025"`
WMD_C	Currently married	World Migration Data	`"v2019"`
WMD_E	Ever married	World Migration Data	`"v2019"`

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,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  n_coef = 2,
```

```
  s = 1,
  shape1 = 5,
  shape2 = 5,
  con = c("none", "by")
)

SVD_AR1(
  ssvd,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  min = 0.8,
  max = 0.98,
  s = 1,
  shape1 = 5,
  shape2 = 5,
  con = c("none", "by")
)

SVD_DRW(
  ssvd,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  s = 1,
  sd = 1,
  min = 0.8,
  max = 0.98,
  shape1 = 5,
  shape2 = 5,
  con = c("none", "by")
)

SVD_DRW2(
  ssvd,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  s = 1,
  sd = 1,
  sd_slope = 1,
  min = 0.8,
  max = 0.98,
  shape1 = 5,
  shape2 = 5,
  con = c("none", "by")
)
```

```
SVD_Lin(
  ssvd,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  s = 1,
  mean_slope = 0,
  sd_slope = 1,
  con = c("none", "by")
)

SVD_RW(
  ssvd,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  s = 1,
  sd = 1,
  con = c("none", "by")
)

SVD_RW2(
  ssvd,
  v = NULL,
  n_comp = 3,
  indep = TRUE,
  s = 1,
  sd = 1,
  sd_slope = 1,
  con = c("none", "by")
)
```

Arguments

ssvd	Object of class "bage_ssvd" holding a scaled SVD. See below for scaled SVDs of databases currently available in **bage**.
v	Version of scaled SVD components to use. If no value is suppled, the most recent version is used.
n_comp	Number of components from scaled SVD to use in modelling. The default is 3.
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. Default is 1. Can be 0 in SVD_Lin().
shape1, shape2	Parameters for prior for coefficients in SVD_AR(), SVD_AR1(), SVD_DRW(), and SVD_DRW2(). Defaults are 5 and 5.
con	Constraints on parameters. Current choices are "none" and "by". Default is "none". See below for details.

`min`, `max`	Minimum and maximum values for autocorrelation coefficient in `SVD_AR1()`, `SVD_DRW()`, and `SVD_DRW2()`. Defaults are `0.8` and `0.98`.
`sd`	Standard deviation of initial value for random walks. Default is `1`. Can be `0`.
`sd_slope`	Standard deviation in prior for initial slope. Default is `1`.
`mean_slope`	Mean in prior for initial slope. Default is `0`.

Details

`SVD_AR()`, `SVD_AR1()`, `SVD_Lin()`, `SVD_RW()`, `SVD_RW2()`, `SVD_DRW()`, 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 inheriting from class `"bage_prior"`.

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;
- $\boldsymbol{F}$ is a known matrix; and
- $\boldsymbol{g}$ is a known vector.

$\boldsymbol{F}$ and $\boldsymbol{g}$ are constructed from a large database of age-specific demographic estimates by applying the singular value decomposition, and then 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_Lin()`, it is

$$\alpha_{k,t} = (t - (T+1)/2)\eta + \epsilon_{k,t}$$

or

$$\alpha_{k,u,t} = (t - (T+1)/2)\eta + \epsilon_{k,u,t};$$

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

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

with `SVD_DRW()`, it is

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

or

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

and with `SVD_DRW2()`, it is

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

or

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

SVD_AR1() and SVD_DRW() are almost but not quite identical. In SVD_AR1(), the variance of ϵ_t is chosen so that α_t has marginal variance τ^2, while in SVD_DRW(), ϵ_t has variance τ^2.

For details on the time series models, see AR(), AR1(), Lin(), RW(), RW2(), DRW(), and DRW2().

Constraints

With some combinations of terms and priors, the values of the intercept, main effects, and interactions are only weakly identified. This weak identifiability is typically harmless. However, in some applications, such as when trying to obtain interpretable values for main effects and interactions, it can be helpful to increase identifiability through the use of constraints, specified through the con argument.

Current options for con are:

- "none" No constraints. The default.
- "by" Only used in interaction terms that include 'along' and 'by' dimensions. Within each value of the 'along' dimension, terms across each 'by' dimension are constrained to sum to 0.

Scaled SVDs of demographic databases in bage

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

References

- For details of the construction of scaled SVDS see the Mathematical Details article

See Also

- SVD() SVD prior for non-time-varying terms
- Lin() Linear with independent errors
- RW() Smoothing via random walk
- RW2() Smoothing via second-order random walk
- DRW() Smoothing via damped random walk
- DRW2() Smoothing via damped second-order random walk
- Sp() Smoothing via splines
- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- 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 = 2)
SVD_RW2(HMD, indep = FALSE)
SVD_Lin(HMD, s = 0)
```

---

swe_infant *Infant Mortality in Sweden*

Description

Counts of births and infant deaths in Sweden by county and year, 1995-2015

Usage

```
swe_infant
```

Format

A tibble with 441 rows and the following columns:

- `county`: A factor with 21 levels, where the levels are ordered by number of births, from `"Stockholm"` down to `"Gotland"`
- ‘time: Calendar year
- `births`: Count of births
- `deaths`: Count of infant deaths

Details

Dataset used in Chapter 11 of the book *Bayesian Demographic Estimation and Forecasting*.

Source

Database "Live births by region, mother’s age and child’s sex. Year 1968 - 2017" and database "Deaths by region, age (during the year) and sex. Year 1968 - 2017" on the Statistics Sweden website. Downloaded on 13 July 2018.

References

Bryant J and Zhang J. 2018. *Bayesian Demographic Estimation and Forecasting*. CRC Press.

See Also

- datasets Overview of datasets in **bage**

`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
- `prior` 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 values for rates, probabilities, or means, together with original data
- `components()` Extract values for hyper-parameters
- `dispersion()` Extract values for dispersion

Examples

```
mod <- mod_pois(injuries ~ age + sex + year,
                data = nzl_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
- `set_seeds()` Reset random seeds
- 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 = nzl_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)
```

---

usa_deaths *Accidental Deaths in the USA*

Description

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

Usage

```
usa_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`.

See Also

- datasets Overview of datasets in **bage**

WMD_C *Scaled SVD Components from World Marriage Database*

Description

Object of class "bage_ssvd" holding scaled SVD components derived from data from the census and survey data on marriage assembled by the United Nations Population Division. WMD_C and WMD_E each hold 5 components.

Usage

```
WMD_C

WMD_E
```

Format

Object of class "bage_ssvd".

Versions:

- "v2019" (default) Data published in 2019

Details

- WMD_C is based on data on the proportion of the population that is currently married. It should be used for modelling the proportion of people whose marital status is "Currently Married"
- WMD_E is based on data on the proportion of the population that has ever been married. It should be used for modelling the proportion of people whose marital status is "Ever Married".

In both cases "marriage" includes de facto marriages and consensual unions, in addition to legal marriages.

Source

Derived from data from the *World Marriage Data 2019* database available on the United Nations Population Division website, and assembled by the UNPD from national census and survey data. Code to create WMD is in folder 'data-raw/ssvd_wmd' in the source code for thet **bage** package.

See Also

- Scaled SVDs Overview of scaled SVDs implemented in **bage**
- SVD() A prior based on a scaled SVD

Index