# Package ‘kDGLM’

July 22, 2025

**Type** Package

**Title** Bayesian Analysis of Dynamic Generalized Linear Models

**Version** 1.2.7

**Description** Provide routines for filtering and smoothing, forecasting, sampling and Bayesian analysis of Dynamic Generalized Linear Models using the methodology described in Alves et al. (2024)<doi:10.48550/arXiv.2201.05387> and dos Santos Jr. et al. (2024)<doi:10.48550/arXiv.2403.13069>.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Imports** extraDistr (>= 1.9.1), Rfast (>= 2.0.8), generics (>= 0.1.3), Rdpack

**RdMacros** Rdpack

**Suggests** knitr, rmarkdown, ggplot2, plotly, tidyverse, spdep, sf, geobr

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**URL** https://silvaneojunior.github.io/kDGLM/

**Repository** CRAN

**BugReports** https://github.com/silvaneojunior/kDGLM/issues

**Depends** R (>= 4.1.0)

**NeedsCompilation** no

**Author** Silvaneo dos Santos Jr. [aut, cre], Mariane Branco Alves [aut], Hélio dos Santos Migon [aut]

**Maintainer** Silvaneo dos Santos Jr. <silvaneojunior@utexas.edu>

**Date/Publication** 2025-03-20 01:00:03 UTC

# Contents



---

`block_mult` *Auxiliary function to replicate blocks*

---

### Description

An auxiliary function to replicate blocks.

### Usage

```
block_mult(block, k)
```

### Arguments

| | |
|---|---|
| `block` | dlm_block: A block to be replicated |
| `k` | Integer: The number of blocks to generate. |

### Value

The combined replicated blocks as a dlm_block.

### See Also

Other auxiliary functions for structural blocks: `TF_block()`, `block_rename()`, `block_superpos()`, `ffs_block()`, `harmonic_block()`, `intervention()`, `noise_block()`, `polynomial_block()`, `regression_block()`, `specify.dlm_block()`, `summary.dlm_block()`

### Examples

```
# Long way
level <- polynomial_block(alpha = 1, order = 1)

final.block <- block_mult(level, 5)

# Short way
final.block <- 5 * polynomial_block(alpha = 1, order = 1)
```

---

## `block_rename` *block_rename*

---

### Description

block_rename

### Usage

```
block_rename(block, pred.names)
```

### Arguments

| | |
|---|---|
| `block` | A dlm_block object. |
| `pred.names` | A vector of string with names for each linear predictor in block. |

### Value

A dlm_block with the linear predictors renamed to the values passed in names.

### See Also

Other auxiliary functions for structural blocks: `TF_block()`, `block_mult()`, `block_superpos()`, `ffs_block()`, `harmonic_block()`, `intervention()`, `noise_block()`, `polynomial_block()`, `regression_block()`, `specify.dlm_block()`, `summary.dlm_block()`

### Examples

```
base.block <- polynomial_block(
  eta = 1,
  order = 1,
  name = "Poly",
  D = 0.95
)

final.block <- block_rename(2 * base.block, c("mu", "sigma"))
```

---

## block_superpos — *Auxiliary function for block superposition*

---

### Description

An auxiliary function for block superposition.

### Usage

```
block_superpos(...)
```

### Arguments

| | |
|---|---|
| `...` | dlm_block: A sequence of block to be combine. |

### Details

Additional details can be found in West and Harrison (1997), section 6.2.

### Value

The combined blocks as a dlm_block.

### References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

### See Also

Other auxiliary functions for structural blocks: `TF_block()`, `block_mult()`, `block_rename()`, `ffs_block()`, `harmonic_block()`, `intervention()`, `noise_block()`, `polynomial_block()`, `regression_block()`, `specify.dlm_block()`, `summary.dlm_block()`

### Examples

```
# Long way
level.1 <- polynomial_block(alpha1 = 1, order = 1)
level.2 <- polynomial_block(alpha2 = 1, order = 2)
season.2 <- harmonic_block(alpha2 = 1, period = 20)

final.block <- block_superpos(level.1, level.2, season.2)

# Short way
final.block <- polynomial_block(alpha1 = 1, order = 1) +
  polynomial_block(alpha2 = 1, order = 2) +
  harmonic_block(alpha2 = 1, period = 20)
```

---

## CAR_prior *CAR prior*

---

### Description

Defines the prior of a structural block as a Conditional Autoregressive (CAR) prior.

### Usage

```
CAR_prior(
  block,
  adj.matrix,
  scale,
  rho,
  sum.zero = FALSE,
  var.index = 1:block$n
)
```

### Arguments

| | |
|---|---|
| `block` | dlm_block object: The structural block. |
| `adj.matrix` | matrix: The adjacency matrix. |
| `scale` | numeric: The tau parameter for the CAR model (see references). |
| `rho` | numeric: The rho parameter for the CAR model (see references). |
| `sum.zero` | Bool: If true, all latent states will add to 0. |
| `var.index` | integer: The index of the variables from which to set the prior. |

### Details

The filtering algorithm used in this package requires a proper prior for the latent space. As such, this implementation of the CAR prior imposes a zero-sum constraint in the regional effects. The discount factor must be the same for all variables whose prior is being modified.

For a revision of the CAR prior, see Schmidt and Nobre (2018).

For the details about the implementation see dos Santos et al. (2024).

### Value

A dlm_block object with the desired prior.

### References

Alexandra M. Schmidt, Widemberg S. Nobre (2018). "Conditional Autoregressive (CAR) Model." In *Wiley StatsRef: Statistics Reference Online*, chapter Conditional Autoregressive (CAR) Model, 1-11. John Wiley & Sons, Ltd. ISBN 9781118445112, doi:10.1002/9781118445112.stat08048, https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat08048, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat08048.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

### See Also

Auxiliary functions for creating structural blocks `polynomial_block`, `regression_block`, `harmonic_block`, `TF_block`.

Other auxiliary functions for defining priors.: `joint_prior()`, `zero_sum_prior()`

### Examples

```
# Creating an arbitrary adjacency matrix
adj.matrix <- matrix(
  c(
    0, 1, 1, 0, 0,
    1, 0, 1, 0, 0,
    1, 1, 0, 0, 0,
    0, 0, 0, 0, 1,
    0, 0, 0, 1, 0
  ),
  5, 5,
  byrow = TRUE
)

polynomial_block(mu = 1, D = 0.95) |>
  block_mult(5) |>
  CAR_prior(scale = 9, rho = 1, adj.matrix = adj.matrix)
```

---

`chickenPox` *Hospital admissions by chicken pox in Brazil*

---

### Description

Monthly hospital admissions by chicken pox in Brazil from January 2010 to December 2019.

### Usage

```
chickenPox
```

### Format

A data frame with 120 rows and 6 columns:

**date** The date of the observations.

**< 5 year, 5 to 9 years, 10 to 14 years, 15 to 49 years, 50 years or more** The number of admissions for each age group.

### Source

https://datasus.saude.gov.br/informacoes-de-saude-tabnet/

---

## `coef.fitted_dlm` *coef.fitted_dlm*

---

### Description

Evaluates the predictive values for the observed values used to fit the model and its latent states. Predictions can be made with smoothed values, with filtered values or h-steps ahead.

### Usage

```
## S3 method for class 'fitted_dlm'
coef(
  object,
  t.eval = seq_len(object$t),
  lag = -1,
  pred.cred = 0.95,
  eval.pred = FALSE,
  eval.metric = FALSE,
  ...
)
```

### Arguments

| | |
|---|---|
| `object` | fitted_dlm: The fitted model to be use for evaluation. |
| `t.eval` | numeric: A vector of positive integers indicating the time index from which to extract predictions. The default is to extract to evaluate the model at all observed times. |
| `lag` | integer: The relative offset for forecast. Values for time t will be calculated based on the filtered values of time t-h. If lag is negative, then the smoothed distribution for the latent states will be used. |
| `pred.cred` | numeric: The credibility level for the C.I.. |

| | |
|---|---|
| `eval.pred` | boolean: A flag indicating if the predictions should be calculated. |
| `eval.metric` | boolean: A flag indicating if the model density (f(M\|y)) should be calculated. Only used when lag<0. |
| `...` | Extra arguments passed to the coef method. |

## Value

A list containing:

- data data.frame: A table with the model evaluated at each observed time.
- theta.mean matrix: The mean of the latent states at each time. Dimensions are n x t, where t is the size of t.eval and n is the number of latent states.
- theta.cov array: A 3D-array containing the covariance matrix of the latent states at each time. Dimensions are n x n x t, where t is the size of t.eval and n is the number of latent states.
- lambda.mean matrix: The mean of the linear predictor at each time. Dimensions are k x t, where t is the size of t.eval and k is the number of linear predictors.
- lambda.cov array: A 3D-array containing the covariance matrix for the linear predictor at each time. Dimensions are k x k x t, where t is the size of t.eval and k is the number of linear predictors.
- log.like, mae, mase, rae, mse, interval.score: The metric value at each time.
- conj.param list: A list containing, for each outcome, a data.frame with the parameter of the conjugated distribution at each time.

## See Also

Other auxiliary functions for fitted_dlm objects: `eval_dlm_norm_const()`, `fit_model()`, `forecast.fitted_dlm()`, `kdglm()`, `simulate.fitted_dlm()`, `smoothing()`, `update.fitted_dlm()`

## Examples

```
# Poisson case
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, order = 2, D = 0.95)
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

outcome <- Poisson(lambda = "rate", data = data)

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
)

var.vals <- coef(fitted.data)
```

---

`cornWheat` *Corn and wheat prices from 1986 to 2014*

---

### Description

The to prices (in U.S. Dollars) per bushel and the log returns of corn and wheat from 1986-01-03 to 2014-10-10. Each observation corresponds to the price on that day, but not all days are present in this dataset.

### Usage

```
cornWheat
```

### Format

A data frame with 7,253 rows and 5 columns:

**date** The date of the observation.

**corn.price, wheat.price** The price (in U.S. Dollars) per bushel of corn and wheat, respectively.

**corn.log.return, wheat.log.return** The log returns for corn and wheat, respectively.

### Source

https://www.macrotrends.net/charts/commodities

---

`ffs_block` *Structural blocks for free-form seasonal trends and regressions*

---

### Description

Creates the structure for a free-form seasonal (FFS) block with desired periodicity.

### Usage

```
ffs_block(
  ...,
  period,
  sum.zero = FALSE,
  name = "Var.FFS",
  D = 1,
  h = 0,
  H = 0,
  a1 = 0,
  R1 = 4,
  monitoring = FALSE
)

ffs(period, D = 0.95, a1 = 0, R1 = 9, name = "Var.FFS", X = 1)
```

### Arguments

| | |
|---|---|
| `...` | Named values for the planning matrix. |
| `period` | Positive integer: The size of the seasonal cycle. This block has one latent state for each element of the cycle, such that the number of latent states n is equal to the period. |
| `sum.zero` | Bool: If true, all latent states will add to 0 and will have a correlated temporal evolution. If false, the first observation is considered the base line level and the states will represent the deviation from the baseline. |
| `name` | String: An optional argument providing the name for this block. Can be useful to identify the models with meaningful labels, also, the name used will be used in some auxiliary functions. |
| `D` | Vector or scalar: The values for the discount factors associated with the first latent state (the current effect) at each time. If D is a vector, it should have size t and it is interpreted as the discount factor at each observed time. If D is a scalar, the same discount will be used at all times. |
| `h` | Vector or scalar: A drift to be add after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a vector, it should have size t, and each value will be applied to the first latent state (the one which affects the linear predictors) in their respective time. If a scalar, the passed value will be used for the first latent state at each time. |
| `H` | Vector or scalar: The values for the covariance matrix for the noise factor at each time. If a vector, it should have size t, and each value will represent the variance of the temporal evolution at each time. If a scalar, the passed value will be used for the first latent state at each time. |
| `a1` | Vector or scalar: The prior mean for the latent states associated with this block at time 1. If a1 is a vector, its dimension should be equal to the period of the FFS block. If a1 is a scalar, its value will be used for all latent states. |
| `R1` | Matrix, vector or scalar: The prior covariance matrix for the latent states associated with this block at time 1. If R1 is a matrix, its dimensions should be period x period. If R1 is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1 in the diagonal. |
| `monitoring` | Bool: A indicator if the first latent state should be monitored (if automated monitoring is used). |
| `X` | Vector or scalar: An argument providing the values of the covariate X_t. |

### Details

For the ..., D, H, a1 and R1 arguments, the user may set one or more of its values as a string. By doing so, the user will leave the block partially undefined. The user must then pass the undefined parameter values as named arguments to the `fit_model` function. Also, multiple values can be passed, allowing for a sensitivity analysis for the value of this parameter.

For the details about the implementation see dos Santos et al. (2024).

For the details about the free-form seasonal trends in the context of DLM's, see West and Harrison (1997), chapter 8.

For the details about dynamic regression models in the context of DLM's, see West and Harrison (1997), chapters 6 and 9.

### Value

A dlm_block object containing the following values:

- FF Array: A 3D-array containing the regression matrix for each time. Its dimension should be n x k x t, where n is the number of latent states, k is the number of linear predictors in the model and t is the time series length.
- FF.labs Matrix: A n x k character matrix describing the type of value of each element of FF.
- G Matrix: A 3D-array containing the evolution matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- G.labs Matrix: A n x n character matrix describing the type of value of each element of G.
- G.idx Matrix: A n x n character matrix containing the index each element of G.
- D Array: A 3D-array containing the discount factor matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- h Matrix: The mean for the random noise of the temporal evolution. Its dimension should be n x t.
- H Array: A 3D-array containing the covariance matrix of the noise for each time. Its dimension should be the same as D.
- a1 Vector: The prior mean for the latent vector.
- R1 Matrix: The prior covariance matrix for the latent vector.
- var.names list: A list containing the variables indexes by their name.
- period Positive integer: Same as argument.
- n Positive integer: The number of latent states associated with this block (2).
- t Positive integer: The number of time steps associated with this block. If 1, the block is compatible with blocks of any time length, but if t is greater than 1, this block can only be used with blocks of the same time length.
- k Positive integer: The number of outcomes associated with this block. This block can only be used with blocks with the same outcome length.
- pred.names Vector: The name of the linear predictors associated with this block.
- monitoring Vector: Same as argument.
- type Character: The type of block (Harmonic).

### References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). “kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models.”

### See Also

`fit_model`

Other auxiliary functions for structural blocks: `TF_block()`, `block_mult()`, `block_rename()`, `block_superpos()`, `harmonic_block()`, `intervention()`, `noise_block()`, `polynomial_block()`, `regression_block()`, `specify.dlm_block()`, `summary.dlm_block()`

### Examples

```
# Creating a first order structure for a model with 2 outcomes.
# One block is created for each outcome
# with each block being associated with only one of the outcomes.
season.1 <- ffs_block(alpha1 = 1, period = 12)
season.2 <- ffs_block(alpha2 = 1, period = 12)

# Creating a block with shared effect between the outcomes
season.3 <- ffs_block(alpha1 = 1, alpha2 = 1, period = 12)
```

---

## fit_model *Fitting kDGLM models*

---

### Description

Fit a model given its structure and the observed data. This function can be used for any supported family (see vignette).

### Usage

```
fit_model(
  ...,
  smooth = TRUE,
  p.monit = NA,
  condition = "TRUE",
  metric = "log.like",
  lag = 1,
  pred.cred = 0.95,
  metric.cutoff = NA,
  save.models = FALSE,
  silent = FALSE
)
```

### Arguments

| | |
|---|---|
| `...` | dlm_block or dlm_distr objects or named values: The structural blocks of the model (dlm_block objects), alongside the model outcomes (dlm_distr object). If at least one block is undefined, the user must also provide its value in this argument (see last example). |
| `smooth` | boolean: A flag indicating if the smoothed distribution for the latent states should be calculated. |
| `p.monit` | numeric (optional): The prior probability of changes in the latent space variables that are not part of its dynamic. Only used when performing sensitivity analysis. |
| `condition` | character: A character defining which combinations of undefined hyper parameter should be tested. See example for details. Only used when performing sensitivity analysis. |

`metric` character: The name of the metric to use for model selection. One of log-likelihood for the one-step-ahead prediction ("log.like"), Mean Absolute Scaled Error ("mase") (Hyndman and Koehler 2006) or Interval Score ("interval.score") (Bracher et al. 2021). Only used when performing sensitivity analysis.

`lag` integer: The number of steps ahead used for the prediction when calculating the metrics. If lag<0, predictions are made using the smoothed distribution of the latent states. Only used when performing sensitivity analysis.

`pred.cred` numeric: A number between 0 and 1 (not included) indicating the credibility interval for predictions. If not within the valid range of values, 0.95 will be used. Only used when performing sensitivity analysis.

`metric.cutoff` integer: The number of observations to ignore when calculating the metrics. Default is 1/10 of the number of observations (rounded down). Only used when performing sensitivity analysis.

`save.models` boolean: A flag indicating if all evaluated models should be saved. If FALSE, only the best model (according to the chosen metric) will be saved. Only used when performing sensitivity analysis.

`silent` boolean: A flag indicating if a progress bar should be printed. Only used when performing sensitivity analysis.

## Details

This is the main function of the kDGLM package, as it is used to fit all models.

For the details about the implementation see dos Santos et al. (2024).

For the details about the methodology see Alves et al. (2024).

For the details about the Dynamic Linear Models see West and Harrison (1997) and Petris et al. (2009).

## Value

A fitted_dlm object.

## See Also

auxiliary functions for creating outcomes `Poisson`, `Multinom`, `Normal`, `Gamma`

auxiliary functions for creating structural blocks `polynomial_block`, `regression_block`, `harmonic_block`, `TF_block`

auxiliary functions for defining priors `zero_sum_prior`, `CAR_prior`

Other auxiliary functions for fitted_dlm objects: `coef.fitted_dlm()`, `eval_dlm_norm_const()`, `forecast.fitted_dlm()`, `kdglm()`, `simulate.fitted_dlm()`, `smoothing()`, `update.fitted_dlm()`

## Examples

```
# Poisson case
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, order = 2, D = 0.95)
```

```
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

outcome <- Poisson(lambda = "rate", data = data)

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
)
summary(fitted.data)

plot(fitted.data, plot.pkg = "base")

##################################################################

# Multinomial case
structure <- (
  polynomial_block(p = 1, order = 2, D = 0.95) +
    harmonic_block(p = 1, period = 12, D = 0.975) +
    noise_block(p = 1, R1 = 0.1) +
    regression_block(p = chickenPox$date >= as.Date("2013-09-01"))
  # Vaccine was introduced in September of 2013
) * 4

outcome <- Multinom(p = structure$pred.names, data = chickenPox[, c(2, 3, 4, 6, 5)])
fitted.data <- fit_model(structure, chickenPox = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")

##################################################################

# Univariate Normal case
structure <- polynomial_block(mu = 1, D = 0.95) +
  polynomial_block(V = 1, D = 0.95)

outcome <- Normal(mu = "mu", V = "V", data = cornWheat$corn.log.return[1:500])
fitted.data <- fit_model(structure, corn = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")

##################################################################

# Bivariate Normal case
structure <- (polynomial_block(mu = 1, D = 0.95) +
  polynomial_block(V = 1, D = 0.95)) * 2 +
  polynomial_block(rho = 1, D = 0.95)

outcome <- Normal(
  mu = c("mu.1", "mu.2"),
  V = matrix(c("V.1", "rho", "rho", "V.2"), 2, 2),
  data = cornWheat[1:500, c(4, 5)]
)
fitted.data <- fit_model(structure, cornWheat = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")
```

```
###################################################################

# Gamma case
structure <- polynomial_block(mu = 1, D = 0.95)

Y <- (cornWheat$corn.log.return[1:500] - mean(cornWheat$corn.log.return[1:500]))**2
outcome <- Gamma(phi = 0.5, mu = "mu", data = Y)
fitted.data <- fit_model(structure, corn = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")

###################################################################

# Sensitivity analysis
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, order = 2, D = "D.level")
season <- harmonic_block(rate = "sazo.effect", order = 2, period = 12, D = "D.sazo")

outcome <- Poisson(lambda = "rate", data = data)

fit_model(level, season, outcome,
  sazo.effect = c(0, 1),
  D.level = c(seq.int(0.8, 1, l = 11)),
  D.sazo = c(seq.int(0.95, 1, l = 11)),
  condition = "sazo.effect==1 | D.sazo==1"
)
```

## fit_model_single — *Fitting one kDGLM models*

### Description

Fits one model given its structure and the observed data. This function can be used for any supported family (see vignette).

### Usage

```
fit_model_single(structure, outcomes, smooth = TRUE, p.monit = NA)
```

### Arguments

| | |
|---|---|
| `structure` | dlm_block: The structural blocks of the model. All block must be completely defined. |
| `outcomes` | dlm_distr or list of dlm_distr objects: The model outcomes. |

| | |
|---|---|
| smooth | boolean: A flag indicating if the smoothed distribution for the latent states should be calculated. |
| p.monit | numeric (optional): The prior probability of changes in the latent space variables that are not part of its dynamic. |

## Value

A fitted_dlm object.

---

forecast.fitted_dlm *Auxiliary function for forecasting*

---

## Description

Auxiliary function for forecasting

## Usage

```
## S3 method for class 'fitted_dlm'
forecast(
  object,
  t = 1,
  plot = ifelse(requireNamespace("plotly", quietly = TRUE), "plotly",
    ifelse(requireNamespace("ggplot2", quietly = TRUE), "ggplot2", "base")),
  pred.cred = 0.95,
  ...
)
```

## Arguments

| | |
|---|---|
| object | fitted_dlm object: The fitted model to be use for predictions. |
| t | numeric: Time window for prediction. |
| plot | boolean or character: A flag indicating if a plot should be produced. Should be one of FALSE, TRUE, 'base', 'ggplot2' or 'plotly'. |
| pred.cred | numeric: The credibility level for the C.I.. |
| ... | Extra variables necessary for prediction (covariates, etc.). |

## Details

If an a covariate is necessary for forecasting, it should be passed as a named argument. Its name must follow this structure: <block name>.Covariate<.index>. If there is only one covariate in the associated block the index is omitted. If an a pulse is necessary for forecasting, it should be passed as a named argument. Its name must follow this structure: <block name>.Pulse<.index>. If there is only one pulse in the associated block the index is omitted. The user may pass the observed values at the prediction windows (optional). See example. As an special case, if the model has an Multinomial outcome, the user may pass the N parameter instead of the observations. If an offset is necessary for forecasting, it should be passed with the same syntax as the observed data. See example.

### Value

A list containing:

- data data.frame: A data frame contain the mean, variance and credibility intervals for the outcomes, including both the observed data and the predictions for future observations.
- forecast data.frame: Same as data, but restricted to predictions for future observations.
- outcomes list: A named list containing predictions for each outcome. Each element of this list is a list containing predictions (mean, variance and credibility intervals), the distribution of the linear predictor for the parameter of the observational model and the parameters of the predictive distribution (if available).
- theta.mean matrix: A matrix with the values for the latent states at each time. Dimensions are n x t, where n is the number of latent states
- theta.cov array: A 3D-array with the covariance of the latent states at each time. Dimensions are n x n x t, where n is the number of latent predictors.
- lambda.mean matrix: A matrix with the values for the linear predictors at each time. Dimensions are k x t, where k is the number of linear predictors
- lambda.cov array: A 3D-array with the covariance of the linear predictors at each time. Dimensions are k x k x t, where k is the number of linear predictors.
- plot (if so chosen): A plotly or ggplot object.

A list containing:

- data data.frame: A table with the model evaluated at each observed time, plus the forecasted period.
- forecast data.frame: A table with the model evaluated at the forecasted period.
- outcomes list: A list containing the parameters of the predictive distribution for each outcome at the forecasted period.
- theta.mean matrix: The mean of the latent states at each forecasted time. Dimensions are n x t.forecast, where t.forecast is the size of the forecast windows and n is the number of latent states.
- theta.cov array: A 3D-array containing the covariance matrix of the latent states at each forecasted time. Dimensions are n x n x t.forecast, where t.forecast is the size of the forecast windows and n is the number of latent states.
- lambda.mean matrix: The mean of the linear predictor at each forecasted time. Dimensions are k x t.forecast, where t.forecast is the size of the forecast windows and k is the number of linear predictors.
- lambda.cov array: A 3D-array containing the covariance matrix for the linear predictor at each forecasted time. Dimensions are k x k x t.forecast, where t.forecast is the size of the forecast windows and k is the number of linear predictors.

### See Also

Other auxiliary functions for fitted_dlm objects: `coef.fitted_dlm()`, `eval_dlm_norm_const()`, `fit_model()`, `kdglm()`, `simulate.fitted_dlm()`, `smoothing()`, `update.fitted_dlm()`

### Examples

```
structure <-
  polynomial_block(p = 1, order = 2, D = 0.95) +
  harmonic_block(p = 1, period = 12, D = 0.975) +
  noise_block(p = 1, R1 = 0.1) +
  regression_block(
    p = chickenPox$date >= as.Date("2013-09-1"),
    # Vaccine was introduced in September of 2013
    name = "Vaccine"
  )

outcome <- Multinom(p = c("p.1", "p.2"), data = chickenPox[, c(2, 3, 5)])
fitted.data <- fit_model(structure * 2,
  chickenPox = outcome
)

forecast(fitted.data, 24,
  chickenPox = list(Total = rep(175, 24)), # Optional
  Vaccine.1.Covariate = rep(TRUE, 24),
  Vaccine.2.Covariate = rep(TRUE, 24)
)
```

---

## Gamma — *Gamma outcome for kDGLM models*

---

### Description

Creates an outcome with gamma distribution with the chosen parameters (can only specify 2).

### Usage

```
Gamma(
  phi = NA,
  mu = NA,
  alpha = NA,
  beta = NA,
  sigma = NA,
  data,
  offset = as.matrix(data)^0
)
```

### Arguments

| | |
|---|---|
| phi | character or numeric: Name of the linear predictor associated with the shape parameter of the gamma distribution. If numeric, this parameter is treated as known and equal to the value passed. If a character, the parameter is treated as unknown and equal to the exponential of the associated linear predictor. It cannot be specified with alpha. |

| | |
|---|---|
| mu | character: Name of the linear predictor associated with the mean parameter of the gamma distribution. The parameter is treated as unknown and equal to the exponential of the associated linear predictor. |
| alpha | character: Name of the linear predictor associated with the shape parameter of the gamma distribution. The parameter is treated as unknown and equal to the exponential of the associated linear predictor. It cannot be specified with phi. |
| beta | character: Name of the linear predictor associated with the rate parameter of the gamma distribution. The parameter is treated as unknown and equal to the exponential of the associated linear predictor. It cannot be specified with sigma. |
| sigma | character: Name of the linear predictor associated with the scale parameter of the gamma distribution. The parameter is treated as unknown and equal to the exponential of the associated linear predictor. It cannot be specified with beta. |
| data | numeric: Values of the observed data. |
| offset | numeric: The offset at each observation. Must have the same shape as data. |

## Details

For evaluating the posterior parameters, we use the method proposed in Alves et al. (2024).

For the details about the implementation see dos Santos et al. (2024).

## Value

An object of the class dlm_distr

## References

Mariane Branco Alves, Helio S. Migon, Raíra Marotta, Junior, Silvaneo Vieira dos Santos (2024). “k-parametric Dynamic Generalized Linear Models: a sequential approach via Information Geometry.” 2201.05387.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). “kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models.”

## See Also

fit_model

Other auxiliary functions for a creating outcomes: Multinom(), Normal(), Poisson(), summary.dlm_distr()

## Examples

```
structure <- polynomial_block(mu = 1, D = 0.95)

Y <- (cornWheat$corn.log.return[1:500] - mean(cornWheat$corn.log.return[1:500]))**2
outcome <- Gamma(phi = 0.5, mu = "mu", data = Y)
fitted.data <- fit_model(structure, corn = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")
```

gastroBR *Hospital admissions from gastroenteritis in Brazil*

## Description

A dataset containing the number of Hospital admissions from gastroenteritis in Brazil, per state, from 2010 to 2022 by month.

## Usage

```
gastroBR
```

## Format

A data frame with 4212 rows and 4 variables:

**UF** The abbreviated state name.

**Date** The date of the observation. Note that the day is only a placeholder and is just a placeholder.

**Admissions** The number hospital admissions.

**Population** The estimated population.

## Source

Admissions: https://datasus.saude.gov.br/informacoes-de-saude-tabnet/

Population: https://www.ibge.gov.br/estatisticas/sociais/populacao.html

harmonic_block *Structural blocks for seasonal trends and regressions*

## Description

Creates the structure for a harmonic block with desired periodicity.

## Usage

```
harmonic_block(
  ...,
  period,
  order = 1,
  name = "Var.Sazo",
  D = 1,
  h = 0,
  H = 0,
  a1 = 0,
  R1 = 4,
```

```
  monitoring = rep(FALSE, order * 2)
)

har(period, order = 1, D = 0.98, a1 = 0, R1 = 4, name = "Var.Sazo", X = 1)
```

## Arguments

`...` Named values for the planning matrix.

`period` Positive integer: The size of the harmonic cycle.

`order` Positive integer: The order of the harmonic structure.

`name` String: An optional argument providing the name for this block. Can be useful to identify the models with meaningful labels, also, the name used will be used in some auxiliary functions.

`D` Array, Matrix, vector or scalar: The values for the discount factors associated with the latent states at each time. If D is an array, its dimensions should be (2n) x (2n) x t, where n is the order of the harmonic block and t is the length of the outcomes. If D is a matrix, its dimensions should be (2n) x (2n) and the same discount matrix will be used in all observations. If D is a vector, it should have size t and it is interpreted as the discount factor at each observed time (same discount for all variable). If D is a scalar, the same discount will be used for all latent states at all times.

`h` Matrix, vector or scalar: A drift to be add after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a matrix, its dimension should be (2n) x t, where n is the order of the harmonic_block and t is the length of the series. If a vector, it should have size t, and each value will be applied to the first latent state (the one which affects the linear predictors) in their respective time. If a scalar, the passed value will be used for the first latent state at each time.

`H` Array, Matrix, vector or scalar: The values for the covariance matrix for the noise factor at each time. If H is an array, its dimensions should be (2n) x (2n) x t, where n is the order of the harmonic block and t is the length of the series. If H is a matrix, its dimensions should be (2n) x (2n) and its values will be used for each time. If H is a vector or scalar, a discount factor matrix will be created as a diagonal matrix with the values of H in the diagonal.

`a1` Vector or scalar: The prior mean for the latent states associated with this block at time 1. If a1 is a vector, its dimension should be equal to two times the order of the harmonic block. If a1 is a scalar, its value will be used for all latent states.

`R1` Matrix, vector or scalar: The prior covariance matrix for the latent states associated with this block at time 1. If R1 is a matrix, its dimensions should be (2n) x (2n). If R1 is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1 in the diagonal.

`monitoring` Vector: A vector of flags indicating which variables should be monitored (if automated monitoring is used). Its size should be 2n. The default is that only the first order component of this structure should be monitored.

`X` Vector or scalar: An argument providing the values of the covariate X_t.

### Details

For the ..., D, H, a1 and R1 arguments, the user may set one or more of its values as a string. By doing so, the user will leave the block partially undefined. The user must then pass the undefined parameter values as named arguments to the `fit_model` function. Also, multiple values can be passed, allowing for a sensitivity analysis for the value of this parameter.

For the details about the implementation see dos Santos et al. (2024).

For the details about the modelling of seasonal trends using harmonics in the context of DLM's, see West and Harrison (1997), chapter 8.

For the details about dynamic regression models in the context of DLM's, see West and Harrison (1997), chapters 6 and 9.

### Value

A dlm_block object containing the following values:

- FF Array: A 3D-array containing the regression matrix for each time. Its dimension should be n x k x t, where n is the number of latent states, k is the number of linear predictors in the model and t is the time series length.
- FF.labs Matrix: A n x k character matrix describing the type of value of each element of FF.
- G Matrix: A 3D-array containing the evolution matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- G.labs Matrix: A n x n character matrix describing the type of value of each element of G.
- G.idx Matrix: A n x n character matrix containing the index each element of G.
- D Array: A 3D-array containing the discount factor matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- h Matrix: The mean for the random noise of the temporal evolution. Its dimension should be n x t.
- H Array: A 3D-array containing the covariance matrix of the noise for each time. Its dimension should be the same as D.
- a1 Vector: The prior mean for the latent vector.
- R1 Matrix: The prior covariance matrix for the latent vector.
- var.names list: A list containing the variables indexes by their name.
- period Positive integer: Same as argument.
- n Positive integer: The number of latent states associated with this block (2).
- t Positive integer: The number of time steps associated with this block. If 1, the block is compatible with blocks of any time length, but if t is greater than 1, this block can only be used with blocks of the same time length.
- k Positive integer: The number of outcomes associated with this block. This block can only be used with blocks with the same outcome length.
- pred.names Vector: The name of the linear predictors associated with this block.
- monitoring Vector: Same as argument.
- type Character: The type of block (Harmonic).

### References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

### See Also

fit_model

Other auxiliary functions for structural blocks: TF_block(), block_mult(), block_rename(), block_superpos(), ffs_block(), intervention(), noise_block(), polynomial_block(), regression_block(), specify.dlm_block(), summary.dlm_block()

### Examples

```
# Creating seasonal structure for a model with 2 outcomes.
# One block is created for each outcome
# with each block being associated with only one of the outcomes.
season.1 <- harmonic_block(alpha1 = 1, period = 3)
season.2 <- harmonic_block(alpha2 = 1, period = 6)

# Creating a block with shared effect between the outcomes
season.3 <- harmonic_block(alpha = 1, alpha2 = 1, period = 12)
```

---

## intervention — *An auxiliary function for model intervention*

---

### Description

This function adds timely modifications to a dlm_block, such that in the specified time the model will override the usual value of the each variable to the value chosen by the user.

### Usage

```
intervention(
  block,
  time,
  var.index = 1:block$n,
  FF = NULL,
  D = NULL,
  h = NULL,
  H = NULL,
  G = NULL
)
```

### Arguments

- `block` dlm_block: The block to add the intervention.
- `time` Vector: A sequence of integers indicating the time of the intervention.
- `var.index` Vector: A sequence of integers indicating which variables should be modified in the intervention.
- `FF` Array: A n x k x t array with the modified FF to be used during the intervention, where n is the length of var.index, k is the number of linear predictors in the block and t is the size of time (can be omitted if time is a scalar).
- `D` Array: A n x n x t array with the modified D to be used during the intervention, where n is the length of var.index and t is the size of time (can be omitted if time is a scalar).
- `h` matrix: A n x t matrix with the modified h to be used during the intervention, where n is the length of var.index and t is the size of time (can be omitted if time is a scalar).
- `H` Array: A n x n x t array with the modified H to be used during the intervention, where n is the length of var.index and t is the size of time (can be omitted if time is a scalar).
- `G` Array: A n x n x t array with the modified G to be used during the intervention, where n is the length of var.index and t is the size of time (can be omitted if time is a scalar).

### Value

A dlm_block with the added intervention.

### See Also

Other auxiliary functions for structural blocks: `TF_block()`, `block_mult()`, `block_rename()`, `block_superpos()`, `ffs_block()`, `harmonic_block()`, `noise_block()`, `polynomial_block()`, `regression_block()`, `specify.dlm_block()`, `summary.dlm_block()`

### Examples

```
data <- c(AirPassengers)
# Adding an artificial change, so that we can make an intervention on the data at that point
# Obviously, one should NOT change their own data.
data[60:144] <- data[60:144] + 500

level <- polynomial_block(rate = 1, order = 2, D = 0.95)
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

# Reducing the discount factor so that the model can capture the expected change.
level <- level |> intervention(time = 60, H = 1, var.index = 1)
# Comment the line above to see the fit without the intervention

outcome <- Poisson(lambda = "rate", data = data)

fitted.data <- fit_model(level, season,
```

```
  AirPassengers = outcome
)

plot(fitted.data, plot.pkg = "base")
```

---

`joint_prior` *Joint prior*

---

### Description

Defines the joint prior of a structural block.

### Usage

```
joint_prior(
  block,
  var.index = 1:block$n,
  a1 = block$a1[var.index],
  R1 = block$R1[var.index, var.index]
)
```

### Arguments

| | |
|---|---|
| `block` | dlm_block object: The structural block. |
| `var.index` | Integer: The index of the variables from which to set the prior. |
| `a1` | Numeric: The prior mean. |
| `R1` | Matrix: The prior covariance matrix. |

### Details

The discount factor must be the same for all variables whose prior is being modified. For the details about the implementation see dos Santos et al. (2024).

### Value

A dlm_block object with the desired prior.

### References

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). “kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models.”

### See Also

Other auxiliary functions for defining priors.: `CAR_prior()`, `zero_sum_prior()`

## Examples

```
polynomial_block(mu = 1, D = 0.95) |>
  block_mult(5) |>
  joint_prior(var.index = 1:2, R1 = matrix(c(1, 0.5, 0.5, 1), 2, 2))
```

---

`kdglm` *Fitting kDGLM models*

---

## Description

Fit a model given its structure and the observed data. This function can be used for any supported family (see vignette).

## Usage

```
kdglm(formula, ..., family, data = NULL, offset = NULL, p.monit = NA)
```

## Arguments

| | |
|---|---|
| `formula` | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. |
| `...` | Extra arguments, including extra formulas (multinomial case) or extra parameters (normal and gamma cases). |
| `family` | a description of the error distribution to be used in the model. For kdglm this can be a character string naming a family function or a family function. |
| `data` | an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called. |
| `offset` | this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead. |
| `p.monit` | numeric (optional): The prior probability of changes in the latent space variables that are not part of its dynamic. Only used when performing sensitivity analysis. |

## Details

This is the main function of the kDGLM package, as it is used to fit all models.

For the details about the implementation see dos Santos et al. (2024).

For the details about the methodology see Alves et al. (2024).

For the details about the Dynamic Linear Models see West and Harrison (1997) and Petris et al. (2009).

### Value

A fitted_dlm object.

### See Also

auxiliary functions for creating outcomes Poisson, Multinom, Normal, Gamma

auxiliary functions for creating structural blocks polynomial_block, regression_block, harmonic_block, TF_block

auxiliary functions for defining priors zero_sum_prior, CAR_prior

Other auxiliary functions for fitted_dlm objects: coef.fitted_dlm(), eval_dlm_norm_const(), fit_model(), forecast.fitted_dlm(), simulate.fitted_dlm(), smoothing(), update.fitted_dlm()

### Examples

```
# Poisson case
fitted.data <- kdglm(c(AirPassengers) ~ pol(2) + har(12, order = 2), family = Poisson)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")

##################################################################

# Multinomial case
chickenPox$Total <- rowSums(chickenPox[, c(2, 3, 4, 6, 5)])
chickenPox$Vaccine <- chickenPox$date >= as.Date("2013-09-01")
fitted.data <- kdglm(`< 5 year` ~ pol(2, D = 0.95) + har(12, D = 0.975) + noise(R1 = 0.1) + Vaccine,
  `5 to 9 years` ~ pol(2, D = 0.95) + har(12, D = 0.975) + noise(R1 = 0.1) + Vaccine,
  `10 to 14 years` ~ pol(2, D = 0.95) + har(12, D = 0.975) + noise(R1 = 0.1) + Vaccine,
  `50 years or more` ~ pol(2, D = 0.95) + har(12, D = 0.975) + noise(R1 = 0.1) + Vaccine,
  N = chickenPox$Total,
  family = Multinom,
  data = chickenPox
)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")

##################################################################

# Univariate Normal case
fitted.data <- kdglm(corn.log.return ~ 1, V = ~1, family = Normal, data = cornWheat[1:500, ])
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")

##################################################################

# Gamma case
Y <- (cornWheat$corn.log.return[1:500] - mean(cornWheat$corn.log.return[1:500]))**2
fitted.data <- kdglm(Y ~ 1, phi = 0.5, family = Gamma, data = cornWheat)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")
```

Multinom *Multinom outcome for kDGLM models*

## Description

Creates an outcome with Multinomial distribution with the chosen parameters.

## Usage

```
Multinom(p, data, offset = as.matrix(data)^0, base.class = NULL)
```

## Arguments

| | |
|---|---|
| `p` | character: a vector with the name of the linear predictor associated with the probability of each category (except the base one, which is assumed to be the last). |
| `data` | vector: Values of the observed data. |
| `offset` | vector: The offset at each observation. Must have the same shape as data. |
| `base.class` | character or integer: The name or index of the base class. Default is to use the last column of data. |

## Details

For evaluating the posterior parameters, we use the method proposed in Alves et al. (2024).

For the details about the implementation see dos Santos et al. (2024).

## Value

A object of the class dlm_distr

## References

Mariane Branco Alves, Helio S. Migon, Raíra Marotta, Junior, Silvaneo Vieira dos Santos (2024). "k-parametric Dynamic Generalized Linear Models: a sequential approach via Information Geometry." 2201.05387.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

`fit_model`

Other auxiliary functions for a creating outcomes: `Gamma()`, `Normal()`, `Poisson()`, `summary.dlm_distr()`

## Examples

```
structure <- (
  polynomial_block(p = 1, order = 2, D = 0.95) +
    harmonic_block(p = 1, period = 12, D = 0.975) +
    noise_block(p = 1, R1 = 0.1) +
    regression_block(p = chickenPox$date >= as.Date("2013-09-01"))
  # Vaccine was introduced in September of 2013
) * 4

outcome <- Multinom(p = structure$pred.names, data = chickenPox[, c(2, 3, 4, 6, 5)])
fitted.data <- fit_model(structure, chickenPox = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")
```

---

`noise_block` *noise_block*

---

## Description

Creates the structure for a Noise block. This block represents an independent random noise that should be added to the linear predictor. The variance of the noise cannot be formally estimated, as such we use a discount strategy similar to that of West and Harrison (1997) to specify it.

## Usage

```
noise_block(..., name = "Noise", D = 0.99, R1 = 0.1, H = 0)

noise(name = "Noise", D = 0.99, R1 = 0.1, H = 0, X = 1)
```

## Arguments

| | |
|---|---|
| `...` | Named values for the planning matrix. |
| `name` | String: An optional argument providing the name for this block. Can be useful to identify the models with meaningful labels, also, the name used will be used in some auxiliary functions. |
| `D` | scalar or vector: A sequence of values specifying the desired discount factor for each time. It should have length 1 or t, where t is the size of the series. If both D and H are specified, the value of D is ignored. |
| `R1` | scalar: The prior variance of the noise. |
| `H` | scalar: The variance of the noise. If both D and H are specified, the value of D is ignored. |
| `X` | Vector or scalar: An argument providing the values of the covariate X_t. |

## Details

For the details about the implementation see dos Santos et al. (2024).

For the details about dynamic regression models in the context of DLMs, see West and Harrison (1997), chapters 6 and 9.

## Value

A dlm_block object containing the following values:

- FF Array: A 3D-array containing the regression matrix for each time. Its dimension should be n x k x t, where n is the number of latent states, k is the number of linear predictors in the model and t is the time series length.
- FF.labs Matrix: A n x k character matrix describing the type of value of each element of FF.
- G Matrix: A 3D-array containing the evolution matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- G.labs Matrix: A n x n character matrix describing the type of value of each element of G.
- D Array: A 3D-array containing the discount factor matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- H Array: A 3D-array containing the covariance matrix of the noise for each time. Its dimension should be the same as D.
- a1 Vector: The prior mean for the latent vector.
- R1 Matrix: The prior covariance matrix for the latent vector.
- var.names list: A list containing the variables indexes by their name.
- order Positive integer: Same as argument.
- n Positive integer: The number of latent states associated with this block (2).
- t Positive integer: The number of time steps associated with this block. If 1, the block is compatible with blocks of any time length, but if t is greater than 1, this block can only be used with blocks of the same time length.
- k Positive integer: The number of outcomes associated with this block. This block can only be used with blocks with the same outcome length.
- pred.names Vector: The name of the linear predictors associated with this block.
- monitoring Vector: The combination of monitoring, monitoring and monitoring.pulse.
- type Character: The type of block (Noise).

## References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). “kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models.”

### See Also

fit_model

Other auxiliary functions for structural blocks: TF_block(), block_mult(), block_rename(), block_superpos(), ffs_block(), harmonic_block(), intervention(), polynomial_block(), regression_block(), specify.dlm_block(), summary.dlm_block()

### Examples

```
noise_block(mu = 1, D = 0.99, R1 = 1e-2)
```

---

## Normal — *Normal outcome for kDGLM models*

---

### Description

Creates an outcome with Normal distribution with the chosen parameters (can only specify 2).

### Usage

```
Normal(mu, V = NA, Tau = NA, Sd = NA, data)
```

### Arguments

- `mu` — character: Name of the linear predictor associated with the mean parameter of the Normal distribution. The parameter is treated as unknown and equal to the associated linear predictor.
- `V` — character or numeric: If V is a character, it is interpreted as the names of the linear predictors associated with the variance parameter of the Normal distribution. If V is numeric, the variance is considered known and equal to the value of V, otherwise, the variance is considered unknown and equal to the exponential of the linear predictor informed in V. If the outcome is a Multivariate Normal, then V must be a matrix and, if the variance is unknown, the elements outside its main diagonal are treated as the linear predictor associated with the correlation between each coordinate of the outcome, otherwise V is treated as the covariance matrix. The user cannot specify V with Tau or Sd.
- `Tau` — character or numeric: If Tau is a character, it is interpreted as the names of the linear predictors associated with the precisions parameter of the Normal distribution. If Tau is numeric, the precision is considered known and equal to the value of Tau, otherwise, the precision is considered unknown and equal to the exponential of the linear predictor informed in Tau. If the outcome is a Multivariate Normal, then Tau must be a matrix and, if the precision is unknown, the elements outside its main diagonal are treated as the linear predictor associated with the correlation between each coordinate of the outcome, otherwise Tau is treated as the precision matrix. The user cannot specify Tau with V or Sd.

| | |
|---|---|
| `Sd` | character or numeric: If Sd is a character, it is interpreted as the names of the linear predictors associated with the standard deviation parameter of the Normal distribution. If Sd is numeric, the standard deviation is considered known and equal to the value of Sd, otherwise, the precision is considered unknown and equal to the exponential of the linear predictor informed by in Sd. If the outcome is a Multivariate Normal, then Tau must be a matrix and the elements outside its main diagonal are treated as the correlation (or the name of the linear predictor associated) between each coordinate of the outcome. The user cannot specify Sd with V or Tau. |
| `data` | numeric: Values of the observed data. |

## Details

If V/Sigma/Tau/Sd is a string, we use the method proposed in Alves et al. (2024). Otherwise, if V/Sigma/Tau/Sd is numeric, we follow the theory presented in West and Harrison (1997).

For the details about the implementation see dos Santos et al. (2024).

## Value

A object of the class dlm_distr

## References

Mariane Branco Alves, Helio S. Migon, Raíra Marotta, Junior, Silvaneo Vieira dos Santos (2024). "k-parametric Dynamic Generalized Linear Models: a sequential approach via Information Geometry." 2201.05387.

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

`fit_model`

Other auxiliary functions for a creating outcomes: `Gamma()`, `Multinom()`, `Poisson()`, `summary.dlm_distr()`

## Examples

```
# Univariate Normal case
structure <- polynomial_block(mu = 1, D = 0.95) +
  polynomial_block(V = 1, D = 0.95)

outcome <- Normal(mu = "mu", V = "V", data = cornWheat$corn.log.return[1:500])
fitted.data <- fit_model(structure, corn = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")
```

```
# Bivariate Normal case
structure <- (polynomial_block(mu = 1, D = 0.95) +
  polynomial_block(V = 1, D = 0.95)) * 2 +
  polynomial_block(rho = 1, D = 0.95)

outcome <- Normal(
  mu = c("mu.1", "mu.2"),
  V = matrix(c("V.1", "rho", "rho", "V.2"), 2, 2),
  data = cornWheat[1:500, c(4, 5)]
)
fitted.data <- fit_model(structure, cornWheat = outcome)
summary(fitted.data)
plot(fitted.data, plot.pkg = "base")
```

---

`noticeSARI` *SARI data from Belo Horizonte*

---

### Description

A dataset containing reports from Severe Acute Respiratory Illness (SARI) from 2020 to April 2022 by week.

### Usage

```
noticeSARI
```

### Format

A data frame with 65404 rows and 7 variables:

**ref.week** The reference week, counting since the monitoring begun.

**reported.1.week** The number of cases occurred in the period and reported until the 1 week after the reference week.

**reported.2.week** The number of cases occurred in the period and reported until the 2 weeks after the reference week.

**reported.4.week** The number of cases occurred in the period and reported until the 4 weeks after the reference week.

**reported.6.week** The number of cases occurred in the period and reported until the 6 weeks after the reference week.

**reported.8.week** The number of cases occurred in the period and reported until the 8 weeks after the reference week.

**reported.12.week** The number of cases occurred in the period and reported until the 12 weeks after the reference week.

**occured** The total number of cases reported (at any time). ...

### Source

https://datasus.saude.gov.br/informacoes-de-saude-tabnet/

## plot.dlm_coef — *Visualizing latent states in a fitted kDGLM model*

### Description

Visualizing latent states in a fitted kDGLM model

### Usage

```
## S3 method for class 'dlm_coef'
plot(
  x,
  var = rownames(x$theta.mean)[x$dynamic],
  cutoff = floor(t/10),
  pred.cred = 0.95,
  plot.pkg = "auto",
  ...
)
```

### Arguments

| | |
|---|---|
| x | dlm_coef object: The coefficients of a fitted DGLM model. |
| var | character: The name of the variables to plot (same value passed while creating the structure). Any variable whose name partially match this variable will be plotted. |
| cutoff | integer: The number of initial steps that should be skipped in the plot. Usually, the model is still learning in the initial steps, so the estimated values are not reliable. |
| pred.cred | numeric: The credibility value for the credibility interval. |
| plot.pkg | character: A flag indicating if a plot should be produced. Should be one of 'auto', 'base', 'ggplot2' or 'plotly'. |
| ... | Extra arguments passed to the plot method. |

### Value

ggplot or plotly object: A plot showing the predictive mean and credibility interval with the observed data.

### See Also

fit_model,coef

Other auxiliary visualization functions for the fitted_dlm class: plot.fitted_dlm(), summary.fitted_dlm(), summary.searched_dlm()

## Examples

```
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, order = 2, D = 0.95)
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

outcome <- Poisson(lambda = "rate", data)

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
)

model.coef <- coef(fitted.data)

plot(model.coef)$plot
```

---

`plot.fitted_dlm` *Visualizing a fitted kDGLM model*

---

## Description

Calculate the predictive mean and some quantile for the observed data and show a plot.

## Usage

```
## S3 method for class 'fitted_dlm'
plot(
  x,
  outcomes = NULL,
  latent.states = NULL,
  linear.predictors = NULL,
  pred.cred = 0.95,
  lag = NA,
  cutoff = floor(x$t/10),
  plot.pkg = "auto",
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | fitted_dlm object: A fitted DGLM. |
| `outcomes` | character: The name of the outcomes to plot. |
| `latent.states` | character: The name of the latent states to plot. |
| `linear.predictors` | character: The name of the linear predictors to plot. |
| `pred.cred` | numeric: The credibility value for the credibility interval. |

| | |
|---|---|
| `lag` | integer: The number of steps ahead to be used for prediction. If lag<0, the smoothed distribution is used and, if lag==0, the filtered interval.score is used. |
| `cutoff` | integer: The number of initial steps that should be skipped in the plot. Usually, the model is still learning in the initial steps, so the predictions are not reliable. |
| `plot.pkg` | character: A flag indicating if a plot should be produced. Should be one of 'auto', 'base', 'ggplot2' or 'plotly'. |
| `...` | Extra arguments passed to the plot method. |

## Value

ggplot or plotly object: A plot showing the predictive mean and credibility interval with the observed data.

## See Also

`fit_model`

Other auxiliary visualization functions for the fitted_dlm class: `plot.dlm_coef()`, `summary.fitted_dlm()`, `summary.searched_dlm()`

## Examples

```
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, order = 2, D = 0.95)
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

outcome <- Poisson(lambda = "rate", data)

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
)

plot(fitted.data, plot.pkg = "base")
```

---

# `Poisson` *Poisson outcome for kDGLM models*

---

## Description

Creates an outcome with Poisson distribution with the chosen parameter.

## Usage

```
Poisson(lambda, data, offset = as.matrix(data)^0)
```

## Arguments

lambda
: character: The name of the linear predictor associated with the rate (mean) parameter of the Poisson distribution. The parameter is treated as unknown and equal to the exponential of the associated linear predictor.

data
: numeric: The values of the observed data.

offset
: numeric: The offset at each observation. Must have the same shape as data.

## Details

For evaluating the posterior parameters, we use the method proposed in Alves et al. (2024).

For the details about the implementation see dos Santos et al. (2024).

## Value

A object of the class dlm_distr

## References

Mariane Branco Alves, Helio S. Migon, Raíra Marotta, Junior, Silvaneo Vieira dos Santos (2024). "k-parametric Dynamic Generalized Linear Models: a sequential approach via Information Geometry." 2201.05387.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

fit_model

Other auxiliary functions for a creating outcomes: Gamma(), Multinom(), Normal(), summary.dlm_distr()

## Examples

```
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, D = 0.95, order = 2)
season <- harmonic_block(rate = 1, period = 12, D = 0.975)

outcome <- Poisson(lambda = "rate", data = data)

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
)
summary(fitted.data)

plot(fitted.data, plot.pkg = "base")
```

`polynomial_block` *Structural blocks for polynomial trends and regressions*

## Description

Creates the structure for a polynomial block with desired order.

## Usage

```
polynomial_block(
  ...,
  order = 1,
  name = "Var.Poly",
  D = 1,
  h = 0,
  H = 0,
  a1 = 0,
  R1 = c(9, rep(1, order - 1)),
  monitoring = c(TRUE, rep(FALSE, order - 1))
)

pol(order = 1, D = 0.95, a1 = 0, R1 = 9, name = "Var.Poly", X = 1)
```

## Arguments

| | |
|---|---|
| `...` | Named values for the planning matrix. |
| `order` | Positive integer: The order of the polynomial structure. |
| `name` | String: An optional argument providing the name for this block. Can be useful to identify the models with meaningful labels, also, the name used will be used in some auxiliary functions. |
| `D` | Array, Matrix, vector or scalar: The values for the discount factors associated with the latent states at each time. If D is an array, its dimensions should be n x n x t, where n is the order of the polynomial block and t is the length of the outcomes. If D is a matrix, its dimensions should be n x n and the same discount matrix will be used in all observations. If D is a vector, it should have size t and it is interpreted as the discount factor at each observed time (same discount for all variable). If D is a scalar, the same discount will be used for all latent states at all times. |
| `h` | Matrix, vector or scalar: A drift to be add after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a matrix, its dimension should be n x t, where n is the number of latent states (i.e., the order) and t is the length of the series. If a vector, it should have size t, and each value will be applied to the first latent state (the one which affects the linear predictors) in their respective time. If a scalar, the passed value will be used for the first latent state at each time. |

| | |
|---|---|
| H | Array, Matrix, vector or scalar: The values for the covariance matrix for the noise factor at each time. If H is an array, its dimensions should be n x n x t, where n is the order of the polynomial block and t is the length of the series. If H is a matrix, its dimensions should be n x n and its values will be used for each time. If H is a vector or scalar, a discount factor matrix will be created as a diagonal matrix with the values of H in the diagonal. |
| a1 | Vector or scalar: The prior mean for the latent states associated with this block at time 1. If a1 is a vector, its dimension should be equal to the order of the polynomial block. If a1 is a scalar, its value will be used for all latent states. |
| R1 | Matrix, vector or scalar: The prior covariance matrix for the latent states associated with this block at time 1. If R1 is a matrix, its dimensions should be n x n. If R1 is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1 in the diagonal. |
| monitoring | Vector: A vector of flags indicating which variables should be monitored (if automated monitoring is used). Its size should be n. The default is that only the first order component of this structure should be monitored. |
| X | Vector or scalar: An argument providing the values of the covariate X_t. |

## Details

For the ..., D, H, a1 and R1 arguments, the user may set one or more of its values as a string. By doing so, the user will leave the block partially undefined. The user must then pass the undefined parameter values as named arguments to the `fit_model` function. Also, multiple values can be passed, allowing for a sensitivity analysis for the value of this parameter.

For the details about the implementation see dos Santos et al. (2024).

For the details about polynomial trend in the context of DLM's, see West and Harrison (1997), chapter 7.

For the details about dynamic regression models in the context of DLM's, see West and Harrison (1997), chapters 6 and 9.

## Value

A dlm_block object containing the following values:

- FF Array: A 3D-array containing the regression matrix for each time. Its dimension should be n x k x t, where n is the number of latent states, k is the number of linear predictors in the model and t is the time series length.
- FF.labs Matrix: A n x k character matrix describing the type of value of each element of FF.
- G Matrix: A 3D-array containing the evolution matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- G.labs Matrix: A n x n character matrix describing the type of value of each element of G.
- G.idx Matrix: A n x n character matrix containing the index each element of G.
- D Array: A 3D-array containing the discount factor matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- h Matrix: The mean for the random noise of the temporal evolution. Its dimension should be n x t.

- H Array: A 3D-array containing the covariance matrix of the noise for each time. Its dimension should be the same as D.
- a1 Vector: The prior mean for the latent vector.
- R1 Matrix: The prior covariance matrix for the latent vector.
- var.names list: A list containing the variables indexes by their name.
- order Positive integer: Same as argument.
- n Positive integer: The number of latent states associated with this block (same value as order).
- t Positive integer: The number of time steps associated with this block. If 1, the block is compatible with blocks of any time length, but if t is greater than 1, this block can only be used with blocks of the same time length.
- k Positive integer: The number of outcomes associated with this block. This block can only be used with blocks with the same outcome length.
- pred.names Vector: The name of the linear predictors associated with this block.
- monitoring Vector: Same as argument.
- type Character: The type of block (polynomial).

## References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

`fit_model`

Other auxiliary functions for structural blocks: `TF_block`(), `block_mult`(), `block_rename`(), `block_superpos`(), `ffs_block`(), `harmonic_block`(), `intervention`(), `noise_block`(), `regression_block`(), `specify.dlm_block`(), `summary.dlm_block`()

## Examples

```
# Creating a first order structure for a model with 2 outcomes.
# One block is created for each outcome
# with each block being associated with only one of the outcomes.
level.1 <- polynomial_block(alpha1 = 1, order = 1)
level.2 <- polynomial_block(alpha2 = 1, order = 1)

# Creating a block with shared effect between the outcomes
level.3 <- polynomial_block(alpha1 = 1, alpha2 = 1, order = 2)
```

---

`regression_block` *Structural blocks for regressions*

---

### Description

Creates a block for a (dynamic) regression for a covariate X_t.

### Usage

```
regression_block(
  ...,
  max.lag = 0,
  zero.fill = TRUE,
  name = "Var.Reg",
  D = 1,
  h = 0,
  H = 0,
  a1 = 0,
  R1 = 9,
  monitoring = rep(FALSE, max.lag + 1)
)

reg(
  X,
  max.lag = 0,
  zero.fill = TRUE,
  D = 0.95,
  a1 = 0,
  R1 = 9,
  name = "Var.Reg"
)
```

### Arguments

| | |
|---|---|
| `...` | Named values for the planning matrix. |
| `max.lag` | Non-negative integer: An optional argument providing the maximum lag for the explanatory variables. If a positive value is provided, this block will create additional latent states to measure the lagged effect of X_t up until the given value. See West and Harrison (1997), subsection 9.2.2 item (3). |
| `zero.fill` | boolean: A Boolean indicating if the block should fill the initial delay values with 0's. If TRUE and max.lag is positive, the block assumes that X_t=0 for all t<1. If FALSE, the block assumes the user will provide X_t for all t, such that X_t will have size t+propagation_size |
| `name` | String: An optional argument providing the name for this block. Can be useful to identify the models with meaningful labels, also, the name used will be used in some auxiliary functions. |

D — Array, Matrix, vector or scalar: The values for the discount factors at each time. If D is a array, its dimensions should be n x n x t, where n is the order of the polynomial block and t is the length of the outcomes. If D is a matrix, its dimensions should be n x n and its values will be used for each time. If D is a vector or scalar, a discount factor matrix will be created as a diagonal matrix with the values of D in the diagonal.

h — Matrix, vector or scalar: A drift to be add after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a matrix, its dimension should be 2 x t, where t is the length of the series. If a vector, it should have size t, and each value will be applied to the first latent state (the one which affects the linear predictors) in their respective time. If a scalar, the passed value will be used for the first latent state at each time.

H — Array, Matrix, vector or scalar: The values for the covariance matrix for the noise factor at each time. If H is a array, its dimensions should be n x n x t, where n is the order of the polynomial block and t is the length of the outcomes. If H is a matrix, its dimensions should be n x n and its values will be used for each time. If H is a vector or scalar, a discount factor matrix will be created as a diagonal matrix with the values of H in the diagonal.

a1 — Vector or scalar: The prior mean for the latent states associated with this block at time 1. If a1 is a vector, its dimension should be equal to the order of the polynomial block. If a1 is a scalar, its value will be used for all latent states.

R1 — Matrix, vector or scalar: The prior covariance matrix for the latent states associated with this block at time 1. If R1 is a matrix, its dimensions should be n x n. If R1 is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1 in the diagonal.

monitoring — Vector: A vector of flags indicating which variables should be monitored (if automated monitoring is used). Its size should be n. The default is that no variable should be monitored.

X — Vector or scalar: An argument providing the values of the covariate X_t.

## Details

For the ..., D, H, a1 and R1 arguments, the user may set one or more of its values as a string. By doing so, the user will leave the block partially undefined. The user must then pass the undefined parameter values as named arguments to the `fit_model` function. Also, multiple values can be passed, allowing for a sensitivity analysis for the value of this parameter.

For the details about the implementation see dos Santos et al. (2024).

For the details about dynamic regression models in the context of DLM's, see West and Harrison (1997), chapters 6 and 9.

## Value

A dlm_block object containing the following values:

- FF Array: A 3D-array containing the regression matrix for each time. Its dimension should be n x k x t, where n is the number of latent states, k is the number of linear predictors in the model and t is the time series length.

- FF.labs Matrix: A n x k character matrix describing the type of value of each element of FF.
- G Matrix: A 3D-array containing the evolution matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- G.labs Matrix: A n x n character matrix describing the type of value of each element of G.
- G.idx Matrix: A n x n character matrix containing the index each element of G.
- D Array: A 3D-array containing the discount factor matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- h Matrix: The mean for the random noise of the temporal evolution. Its dimension should be n x t.
- H Array: A 3D-array containing the covariance matrix of the noise for each time. Its dimension should be the same as D.
- a1 Vector: The prior mean for the latent vector.
- R1 Matrix: The prior covariance matrix for the latent vector.
- var.names list: A list containing the variables indexes by their name.
- max.lag Positive integer: Same as argument.
- n Positive integer: The number of latent states associated with this block (2).
- t Positive integer: The number of time steps associated with this block. If 1, the block is compatible with blocks of any time length, but if t is greater than 1, this block can only be used with blocks of the same time length.
- k Positive integer: The number of outcomes associated with this block. This block can only be used with blocks with the same outcome length.
- pred.names Vector: The name of the linear predictors associated with this block.
- monitoring Vector: Same as argument.
- type Character: The type of block (Harmonic).

## References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

`fit_model`

Other auxiliary functions for structural blocks: `TF_block()`, `block_mult()`, `block_rename()`, `block_superpos()`, `ffs_block()`, `harmonic_block()`, `intervention()`, `noise_block()`, `polynomial_block()`, `specify.dlm_block()`, `summary.dlm_block()`

### Examples

```
structure <- (
  polynomial_block(p = 1, order = 2, D = 0.95) +
    harmonic_block(p = 1, period = 12, D = 0.95) +
    regression_block(p = chickenPox$date >= as.Date("2013-09-01"))
  # Vaccine was introduced in September of 2013
) * 4

outcome <- Multinom(p = structure$pred.names, data = chickenPox[, c(2, 3, 4, 6, 5)])
fitted.data <- fit_model(structure, chickenPox = outcome)
summary(fitted.data)
plot(coef(fitted.data), plot.pkg = "base")
```

---

## simulate.fitted_dlm *Draw samples from the distribution of the latent states*

---

### Description

This is function draws samples from the latent states using the backward sampling algorithm. See West and Harrison (1997), chapter 15, for details.

### Usage

```
## S3 method for class 'fitted_dlm'
simulate(object, nsim, seed = NULL, lag = -1, ...)
```

### Arguments

| | |
|---|---|
| `object` | fitted_dlm: A fitted model from which to sample. |
| `nsim` | integer: The number of samples to draw. |
| `seed` | integer: An object specifying if and how the random number generator should be initialized. |
| `lag` | integer: The relative offset for forecast. Values for time t will be calculated based on the filtered values of time t-h. If lag is negative, then the smoothed distribution for the latent states will be used. |
| `...` | Extra arguments passed to the plot method. |

### Value

A list containing the following values:

- theta array: An array containing a sample of the latent states. Dimensions are n x t x nsim, where n is the number of latent states in the model and t is the number of observed values.
- lambda array: An array containing a sample of the linear predictors. Dimensions are k x t x nsim, where k is the number of linear predictors in the model and t is the number of observed values.

- param list: A named list containing, for each model outcome, an array with the samples of the parameters of the observational model. Each array will have dimensions l x t x nsim, where l is the number of parameters in the observational model and t is the number of observed values.

### See Also

Other auxiliary functions for fitted_dlm objects: coef.fitted_dlm(), eval_dlm_norm_const(), fit_model(), forecast.fitted_dlm(), kdglm(), smoothing(), update.fitted_dlm()

### Examples

```
structure <- polynomial_block(mu = 1, D = 0.95) +
  polynomial_block(V = 1, D = 0.95)

outcome <- Normal(mu = "mu", V = "V", data = cornWheat$corn.log.return[1:500])
fitted.data <- fit_model(structure, corn = outcome)

sample <- simulate(fitted.data, 5000)
```

---

## smoothing — *Auxiliary function for model smoothing*

---

### Description

Auxiliary function for model smoothing

### Usage

```
smoothing(model)
```

### Arguments

| | |
|---|---|
| model | A fitted_dlm object. |

### Value

A fitted_dlm object with smoothed means (mts) and covariance matrix (Cts) for each observation.

### See Also

Other auxiliary functions for fitted_dlm objects: coef.fitted_dlm(), eval_dlm_norm_const(), fit_model(), forecast.fitted_dlm(), kdglm(), simulate.fitted_dlm(), update.fitted_dlm()

specify.dlm_block *Specify method for dlm blocks*

## Description

Sets the values of undefined parameters in a block to those passed by the user.

## Usage

```
## S3 method for class 'dlm_block'
specify(x, ...)
```

## Arguments

x dlm_block: A undefined dlm_block object from which the undefined parameters shall be substituted.

... A set of named values for each unknown parameter.

## Value

The initual block, but with the undefined parameters set to the chosen values.

## See Also

Other auxiliary functions for structural blocks: TF_block(), block_mult(), block_rename(), block_superpos(), ffs_block(), harmonic_block(), intervention(), noise_block(), polynomial_block(), regression_block(), summary.dlm_block()

## Examples

```
season <- harmonic_block(rate = 1, period = 12, D = "D.sazo") |>
  specify(D.sazo = 0.975)
```

summary.fitted_dlm *Summary for a fitted kDGLM model*

## Description

Prints a report for a fitted_dlm object.

### Usage

```
## S3 method for class 'fitted_dlm'
summary(
  object,
  t = object$t,
  lag = -1,
  metric.lag = 1,
  metric.cutoff = floor(object$t/10),
  pred.cred = 0.95,
  ...
)
```

### Arguments

| | |
|---|---|
| `object` | A fitted_dlm object. |
| `t` | Integer: The time index for the latent states. |
| `lag` | Integer: The number of steps ahead used for the evaluating the latent states. Use lag<0 for the smoothed distribution, If lag==0 for the filtered distribution and lag=h for the h-step-ahead prediction. |
| `metric.lag` | Integer: The number of steps ahead used for the evaluating the predictions used when calculating metrics. Use metric.lag<0 for the smoothed distribution, If metric.lag==0 for the filtered distribution and metric.lag=h for the h-step-ahead prediction. |
| `metric.cutoff` | Integer: The cutoff time index for the metric calculation. Values before that time will be ignored. |
| `pred.cred` | numeric: The credibility interval to be used for the interval score. |
| `...` | Extra arguments passed to the coef method.#' |

### Value

No return value, called to print a summary of the fitted kDGLM model.

### See Also

Other auxiliary visualization functions for the fitted_dlm class: `plot.dlm_coef()`, `plot.fitted_dlm()`, `summary.searched_dlm()`

### Examples

```
data <- c(AirPassengers)

level <- polynomial_block(rate = 1, order = 2, D = 0.95)
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

outcome <- Poisson(lambda = "rate", data)

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
```

```
)
summary(fitted.data)
```

---

TF_block *Structural blocks for auto regressive trends and regressions*

---

## Description

Creates the structure for a Auto Regressive (AR) block (see West and Harrison (1997), chapter 9) with desired order. As the package suppose that the structure of the model is linear, a linearization is applied to the evolution equation, as described in West and Harrison (1997), chapter 13. This block also supports Transfer Functions, being necessary to specify the associated pulse when calling the TF_block function (see arg.).

## Usage

```
TF_block(
  ...,
  order,
  noise.var = NULL,
  noise.disc = NULL,
  pulse = 0,
  name = "Var.AR",
  AR.support = "free",
  h = 0,
  a1 = 0,
  R1 = 4,
  monitoring = TRUE,
  multi.states = FALSE,
  D.coef = 1,
  h.coef = 0,
  H.coef = 0,
  a1.coef = c(1, rep(0, order - 1)),
  R1.coef = c(1, rep(0.25, order - 1)),
  monitoring.coef = rep(FALSE, order),
  D.pulse = 1,
  h.pulse = 0,
  H.pulse = 0,
  a1.pulse = 0,
  R1.pulse = 4,
  monitoring.pulse = FALSE
)

AR(
  order = 1,
  noise.var = NULL,
```

```
    noise.disc = NULL,
    a1 = 0,
    R1 = 9,
    a1.coef = c(1, rep(0, order - 1)),
    R1.coef = c(1, rep(0.25, order - 1)),
    name = "Var.AR",
    X = 1
  )

  TF(
    pulse,
    order = 1,
    noise.var = NULL,
    noise.disc = NULL,
    a1 = 0,
    R1 = 9,
    a1.coef = c(1, rep(0, order - 1)),
    R1.coef = c(1, rep(0.25, order - 1)),
    a1.pulse = 0,
    R1.pulse = 4,
    name = "Var.AR",
    X = 1
  )
```

## Arguments

| | |
|---|---|
| `...` | Named values for the planning matrix. |
| `order` | Positive integer: The order of the AR block. |
| `noise.var` | Non-negative scalar: The variance of the white noise added to the latent state. |
| `noise.disc` | Vector or scalar: The value for the discount factor associated with the current latent state. If noise.disc is a vector, it should have size t and it is interpreted as the discount factor at each observed time. If D is a scalar, the same discount will be used for all observation. |
| `pulse` | Vector or scalar: An optional argument providing the values for the pulse for a Transfer Function. Default is 0 (no Transfer Function). |
| `name` | String: An optional argument providing the name for this block. Can be useful to identify the models with meaningful labels, also, the name used will be used in some auxiliary functions. |
| `AR.support` | String: Either "constrained" or "free" (default). If AR.support is "constrained", then the AR coefficients will be forced to be on the interval (-1,1), otherwise, the coefficients will be unrestricted. Beware that, under no restriction on the coefficients, there is no guarantee that the estimated coefficients will imply in a stationary process, furthermore, if the order of the AR block is greater than 1. As such the restriction of the coefficients support is only available for AR blocks with order equal to 1. |
| `h` | Vector or scalar: A drift to be add in the states after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a vector, it |

should have size t, and each value will be applied in their respective time. If a scalar, the passed value will be used for all observations.

`a1` Vector or scalar: The prior mean for the states associated with this block at time 1. If a1 is a vector, its dimension should be equal to the order of the AR block. If a1 is a scalar, its value will be used for all coefficients.

`R1` Matrix, vector or scalar: The prior covariance matrix for the states associated with this block at time 1. If R1 is a matrix, its dimensions should be n x n, where n is the order of the AR block. If R1 is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1 in the diagonal.

`monitoring` bool: A flag indicating if the latent state should be monitored (if automated monitoring is used). The default is TRUE.

`multi.states` bool: If FALSE (default) a single latent state will be created affecting all linear predictor and being affected by all pulses. If TRUE, each linear predictor will have its own latent state, but all latent states will share the same AR coefficients and all pulse effects (each state will have its own pulse though).

`D.coef` Array, Matrix, vector or scalar: The values for the discount factors associated with the AR coefficients at each time. If D.coef is an array, its dimensions should be n x n x t, where n is the order of the AR block and t is the length of the outcomes. If D.coef is a matrix, its dimensions should be n x n and the same discount matrix will be used in all observations. If D.coef is a vector, it should have size t and it is interpreted as the discount factor at each observed time (same discount for all variable). If D.coef is a scalar, the same discount will be used for all AR coefficients at all times.

`h.coef` Matrix, vector or scalar: A drift to be add in the AR coefficients after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a matrix, its dimension should be n x t, where n is the order of the AR block and t is the length of the series. If a scalar, the passed value will be used for all coefficients at each time.

`H.coef` Array, Matrix, vector or scalar: The values for the covariance matrix for the noise factor associated with the AR coefficients at each time. If H.coef is a array, its dimensions should be n x n x t, where n is the order of the AR block and t is the length of the outcomes. If H.coef is a matrix, its dimensions should be n x n and its values will be used for each time. If H.coef is a vector or scalar, a discount factor matrix will be created as a diagonal matrix with the values of H.coef in the diagonal.

`a1.coef` Vector or scalar: The prior mean for the AR coefficients associated with this block at time 1. If a1.coef is a vector, its dimension should be equal to the order of the AR block. If a1.coef is a scalar, its value will be used for all coefficients. If the coefficients are restricted to the interval (-1,1), the a1.coef is interpreted as the mean for atanh(rho), where rho is the AR coefficient.

`R1.coef` Matrix, vector or scalar: The prior covariance matrix for the coefficients associated with this block at time 1. If R1.coef is a matrix, its dimensions should be n x n, where n is the order of the AR block. If R1.coef is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1.coef in the diagonal. If the coefficients are restricted to the interval (-1,1), the R1.coef is interpreted as the covariance matrix for atanh(rho), where rho is the AR coefficient.

`monitoring.coef`
: Vector: A vector of flags indicating which AR coefficients should be monitored (if automated monitoring is used). Its size should be n, where n is the order of the AR block. The default is that no coefficient should be monitored.

`D.pulse`
: Array, Matrix, vector or scalar: The values for the discount factors associated with the pulse coefficients at each time. If D.pulse is an array, its dimensions should be n x n x t, where n is the number of pulses and t is the length of the outcomes. If D.pulse is a matrix, its dimensions should be n x n and the same discount matrix will be used in all observations. If D.pulse is a vector, it should have size t and it is interpreted as the discount factor at each observed time (same discount for all variable). If D is a scalar, the same discount will be used for all pulse coefficients at all times.

`h.pulse`
: Matrix, vector or scalar: A drift to be add in the pulse effect after the temporal evolution (can be interpreted as the mean of the random noise at each time). If a matrix, its dimension should be n x t, where n is the number of pulses and t is the length of the series. If a scalar, the passed value will be used for all latent state at each time.

`H.pulse`
: Array, Matrix, vector or scalar: The values for the covariance matrix for the noise factor associated with pulse coefficients at each time. If H.pulse is an array, its dimensions should be n x n x t, where n is the number of pulses and t is the length of the outcomes. If H.pulse is a matrix, its dimensions should be n x n and its values will be used for each time. If H.pulse is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of H.pulse in the diagonal.

`a1.pulse`
: Vector or scalar: The prior mean for the coefficients associated with the pulses at time 1. If a1.pulse is a vector, its dimension should be equal to the number of pulses. If a1.pulse is a scalar, its value will be used for all coefficients.

`R1.pulse`
: Matrix, vector or scalar: The prior covariance matrix for the coefficients associated with the pulses at time 1. If R1.pulse is a matrix, its dimensions should be n x n, where n is the number of pulses. If R1.pulse is a vector or scalar, a covariance matrix will be created as a diagonal matrix with the values of R1.pulse in the diagonal.

`monitoring.pulse`
: Vector: A vector of flags indicating which pulse coefficients should be monitored (if automated monitoring is used). Its size should be n, where n is the number of pulses. The default is that no pulse coefficient should be monitored.

`X`
: Vector or scalar: An argument providing the values for the pulse for a Transfer Function.

## Details

For the ..., noise.var, noise.disc, D, H, a1, R1, a1, R1, a1.pulse, R1.pulse, D.pulse, h.pulse, H.pulse arguments, the user may set one or more of its values as a string. By doing so, the user will leave the block partially undefined. The user must then pass the undefined parameter values as named arguments to the `fit_model` function. Also, multiple values can be passed, allowing for a sensitivity analysis for the value of this parameter.

For the details about the implementation see dos Santos et al. (2024).

For the details about Auto regressive models in the context of DLM's, see West and Harrison (1997), chapter 9.

For the details about the linearization of non-linear evolution equations in the context of DLM's, see West and Harrison (1997), chapter 13.

For the details about dynamic regression models in the context of DLM's, see West and Harrison (1997), chapters 6 and 9.

## Value

A dlm_block object containing the following values:

- FF Array: A 3D-array containing the regression matrix for each time. Its dimension should be n x k x t, where n is the number of latent states, k is the number of linear predictors in the model and t is the time series length.
- FF.labs Matrix: A n x k character matrix describing the type of value of each element of FF.
- G Matrix: A 3D-array containing the evolution matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- G.labs Matrix: A n x n character matrix describing the type of value of each element of G.
- G.idx Matrix: A n x n character matrix containing the index each element of G.
- D Array: A 3D-array containing the discount factor matrix for each time. Its dimension should be n x n x t, where n is the number of latent states and t is the time series length.
- H Array: A 3D-array containing the covariance matrix of the noise for each time. Its dimension should be the same as D.
- a1 Vector: The prior mean for the latent vector.
- R1 Matrix: The prior covariance matrix for the latent vector.
- var.names list: A list containing the variables indexes by their name.
- order Positive integer: Same as argument.
- n Positive integer: The number of latent states associated with this block (2).
- t Positive integer: The number of time steps associated with this block. If 1, the block is compatible with blocks of any time length, but if t is greater than 1, this block can only be used with blocks of the same time length.
- k Positive integer: The number of outcomes associated with this block. This block can only be used with blocks with the same outcome length.
- pred.names Vector: The name of the linear predictors associated with this block.
- monitoring Vector: The combination of monitoring, monitoring and monitoring.pulse.
- type Character: The type of block (AR).
- AR.support Character: Same as argument.

## References

Mike West, Jeff Harrison (1997). *Bayesian Forecasting and Dynamic Models (Springer Series in Statistics)*. Springer-Verlag. ISBN 0387947256.

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

fit_model

Other auxiliary functions for structural blocks: block_mult(), block_rename(), block_superpos(), ffs_block(), harmonic_block(), intervention(), noise_block(), polynomial_block(), regression_block(), specify.dlm_block(), summary.dlm_block()

## Examples

```
#### AR block ####
TF_block(mu = 1, order = 2, noise.disc = 0.9)

#### Transfer function ####
TF_block(mu = 1, pulse = beaver1$activ, order = 1, noise.disc = 0.9)
```

---

# update.fitted_dlm *update.fitted_dlm*

---

## Description

update.fitted_dlm

## Usage

```
## S3 method for class 'fitted_dlm'
update(object, ...)
```

## Arguments

| | |
|---|---|
| object | fitted_dlm: The fitted model to be updated. |
| ... | Extra variables necessary for updating (covariates, observed values, etc.). |

## Details

If an a covariate is necessary for updating, it should be passed as a named argument. Its name must follow this structure: <block name>.Covariate<.index>. If there is only one pulse in the associated block the index is omitted. If an a pulse is necessary for updating, it should be passed as a named argument. Its name must follow this structure: <block name>.Pulse<.index>. If there is only one pulse in the associated block the index is omitted. If an offset is necessary for updating, it should be passed along with the observed data. See example.

## Value

A fitted_dlm object.

### See Also

Other auxiliary functions for fitted_dlm objects: `coef.fitted_dlm()`, `eval_dlm_norm_const()`, `fit_model()`, `forecast.fitted_dlm()`, `kdglm()`, `simulate.fitted_dlm()`, `smoothing()`

### Examples

```
level <- polynomial_block(rate = 1, order = 2, D = 0.95)
season <- harmonic_block(rate = 1, order = 2, period = 12, D = 0.975)

# Only first 100 observations (for the sake of the example)
outcome <- Poisson(lambda = "rate", data = c(AirPassengers)[1:100])

fitted.data <- fit_model(level, season,
  AirPassengers = outcome
)

updated.fit <- update(fitted.data, AirPassengers = list(data = c(AirPassengers)[101:144]))
# If a offset was present, the user should pass its value when updating
# updated.fit=update(fitted.data,
#                      AirPassengers=list(
#                       data=c(AirPassengers)[101:144],
#                       offset= ... ))
```

---

## zero_sum_prior    *Zero sum prior*

---

### Description

Defines the prior of a structural block to be such that the latent states sum zero with probability one.

### Usage

```
zero_sum_prior(
  block,
  var.index = 1:block$n,
  weights = rep(1, length(var.index))
)
```

### Arguments

| | |
|---|---|
| `block` | dlm_block object: The structural block. |
| `var.index` | integer: The index of the variables from which to set the prior. |
| `weights` | numeric: A vector indicating which linear transformation of the data is 0 with probability 1. Default is equivalent to a zero-sum restriction. |

## Details

The covariance matrix of the evolution and the drift parameter are also altered to guarantee that the zero sum condition will always hold. The discount factor must be the same for all variables whose prior is being modified. For the details about the implementation see dos Santos et al. (2024).

## Value

A dlm_block object with the desired prior.

## References

Junior, Silvaneo Vieira dos Santos, Mariane Branco Alves, Helio S. Migon (2024). "kDGLM: an R package for Bayesian analysis of Dynamic Generialized Linear Models."

## See Also

Other auxiliary functions for defining priors.: `CAR_prior()`, `joint_prior()`

## Examples

```
polynomial_block(mu = 1, D = 0.95) |>
  block_mult(5) |>
  zero_sum_prior()
```

# Index