Package ‘sparsesurv’

September 9, 2025

Title Forecasting and Early Outbreak Detection for Sparse Count Data

Version 0.1.1

Description Functions for fitting, forecasting, and early detection of outbreaks in sparse surveillance count time series. Supports negative binomial (NB), self-exciting NB, generalise autoregressive moving average (GARMA) NB , zero-inflated NB (ZINB), self-exciting ZINB, generalise autoregressive moving average ZINB, and hurdle formulations. Climatic and environmental covariates can be included in the regression component and/or the zero-modified components. Includes outbreak-detection algorithms for NB, ZINB, and hurdle models, with utilities for prediction and diagnostics.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1)

Imports R2jags, coda, stats

Suggests testthat (>= 3.0.0), knitr, rjags, rmarkdown, ggplot2, reshape2

Config/testthat/edition 3

SystemRequirements JAGS (>= 4.x)

URL `https://github.com/alexangelakis-ang/sparsesurv`

BugReports `https://github.com/alexangelakis-ang/sparsesurv/issues`

RoxygenNote 7.3.2

NeedsCompilation no

Author Alexandros Angelakis [aut, cre],
Bryan Nyawanda [aut],
Penelope Vounatsou [aut]

Maintainer Alexandros Angelakis `<alexandros.angelakis@swisstph.ch>`

Repository CRAN

Date/Publication 2025-09-09 14:10:02 UTC

Contents

`GARMA_NB` *Fit Negative Binomial GARMA Model with Prediction*

Description

This function fits a generalized autoregressive moving average (GARMA-NB) model for count data using a negative binomial distribution, and optionally generates posterior predictive counts for future covariate inputs.

Usage

```
GARMA_NB(
  cases,
  pop = NULL,
  covariates = NULL,
  p = 2,
  q = 2,
  c = 1,
  beta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("r", "beta", "phi", "theta"),
  covariatespred = NULL,
  poppred = NULL,
  casespred = NULL
)
```

Arguments

`cases`	Vector of observed counts (length N)
`pop`	Optional vector of population offsets (length N)
`covariates`	Optional numeric matrix (N x P) of covariates for the count component.
`p`	Integer, autoregressive order
`q`	Integer, moving average order
`c`	Constant added before log (default 1)
`beta_init`	Optional list of length n_chains for beta, count coefficients initial values.
`r_init`	Optional numeric vector of length n_chains for dispersion parameter.
`beta_prior_mean`	Mean for beta prior (default: 0)
`beta_prior_sd`	SD for beta prior (default: 10)
`r_prior_shape`	Shape for r ~ dgamma (default: 1)
`r_prior_rate`	Rate for r ~ dgamma (default: 1)
`n_iter`	Total MCMC iterations (default: 100000)
`n_burnin`	Burn-in iterations (default: 10000)
`n_chains`	Number of chains (default: 3)
`n_thin`	Thinning interval (default: 1)
`save_params`	Character vector of parameters to save (default c("beta","delta","r"))
`covariatespred`	Optional numeric matrix (M x P) of new covariates for count prediction.
`poppred`	Optional vector of population offsets (length M) for prediction.
`casespred`	Optional vector of true counts (length M) for prediction performance.

Value

A list with MCMC summary, samples, DIC, and if prediction data provided: pred_matrix, pred_mean, mae, rmse

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(2)
cases <- rnbinom(100, size = 5, mu = 8)  # toy NB series



# ---- actually fit the model, but only when JAGS is available ----

fit <- GARMA_NB(
  cases = cases,
  p = 1, q = 1,          #
  beta_prior_mean = 0,
  beta_prior_sd   = 5,
  r_prior_shape   = 2,
```

```
  r_prior_rate    = 0.5,
  n_iter   = 100,        # keep fast
  n_burnin = 10,
  n_chains = 1,
  n_thin   = 1
)
print(fit)


# ---- longer user-facing demo (skipped on checks) ----
# add a simple seasonal covariate and slightly higher orders
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  x <- sin(2*pi*seq_along(cases)/12)
  fit2 <- GARMA_NB(
    cases = cases,
    p = 2, q = 1,
    beta_prior_mean = 0,
    beta_prior_sd   = 5,
    r_prior_shape   = 2,
    r_prior_rate    = 0.5,
    n_iter   = 1000,
    n_burnin = 100,
    n_chains = 2,
    n_thin   = 2
  )
  print(fit2)
  # if a plot method exists:  # plot(fit2)
}


## Not run:
# ---- time-consuming / full demo ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit_full <- GARMA_NB(
    cases = cases,
    p = 2, q = 2,
    n_iter   = 100000,
    n_burnin = 10000,
    n_chains = 4,
    n_thin   = 5
  )
  print(fit_full)
}

## End(Not run)

if (interactive()) {
  # e.g., plot(fit)
}
```

`GARMA_ZINB` *Fit Zero-Inflated Negative Binomial GARMA Model with Prediction*

Description

This function fits a generalized autoregressive moving average (GARMA-ZINB) model for count data using a zero-inflated negative binomial distribution, allowing separate covariates for the count and zero-inflation parts, and optionally generates posterior predictive counts for future covariate inputs.

Usage

```
GARMA_ZINB(
  cases,
  pop = NULL,
  covariates_count = NULL,
  covariates_zero = NULL,
  p = 2,
  q = 2,
  c = 1,
  beta_init = NULL,
  delta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  delta_prior_mean = 0,
  delta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("r", "beta", "phi", "theta", "delta"),
  covariatespred_count = NULL,
  covariatespred_zero = NULL,
  poppred = NULL,
  casespred = NULL
)
```

Arguments

`cases`	Vector of observed counts (length N)
`pop`	Optional vector of population offsets (length N)
`covariates_count`	Optional numeric matrix (N x P) of covariates for the count component.

`covariates_zero`	Optional numeric matrix (N x Q) of covariates for the zero-inflation component.
`p`	Integer, autoregressive order
`q`	Integer, moving average order
`c`	Constant added before log (default 1)
`beta_init`	Optional list of length n_chains for beta, count coefficients initial values.
`delta_init`	Optional list of length n_chains for delta, zero-inflation coefficients.
`r_init`	Optional numeric vector of length n_chains for dispersion parameter.
`beta_prior_mean`	Mean for beta prior (default: 0)
`beta_prior_sd`	SD for beta prior (default: 10)
`delta_prior_mean`	Mean for delta prior (default: 0)
`delta_prior_sd`	SD for delta prior (default: 10)
`r_prior_shape`	Shape for r ~ dgamma (default: 1)
`r_prior_rate`	Rate for r ~ dgamma (default: 1)
`n_iter`	Total MCMC iterations (default: 100000)
`n_burnin`	Burn-in iterations (default: 10000)
`n_chains`	Number of chains (default: 3)
`n_thin`	Thinning interval (default: 1)
`save_params`	Character vector of parameters to save (default c("beta","delta","r"))
`covariatespred_count`	Optional numeric matrix (M x P) of new covariates for count prediction.
`covariatespred_zero`	Optional numeric matrix (M x Q) of new covariates for zero-inflation prediction.
`poppred`	Optional vector of population offsets (length M) for prediction.
`casespred`	Optional vector of true counts (length M) for prediction performance.

Value

A list with MCMC summary, samples, DIC, and if prediction data provided: pred_matrix, pred_mean, mae, rmse

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(3)
n <- 100
# toy NB counts with extra zeros to mimic zero-inflation
base <- rnbinom(n, size = 5, mu = 6)
zeros <- rbinom(n, size = 1, prob = 0.30)
cases <- ifelse(zeros == 1, 0L, base)
```

```
# ---- actually fit the model, but only when JAGS is available ----

fit <- GARMA_ZINB(
  cases = cases,
  p = 1, q = 1,          # rename if your args are ar_order/ma_order
  # keep priors at defaults unless you need to tweak
  n_iter   = 100,        # keep fast for examples
  n_burnin = 10,
  n_chains = 1,
  n_thin   = 1
)
print(fit)


# ---- longer user-facing demo (skipped on checks) ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  # simple seasonal covariate (use only if your function supports 'covariates')
  # x <- sin(2*pi*seq_along(cases)/12)
  fit2 <- GARMA_ZINB(
    cases = cases,
    p = 2, q = 1,
    # covariates = cbind(x),        # uncomment if supported
    # z_covariates = cbind(x),      # uncomment if zero-part covariates are supported
    n_iter   = 1000,
    n_burnin = 100,
    n_chains = 2,
    n_thin   = 2
  )
  print(fit2)
  # if a plot method exists:  # plot(fit2)
}


## Not run:
# ---- time-consuming / full demo ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit_full <- GARMA_ZINB(
    cases = cases,
    p = 2, q = 2,
    n_iter   = 100000,
    n_burnin = 10000,
    n_chains = 4,
    n_thin   = 1
  )
  print(fit_full)
}

## End(Not run)

if (interactive()) {
  # e.g., plot(fit)
}
```

NB *Fit Negative Binomial Model with Arbitrary Covariates*

Description

Fits a negative binomial (NB) model using JAGS, with an optional design matrix of covariates and full inprod for mean structure, and can generate posterior predictive counts for new covariate data.

Usage

```
NB(
  cases,
  pop = NULL,
  casespred = NULL,
  covariates = NULL,
  covariatespred = NULL,
  poppred = NULL,
  beta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("beta", "r")
)
```

Arguments

`cases`	Vector of observed counts (length N)
`pop`	Optional vector of population offsets (length N)
`casespred`	Optional vector of true counts (length M) for prediction performance.
`covariates`	Optional numeric matrix (N x P) of covariates for the count component.
`covariatespred`	Optional numeric matrix (M x P) of new covariates for count prediction.
`poppred`	Optional vector of population offsets (length M) for prediction.
`beta_init`	Optional list of length n_chains for beta, count coefficients initial values.
`r_init`	Optional numeric vector of length n_chains for dispersion parameter.
`beta_prior_mean`	Mean for beta prior (default: 0)

`beta_prior_sd`	SD for beta prior (default: 10)
`r_prior_shape`	Shape for r ~ dgamma (default: 1)
`r_prior_rate`	Rate for r ~ dgamma (default: 1)
`n_iter`	Total MCMC iterations (default: 100000)
`n_burnin`	Burn-in iterations (default: 10000)
`n_chains`	Number of chains (default: 3)
`n_thin`	Thinning interval (default: 1)
`save_params`	Character vector of parameters to save (default c("beta","delta","r"))

Value

A list with MCMC summary, samples, DIC, and if prediction data provided: prediction_matrix, prediction_mean, mae, rmse

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(4)
cases <- rnbinom(80, size = 5, mu = 7)  # toy NB series


# ---- actually fit the model, but only when JAGS is available ----

fit <- NB(
  cases = cases,
  # add pop = ... here if your NB function supports offsets
  beta_prior_mean = 0,
  beta_prior_sd   = 5,
  r_prior_shape   = 2,
  r_prior_rate    = 0.5,
  n_iter   = 400,           # keep fast
  n_burnin = 200,
  n_chains = 1,
  n_thin   = 2
)
print(fit)


# ---- longer user-facing demo (skipped on checks) ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  x <- sin(2*pi*seq_along(cases)/12)
  fit2 <- NB(
    cases = cases,
    covariates = cbind(x),  # simple seasonal regressor
    beta_prior_mean = 0,
    beta_prior_sd   = 5,
    r_prior_shape   = 2,
    r_prior_rate    = 0.5,
    n_iter   = 1000,
    n_burnin = 100,
```

```
    n_chains = 2,
    n_thin   = 2
  )
  print(fit2)
  # if a plot method exists:  # plot(fit2)
}


## Not run:
# ---- time-consuming / full demo ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit_full <- NB(
    cases = cases,
    covariates = cbind(x),  # simple seasonal regressor
    n_iter   = 100000,
    n_burnin = 10000,
    n_chains = 4,
    n_thin   = 5
  )
  print(fit_full)
}

## End(Not run)

if (interactive()) {
  # e.g., plot(fit)
}
```

`outbreakHNB` *Fit an outbreak detection Hurdle negative binomial outbreak model*

Description

Fit an outbreak detection Hurdle negative binomial outbreak model

Usage

```
outbreakHNB(
  cases,
  pop = NULL,
  covariates_count = NULL,
  covariates_zero = NULL,
  beta_init = NULL,
  delta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  delta_prior_mean = 0,
```

```
    delta_prior_sd = 10,
    r_prior_shape = 1,
    r_prior_rate = 1,
    p_priors = 1,
    n_iter = 1e+05,
    n_burnin = 10000,
    n_chains = 3,
    n_thin = 1,
    save_params = c("beta", "delta", "r", "Z"),
    dates = NULL,
    plot_Z = FALSE
)
```

Arguments

`cases`	Integer or numeric vector of observed case counts (length N).
`pop`	(Optional) Numeric vector of population offsets (length N). If NULL, offset = 1.
`covariates_count`	(Optional) Data.frame or matrix of covariates for the count model (N x p_c).
`covariates_zero`	(Optional) Data.frame or matrix of covariates for the Hurdle model (N x p_z).
`beta_init`	(Optional) List of length `n_chains` giving initial values for beta (each a vector of length p_c+1).
`delta_init`	(Optional) List of length `n_chains` giving initial values for delta (each a vector of length p_z+1).
`r_init`	(Optional) Numeric vector of length `n_chains` giving initial values for the NB dispersion parameter.
`beta_prior_mean`	Prior mean for beta coefficients of the Negative binomial part (default = 0).
`beta_prior_sd`	Prior SD for beta coefficients of the Negative binomial part (default = 10).
`delta_prior_mean`	Prior mean for delta coefficients of the Hurdle part (default = 0).
`delta_prior_sd`	Prior SD for delta coefficients of the Hurdle part (default = 10).
`r_prior_shape`	Shape parameter of a prior on r (default = 1).
`r_prior_rate`	Rate parameter of b prior on r (default = 1).
`p_priors`	Alpha parameters for the binomial priors on p00 and p11 (default = 1).
`n_iter`	Total number of MCMC iterations per chain (default = 100000).
`n_burnin`	Number of burn-in iterations (default = 10000).
`n_chains`	Number of MCMC chains (default = 3).
`n_thin`	Thinning interval for MCMC samples (default = 1).
`save_params`	Character vector of parameter names to save (must include "Z").
`dates`	(Optional) Vector of Date or POSIX dates for plotting Z; if NULL, uses index 1:N.
`plot_Z`	Logical; if TRUE, returns a ggplot2 object of the posterior mean Z over time.

Value

A list with MCMC summary, samples, DIC, WAIC, and plot of the probability of being in an epidemic state.

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(13)
n <- 120
# baseline hurdle-like series: generate NB counts then zero out some days
base  <- rnbinom(n, size = 6, mu = 8)
zeros <- rbinom(n, 1, 0.30)
cases <- ifelse(zeros == 1, 0L, base)
# inject a small "outbreak" window
cases[70:74] <- cases[70:74] + rnbinom(5, size = 6, mu = 25)
dates <- as.Date("2020-01-01") + seq_len(n) - 1L



# ---- actually run the detector, but only when JAGS is available ----


fit <- outbreakHNB(
  cases   = cases,
  dates   = dates,
  n_iter  = 10,   # keep fast for examples
  n_burnin= 1,
  n_chains= 1,
  n_thin  = 1,
  plot_Z  = FALSE  # avoid plotting in examples (rename/omit if not applicable)
)
print(fit)



# ---- longer user-facing demo (skipped on checks) ----
# Increase iterations a bit for a stabler run (still JAGS-gated by @examplesIf above)
# fit2 <- outbreakHNB(
#   cases   = cases,
#   dates   = dates,
#   n_iter  = 1500,
#   n_burnin= 500,
#   n_chains= 2,
#   n_thin  = 2,
#   plot_Z  = FALSE
# )
# print(fit2)


## Not run:
# ---- time-consuming / full demo (not run anywhere) ----
# Here you might use larger MCMC and produce figures/tables of alerts.
```

```
# fit_full <- outbreakHNB(
#   cases   = cases,
#   dates   = dates,
#   n_iter  = 10000,
#   n_burnin= 5000,
#   n_chains= 4,
#   n_thin  = 1,
#   plot_Z  = TRUE
# )
# print(fit_full)

## End(Not run)

if (interactive()) {
  # e.g., if a plot method exists:  # plot(fit)
}
```

`outbreakNB` *Fit an outbreak detection negative binomial outbreak model*

Description

Fit an outbreak detection negative binomial outbreak model

Usage

```
outbreakNB(
  cases,
  pop = NULL,
  covariates = NULL,
  beta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  p_priors = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("beta", "r", "Z"),
  dates = NULL,
  plot_Z = FALSE
)
```

Arguments

`cases`	Integer or numeric vector of observed case counts (length N).
`pop`	(Optional) Numeric vector of population offsets (length N). If NULL, offset = 1.
`covariates`	(Optional) Data.frame or matrix of covariates for the count model (N x p_c).
`beta_init`	(Optional) List of length `n_chains` giving initial values for beta (each a vector of length p_c+1).
`r_init`	(Optional) Numeric vector of length `n_chains` giving initial values for the NB dispersion parameter.
`beta_prior_mean`	Prior mean for beta coefficients of the Negative binomial part (default = 0).
`beta_prior_sd`	Prior SD for beta coefficients of the Negative binomial part (default = 10).
`r_prior_shape`	Shape parameter of a prior on r (default = 1).
`r_prior_rate`	Rate parameter of b prior on r (default = 1).
`p_priors`	Alpha parameters for the binomial priors on p00 and p11 (default = 1).
`n_iter`	Total number of MCMC iterations per chain (default = 100000).
`n_burnin`	Number of burn-in iterations (default = 10000).
`n_chains`	Number of MCMC chains (default = 3).
`n_thin`	Thinning interval for MCMC samples (default = 1).
`save_params`	Character vector of parameter names to save (must include "Z").
`dates`	(Optional) Vector of Date or POSIX dates for plotting Z; if NULL, uses index 1:N.
`plot_Z`	Logical; if TRUE, returns a ggplot2 object of the posterior mean Z over time.

Value

A list with MCMC summary, samples, DIC, WAIC, and plot of the probability of being in an epidemic state.

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(12)
n <- 120
# baseline NB counts with an injected ”outbreak” window
cases <- rnbinom(n, size = 6, mu = 8)
cases[70:74] <- cases[70:74] + rnbinom(5, size = 6, mu = 25)
dates <- as.Date(”2020-01-01”) + seq_len(n) - 1L



# ---- actually run the detector, but only when JAGS is available ----


fit <- outbreakNB(
```

```
  cases   = cases,
  dates   = dates,
  n_iter  = 10,   # keep fast for examples
  n_burnin= 1,
  n_chains= 1,
  n_thin  = 1,
  plot_Z  = FALSE  # avoid plotting in examples (rename/omit if not applicable)
)
print(fit)



# ---- longer user-facing demo (skipped on checks) ----
# Increase iterations a bit for a stabler run (still JAGS-gated by @examplesIf above)
# fit2 <- outbreakNB(
#   cases   = cases,
#   dates   = dates,
#   n_iter  = 1500,
#   n_burnin= 500,
#   n_chains= 2,
#   n_thin  = 2,
#   plot_Z  = FALSE
# )
# print(fit2)


## Not run:
# ---- time-consuming / full demo (not run anywhere) ----
# Here you might use larger MCMC and produce figures/tables of alerts.
# fit_full <- outbreakNB(
#   cases   = cases,
#   dates   = dates,
#   n_iter  = 10000,
#   n_burnin= 5000,
#   n_chains= 4,
#   n_thin  = 5,
#   plot_Z  = TRUE
# )
# print(fit_full)

## End(Not run)

if (interactive()) {
  # e.g., if a plot method exists:  # plot(fit)
}
```

`outbreakZINB` *Fit an outbreak detection zero-inflated negative binomial outbreak model*

Description

Fit an outbreak detection zero-inflated negative binomial outbreak model

Usage

```
outbreakZINB(
  cases,
  pop = NULL,
  covariates_count = NULL,
  covariates_zero = NULL,
  beta_init = NULL,
  delta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  delta_prior_mean = 0,
  delta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  p_priors = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("beta", "delta", "r", "Z"),
  dates = NULL,
  plot_Z = FALSE
)
```

Arguments

`cases`	Integer or numeric vector of observed case counts (length N).
`pop`	(Optional) Numeric vector of population offsets (length N). If NULL, offset = 1.
`covariates_count`	(Optional) Data.frame or matrix of covariates for the count model (N x p_c).
`covariates_zero`	(Optional) Data.frame or matrix of covariates for the zero-inflation model (N x p_z).
`beta_init`	(Optional) List of length `n_chains` giving initial values for beta (each a vector of length p_c+1).
`delta_init`	(Optional) List of length `n_chains` giving initial values for delta (each a vector of length p_z+1).
`r_init`	(Optional) Numeric vector of length `n_chains` giving initial values for the NB dispersion parameter.
`beta_prior_mean`	Prior mean for beta coefficients of the Negative binomial part (default = 0).

`beta_prior_sd`	Prior SD for beta coefficients of the Negative binomial part (default = 10).
`delta_prior_mean`	Prior mean for delta coefficients of the zero inflation part (default = 0).
`delta_prior_sd`	Prior SD for delta coefficients of the zero inflation part (default = 10).
`r_prior_shape`	Shape parameter of a prior on r (default = 1).
`r_prior_rate`	Rate parameter of b prior on r (default = 1).
`p_priors`	Alpha parameters for the binomial priors on p00 and p11 (default = 1).
`n_iter`	Total number of MCMC iterations per chain (default = 100000).
`n_burnin`	Number of burn-in iterations (default = 10000).
`n_chains`	Number of MCMC chains (default = 3).
`n_thin`	Thinning interval for MCMC samples (default = 1).
`save_params`	Character vector of parameter names to save (must include "Z").
`dates`	(Optional) Vector of Date or POSIX dates for plotting Z; if NULL, uses index 1:N.
`plot_Z`	Logical; if TRUE, returns a ggplot2 object of the posterior mean Z over time.

Value

A list with MCMC summary, samples, DIC, WAIC, and plot of the probability of being in an epidemic state.

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(11)
n <- 120
# baseline ZINB-like counts with an injected ”outbreak” window
base  <- rnbinom(n, size = 6, mu = 8)
zeros <- rbinom(n, 1, 0.25)
cases <- ifelse(zeros == 1, 0L, base)
dates <- as.Date(”2020-01-01”) + seq_len(n) - 1L



# ---- actually run the detector, but only when JAGS is available ----


fit <- outbreakZINB(
  cases   = cases,
  dates   = dates,
  n_iter  = 10,   # keep fast for examples
  n_burnin= 1,
  n_chains= 1,
  n_thin  = 1,
  plot_Z  = FALSE  # avoid plotting in examples
)
print(fit)
```

```
# ---- longer user-facing demo (skipped on checks) ----
# Increase iterations a bit for a stabler run (still JAGS-gated by @examplesIf above)
# fit2 <- outbreakZINB(
#   cases   = cases,
#   dates   = dates,
#   n_iter  = 1500,
#   n_burnin= 500,
#   n_chains= 2,
#   n_thin  = 2,
#   plot_Z  = FALSE
# )
# print(fit2)


## Not run:
# ---- time-consuming / full demo (not run anywhere) ----
# Here you might use larger MCMC and produce figures/tables of alerts.
# fit_full <- outbreakZINB(
#   cases   = cases,
#   dates   = dates,
#   n_iter  = 10000,
#   n_burnin= 5000,
#   n_chains= 4,
#   n_thin  = 5,
#   plot_Z  = TRUE
# )
# print(fit_full)

## End(Not run)

if (interactive()) {
  # e.g., if a plot method exists:  # plot(fit)
}
```

`SENB` *Fit Self-Exciting Negative Binomial Model with Prediction*

Description

Fits a self-exciting negative binomial (SE-NB) model using JAGS, with an optional design matrix of covariates and full inprod for mean structure, and can generate posterior predictive counts including self-excitation.

Usage

```
SENB(
```

```
  cases,
  pop = NULL,
  casesoldold = 0,
  covariates = NULL,
  covariatespred = NULL,
  poppred = NULL,
  casesoldpred = 0,
  casespred = NULL,
  beta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("beta", "r", "eta")
)
```

Arguments

`cases`	Vector of observed counts (length N)
`pop`	Optional vector of population offsets (length N)
`casesoldold`	Optional parameter of the cases of 1 timepoint previous than the start of timepoints fit.
`covariates`	Optional numeric matrix (N x P) of covariates for the count component.
`covariatespred`	Optional numeric matrix (M x P) of new covariates for count prediction.
`poppred`	Optional vector of population offsets (length M) for prediction.
`casesoldpred`	Optional parameter of the cases of 1 timepoint previous than the start of the prediction.
`casespred`	Optional vector of true counts (length M) for prediction performance.
`beta_init`	Optional list of length n_chains for beta, count coefficients initial values.
`r_init`	Optional numeric vector of length n_chains for dispersion parameter.
`beta_prior_mean`	Mean for beta prior (default: 0)
`beta_prior_sd`	SD for beta prior (default: 10)
`r_prior_shape`	Shape for r ~ dgamma (default: 1)
`r_prior_rate`	Rate for r ~ dgamma (default: 1)
`n_iter`	Total MCMC iterations (default: 100000)
`n_burnin`	Burn-in iterations (default: 10000)
`n_chains`	Number of chains (default: 3)
`n_thin`	Thinning interval (default: 1)
`save_params`	Character vector of parameters to save (default c("beta","r","eta"))

Value

A list with MCMC summary, samples, DIC, and if prediction data provided: prediction_matrix, prediction_mean, mae, rmse

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(1)
cases <- rnbinom(72, size = 5, mu = 8)  # toy NB series



# ---- actually fit the model, but only when JAGS is available ----

fit <- SENB(
  cases = cases,
  beta_prior_mean = 0,
  beta_prior_sd   = 5,
  r_prior_shape   = 2,
  r_prior_rate    = 0.5,
  n_iter   = 400,  # keep fast
  n_burnin = 200,
  n_chains = 1,
  n_thin   = 2
)
print(fit)


# ---- longer user-facing demo (skipped on checks) ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit2 <- SENB(
    cases = cases,
    beta_prior_mean = 0,
    beta_prior_sd   = 5,
    r_prior_shape   = 2,
    r_prior_rate    = 0.5,
    n_iter   = 1500,
    n_burnin = 500,
    n_chains = 2,
    n_thin   = 2
  )
  print(fit2)
}


## Not run:
# ---- time-consuming / full demo ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit_full <- SENB(
    cases = cases,
    beta_prior_mean = 0,
    beta_prior_sd   = 5,
```

```
    r_prior_shape   = 2,
    r_prior_rate    = 0.5,
    n_iter   = 10000,
    n_burnin = 5000,
    n_chains = 4,
    n_thin   = 5
  )
  print(fit_full)
}

## End(Not run)

if (interactive()) {
  # e.g., plot(fit)
}
```

`SEZINB` *Fit Self exciting Zero-Inflated Negative Binomial Model with Arbitrary Covariates and Prediction*

Description

Fits a Self exciting zero-inflated negative binomial (SE-ZINB) model using JAGS, with an optional design matrix of covariates and full inprod for mean structure, and can generate posterior predictive counts for new covariate data.

Usage

```
SEZINB(
  cases,
  pop = NULL,
  casesoldold = 0,
  covariates_count = NULL,
  covariates_zero = NULL,
  covariatespred_count = NULL,
  covariatespred_zero = NULL,
  poppred = NULL,
  casesoldpred = 0,
  casespred = NULL,
  beta_init = NULL,
  delta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  delta_prior_mean = 0,
  delta_prior_sd = 10,
  r_prior_shape = 1,
```

```
  r_prior_rate = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("beta", "delta", "r", "eta")
)
```

Arguments

`cases`	Vector of observed counts (length N)
`pop`	Optional vector of population offsets (length N)
`casesoldold`	Optional parameter of the cases of 1 timepoint previous than the start of timepoints fit.
`covariates_count`	Optional numeric matrix (N x P) of covariates for the count component.
`covariates_zero`	Optional numeric matrix (N x Q) of covariates for the zero-inflation component.
`covariatespred_count`	Optional numeric matrix (M x P) of new covariates for count prediction.
`covariatespred_zero`	Optional numeric matrix (M x Q) of new covariates for zero-inflation prediction.
`poppred`	Optional vector of population offsets (length M) for prediction.
`casesoldpred`	Optional parameter of the cases of 1 timepoint previous than the start of the prediction.
`casespred`	Optional vector of true counts (length M) for prediction performance.
`beta_init`	Optional list of length n_chains for beta, count coefficients initial values.
`delta_init`	Optional list of length n_chains for delta, zero-inflation coefficients.
`r_init`	Optional numeric vector of length n_chains for dispersion parameter.
`beta_prior_mean`	Mean for beta prior (default: 0)
`beta_prior_sd`	SD for beta prior (default: 10)
`delta_prior_mean`	Mean for delta prior (default: 0)
`delta_prior_sd`	SD for delta prior (default: 10)
`r_prior_shape`	Shape for r ~ dgamma (default: 1)
`r_prior_rate`	Rate for r ~ dgamma (default: 1)
`n_iter`	Total MCMC iterations (default: 100000)
`n_burnin`	Burn-in iterations (default: 10000)
`n_chains`	Number of chains (default: 3)
`n_thin`	Thinning interval (default: 1)
`save_params`	Character vector of parameters to save (default c("beta","delta","r"))

Value

A list with MCMC summary, samples, DIC, and if prediction data provided: pred_matrix, pred_mean, mae, rmse

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(6)
n <- 100
base  <- rnbinom(n, size = 5, mu = 6)
zeros <- rbinom(n, 1, 0.30)               # extra zeros to mimic zero-inflation
cases <- ifelse(zeros == 1, 0L, base)



# ---- actually fit the model, but only when JAGS is available ----

fit <- SEZINB(
  cases = cases,
  # keep priors at defaults; add them here only if your API requires
  n_iter   = 100,             # keep fast for examples
  n_burnin = 10,
  n_chains = 1,
  n_thin   = 1
)
print(fit)


# ---- longer user-facing demo (skipped on checks) ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  x <- sin(2*pi*seq_len(n)/12)           # simple seasonal regressor
  fit2 <- SEZINB(
    cases = cases,
covariates_count = cbind(x),
covariates_zero  = cbind(x),
    n_iter   = 10000,
    n_burnin = 500,
    n_chains = 2,
    n_thin   = 2
  )
  print(fit2)
  # if a plot method exists:  # plot(fit2)
}


## Not run:
# ---- time-consuming / full demo ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit_full <- SEZINB(
    cases = cases,
    n_iter   = 10000,
    n_burnin = 5000,
```

```
    n_chains = 4,
    n_thin   = 5
  )
  print(fit_full)
}

## End(Not run)

if (interactive()) {
  # e.g., plot(fit)
}
```

`ZINB` *Fit Zero-Inflated Negative Binomial Model with Arbitrary Covariates and Prediction*

Description

Fits a zero-inflated negative binomial (ZINB) model using JAGS, with an optional design matrix of covariates and full inprod for mean structure, and can generate posterior predictive counts for new covariate data.

Usage

```
ZINB(
  cases,
  pop = NULL,
  covariates_count = NULL,
  covariates_zero = NULL,
  covariatespred_count = NULL,
  covariatespred_zero = NULL,
  poppred = NULL,
  casespred = NULL,
  beta_init = NULL,
  delta_init = NULL,
  r_init = NULL,
  beta_prior_mean = 0,
  beta_prior_sd = 10,
  delta_prior_mean = 0,
  delta_prior_sd = 10,
  r_prior_shape = 1,
  r_prior_rate = 1,
  n_iter = 1e+05,
  n_burnin = 10000,
  n_chains = 3,
  n_thin = 1,
  save_params = c("beta", "delta", "r")
)
```

Arguments

`cases`	Vector of observed counts (length N)
`pop`	Optional vector of population offsets (length N)
`covariates_count`	Optional numeric matrix (N x P) of covariates for the count component.
`covariates_zero`	Optional numeric matrix (N x Q) of covariates for the zero-inflation component.
`covariatespred_count`	Optional numeric matrix (M x P) of new covariates for count prediction.
`covariatespred_zero`	Optional numeric matrix (M x Q) of new covariates for zero-inflation prediction.
`poppred`	Optional vector of population offsets (length M) for prediction.
`casespred`	Optional vector of true counts (length M) for prediction performance.
`beta_init`	Optional list of length n_chains for beta, count coefficients initial values.
`delta_init`	Optional list of length n_chains for delta, zero-inflation coefficients.
`r_init`	Optional numeric vector of length n_chains for dispersion parameter.
`beta_prior_mean`	Mean for beta prior (default: 0)
`beta_prior_sd`	SD for beta prior (default: 10)
`delta_prior_mean`	Mean for delta prior (default: 0)
`delta_prior_sd`	SD for delta prior (default: 10)
`r_prior_shape`	Shape for r ~ dgamma (default: 1)
`r_prior_rate`	Rate for r ~ dgamma (default: 1)
`n_iter`	Total MCMC iterations (default: 100000)
`n_burnin`	Burn-in iterations (default: 10000)
`n_chains`	Number of chains (default: 3)
`n_thin`	Thinning interval (default: 1)
`save_params`	Character vector of parameters to save (default c("beta","delta","r"))

Value

A list with MCMC summary, samples, DIC, and if prediction data provided: pred_matrix, pred_mean, mae, rmse

Examples

```
# ---- tiny example for users & CRAN (< 5s) ----
set.seed(5)
n <- 100
base  <- rnbinom(n, size = 5, mu = 7)
zeros <- rbinom(n, 1, 0.25)       # add extra zeros
cases <- ifelse(zeros == 1, 0L, base)
```

```
# ---- actually fit the model, but only when JAGS is available ----

fit <- ZINB(
  cases = cases,
  # optional priors if your API exposes them, e.g.:
  # beta_prior_mean = 0, beta_prior_sd = 5,
  # r_prior_shape = 2,  r_prior_rate  = 0.5,
  n_iter   = 400,            # keep fast
  n_burnin = 200,
  n_chains = 1,
  n_thin   = 2
)
print(fit)


# ---- longer user-facing demo (skipped on checks) ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  x <- sin(2*pi*seq_len(n)/12)       # simple seasonal regressor
  fit2 <- ZINB(
    cases = cases,
    covariates_count   = cbind(x),
    covariates_zero = cbind(x),
    n_iter   = 1000,
    n_burnin = 100,
    n_chains = 2,
    n_thin   = 2
  )
  print(fit2)
  # if a plot method exists:  # plot(fit2)
}


## Not run:
# ---- time-consuming / full demo ----
if (nzchar(Sys.which("jags")) && requireNamespace("R2jags", quietly = TRUE)) {
  fit_full <- ZINB(
    cases = cases,
    n_iter   = 100000,
    n_burnin = 10000,
    n_chains = 4,
    n_thin   = 5
  )
  print(fit_full)
}

## End(Not run)

if (interactive()) {
  # e.g., plot(fit)
}
```

Index