# Package: linelistBayes (via r-universe)

October 25, 2026

**Type** Package

**Title** Bayesian Analysis of Epidemic Data Using Line List and Case Count Approaches

**Version** 1.0

**Date** 2024-04-30

**Maintainer** Chad Milando <cmilando@bu.edu>

**Description** Provides tools for performing Bayesian inference on epidemiological data to estimate the time-varying reproductive number and other related metrics. These methods were published in Li and White (2021) <doi:10.1371/journal.pcbi.1009210>. This package supports analyses based on aggregated case count data and individual line list data, facilitating enhanced surveillance and intervention planning for infectious diseases like COVID-19.

**License** GPL-3

**Imports** Rcpp (>= 1.0.11), magrittr, lubridate, coda, dplyr

**LinkingTo** Rcpp, RcppArmadillo

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

**VignetteBuilder** knitr

**LazyData** true

**Depends** R (>= 2.10)

**Config/testthat/edition** 3

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** Chad Milando [aut, cre], Tenglong Li [ctb], Laura White [ctb]

**Repository** https://cran.r-universe.dev

**Date/Publication** 2024-05-04 02:33:23 UTC

**RemoteUrl** https://github.com/cran/linelistBayes

**RemoteRef** HEAD

**RemoteSha** bb66f1d37c071b2a84f86e11648cb139c838644f

# Contents

backnow_cm . . . 2
convert_to_linelist . . . 3
create_caseCounts . . . 5
create_linelist . . . 6
dnb . . . 7
dummy . . . 7
findmiss . . . 8
get_mu_vec . . . 9
getr . . . 10
lambda . . . 11
logLikNB . . . 12
plot.backnow . . . 13
plot.caseCounts . . . 14
pnb . . . 15
prop . . . 16
rnb . . . 17
run_backnow . . . 17
sample_cases . . . 19
si . . . 20

**Index** **22**

---

`backnow_cm` *Get Bayesian Back-calculation Estimates and Model Diagnostics*

---

## Description

This function performs Bayesian back-calculation, imputation of missing delays, and nowcasting based on the provided data.

## Usage

```
backnow_cm(
  outcome,
  days,
  week,
  weekend,
  iter,
  sigma,
  maxdelay,
  si,
  size,
  workerID,
  printProgress,
  cd = NULL
)
```

### Arguments

| | |
|---|---|
| outcome | Vector of outcomes; difference between report and onset times |
| days | Vector of days when the report is given, aligned from the minimum report day |
| week | Vector indicating the week of the report, assumes no change within the week |
| weekend | Binary vector indicating if the outcome was reported during a weekend |
| iter | Number of iterations for the Bayesian back-calculation algorithm |
| sigma | The standard deviation for the normal distribution |
| maxdelay | The maximum delay parameter for the negative binomial distribution |
| si | Serial interval vector |
| size | The size parameter for the negative binomial distribution |
| workerID | Identifier for the parallel worker |
| printProgress | Flag to print the progress information |
| cd | second size parameter, unused |

### Value

output A list object that contains the back-calculated estimates and model diagnostics

### Examples

```
data("sample_onset_dates")
data("sample_report_dates")
line_list <- create_linelist(sample_report_dates, sample_onset_dates)
sip <- si(14, 4.29, 1.18)
results <- run_backnow(
 line_list,
  MAX_ITER = as.integer(2000),
  norm_sigma = 0.5,
  sip = sip,
  NB_maxdelay = as.integer(20),
  NB_size = as.integer(6),
  workerID = 1,
  printProgress = 1,
 preCalcTime = TRUE)
```

---

## convert_to_linelist *Convert Case Counts to a Line List*

---

### Description

This function takes a data frame of case counts and expands it into a line list format, which is often used for epidemiological analysis. The function validates input data, manages missingness, and assumes additional generation times based on the specified reporting function.

## Usage

```
convert_to_linelist(
  caseCounts,
  reportF = NULL,
  reportF_args = NULL,
  reportF_missP = NULL
)
```

## Arguments

| | |
|---|---|
| caseCounts | A data frame with columns ‘date‘, ‘cases‘, and ‘location‘. The data frame must meet several criteria: - It should only contain data for one location. - Dates must be in Date format. - Case numbers must be non-negative integers. - No missing values are allowed in the necessary columns. |
| reportF | A function used to simulate the delay from case reporting to case onset. Defaults to a negative binomial distribution function (‘rnbinom‘) if NULL. |
| reportF_args | A list of additional arguments to pass to ‘reportF‘. Defaults to ‘list(size = 3, mu = 9)‘ when ‘reportF‘ is NULL. |
| reportF_missP | A numeric probability between 0 and 1 (exclusive) indicating the proportion of missing onset dates. It throws an error if it is out of bounds or not numeric. |

## Details

The function stops and sends error messages for various data integrity issues, such as incorrect data types, negative cases, or missing required columns. It also assumes that the input data is for only one location and handles NA generation according to ‘reportF_missP‘.

## Value

A data frame in line list format, where each row corresponds to a case report. The data frame includes columns for the report date, the delay from report to onset, the onset date, weekend indicator, report interval in days from the first report, and week interval. The returned data frame has additional attributes set, including ‘min_day‘ and the class ‘lineList‘.

## Examples

```
data("sample_dates")
data("sample_location")
data("sample_cases")
case_Counts <- create_caseCounts(sample_dates, sample_location, sample_cases)
line_list <- convert_to_linelist(case_Counts)
```

`create_caseCounts` *Create a Case Counts Data Frame*

## Description

This function constructs a data frame from vectors representing dates, locations, and case numbers, ensuring that all input vectors meet specific data integrity requirements. It checks for the correct data types, non-negative case numbers, and uniformity in vector lengths. The function also ensures no missing values are present and that all data pertain to a single location.

## Usage

```
create_caseCounts(date_vec, location_vec, cases_vec)
```

## Arguments

| | |
|---|---|
| `date_vec` | A vector of dates corresponding to case reports; must be of type Date. |
| `location_vec` | A character vector representing the location of the case reports; all entries must refer to the same location. |
| `cases_vec` | A numeric vector representing the number of cases reported on each date; values must be non-negative integers. |

## Details

The function performs several checks to ensure the integrity of the input: - It verifies that all vectors have the same length. - It confirms that there are no negative numbers in ‘cases_vec‘. - It checks for and disallows any missing values in the data frame. It throws errors if any of these conditions are not met, indicating that the input vectors are not appropriately formatted or contain invalid data.

## Value

A data frame named ‘caseCounts‘ with columns ‘date‘, ‘cases‘, and ‘location‘. Each row corresponds to a unique report of cases on a given date at a specified location. The data frame is assigned a class attribute of ‘caseCounts‘.

## Examples

```
data("sample_dates")
data("sample_location")
data("sample_cases")
case_Counts = create_caseCounts(sample_dates, sample_location, sample_cases)
```

`create_linelist` *Create a Line List from Report and Onset Dates*

## Description

This function constructs a line list data frame using vectors of report and onset dates. It performs several checks and transformations to ensure the data is consistent and appropriate for epidemiological analysis.

## Usage

```
create_linelist(report_dates, onset_dates)
```

## Arguments

| | |
|---|---|
| `report_dates` | A vector of dates representing when cases were reported; must be of type Date. |
| `onset_dates` | A vector of dates representing when symptoms onset occurred; must be of type Date. This vector can contain NA values, but not exclusively or none at all. |

## Details

The function ensures the following: - The length of ‘report_dates‘ and ‘onset_dates‘ must be equal. - There should be no NA values in ‘report_dates‘. - ‘onset_dates‘ must contain some but not all NA values. - Each non-NA onset date must be earlier than or equal to its corresponding report date. If any of these conditions are violated, the function will stop with an error message. Additionally, the function calculates the delay in days between onset and report dates, identifies weekends, and calculates reporting and week intervals based on the earliest date.

## Value

A data frame with the following columns: report_dates, delay_int, onset_dates, is_weekend, report_int, and week_int. This data frame is ordered by report_dates and assigned a class attribute of ‘lineList‘.

## Examples

```
data("sample_onset_dates")
data("sample_report_dates")
line_list <- create_linelist(sample_report_dates, sample_onset_dates)
```

dnb *Calculate Log-Probability Density for Negative Binomial Distribution*

## Description

This function computes the log-probability density function of the negative binomial distribution, given the number of successes, the dispersion parameter, and the mean of the distribution. This is useful for probabilistic models where negative binomial assumptions are applicable, such as certain types of count data.

## Usage

```
dnb(x, r, m)
```

## Arguments

| | |
|---|---|
| x | A numeric value representing the number of successes, which should be a non-negative integer. |
| r | A numeric value representing the dispersion parameter of the negative binomial distribution. A higher value indicates a distribution more tightly concentrated around the mean. |
| m | A numeric value representing the mean of the distribution. |

## Value

A numeric value representing the log-probability density function value of observing 'x' successes given the mean 'm' and dispersion 'r'. This return value is given on the log scale to facilitate calculations that involve very small probabilities.

## Examples

```
dnb(5, 2, 10);
```

dummy *Generate Dummy Variables Matrix for Weeks and Weekends*

## Description

This function creates a matrix of dummy variables based on reported weeks and weekend indicators. Each column in the resulting matrix corresponds to a specific week, except for the last column, which indicates whether the date falls on a weekend. This matrix is typically used in regression models where week-specific effects are to be adjusted along with the effect of weekends.

## Usage

```
dummy(week, weekend)
```

### Arguments

| | |
|---|---|
| week | An integer vector representing the week number of each observation. Each element denotes the week during which a specific event occurred. |
| weekend | A binary integer vector (elements being 0 or 1) indicating whether each observation corresponds to a weekend. Here, ‘1‘ indicates a weekend and ‘0‘ a weekday. |

### Value

A numeric matrix where each row corresponds to an observation and each column to a week, with an additional final column for weekend indicators. The elements of the matrix are dummy variables (0 or 1); each row contains exactly one ’1’ in one of the first several columns corresponding to the week of the observation, and a ’1’ or ’0’ in the last column indicating weekend status.

### Examples

```
week <- c(1, 1, 1, 2, 2, 3, 3, 3, 3, 3)
weekend <- c(0, 1, 0, 0, 1, 0, 0, 1, 1, 0)
dummy_vars <- dummy(week, weekend)
```

---

## findmiss *Locate Missing Values in a Numeric Vector*

---

### Description

This function identifies the indices of missing values (NA) in a given numeric vector. It is useful for data cleaning and preprocessing steps where identification of missing data points is required.

### Usage

```
findmiss(x)
```

### Arguments

| | |
|---|---|
| x | A numeric vector potentially containing NA values. The values can range from -infinity to infinity. |

### Value

An integer vector containing the indices where NA values are found in ‘x‘. These indices can be used directly to reference or manipulate elements in other vectors of the same length or for subsetting the original vector.

### Examples

```
vec <- c(1, 2, NA, 4, NA, 6)
findmiss(vec)
```

get_mu_vec *Calculate Exponential of Linear Combinations*

## Description

Computes the exponential of linear combinations of beta coefficients and a matrix of predictors, typically used in Poisson or logistic regression models for estimating rates or probabilities. This function specifically handles the exponential transformation, which is commonly used to ensure that rates or probabilities are non-negative.

## Usage

```
get_mu_vec(x12, beta)
```

## Arguments

| | |
|---|---|
| x12 | NumericMatrix representing a matrix of predictors, where each row corresponds to an observation and columns correspond to different predictor variables (e.g., weeks and weekends). |
| beta | NumericVector of coefficients corresponding to the predictors in ‘x12‘. This should include coefficients for both weekly effects and potentially an additional coefficient for weekends. |

## Details

The function multiplies the matrix ‘x12‘ by the vector ‘beta‘ to get the linear predictors, then applies the exponential function to convert these linear predictors to a scale suitable for models where the response variable is a count or probability. This is a critical step in generalized linear models where the link function is the natural logarithm.

## Value

NumericVector where each element is the exponential of the linear combination of the predictors and coefficients for a given observation. This vector represents the model-estimated mean values for each observation.

## Examples

```
# Assuming x12 is a matrix with 10 observations and 3 predictors
# and beta is a vector of 3 coefficients
x12 <- matrix(rnorm(30), ncol=3)
beta <- c(0.1, -0.2, 0.05)
get_mu_vec(x12, beta)
```

getr *Calculate Time-Varying Reproduction Number R(t)*

## Description

This function estimates the time-varying reproduction number, R(t), based on the epidemic curve and serial interval distribution. R(t) is calculated for each day using a moving window approach, which involves taking a segment of the epidemic curve and applying a transformation based on the serial interval to estimate how many subsequent cases are generated by cases within the window.

## Usage

```
getr(curve, si, size)
```

## Arguments

| | |
|---|---|
| curve | NumericVector representing the estimated epidemic curve with daily counts. This curve can include both back-calculated and nowcasted counts of infections. |
| si | NumericVector representing the serial interval distribution, expressed as a vector where each element corresponds to the probability of a delay of that many days between successive cases. |
| size | Integer specifying the size of the moving window used to calculate the mean reproduction number. This window size determines how many days are included in the calculation of R(t) at each step. |

## Value

NumericVector containing the estimated mean reproduction numbers (R(t)) for each day. The length of this vector will be the length of ‘curve‘ minus ‘size‘ minus one, reflecting the fact that the last few days do not have enough data to fill the window.

## Examples

```
# Assume curve is a numeric vector of daily case counts and si is the serial interval distribution
curve <- rnorm(100, mean=10, sd=3)  # example epidemic curve
si <- rep(0.1, 10)  # example serial interval distribution
size <- 6  # example size of the moving window
getr(curve, si, size)
```

`lambda` *Calculate Lambda Values for Poisson Distribution Mean*

## Description

This function computes the convolution of the epidemic curve with the serial interval distribution to estimate the mean of the Poisson distribution for each day. This mean is crucial in models where the number of new cases follows a Poisson process. The calculation is effectively a weighted sum of past case counts, where weights are given by the serial interval distribution, representing the expected number of new cases caused by an individual case on each subsequent day.

## Usage

```
lambda(curve, si)
```

## Arguments

| | |
|---|---|
| `curve` | NumericVector representing the daily counts of new cases, estimated from the epidemic data. This vector should include both historically observed data and nowcasted estimates. |
| `si` | NumericVector representing the serial interval distribution, a probability distribution describing the time delay between successive cases in an infectious disease transmission chain. |

## Details

The function applies the serial interval to the epidemic curve via convolution, essentially calculating the expected number of secondary cases generated by each primary case over the serial interval. The length of the resulting vector is one less than that of ‘curve‘ because the last day’s value cannot be calculated without full serial interval data.

## Value

NumericVector containing the estimated mean values for the Poisson distribution at each day, excluding the last day as the serial interval cannot be fully applied.

## Examples

```
curve <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
si <- c(0.5, 0.3, 0.2)  # Declining probability over three days
lambda_values <- lambda(curve, si)
```

`logLikNB` *Compute Log-Likelihood for a Right-Truncated Negative Binomial Model*

## Description

This function calculates the log-likelihood of observing the given data under a right-truncated negative binomial distribution. It is used to assess the fit of a model involving delay times in reporting cases, where each case has a delay modeled by a negative binomial distribution that is truncated at a specified maximum delay.

## Usage

```
logLikNB(delay_vec, x12, disp, betaplus, maxdelay)
```

## Arguments

| | |
|---|---|
| `delay_vec` | NumericVector representing the observed delay times for each case. |
| `x12` | NumericMatrix of covariates used to model the mean of the negative binomial distribution. Each row corresponds to a case and columns correspond to covariates including time since onset and others. |
| `disp` | NumericVector indicating the dispersion parameter for each case. This can affect the distribution of counts and is used to model heterogeneity in reporting delays. |
| `betaplus` | NumericVector containing current estimates of the model parameters, which may include coefficients for covariates and additional parameters. |
| `maxdelay` | Integer specifying the maximum reporting delay, truncating the distribution at this value. |

## Details

This function computes the log-likelihood by calculating the likelihood of each observed delay under the specified model parameters, considering the truncation at ‘maxdelay‘. The parameters ‘disp‘ and ‘betaplus‘ allow for flexibility in modeling different types of heterogeneity and covariate effects.

## Value

Double representing the log-likelihood of the data given the model parameters. This value measures how well the model with the current parameter estimates fits the observed data.

## Examples

```
# Example usage with arbitrary data:
delay_vec <- rnorm(100, mean=10, sd=3)  # Simulated delay times
x12 <- matrix(rnorm(300), ncol=3)       # Simulated covariates
disp <- rep(1, 100)                     # Dispersion parameter, constant for simplicity
```

```
betaplus <- runif(4)                    # Simulated parameter estimates
maxdelay <- 15                          # Maximum delay for truncation
loglik_value <- logLikNB(delay_vec, x12, disp, betaplus, maxdelay)
```

---

`plot.backnow` *Plot Estimates or Reproduction Numbers*

---

## Description

This function plots estimates of case numbers or reproduction numbers (‘r(t)‘) based on the provided object. It can handle two types of plots: ’est’ for estimated case numbers over time, and ’rt’ for estimated reproduction numbers over time.

## Usage

```
## S3 method for class 'backnow'
plot(x, plottype, ...)
```

## Arguments

| | |
|---|---|
| `x` | An object containing the necessary data for plotting. This object should have specific structure depending on the ‘plottype‘: - For ‘plottype = ’est’‘, ‘x‘ should contain ‘report_date‘, ‘report_cases‘, ‘est_back_date‘, and ‘est_back‘, where ‘est_back‘ is expected to be a matrix with three rows representing the lower bound, estimate, and upper bound. - For ‘plottype = ’rt’‘, ‘x‘ should contain ‘est_rt_date‘ and ‘est_rt‘, with ‘est_rt‘ formatted similarly to ‘est_back‘. |
| `plottype` | A character string specifying the type of plot to generate. Valid options are ’est’ for case estimates and ’rt’ for reproduction numbers. |
| `...` | Additional arguments passed to the plot function. |

## Details

Depending on the ‘plottype‘: - ’est’: Plots the reported cases over time with a polygon representing the uncertainty interval and a line showing the central estimate. - ’rt’: Plots the reproduction number over time with a similar style.

## Value

a plot object for an object of class ‘backnow‘

## Examples

```
data("sample_onset_dates")
data("sample_report_dates")
line_list <- create_linelist(sample_report_dates, sample_onset_dates)
sip <- si(14, 4.29, 1.18)
results <- run_backnow(
  line_list,
```

```
  MAX_ITER = as.integer(2000),
  norm_sigma = 0.5,
  sip = sip,
  NB_maxdelay = as.integer(20),
  NB_size = as.integer(6),
  workerID = 1,
  printProgress = 1,
  preCalcTime = TRUE)
plot(results, 'est')
plot(results, 'rt')
```

---

`plot.caseCounts` *Plot Case Counts Over Time*

---

## Description

This function plots the number of cases over time from a data frame object. If the data frame contains multiple locations, a specific location must be specified. The plot displays the total number of cases against dates and annotates one of the earliest points with the location name.

## Usage

```
## S3 method for class 'caseCounts'
plot(x, loc = NULL, ...)
```

## Arguments

| | |
|---|---|
| `x` | A data frame containing the case counts with at least two columns: ‘date‘ and ‘cases‘. The data frame may optionally include a ‘location‘ column, which is required if multiple locations are present. |
| `loc` | An optional string specifying the location to filter the case counts by. If ‘loc‘ is provided and ‘location‘ column exists in ‘x‘, the plot will only show data for the specified location. If multiple locations are present and ‘loc‘ is not specified, the function will stop with an error. |
| `...` | Additional arguments passed to the ‘plot‘ function. |

## Details

If the ‘location‘ column is present in ‘x‘ and contains multiple unique values, the ‘loc‘ parameter must be specified to indicate which location’s data to plot. The function adds a text annotation to the plot, labeling one of the earliest points with the specified location’s name.

## Value

a plot object for an object of class ‘caseCounts‘

## Examples

```
data("sample_dates")
data("sample_location")
data("sample_cases")
case_Counts = create_caseCounts(sample_dates, sample_location, sample_cases)
plot(case_Counts)
```

---

pnb *Compute Cumulative Distribution Function for Negative Binomial Distribution*

---

## Description

This function calculates the cumulative distribution function (CDF) of the negative binomial distribution given a number of successes, a dispersion parameter, and the mean. The negative binomial distribution is commonly used to model count data with overdispersion relative to a Poisson distribution.

## Usage

```
pnb(x, r, m)
```

## Arguments

| | |
|---|---|
| x | Non-negative integer specifying the number of successes for which the CDF is computed. This value must be non-negative as it represents the number of successes in the distribution. |
| r | Dispersion parameter of the distribution, a positive real number. Higher values of 'r' indicate a higher probability of counts clustering around the mean, reducing overdispersion. |
| m | Mean of the distribution, a positive real number indicating the expected number of successes. The mean must be positive. |

## Details

The negative binomial distribution can be parameterized by a dispersion parameter 'r' and a mean 'm', which together determine the shape of the distribution. This function is essential for modeling and probability calculations in various fields such as epidemiology and ecology where the negative binomial distribution is used to model count data.

## Value

y The probability of observing up to 'x' successes in a negative binomial distribution parameterized by 'r' (dispersion) and 'm' (mean). This function returns the CDF value.

## Examples

```
pnb(10, 5, 10)
```

prop *Calculate Proportions of Event Counts Within a Specified Time Range*

## Description

This function calculates the proportion of event counts that occur within each unit of time from a specified starting point (onset time) up to a maximum delay. The proportions are computed relative to the total number of events occurring within the specified time range.

## Usage

```
prop(x, onset, maxdelay, cd)
```

## Arguments

| | |
|---|---|
| x | NumericVector representing the count of events at each time point. |
| onset | NumericVector representing the onset times for each event count. Each element in the vector indicates the time at which an event was initiated. |
| maxdelay | The maximum time delay (inclusive) for which the proportions are calculated. This parameter defines the upper bound of the time interval over which the proportions of event counts are evaluated. |
| cd | The onset time (exclusive) from which to start calculating proportions. Events that start at times less than or equal to this value are excluded from the calculation. |

## Value

NumericVector containing the proportions of the total events falling within each unit of time from the specified 'cd' up to 'maxdelay'. The proportions are cumulative, with each element representing the proportion of events that have occurred by that time point, starting from 'cd'.

## Examples

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
onset <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, 5)
maxdelay <- 5
cd <- 1
prop(x, onset, maxdelay, cd)
```

## rnb *Generate Random Samples from a Negative Binomial Distribution*

### Description

This function generates random samples from a negative binomial distribution with the specified dispersion parameter (r) and success probability (p).

### Usage

```
rnb(r, p)
```

### Arguments

| | |
|---|---|
| r | The number of failures before achieving a specified number of successes in a negative binomial experiment. It also serves as the dispersion parameter which controls the variance of the distribution. |
| p | The probability of success on each independent Bernoulli trial within the negative binomial experiment. |

### Value

A random value sampled from the negative binomial distribution with parameters r and p.

### Examples

```
r <- 2
p <- 0.3
rnb(r, p)
```

## run_backnow *Run Back Calculation and Estimate Reproduction Numbers*

### Description

This function performs a back-calculation based on provided epidemic case count data, estimating the time distribution of infections and reproduction numbers (r(t)). It utilizes extensive input checks and parameter validation to ensure robust model execution.

## Usage

```
run_backnow(
  input,
  MAX_ITER,
  norm_sigma,
  sip,
  NB_maxdelay,
  NB_size,
  n_trunc = NB_size,
  workerID = NULL,
  printProgress = 0,
  preCalcTime = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| input | A data frame or list that includes epidemic data with either class 'caseCounts' or 'lineList'. The input type determines initial processing steps. |
| MAX_ITER | Integer, maximum number of iterations for the back-calculation model. Requires at least 2000 iterations; high numbers can significantly increase runtime. |
| norm_sigma | Numeric, the standard deviation for the normal distribution in the Bayesian framework. |
| sip | Vector of numeric values specifying the serial interval probabilities. |
| NB_maxdelay | Integer, the maximum delay for the right-truncated negative binomial distribution used in modeling. |
| NB_size | Integer, the size parameter for the negative binomial distribution. |
| n_trunc | Integer, the truncation number for the final result matrices (defaults to 'NB_size'). |
| workerID | Optional integer to specify a worker ID for parallel processing frameworks; defaults to 0. |
| printProgress | Binary integer (0 or 1), specifying whether to print progress to console; affects performance. |
| preCalcTime | Boolean, if TRUE, the function calculates a preliminary runtime estimate before full execution. |
| ... | Additional arguments passed to underlying functions when converting input to the required format. |

## Details

The function ensures input data is of the correct class and processes it accordingly. It handles different input classes by either converting 'caseCounts' to 'lineList' or directly using 'lineList'. The function stops with an error if the input doesn't meet expected standards. It performs simulations to estimate both the back-calculation of initial infections and reproduction numbers over time, while checking and adjusting for potential NA values and ensuring that all conditions for the model parameters are met. Output includes estimates of initial infections and reproduction numbers along with diagnostic statistics.

## Value

an object of class ‘backnow‘ with the following structure

- est_back: back-calculated case counts

- est_back_date: dates for back-calculated case counts

- est_rt: back-calculated R(r)

- est_rt_date: dates for back-calculated R(t)

- geweke_back: Geweke diagnostics for the estimated back-calculation of cases

- geweke_rt; Geweke diagnostics for R(t)

- report_date: a vector of dates, matches reported_cases

- report_cases: a vector of reported cases

- MAX_ITER: the input for ‘MAX_ITER‘

- norm_sigma: the input for ‘norm_sigma‘

- NB_maxdelay: the input for ‘NB_maxdelay‘

- si: the input for serial interval ‘si‘

- NB_size: the input for ‘NB_size‘

## Examples

```
data("sample_onset_dates")
data("sample_report_dates")
line_list <- create_linelist(sample_report_dates, sample_onset_dates)
sip <- si(14, 4.29, 1.18)
results <- run_backnow(
  line_list,
  MAX_ITER = as.integer(2000),
  norm_sigma = 0.5,
  sip = sip,
  NB_maxdelay = as.integer(20),
  NB_size = as.integer(6),
  workerID = 1,
  printProgress = 1,
  preCalcTime = TRUE)
```

---

# sample_cases *Sample datasets*

---

## Description

These data sets provide tests for use in converting between case counts and line list data.

## Usage

```
sample_cases

sample_dates

sample_location

sample_onset_dates

sample_report_dates

out_list_demo
```

## Format

Either vectors or a list object (out_list_demo)

## Details

'sample_cases' provides a vector of daily case counts. 'sample_dates' are the dates of the sample case counts. 'sample_location' are the locations of the sample case counts. 'sample_onset_dates' are the same information as in sample_cases, but with one entry per case indicating the date of symptom onset. 'sample_report_dates' are the same information as in sample_cases, but with one entry per case indicating the date of symptom reporting. 'out_list_demo' is a precomputed output from run_backnow, useful for plotting in the vignettes

## Examples

```
sample_cases
sample_dates
sample_location
sample_onset_dates
sample_report_dates
```

---

# si *Calculate Serial Interval Distribution for COVID-19*

---

## Description

This function computes the probability distribution function (PDF) of the serial interval for COVID-19 using a gamma distribution with specified shape and rate parameters. The serial interval is defined as the time between successive cases in a chain of transmission. This implementation generates a discrete PDF over a given number of days.

## Usage

```
si(ndays, alpha, beta)
```

## Arguments

| | |
|---|---|
| `ndays` | Integer, the number of days over which to calculate the serial interval distribution. |
| `alpha` | Numeric, the shape parameter of the gamma distribution. |
| `beta` | Numeric, the rate parameter of the gamma distribution. |

## Details

The function uses the ‘pgamma‘ function to calculate cumulative probabilities for each day up to ‘ndays‘ and then differences these to get daily probabilities. The resulting probabilities are normalized to sum to 1, ensuring that they represent a valid probability distribution.

## Value

Numeric vector representing the serial interval probabilities for each of the first ‘ndays‘ days. The probabilities are normalized so that their sum is 1.

## References

Nishiura, H., Linton, N. M., & Akhmetzhanov, A. R. (2020). Serial interval of novel coronavirus (COVID-19) infections. International Journal of Infectious Diseases, 93, 284-286. [Link to the article](https://www.sciencedirect.com/science/article/pii/S1201971220306111)

## Examples

```
sip <- si(14, 4.29, 1.18)
```

# Index

∗ **datasets**
sample_cases, 19

backnow_cm, 2

convert_to_linelist, 3
create_caseCounts, 5
create_linelist, 6

dnb, 7
dummy, 7

findmiss, 8

get_mu_vec, 9
getr, 10

lambda, 11
logLikNB, 12

out_list_demo *(*sample_cases*)*, 19

plot.backnow, 13
plot.caseCounts, 14
pnb, 15
prop, 16

rnb, 17
run_backnow, 17

sample_cases, 19
sample_dates *(*sample_cases*)*, 19
sample_location *(*sample_cases*)*, 19
sample_onset_dates *(*sample_cases*)*, 19
sample_report_dates *(*sample_cases*)*, 19
si, 20