Package 'bayestransmission'

December 12, 2025

Title Bayesian Transmission Models

Version 0.1.0

Description Provides Bayesian inference methods for infectious disease transmission models in healthcare settings. Implements Markov Chain Monte Carlo (MCMC) algorithms for estimating transmission parameters from patient-level data including admission, discharge, and testing events as described in Thomas 'et al.' (2015) <doi:10.1093/imammb/dqt021>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports assertthat, dplyr, methods, Rcpp (>= 1.0.12), rlang

Config/testthat/edition 3

Depends R (>= 4.2.0)

LazyData true

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++17, BLAS, LAPACK

Suggests checkmate, devtools, ggplot2, testthat (>= 3.0.0), tidyr, knitr, rmarkdown, pillar

RcppModules Infect

URL https://epiforesite.github.io/bayestransmission/, https://github.com/EpiForeSITE/bayestransmission

VignetteBuilder knitr

BugReports https://github.com/EpiForeSITE/bayestransmission/issues

NeedsCompilation yes

Author Andrew Redd [aut, cre] (ORCID: <https://orcid.org/0000-0002-6149-2438>), Alun Thomas [aut], University of Utah [cph], CDC's Center for Forecasting and Outbreak Analytics [fnd] (This project was made possible by cooperative agreement CDC-RFA-FT-23-0069

(grant # NU38FT000009-01-00) from the CDC's Center for Forecasting and Outbreak Analytics. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the Centers for Disease Control and Prevention.)

Maintainer Andrew Redd <andrew.redd@hsc.utah.edu>

Repository CRAN

Date/Publication 2025-12-12 21:10:07 UTC

Contents

`AbxParams` *Antibiotic Parameters*

Description

Antibiotic Parameters

Usage

```
AbxParams(onoff = FALSE, delay = 0, life = 1)
```

Arguments

onoff	If Anti-biotic are used or not.
delay	The delay in using antibiotics.
life	The life of antibiotics.

Value

list of parameters for antibiotic effect

Examples

```
AbxParams()
```

AbxRateParams *Antibiotic Administration Rate Parameters*

Description

Antibiotic Administration Rate Parameters

Usage

```
AbxRateParams(
  uncolonized = Param(1, 0),
  colonized = Param(1, 0),
  latent = Param(0)
)
```

Arguments

uncolonized	Rate of antibiotic administration when the individual is uncolonized.
colonized	Rate of antibiotic administration when the individual is colonized.
latent	Rate of antibiotic administration when the individual is latent.

Value

list of parameters for antibiotic administration.

Examples

```
AbxRateParams()
```

ClearanceParams *Clearance Parameters*

Description

Clearance Parameters

Usage

```
ClearanceParams(rate = Param(0.01), abx = Param(1, 0), ever_abx = Param(1, 0))
```

Arguments

rate	base rate of clearance
abx	effect of antibiotics on clearance
ever_abx	effect of ever having taken antibiotics on clearance

Value

A list of parameters for clearance.

Examples

```
ClearanceParams()
```

CodeToEvent *Convert coded events to string events*

Description

Convert coded events to string events

Usage

```
CodeToEvent(x)
```

Arguments

x	A vector of integers

Value

A vector of strings

Examples

```
CodeToEvent(c(-1:19, 21:23, 31:33, -999))
```

EventToCode *Convert string events to coded events*

Description

Convert string events to coded events

Usage

```
EventToCode(x)
```

Arguments

x	A vector of strings

Value

A vector of integers

Examples

```
EventToCode(c("admission", "discharge", "postest", "negtest"))
EventToCode(c("abxon", "abxoff", "isolon", "isoloff"))
```

getCppModelParams *Extract Model Parameters from C++ Model Object*

Description

Convenience function to extract all parameter values from a C++ model object created with `newCppModel()`. This is essentially a wrapper around accessing the model's parameter properties.

Usage

```
getCppModelParams(model)
```

Arguments

model	A C++ model object created with `newCppModel()`

Value

A named list containing all model parameter values:

- `Insitu` - Named numeric vector of in situ parameter values
- `SurveillanceTest` - Named numeric vector of surveillance test parameter values
- `ClinicalTest` - Named numeric vector of clinical test parameter values
- `OutCol` - Named numeric vector of out-of-unit colonization parameter values
- `InCol` - Named numeric vector of in-unit colonization parameter values
- `Abx` - Named numeric vector of antibiotic parameter values (if applicable)

If a component's names cannot be determined or lengths mismatch, the vector is returned unnamed.

Examples

```
# Create a linear antibiotic model
params <- LinearAbxModel()
model <- newCppModel(params)

# Extract all parameters
all_params <- getCppModelParams(model)

# View specific parameter groups
all_params$InCol  # In-unit colonization parameters
all_params$Insitu # In situ parameters
```

`getExposureFlags` *Get compilation flags for exposed classes*

Description

Returns information about which optional classes are exposed in this build. This allows tests to conditionally skip tests for unexposed classes.

Usage

```
getExposureFlags()
```

Value

List with logical flags:

- `comprehensive_testing` - TRUE if comprehensive testing classes are exposed
- `all_classes` - TRUE if all optional classes are exposed
- `minimal` - TRUE if only critical classes are exposed

`InsituParams` *InSitu Parameters*

Description

InSitu Parameters

Usage

```
InsituParams(probs = NULL, priors = NULL, doit = NULL, nstates = NULL)
```

Arguments

`probs`	The probability of the individual being in each state.
`priors`	The prior probability of the individual being in each state.
`doit`	A flag indicating if the rate(s) should be updated in the MCMC.
`nstates`	The number of states (2 or 3). If NULL, inferred from probs length. For 2-state models, uses `c(uncolonized, latent=0, colonized)`. For 3-state models, uses `c(uncolonized, latent, colonized)`.

Value

A list of parameters for in situ testing.

Examples

```
InsituParams()
InsituParams(nstates = 2)  # c(0.9, 0.0, 0.1)
InsituParams(nstates = 3)  # c(0.98, 0.01, 0.01)
```

`InUnitParams` *In Unit Parameters*

Description

In Unit Parameters

Usage

```
InUnitParams(
  acquisition = LinearAbxAcquisitionParams(),
  progression = ProgressionParams(),
  clearance = ClearanceParams()
)

ABXInUnitParams(
  acquisition = LinearAbxAcquisitionParams(),
  progression = ProgressionParams(),
  clearance = ClearanceParams()
)
```

Arguments

acquisition	Acquisition, for rate of acquisition of the disease moving into colonized(2-State)/latent(3-state) state.
progression	Progression from latent state to colonized state.
clearance	Clearance from colonized state to uncolonized state.

Value

A list of parameters for in unit infection.

Functions

- ABXInUnitParams(): In Unit Parameters with Antibiotics.

Examples

```
InUnitParams(
  acquisition = LinearAbxAcquisitionParams(),
  progression = ProgressionParams(),
  clearance = ClearanceParams()
)
ABXInUnitParams(
  acquisition = LinearAbxAcquisitionParams(),
  progression = ProgressionParams(),
  clearance = ClearanceParams()
)
```

LinearAbxAcquisitionParams

Linear Antibiotic Acquisition Parameters

Description

Acquisition parameters for LinearAbxModel and LinearAbxModel2.

Usage

```
LinearAbxAcquisitionParams(
  base = Param(0.001),
  time = Param(1, 0),
  mass = Param(1, 1),
  freq = Param(1, 1),
  col_abx = Param(1, 0),
  suss_abx = Param(1, 0),
  suss_ever = Param(1, 0)
)
```

Arguments

base	The base rate of acquisition.
time	The time effect on acquisition.
mass	The mass action effect on acquisition.
freq	The frequency effect on acquisition.
col_abx	The effect for colonized on antibiotics.
suss_abx	The effect on susceptible being currently on antibiotics.
suss_ever	The effect on susceptible ever being on antibiotics.

Details

The model for this acquisition model is given by

$$P(\text{Acq(t)}) = \left[e^{\beta_{\text{time}}(t-t_0)}\right] \left\{e^{\beta_0}\left[\left(\frac{\beta_{\text{freq}}}{P(t)} + (1 - e^{\beta_{\text{freq}}})\right) e^{\beta_{\text{mass}}}\left((N_c(t) - N_{ca}(t)) + e^{\beta_{\text{col\_abx}}} N_{ca}(t)\right) + 1 - e^{\beta_{\text{mass}}}\right]\right\} [$$

where P(Acq(t)) is the acquisition probability at time t, with effects from time (beta_time), mass action (beta_mass), frequency dependence (beta_freq), colonized individuals on antibiotics (beta_col_abx), and susceptible individuals currently (beta_suss_abx) or ever (beta_suss_ever) on antibiotics.

Value

A list of parameters for acquisition.

Examples

```
LinearAbxAcquisitionParams()
```

`LogNormalAcquisitionParams` *Log-Normal Acquisition Parameters*

Description

Acquisition parameters for the log-normal model (LogNormalAbxICP). This model has 8 acquisition parameters accessed by index in C++. Note: When accessed via setupLogNormalICPAcquisition, parameters are set by index, so this returns an unnamed list where position matters.

Usage

```
LogNormalAcquisitionParams(
  time = Param(0, 0),
  constant = Param(0.001, 1),
  log_tot_inpat = Param(-1, 0),
  log_col = Param(1, 0),
  col = Param(0, 0),
  abx_col = Param(0, 0),
  onabx = Param(0, 0),
  everabx = Param(0, 0)
)
```

Arguments

`time`	Time parameter (index 0)
`constant`	Constant parameter (index 1)
`log_tot_inpat`	Log total in-patients parameter (index 2)
`log_col`	Log number colonized parameter (index 3)
`col`	Number colonized parameter (index 4)
`abx_col`	Number abx colonized parameter (index 5)
`onabx`	Susceptible patient on Abx effect (index 6)
`everabx`	Susceptible patient ever on Abx effect (index 7)

Value

An unnamed list of 8 parameters in the correct order for LogNormalAbxICP.

Examples

```
LogNormalAcquisitionParams()
```

LogNormalModelParams *Model Parameters for a Log Normal Model*

Description

Model Parameters for a Log Normal Model

Usage

```
LogNormalModelParams(
  modname,
  nstates = 2L,
  nmetro = 1L,
  forward = TRUE,
  cheat = FALSE,
  Insitu = NULL,
  SurveillanceTest = SurveillanceTestParams(),
  ClinicalTest = ClinicalTestParams(),
  OutOfUnitInfection = OutOfUnitInfectionParams(),
  InUnit = NULL,
  Abx = AbxParams(),
  AbxRate = AbxRateParams()
)

LinearAbxModel(..., InUnit = ABXInUnitParams())
```

Arguments

modname	The name of the model used. Usually specified by specification functions.
nstates	The number of states in the model.
nmetro	The number of Metropolis-Hastings steps to take between outputs.
forward	TODO
cheat	TODO
Insitu	In Situ Parameters
SurveillanceTest	Surveillance Testing Parameters
ClinicalTest	Clinical Testing Parameters
OutOfUnitInfection	Out of Unit Infection Parameters
InUnit	In Unit Parameters, should be a list of lists with parameters for the acquisition, progression and clearance of the disease.
Abx	Antibiotic Parameters
AbxRate	Antibiotic Rate Parameters
...	Additional arguments passed to LogNormalModelParams

Value

A list of parameters for the model.

Functions

- `LinearAbxModel()`: Linear Antibiotic Model Alias

Examples

```
LogNormalModelParams("LogNormalModel")
```

`mcmc_to_dataframe` *Convert MCMC Parameters to Data Frame*

Description

Converts the nested list structure of MCMC parameters from `runMCMC` output into a tidy data frame format suitable for analysis and visualization.

Usage

```
mcmc_to_dataframe(mcmc_results)
```

Arguments

`mcmc_results`	The results object returned by `runMCMC()`. Must contain a `Parameters` component and a `LogLikelihood` component.

Details

The function extracts parameters from the nested list structure and handles missing values gracefully by inserting `NA` when a parameter is not present. This is particularly useful for creating trace plots and posterior distributions.

Value

A data frame with one row per MCMC iteration containing:

- `iteration`: The iteration number
- `insitu_*`: In-situ probability parameters
- `surv_test_*`: Surveillance test parameters
- `clin_test_*`: Clinical test parameters and rates
- `outunit_*`: Out of unit infection parameters
- `inunit_*`: In unit LinearAbx model parameters (base, time, mass, freq, colabx, susabx, susever, clr, clrAbx, clrEver)
- `abxrate_*`: Antibiotic rate parameters
- `loglikelihood`: Log likelihood at each iteration

Examples

```
results <- runMCMC(data = simulated.data,
                   modelParameters = LinearAbxModel(),
                   nsims = 10,
                   nburn = 0,
                   outputparam = TRUE,
                   outputfinal = FALSE)
param_df <- mcmc_to_dataframe(results)
head(param_df)
```

---

newCppModel *Create a new C++ model object with parameters*

Description

Creates and initializes a C++ model object based on the provided parameters. This function wraps the underlying C++ model classes (LogNormalModel, LinearAbxModel, LinearAbxModel2, MixedModel) in appropriate R reference classes that expose the model's methods and properties.

Usage

```
newCppModel(modelParameters, verbose = FALSE)
```

Arguments

`modelParameters`

List of model parameters created using functions from constructors.R, such as:

- `LogNormalModelParams()` - Basic log-normal model
- `LinearAbxModel()` - Linear antibiotic model
- Or custom parameter lists containing:
  - `modname`: Model name ("LogNormalModel", "LinearAbxModel", "LinearAbxModel2", "MixedModel")
  - `nstates`: Number of states (2 or 3)
  - `nmetro`: Number of Metropolis-Hastings steps
  - `forward`: Forward simulation flag
  - `cheat`: Cheat flag for debugging
  - `Insitu`: In situ parameters from `InsituParams()`
  - `SurveillanceTest`: Surveillance test parameters from `SurveillanceTestParams()`
  - `ClinicalTest`: Clinical test parameters from `ClinicalTestParams()`
  - `OutCol`: Out-of-unit infection parameters from `OutOfUnitInfectionParams()`
  - `InCol`: In-unit parameters from `InUnitParams()` or `ABXInUnitParams()`
  - `Abx`: Antibiotic parameters from `AbxParams()`
  - `AbxRate`: Antibiotic rate parameters from `AbxRateParams()`

`verbose` Logical flag to print progress messages during model creation and parameter setup (default: FALSE)

Details

The function uses the existing `newModel` C++ function to instantiate the model and configure all parameters, then wraps it in the appropriate R reference class based on the model type specified in `modelParameters$modname`.

Value

A reference class object wrapping the C++ model. The specific class depends on `modelParameters$modname`:

- `CppLogNormalModel` - For "LogNormalModel"
- `CppLinearAbxModel` - For "LinearAbxModel"
- `CppLinearAbxModel2` - For "LinearAbxModel2"
- `CppMixedModel` - For "MixedModel" (if exposed in C++)

All returned objects inherit from `CppBasicModel` and provide access to:

- **Properties:**
  - `InColParams` - In-unit colonization parameters
  - `OutColParams` - Out-of-unit colonization parameters
  - `InsituParams` - In situ parameters
  - `SurveillanceTestParams` - Surveillance test parameters
  - `ClinicalTestParams` - Clinical test parameters
  - `AbxParams` - Antibiotic parameters
- **Methods:**
  - `logLikelihood(hist)` - Calculate log likelihood for a SystemHistory
  - `getHistoryLinkLogLikelihoods(hist)` - Get individual link log likelihoods
  - `forwardSimulate(...)` - Perform forward simulation
  - `initEpisodeHistory(...)` - Initialize episode history
  - `sampleEpisodes(...)` - Sample episodes
  - `setAbx(...)` - Set antibiotic parameters

See Also

- `LogNormalModelParams()` for creating model parameters
- `LinearAbxModel()` for linear antibiotic model parameters
- `InsituParams()`, `SurveillanceTestParams()`, etc. for parameter components
- `newModelExport()` for extracting parameter values from a model

Examples

```
# Create a linear antibiotic model (recommended - stable constructors)
params <- LinearAbxModel()
model <- newCppModel(params)

# Access model properties
inColParams <- model$InColParams
```

```
insituParams <- model$InsituParams

# Get parameter values
paramValues <- inColParams$values

# Get parameter names (if available)
paramNames <- inColParams$names

# Create a log-normal model
params <- LogNormalModelParams("LogNormalModel")
model <- newCppModel(params, verbose = TRUE)
```

`newModelExport` *Create a new model object*

Description

Creates and initializes a model object based on the provided parameters. This allows direct creation and inspection of model objects without running MCMC. Returns a list with all model parameter values for verification.

Usage

```
newModelExport(modelParameters, verbose = FALSE)
```

Arguments

`modelParameters`
: List of model parameters, including:
  - `modname` Name of the model (e.g., "LogNormalModel", "LinearAbxModel", "LinearAbxModel2", "MixedModel")
  - `nstates` Number of states in the model
  - `nmetro` Number of metropolis steps
  - `forward` Forward parameter
  - `cheat` Cheat parameter

`verbose`
: Print progress messages (default: false)

Value

A list containing the initialized model parameters:

- `Insitu` - In situ parameters
- `SurveillanceTest` - Surveillance test parameters
- `ClinicalTest` - Clinical test parameters
- `OutCol` - Out of unit colonization parameters

- `InCol` - In unit colonization parameters
- `Abx` - Antibiotic parameters

`OutOfUnitInfectionParams` *Out of Unit Infection Parameters*

Description

Out of Unit Infection Parameters

Usage

```
OutOfUnitInfectionParams(
  acquisition = Param(0.05),
  clearance = Param(0.01),
  progression = Param(0)
)
```

Arguments

`acquisition`	Rate of acquisition of the disease moving into latent state.
`clearance`	Rate of clearance of the disease moving into uncolonized state.
`progression`	Rate of progression of the disease moving into colonized state.

Value

A list of parameters for out of unit infection.

Examples

```
OutOfUnitInfectionParams()
```

`Param` *Construct a parameter with a prior, weight and an update flag.*

Description

Construct a parameter with a prior, weight and an update flag.

Usage

```
Param(
  init,
  weight = if_else(init == 0, 0, 1),
  update = weight > 0,
  prior = init
)
```

Arguments

init	the initial value of the parameter.
weight	the weight of the prior.
update	a flag indicating if the parameter shouldbe updated in the MCMC.
prior	mean value of the prior distribution, may be used with weight to fully determine prior parameters.

Value

A list with the following elements:

- init the initial value of the parameter.
- weight the weight of the prior.
- update a flag indicating if the parameter shouldbe updated in the MCMC.
- prior mean value of the prior distribution, may be used with weight to fully determine prior parameters.

Examples

```
# Fully specified parameter.
Param(init = 0, weight = 1, update = TRUE, prior = 0.5)
# Fixed parameter
# Weight = 0 implies update=FALSE and prior is ignored.
Param(0, 0)
# Update parameter that starts at zero.
Param(0, weight =1, update=TRUE)
# Parameters specified at zero implies fixed.
Param(0)
```

ParamWRate *Specify a random testing parameter with a rate.*

Description

Specify a random testing parameter with a rate.

Usage

```
ParamWRate(param = Param(), rate = Param())
```

Arguments

param	Values for the positive rate of the test.
rate	Values for the rate of the test.

Value

A list of with param and rate.

Examples

```
ParamWRate(Param(0.5, 0), rate = Param(1, 0))
```

ProgressionParams *Progression Parameters*

Description

Progression Parameters

Usage

```
ProgressionParams(
  rate = Param(0.01),
  abx = Param(1, 0),
  ever_abx = Param(1, 0)
)
```

Arguments

rate	Base progression rate
abx	Effect of current antibiotics on progression
ever_abx	Effect of ever having taken antibiotics on progression

Value

A list of parameters for progression.

Examples

```
ProgressionParams()
```

RandomTestParams *Random Testing Parameter Set*

Description

Random Testing Parameter Set

Usage

```
RandomTestParams(
  uncolonized = ParamWRate(Param(0.5, 0), Param(1, 0)),
  colonized = ParamWRate(Param(0.5, 0), Param(1, 0)),
  latent = ParamWRate(Param(0), Param(0))
)

ClinicalTestParams(
  uncolonized = ParamWRate(Param(0.5, 0), Param(1, 0)),
  colonized = ParamWRate(Param(0.5, 0), Param(1, 0)),
  latent = ParamWRate(Param(0), Param(0))
)
```

Arguments

uncolonized	Testing when the individual is uncolonized.
colonized	Testing when the individual is colonized.
latent	Testing when the individual is latent.

Value

list of parameters for random testing.

Functions

- `ClinicalTestParams()`: Clinical Test Parameters Alias

Examples

```
RandomTestParams()
```

runMCMC *Run Bayesian Transmission MCMC*

Description

Run Bayesian Transmission MCMC

Usage

```
runMCMC(
  data,
  modelParameters,
  nsims,
  nburn = 100L,
  outputparam = TRUE,
  outputfinal = FALSE,
  verbose = FALSE
)
```

Arguments

data	Data frame with columns, in order: facility, unit, time, patient, and event type.
modelParameters	List of model parameters, see LogNormalModelParams().
nsims	Number of MCMC samples to collect after burn-in.
nburn	Number of burn-in iterations.
outputparam	Whether to output parameter values at each iteration.
outputfinal	Whether to output the final model state.
verbose	Print progress messages.

Value

A list with the following elements:

- Parameters the MCMC chain of model parameters (if outputparam=TRUE)
- LogLikelihood the log likelihood of the model at each iteration (if outputparam=TRUE)
- MCMCParameters the MCMC parameters used
- ModelParameters the model parameters used
- ModelName the name of the model
- nstates the number of states in the model
- waic1 the WAIC1 estimate
- waic2 the WAIC2 estimate
- and optionally (if outputfinal=TRUE) FinalModel the final model state.

See Also

mcmc_to_dataframe

Examples

```
# Minimal example: create parameters and run a very short MCMC
params <- LinearAbxModel(nstates = 2)
data(simulated.data_sorted, package = "bayestransmission")
results <- runMCMC(
  data = simulated.data_sorted,
  modelParameters = params,
  nsims = 3,
  nburn = 0,
  outputparam = TRUE,
  outputfinal = FALSE,
  verbose = FALSE
)
str(results)
```

---

simulated.data *Simulated Transmission Data*

Description

This data set contains simulated transmission data for a hypothetical infectious disease. The data set contains the following columns:

Usage

```
simulated.data
```

Format

An object of class `data.frame` with 8360 rows and 5 columns.

Details

- `facility`: The facility where the event occurred.
- `unit`: The unit within the facility where the event occurred.
- `time`: The time at which the event occurred.
- `patient`: The patient involved in the event.
- `type`: The type of event.

simulated.data_sorted *Simulated Transmission Data (Sorted)*

Description

This is a sorted version of the simulated transmission data. The data is sorted by time, then by patient. See simulated.data for column descriptions.

Usage

```
simulated.data_sorted
```

Format

An object of class data.frame with 8360 rows and 5 columns.

SurveillanceTestParams *Surveillance Test Parameters*

Description

Specify the rates of positive tests for each state of the model.

Usage

```
SurveillanceTestParams(
  colonized = Param(init = 0.8, weight = 1),
  uncolonized = Param(init = 1e-10, weight = 0),
  latent = Param(init = 0, weight = 0)
)
```

Arguments

colonized	Also known as the true positive rate for a two state model.
uncolonized	Also known as the false positive rate for a two state model.
latent	The rate of positive tests when the individual is in the (optional) latent state.

Value

A list of parameters for surveillance testing.

Examples

```
SurveillanceTestParams()
```

Index