Package: EpiAwareR (via r-universe)

May 14, 2026

Type Package

Title R Interface to the 'EpiAware' Infectious Disease Modelling Framework

Version 0.2.0

Description Provides an R interface to the 'Julia'-based 'EpiAware' compositional infectious disease modelling framework. Enables users to build flexible epidemiological models by composing reusable components for latent processes, infection dynamics, and observation models without requiring 'Julia' expertise.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports JuliaCall (>= 0.17.0), tibble, posterior, ggplot2, checkmate, rlang

Suggests testthat (>= 3.0.0), knitr, rmarkdown, readr, bayesplot, patchwork, EpiNow2

VignetteBuilder knitr

RoxygenNote 7.3.3

SystemRequirements Julia (>= 1.9.0), Julia packages: EpiAware, Turing, Distributions

URL https://github.com/sbfnk/EpiAwareR

BugReports https://github.com/sbfnk/EpiAwareR/issues

Repository https://epiforecasts.r-universe.dev

Date/Publication 2026-04-14 19:05:21 UTC

RemoteUrl https://github.com/sbfnk/EpiAwareR

RemoteRef v0.2.0

RemoteSha 42a4c07ec7ed866539aaa899a159a32867201854

Contents

`AR` *Autoregressive Process for Latent Dynamics*

Description

Constructs an autoregressive (AR) latent process of order p for modeling time-varying log reproduction numbers or other epidemiological parameters.

Usage

```
AR(order = 1, damp_priors, init_priors, std_prior)
```

Arguments

`order`	Integer. The order of the autoregressive process (p).
`damp_priors`	List of distribution specifications for damping coefficients. Length must equal `order`.
`init_priors`	List of distribution specifications for initial states. Length must equal `order`.
`std_prior`	Distribution specification for innovation standard deviation.

Value

An S3 object of class `c("epiaware_ar", "epiaware_latent", "epiaware_model")` containing:

julia_ref Reference to the Julia AR object

spec List of model specifications

See Also

`epiaware_call` for accessing other latent models

Examples

```
## Not run:
# AR(2) model with truncated normal priors (Mishra et al. 2020)
ar2 <- AR(
  order = 2,
  damp_priors = list(
    truncnorm(0.2, 0.2, 0, 1),
    truncnorm(0.1, 0.05, 0, 1)
  ),
  init_priors = list(norm(0, 0.2), norm(0, 0.2)),
  std_prior = halfnorm(0.1)
)
print(ar2)

## End(Not run)
```

`as_draws_array.epiaware_fit`

Convert epiaware_fit to draws_array

Description

Convert epiaware_fit to draws_array

Usage

```
## S3 method for class 'epiaware_fit'
as_draws_array(x, ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments passed to `as_draws_array`.

Value

A `posterior::draws_array` object.

`as_draws_df.epiaware_fit`

Convert epiaware_fit to draws_df

Description

Convert epiaware_fit to draws_df

Usage

```
## S3 method for class 'epiaware_fit'
as_draws_df(x, ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments passed to `as_draws_df`.

Value

A `posterior::draws_df` object.

`as_draws_list.epiaware_fit`

Convert epiaware_fit to draws_list

Description

Convert epiaware_fit to draws_list

Usage

```
## S3 method for class 'epiaware_fit'
as_draws_list(x, ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments passed to `as_draws_list`.

Value

A `posterior::draws_list` object.

`as_draws_matrix.epiaware_fit`

Convert epiaware_fit to draws_matrix

Description

Convert epiaware_fit to draws_matrix

Usage

```
## S3 method for class 'epiaware_fit'
as_draws_matrix(x, ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments passed to `as_draws_matrix`.

Value

A `posterior::draws_matrix` object.

`as_draws_rvars.epiaware_fit` *Convert epiaware_fit to draws_rvars*

Description

Convert epiaware_fit to draws_rvars

Usage

```
## S3 method for class 'epiaware_fit'
as_draws_rvars(x, ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments passed to `as_draws_rvars`.

Value

A `posterior::draws_rvars` object.

`epiaware_available` *Check if Julia and EpiAware are available*

Description

Tests whether Julia is configured and EpiAware packages are accessible.

Usage

```
epiaware_available()
```

Value

Logical. TRUE if Julia and EpiAware are available, FALSE otherwise.

Examples

```
## Not run:
if (epiaware_available()) {
  # Run EpiAware analysis
} else {
  epiaware_setup_julia()
}

## End(Not run)
```

`epiaware_call` *Generic wrapper for calling Julia EpiAware functions*

Description

Provides access to Julia EpiAware functions not yet wrapped in R. This is an ”escape hatch” for accessing newer or experimental EpiAware features before they get explicit R wrappers.

Usage

```
epiaware_call(
  fn_name,
  ...,
  .param_map = NULL,
  .class = c("epiaware_generic", "epiaware_model")
)
```

Arguments

`fn_name`	Character string. Name of the Julia function to call.
`...`	Named arguments to pass to the Julia function.
`.param_map`	Named character vector mapping R parameter names to Julia names. Use this when Julia uses Greek letters or other characters not easily typed in R. Example: c(theta_priors = ” _priors”, eps_t = ” _t”).
`.class`	Character vector. S3 classes to add to the returned object. Default is c(”epiaware_generic”, ”epiaware_model”).

Value

An S3 object containing the Julia reference and specifications.

Examples

```
## Not run:
# Call a hypothetical new EpiAware component not yet wrapped
custom_model <- epiaware_call(
  "NewLatentModel",
  param1 = 0.5,
  param2 = norm(0, 1)
)

# For Julia functions with Greek letters, use parameter mapping
eps_model <- epiaware_call("HierarchicalNormal", halfnorm(0.1))
ma2 <- epiaware_call(
  "MA",
  theta_priors = list(norm(0, 0.1), norm(0, 0.1)),
  eps_t = eps_model,
  .param_map = c(theta_priors = " _priors", eps_t = " _t")
```

```
)

## End(Not run)
```

`epiaware_setup_julia` *Setup Julia and EpiAware*

Description

Configures Julia and installs required Julia packages for EpiAwareR. This function should be run on first use if automatic setup fails.

Usage

```
epiaware_setup_julia(verbose = TRUE)
```

Arguments

`verbose`	Logical. If TRUE, prints progress messages. Default is TRUE.

Value

Invisible TRUE if setup succeeds, otherwise throws an error.

Examples

```
## Not run:
# Setup Julia and install EpiAware packages
epiaware_setup_julia()

## End(Not run)
```

`EpiProblem` *Compose a Complete Epidemiological Model*

Description

Creates an EpiProblem that composes infection, latent, and observation models into a complete epidemiological model specification. The latent model generates time-varying epidemiological parameters, the infection model simulates disease transmission, and the observation model links latent infections to observed data.

Usage

```
EpiProblem(epi_model, latent_model, observation_model, tspan)
```

Arguments

`epi_model`	An infection model object (e.g., from `Renewal`).
`latent_model`	A latent process model (e.g., from `AR`).
`observation_model`	An observation model (e.g., from `NegativeBinomialError`).
`tspan`	A numeric vector of length 2 specifying the time span for inference or simulation: c(start_time, end_time).

Value

An S3 object of class `c("epiaware_problem", "epiaware_model")` containing:

julia_ref Reference to the Julia EpiProblem object

components List containing the three model components

tspan The time span

Examples

```
## Not run:
# Compose a complete epidemiological model (Mishra et al. 2020)
ar2 <- AR(
  order = 2,
  damp_priors = list(truncnorm(0.2, 0.2, 0, 1), truncnorm(0.1, 0.05, 0, 1)),
  init_priors = list(norm(0, 0.2), norm(0, 0.2)),
  std_prior = halfnorm(0.1)
)

renewal <- Renewal(
  gen_distribution = gamma_dist(6.5, 0.62),
  initialisation_prior = norm(log(1.0), 0.1)
)

negbin <- NegativeBinomialError(
  cluster_factor_prior = halfnorm(0.1)
)

model <- EpiProblem(
  epi_model = renewal,
  latent_model = ar2,
  observation_model = negbin,
  tspan = c(45, 80)
)

print(model)

## End(Not run)
```

`exponential` *Exponential distribution*

Description

Specifies an exponential distribution.

Usage

```
exponential(rate)
```

Arguments

`rate`	Numeric. Rate parameter (inverse of mean).

Value

A list specification for an exponential distribution.

Examples

```
## Not run:
# Exponential prior with mean 10
prior <- exponential(1 / 10)

## End(Not run)
```

`fit` *Fit an EpiAware Model to Data*

Description

Performs Bayesian inference on an epidemiological model using MCMC sampling. This function generates a Turing.jl model from the EpiProblem specification, runs the specified inference method, and returns posterior samples with diagnostics.

Usage

```
fit(model, data, method = nuts_sampler(), ...)
```

Arguments

`model`	An `EpiProblem` object specifying the complete model.
`data`	A data frame or list containing observed data. Must have a column/element named `y_t` or `cases` with case counts. Optionally can include `dates`.
`method`	A sampler configuration object (e.g., from `nuts_sampler`). Default: `nuts_sampler()`.
`...`	Additional arguments (currently unused).

Value

An S3 object of class `epiaware_fit` containing:

samples posterior::draws_df object with MCMC samples

summary tibble with parameter summaries

diagnostics tibble with convergence diagnostics (Rhat, ESS)

generated_quantities List with generated quantities (Rt, infections, etc.)

model The original EpiProblem

data The data used for inference

method The inference method used

Examples

```
## Not run:
# Load data
data <- read.csv("south_korea_data.csv")
training_data <- data[45:80, ]

# Fit model
results <- fit(
  model = mishra_model,
  data = training_data,
  method = nuts_sampler(warmup = 1000, draws = 1000, chains = 4)
)

# Examine results
print(results)
summary(results)
plot(results, type = "Rt")

## End(Not run)
```

`gamma_dist` *Gamma distribution*

Description

Specifies a gamma distribution using shape and scale parameterization.

Usage

```
gamma_dist(shape, scale)
```

Arguments

`shape`	Numeric. Shape parameter (alpha).
`scale`	Numeric. Scale parameter (theta). Note: this is scale, not rate.

Value

A list specification for a gamma distribution.

Examples

```
## Not run:
# Generation time distribution with mean 6.5 and scale 0.62
gen_time <- gamma_dist(6.5, 0.62)

## End(Not run)
```

`halfnorm` *Half-normal distribution*

Description

Specifies a half-normal distribution (normal distribution truncated to positive values).

Usage

```
halfnorm(sd)
```

Arguments

`sd`	Numeric. Scale parameter (standard deviation of underlying normal).

Value

A list specification for a half-normal distribution.

Examples

```
## Not run:
# Prior for standard deviation parameters
sigma_prior <- halfnorm(0.1)

## End(Not run)
```

`LatentDelay` *Latent Delay Observation Model*

Description

Wraps an observation model with a reporting delay, convolving the latent infections with a delay distribution. This is used to model delays between infection and observation (e.g., incubation period, reporting delays).

Usage

```
LatentDelay(model, delay_distribution)
```

Arguments

`model`	An observation model object (e.g., from `NegativeBinomialError`).
`delay_distribution`	Distribution specification for the delay. Can be any continuous distribution that will be discretized.

Value

An S3 object of class `c("epiaware_delay", "epiaware_observation", "epiaware_model")` containing:

julia_ref Reference to the Julia LatentDelay object

base_model The wrapped observation model

spec List of model specifications

Examples

```
## Not run:
# Add incubation and reporting delays to observation model
negbin <- NegativeBinomialError(halfnorm(0.1))

# Add incubation delay
incubation_model <- LatentDelay(
  negbin,
  delay_distribution = lognorm(1.6, 0.42)
)

# Add reporting delay
full_model <- LatentDelay(
  incubation_model,
  delay_distribution = lognorm(0.58, 0.47)
)

## End(Not run)
```

`lognorm` *Log-normal distribution*

Description

Specifies a log-normal distribution.

Usage

```
lognorm(meanlog, sdlog)
```

Arguments

`meanlog`	Numeric. Mean of the distribution on the log scale.
`sdlog`	Numeric. Standard deviation of the distribution on the log scale.

Value

A list specification for a log-normal distribution.

Examples

```
## Not run:
# Delay distribution
delay <- lognorm(1.6, 0.42)

## End(Not run)
```

`NegativeBinomialError` *Negative Binomial Observation Error Model*

Description

Links latent infections to observed case counts using a negative binomial distribution. The cluster_factor parameterizes the overdispersion as the coefficient of variation (sqrt(1/phi)), which is more intuitive for setting priors than the dispersion parameter phi directly.

Usage

```
NegativeBinomialError(cluster_factor_prior)
```

Arguments

`cluster_factor_prior`	Distribution specification for the cluster factor (sqrt(1/phi)), which represents the coefficient of variation of observation noise.

Value

An S3 object of class `c("epiaware_negbin", "epiaware_observation", "epiaware_model")` containing:

julia_ref Reference to the Julia NegativeBinomialError object

spec List of model specifications

Examples

```
## Not run:
# Negative binomial observation model
negbin <- NegativeBinomialError(
  cluster_factor_prior = halfnorm(0.1)
)
print(negbin)

## End(Not run)
```

`norm` *Normal distribution*

Description

Specifies a normal (Gaussian) distribution.

Usage

```
norm(mean, sd)
```

Arguments

`mean`	Numeric. Mean of the distribution.
`sd`	Numeric. Standard deviation of the distribution.

Value

A list specification for a normal distribution.

Examples

```
## Not run:
# Standard normal
prior1 <- norm(0, 1)

# Prior for log Rt
prior2 <- norm(log(1.5), 0.2)

## End(Not run)
```

`nuts_sampler` *Configure NUTS Sampler*

Description

Creates a configuration for the No-U-Turn Sampler (NUTS), a variant of Hamiltonian Monte Carlo that automatically tunes step size and number of steps.

Usage

```
nuts_sampler(warmup = 1000, draws = 1000, chains = 4, target_acceptance = 0.9)
```

Arguments

`warmup`	Integer. Number of warmup/adaptation iterations. Default: 1000.
`draws`	Integer. Number of post-warmup samples to draw. Default: 1000.
`chains`	Integer. Number of MCMC chains to run. Default: 4.
`target_acceptance`	Numeric. Target acceptance rate for adaptation. Default: 0.9 (recommended for complex models).

Value

An S3 object of class `c("epiaware_nuts", "epiaware_sampler")` containing sampler configuration.

Examples

```
## Not run:
# Default NUTS configuration
sampler <- nuts_sampler()

# Custom configuration for faster testing
test_sampler <- nuts_sampler(
  warmup = 100,
  draws = 100,
  chains = 2
)

## End(Not run)
```

`plot.epiaware_fit` *Plot Method for Fitted EpiAware Models*

Description

Creates visualizations of fitted model results, including reproduction number trajectories, infection curves, and posterior predictive distributions.

Usage

```
## S3 method for class 'epiaware_fit'
plot(x, type = c("Rt", "cases", "posterior"), ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`type`	Character string specifying plot type. Options: • `"Rt"`: Time-varying reproduction number with credible intervals • `"cases"`: Observed vs predicted cases with credible intervals • `"posterior"`: Posterior distributions for key parameters
`...`	Additional arguments passed to plotting functions.

Value

A ggplot2 object.

Examples

```
## Not run:
# Plot Rt trajectory
plot(results, type = "Rt")

# Plot posterior predictive for cases
plot(results, type = "cases")

# Plot parameter posteriors
plot(results, type = "posterior")

## End(Not run)
```

`print.epiaware_ar` *Print method for AR latent models*

Description

Print method for AR latent models

Usage

```
## S3 method for class 'epiaware_ar'
print(x, ...)
```

Arguments

`x`	An `epiaware_ar` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_delay` *Print method for latent delay observation models*

Description

Print method for latent delay observation models

Usage

```
## S3 method for class 'epiaware_delay'
print(x, ...)
```

Arguments

`x`	An `epiaware_delay` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_fit` *Print Method for Fitted EpiAware Models*

Description

Print Method for Fitted EpiAware Models

Usage

```
## S3 method for class 'epiaware_fit'
print(x, ...)
```

Arguments

`x`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_generic` *Print method for generic EpiAware calls*

Description

Print method for generic EpiAware calls

Usage

```
## S3 method for class 'epiaware_generic'
print(x, ...)
```

Arguments

`x`	An `epiaware_generic` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_negbin` *Print method for negative binomial observation models*

Description

Print method for negative binomial observation models

Usage

```
## S3 method for class 'epiaware_negbin'
print(x, ...)
```

Arguments

`x`	An `epiaware_negbin` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_nuts` *Print method for NUTS sampler configuration*

Description

Print method for NUTS sampler configuration

Usage

```
## S3 method for class 'epiaware_nuts'
print(x, ...)
```

Arguments

`x`	An `epiaware_nuts` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_problem` *Print method for EpiAware problems*

Description

Print method for EpiAware problems

Usage

```
## S3 method for class 'epiaware_problem'
print(x, ...)
```

Arguments

`x`	An `epiaware_problem` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`print.epiaware_renewal` *Print method for renewal infection models*

Description

Print method for renewal infection models

Usage

```
## S3 method for class 'epiaware_renewal'
print(x, ...)
```

Arguments

`x`	An `epiaware_renewal` object.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`Renewal` *Renewal Process Infection Model*

Description

Constructs a renewal process model for infections, where new infections arise from previous infections weighted by a generation time distribution. This implements the renewal equation I_t = R_t * sum(g_s * I_{t-s}).

Usage

```
Renewal(gen_distribution, initialisation_prior = NULL)
```

Arguments

`gen_distribution`	Distribution specification for the generation time (or serial interval). Can be any continuous distribution that will be discretized using double interval censoring.
`initialisation_prior`	Optional distribution specification for the initial infection level (on log scale). If NULL, uses a default prior.

Value

An S3 object of class `c("epiaware_renewal", "epiaware_epi", "epiaware_model")` containing:

julia_ref Reference to the Julia Renewal object

spec List of model specifications

See Also

`epiaware_call` for accessing other infection models

Examples

```
## Not run:
# Renewal model with Gamma generation time (Mishra et al. 2020)
renewal <- Renewal(
  gen_distribution = gamma_dist(6.5, 0.62),
  initialisation_prior = norm(log(1.0), 0.1)
)
print(renewal)

# For advanced features, use the generic wrapper
# to access newer EpiAware infection models
custom_model <- epiaware_call("NewInfectionModel", param1 = ...)
```

```
## End(Not run)
```

`summary.epiaware_fit` *Summary Method for Fitted EpiAware Models*

Description

Summary Method for Fitted EpiAware Models

Usage

```
## S3 method for class 'epiaware_fit'
summary(object, ...)
```

Arguments

`object`	An `epiaware_fit` object from `fit()`.
`...`	Additional arguments (currently unused).

Value

A tibble with parameter summaries.

`truncnorm` *Truncated normal distribution*

Description

Specifies a normal distribution truncated to a specified range.

Usage

```
truncnorm(mean, sd, lower, upper)
```

Arguments

`mean`	Numeric. Mean of the underlying normal distribution.
`sd`	Numeric. Standard deviation of the underlying normal distribution.
`lower`	Numeric. Lower truncation bound.
`upper`	Numeric. Upper truncation bound.

Value

A list specification for a truncated normal distribution.

Examples

```
## Not run:
# AR damping coefficient bounded to [0, 1]
damp_prior <- truncnorm(0.8, 0.2, 0, 1)

## End(Not run)
```

Index