

# Package: ern (via r-universe)

August 21, 2024

**Title** Effective Reproduction Number Estimation

**Version** 2.0.0

**Maintainer** David Champredon <david.champredon@canada.ca>

**Description** Estimate the effective reproduction number from wastewater and clinical data sources.

**License** MIT + file LICENSE

**Imports** assertthat, coda, dplyr, EpiEstim, ggplot2, lubridate, patchwork, rjags, runjags, stats, stringr, tibble, tidyr, zoo

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

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Depends** R (>= 4.1.0)

**LazyData** true

**NeedsCompilation** no

**Author** David Champredon [aut, cre]  
(<<https://orcid.org/0000-0002-7090-8757>>), Warsame Yusuf [aut]  
(<<https://orcid.org/0000-0001-5571-8122>>), Irena Papst [aut]  
(<<https://orcid.org/0000-0001-5901-7585>>)

**Date/Publication** 2024-04-22 13:22:33 UTC

**Repository** <https://davidchampredon.r-universe.dev>

**RemoteUrl** <https://github.com/cran/ern>

**RemoteRef** HEAD

**RemoteSha** a5ac4c421387353d0ac53dace9e2f294a50de4b3

## Contents

agg_to_daily	2
cl.data	4
def_dist	5
estimate_R_cl	6
estimate_R_ww	10
extract_mcmc_values	12
get_discrete_dist	13
linear_int_daily	14
plot_diagnostic_cl	14
plot_diagnostic_ww	16
plot_dist	18
plot_gelman_rubin	19
plot_traces	19
ww.data	20

<b>Index</b>	<b>21</b>
--------------	-----------

---

agg_to_daily	<i>Infer daily counts from aggregates</i>
--------------	---

---

### Description

Infer daily counts from aggregates

### Usage

```
agg_to_daily(cl.data, dist.gi, prm.daily, silent = FALSE)
```

### Arguments

cl.data	Data frame. Must have variables: <ul style="list-style-type: none"> <li>• date: calendar date of report</li> <li>• value: count of reported cases</li> </ul>
dist.gi	List. Parameters for the generation interval distribution in the same format as returned by <code>def_dist()</code> .
prm.daily	List. Parameters for daily report inference via MCMC. Elements include: <ul style="list-style-type: none"> <li>• method: String. Method name to infer the daily incidence reports from aggregated ones. Either <code>linear</code> or <code>renewal</code> is currently implemented. The <code>linear</code> method simply performs a linear interpolation that matches the aggregated values. The <code>renewal</code> method fits a SIR-like model using a renewal equation to infer the daily incidence. In this case, the fitting algorithm is a Markov Chain Monte Carlo (MCMC) implemented in JAGS and needs the parameters below (e.g., <code>burn</code>, <code>iter</code>, <code>chains</code>, ...). The <code>renewal</code> method is more adapted for short single wave epidemics as this models i) naturally fits a single wave and ii) has longer computing time. For longer time series, user may prefer the <code>linear</code> method.</li> </ul>

- popsize: Integer. Population size to use in MCMC simulation to infer daily observations from aggregated input data.
- burn: Numeric. Length of burn-in period (number of days).
- iter: Numeric. Number of iterations after burn-in period (number of days).
- chains: Numeric. Number of chains to simulate.
- prior\_R0\_shape: Shape of the (hyper-)parameter for the prior Gamma distribution for R0.
- prior\_R0\_rate: Rate of the (hyper-)parameter for the prior Gamma distribution for R0.
- prior\_alpha\_shape: Shape of the (hyper-)parameter for the prior Gamma distribution for alpha.
- prior\_alpha\_rate: Rate of the (hyper-)parameter for the prior Gamma distribution for alpha.
- first\_agg\_period: length of aggregation period for first aggregated observation (number of days); if NULL, assume same aggregation period as observed for second observation (gap between first and second observations)

silent            Logical. Flag to suppress all output messages, warnings, and progress bars.

### Value

A list containing a data frame with individual realizations of daily reported cases and the JAGS object.

### Examples

```
# Importing data attached to the `ern` package
# and selecting the Omicron wave in Ontario, Canada.
# This is *weekly* incidence.
data(cl.data)
data = cl.data[cl.data$pt == 'on' &
              cl.data$date > as.Date('2021-11-30') &
              cl.data$date < as.Date('2021-12-31'),]

head(data)
dist.gi = ern::def_dist(
  dist      = "gamma",
  mean      = 6.84,
  mean_sd   = 0.7486,
  shape     = 2.39,
  shape_sd  = 0.3573,
  max       = 15
)

a = agg_to_daily(
  cl.data = data,
  dist.gi = dist.gi,
  prn.daily = list(
    method = "renewal",
    popsize = 14e6,
  )
)
```

```

# MCMC parameters.
# small values for computation speed for this example.
# Increase for better accuracy
burn = 100,
iter = 100,
chains = 2,
# - - - -
prior_R0_shape = 2,
prior_R0_rate = 0.6,
prior_alpha_shape = 1,
prior_alpha_rate = 1
))
# This is a Bayesian inference, so we
# have a posterior distribution of
# daily incidences. Here we just plot
# one single draw:

df = a$df
df1 = df[df$id==1,]
plot(x = df1$t, y = df1$value, typ = 'o',
      xlab = 'days', ylab = 'daily incidence',
      main = 'Posterior daily incidence infered from weekly incidence')

# Extract of the parameters values from the first chain
a$jags.object[[1]][1:9,1:9]

```

---

cl.data

*Sample of aggregated clinical reports*


---

## Description

A subset of COVID-19 weekly reports in the Government of Canada Health Infobase. See <https://health-infobase.canada.ca/covid-19/>

## Usage

```
cl.data
```

## Format

```
cl.data:
```

A data frame with 96 rows and 3 columns:

- pt: standard two-character abbreviation (lowercase) of the province name (based on Statistics Canada 2021 census abbreviations)
- date: report date
- value: count of reported cases for the previous week

Filter indicating a specific province to extract a sample dataset for use with `estimate_R_cl()`, e.g.

```
estimate_R_cl(cl.data = dplyr::filter(cl.data, pt == 'bc'), ...)
```

---

def\_dist *Define a family of distributions.*

---

## Description

Define a family of distributions.

## Usage

```
def_dist(dist, ...)
```

## Arguments

`dist` distribution type. Distributions currently supported are:

- `norm` = normal,
- `lnorm` = log-normal,
- `gamma` = Gamma,
- `unif` = uniform

`...` a series of distribution parameters. Included should be the following:

- `mean` distribution mean (only for `dist` = `lnorm` or `gamma`).
- `mean_sd` standard deviation of the mean (only for `dist` = `lnorm` or `gamma`).
- `sd` standard deviation (only for `dist` = `lnorm` or `gamma`).
- `sd_sd` standard deviation of the standard deviation (only for `dist` = `lnorm` or `gamma`).
- `min` minimum value of the random variable modelled by this distribution (only for `dist` = `unif`).
- `max` maximum value of the random variable modelled by this distribution.

## Value

List with components specified in the parameters.

## Examples

```
d = def_dist(  
  dist = "gamma",  
  mean = 3.49,  
  mean_sd = 0.1477,  
  shape = 8.5,  
  shape_sd = 1.8945,  
  max = 8  
)  
print(d)
```

estimate\_R\_cl

*Estimate the effective reproduction from clinical report data***Description**

Estimate the effective reproduction from clinical report data

**Usage**

```
estimate_R_cl(
  cl.data,
  dist.repdelay,
  dist.repfrac,
  dist.incub,
  dist.gi,
  prm.daily = list(method = "linear", popsize = NULL, burn = 500, iter = 2000, chains =
    3, prior_R0_shape = 2, prior_R0_rate = 0.6, prior_alpha_shape = 1, prior_alpha_rate =
    1, first.agg.period = NULL),
  prm.daily.check = list(agg.reldiff.tol = 10),
  prm.smooth = list(method = "rollmean", align = "right", window = 7),
  prm.R = list(iter = 10, CI = 0.95, window = 7, config.EpiEstim = NULL),
  RL.max.iter = 10,
  silent = FALSE
)
```

**Arguments**

<code>cl.data</code>	Data frame. Must have variables: <ul style="list-style-type: none"> <li>• <code>date</code>: calendar date of report</li> <li>• <code>value</code>: count of reported cases</li> </ul>
<code>dist.repdelay</code>	List. Parameters for the reporting delay distribution in the same format as returned by <code>def_dist()</code> .
<code>dist.repfrac</code>	List. Parameters for the reporting fraction distribution in the same format as returned by <code>def_dist()</code> .
<code>dist.incub</code>	List. Parameters for the incubation period distribution in the same format as returned by <code>def_dist()</code> .
<code>dist.gi</code>	List. Parameters for the generation interval distribution in the same format as returned by <code>def_dist()</code> .
<code>prm.daily</code>	List. Parameters for daily report inference via MCMC. Elements include: <ul style="list-style-type: none"> <li>• <code>method</code>: String. Method name to infer the daily incidence reports from aggregated ones. Either <code>linear</code> or <code>renewal</code> is currently implemented. The <code>linear</code> method simply performs a linear interpolation that matches the aggregated values. The <code>renewal</code> method fits a SIR-like model using a renewal equation to infer the daily incidence. In this case, the fitting algorithm is a Markov Chain Monte Carlo (MCMC) implemented in JAGS and needs the</li> </ul>

parameters below (e.g., `burn`, `iter`, `chains`, ...). The renewal method is more adapted for short single wave epidemics as this models i) naturally fits a single wave and ii) has longer computing time. For longer time series, user may prefer the linear method.

- `popsize`: Integer. Population size to use in MCMC simulation to infer daily observations from aggregated input data.
- `burn`: Numeric. Length of burn-in period (number of days).
- `iter`: Numeric. Number of iterations after burn-in period (number of days).
- `chains`: Numeric. Number of chains to simulate.
- `prior_R0_shape`: Shape of the (hyper-)parameter for the prior Gamma distribution for  $R_0$ .
- `prior_R0_rate`: Rate of the (hyper-)parameter for the prior Gamma distribution for  $R_0$ .
- `prior_alpha_shape`: Shape of the (hyper-)parameter for the prior Gamma distribution for  $\alpha$ .
- `prior_alpha_rate`: Rate of the (hyper-)parameter for the prior Gamma distribution for  $\alpha$ .
- `first_agg_period`: length of aggregation period for first aggregated observation (number of days); if NULL, assume same aggregation period as observed for second observation (gap between first and second observations)

#### `prm.daily.check`

List. Parameters for checking aggregated to daily report inference. Elements include:

- `agg.reldiff.tol`: numerical tolerance (%) for relative error between aggregated inferred daily reports and original aggregated reports; chronological observations are dropped until this tolerance is first achieved (convergence at the start of the timeseries is often the worst, need to maintain uninterrupted daily timeseries for input into  $R_t$  calculation).

Set this entire argument to NULL to use inferred daily reports as is.

#### `prm.smooth`

List. list of smoothing parameters. Parameters should be specified as followed:

- `method`: smoothing method, either 'rollmean' (rolling mean) or 'loess' (LOESS smoothing via `stats::loess()`)
- `window`: for method = 'rollmean only'; width of smoothing window in days
- `align`: for method = 'rollmean only'; smoothing alignment, either 'center', 'left', 'right'
- `span`: for method = 'loess' only; smoothing span (see the documentation for `stats::loess()` for details)
- `floor`: optional call for wastewater concentration smoothing with method = 'loess' only; user defined minimum smoothing concentration

Set this entire list to NULL to turn off smoothing

#### `prm.R`

List. Settings for the ensemble when calculating  $R_t$ . Elements include:

- `iter`: Integer. Number of iterations for the  $R_t$  ensemble

	<ul style="list-style-type: none"> <li>• CI: Numeric between 0 and 1. Confidence interval width for Rt estimates after sampling uncertain distributions.</li> <li>• window: Integer. Number of days defining the window of data used by EpiEstim to estimate Rt. If NULL, will default to 7.</li> <li>• config.EpiEstim: (optional) configuration for EpiEstim defined via <code>EpiEstim::make_config()</code>. If NULL, will use default config from EpiEstim.</li> </ul>
RL.max.iter	Integer. Maximum of iterations for the Richardson-Lucy deconvolution algorithm.
silent	Logical. Flag to suppress all output messages, warnings, and progress bars.

### Value

List. Elements include:

- cl.data: original aggregated reports signal
- cl.daily: reports as input for Rt calculation (inferred daily counts, smoothed)
- inferred.agg: inferred daily reports aggregated on the reporting schedule as input in cl.data
- R: the effective R estimate (summary from ensemble)

### See Also

[plot\\_diagnostic\\_cl\(\)](#) [estimate\\_R\\_ww\(\)](#)

### Examples

```
# -- THIS EXAMPLE TAKES ABOUT 30 SECONDS TO RUN --
# Estimate Rt

## Not run:
# Load SARS-CoV-2 reported cases in Quebec
# during the Summer 2021
dat <- (ern::cl.data
  |> dplyr::filter(
    pt == "qc",
    dplyr::between(date, as.Date("2021-06-01"), as.Date("2021-09-01"))
  )
)
# distributions
dist.repdelay = ern::def_dist(
  dist = 'gamma',
  mean = 5,
  mean_sd = 1,
  sd = 1,
  sd_sd = 0.1,
  max = 10
)
dist.repfrac = ern::def_dist(
  dist = "unif",
  min = 0.1,
  max = 0.3
)
```

```
)
dist.incub = ern::def_dist(
  dist = "gamma",
  mean = 3.49,
  mean_sd = 0.1477,
  shape = 8.5,
  shape_sd = 1.8945,
  max = 8
)
dist.gi = ern::def_dist(
  dist = "gamma",
  mean = 6,
  mean_sd = 0.75,
  shape = 2.4,
  shape_sd = 0.3,
  max = 10
)

# settings
prm.daily <- list(
  method = "renewal",
  popsize = 8.5e6, # Q3 (July 1) 2022 estimate for Quebec
  burn = 500,
  iter = 500,
  chains = 2,
  prior_R0_shape = 1.1, prior_R0_rate = 0.6,
  prior_alpha_shape = 1, prior_alpha_rate = 1
)
prm.daily.check <- list(
  agg.reldiff.tol = 10
)
prm.smooth <- list(
  method = "rollmean",
  align = "center",
  window = 7
)
prm.R <- list(
  iter = 20,
  CI = 0.95,
  window = 7,
  config.EpiEstim = NULL
)

x <- estimate_R_cl(
  dat,
  dist.repdelay,
  dist.repfrac,
  dist.incub,
  dist.gi,
  prm.daily,
  prm.daily.check,
  prm.smooth,
  prm.R
)
```

```

)

# Rt estimates
print(x$R)

## End(Not run)

```

---

estimate_R_ww	<i>Estimate the effective reproduction from wastewater concentration data.</i>
---------------	--

---

### Description

Estimate the effective reproduction from wastewater concentration data.

### Usage

```

estimate_R_ww(
  ww.conc,
  dist.fec,
  dist.gi,
  scaling.factor = 1,
  prm.smooth = list(window = 14, align = "center", method = "loess", span = 0.2),
  prm.R = list(iter = 10, CI = 0.95, window = 7, config.EpiEstim = NULL),
  silent = FALSE,
  RL.max.iter = 9
)

```

### Arguments

ww.conc	Data frame. Must have variables: <ul style="list-style-type: none"> <li>• date: calendar date of wastewater collection</li> <li>• value: pathogen concentration</li> </ul>
dist.fec	List. Parameters for the fecal shedding distribution in the same format as returned by <code>def_dist()</code> .
dist.gi	List. Parameters for the generation interval distribution in the same format as returned by <code>def_dist()</code> .
scaling.factor	Numeric. Scaling from wastewater concentration to prevalence. This value may be assumed or independently calibrated to data.
prm.smooth	List. list of smoothing parameters. Parameters should be specified as followed: <ul style="list-style-type: none"> <li>• method: smoothing method, either 'rollmean' (rolling mean) or 'loess' (LOESS smoothing via <code>stats::loess()</code>)</li> </ul>

- window: for method = 'rollmean only; width of smoothing window in days
- align: for method = 'rollmean only; smoothing alignment, either 'center', 'left', 'right'
- span: for method = 'loess' only; smoothing span (see the documentation for `stats::loess()` for details)
- floor: optional call for wastewater concentration smoothing with method = 'loess' only; user defined minimum smoothing concentration

Set this entire list to NULL to turn off smoothing

prm.R List. Settings for the ensemble when calculating Rt. Elements include:

- iter: Integer. Number of iterations for the Rt ensemble
- CI: Numeric between 0 and 1. Confidence interval width for Rt estimates after sampling uncertain distributions.
- window: Integer. Number of days defining the window of data used by EpiEstim to estimate Rt. If NULL, will default to 7.
- config.EpiEstim: (optional) configuration for EpiEstim defined via `EpiEstim::make_config()`. If NULL, will use default config from EpiEstim.

silent Logical. Flag to suppress all output messages, warnings, and progress bars.

RL.max.iter Integer. Maximum of iterations for the Richardson-Lucy deconvolution algorithm.

## Value

List. Elements include:

- ww.conc: original wastewater signal
- ww.smooth: smoothed wastewater signal
- inc: inferred incidence
- R: the effective reproduction number estimate

## See Also

`plot_diagnostic_ww()` `estimate_R_cl()`

## Examples

```
# Load data of viral concentration in wastewater
data("ww.data")

# Run the estimation of Rt based on the wastewater data
x = estimate_R_ww(
  ww.conc = ww.data,
  dist.fec = ern::def_dist(
    dist = "gamma",
    mean = 12.90215,
    mean_sd = 1.136829,
    shape = 1.759937,
```

```
      shape_sd = 0.2665988,  
      max = 33  
    ),  
    dist.gi = ern::def_dist(  
      dist = "gamma",  
      mean = 6.84,  
      mean_sd = 0.7486,  
      shape = 2.39,  
      shape_sd = 0.3573,  
      max = 15  
    ),  
    silent = TRUE  
  )  
  
  # Rt estimates  
  head(x$R)  
  
  # inferred daily incidence  
  head(x$inc)
```

---

extract\_mcmc\_values    *Extract MCMC chains from a JAGS object*

---

## Description

Extract MCMC chains from a JAGS object

## Usage

```
extract_mcmc_values(chain, jags.obj)
```

## Arguments

chain	Integer. Chain number.
jags.obj	JAGS object as returned by code <code>.sample()</code>

## Value

A dataframe of the chain values for selected parameters.

---

get_discrete_dist	<i>Get a discretized, truncated version of a distribution</i>
-------------------	---

---

**Description**

Get a discretized, truncated version of a distribution

**Usage**

```
get_discrete_dist(params)
```

**Arguments**

params            distribution params (output of def\_dist\_\*( ) function)

**Value**

Numeric. Vector with discretized density.

**Examples**

```
# Define distributions
fec = ern::def_dist(
  dist = "gamma",
  mean = 12.90215,
  mean_sd = 1.136829,
  shape = 1.759937,
  shape_sd = 0.2665988,
  max = 33
)
gi = ern::def_dist(
  dist = "gamma",
  mean = 6.84,
  mean_sd = 0.7486,
  shape = 2.39,
  shape_sd = 0.3573,
  max = 15
)

# Get their (discretized) densities
d.fec = get_discrete_dist(fec)
d.gi = get_discrete_dist(gi)

print(d.fec)
print(d.gi)
```

---

linear_int_daily	<i>Daily incidence from linear interpolation</i>
------------------	--

---

**Description**

Daily incidence from linear interpolation

**Usage**

```
linear_int_daily(cl.data)
```

**Arguments**

cl.data            Aggregated incidence.

**Value**

A dataframe of daily incidence

---

plot_diagnostic_cl	<i>Diagnostic plot for R estimation from clinical report data</i>
--------------------	---

---

**Description**

Diagnostic plot for R estimation from clinical report data

**Usage**

```
plot_diagnostic_cl(r.estim, caption = NULL, wrap.plots = TRUE)
```

**Arguments**

r.estim            List. Output of `estimate_R_cl()`.

caption            String. Caption to be inserted in the plot. Default is `caption = NULL` which disables the caption.

wrap.plots        Logical. Wrap the plots together into a single ggplot object? If `wrap.plots = TRUE` (the default) will return wrapped plots in a single object, else will return a list of separate ggplot objects.

**Value**

Plots of the clinical data used, the inferred daily incidence and Rt estimates. If `wrap.plots = TRUE` (the default) will return wrapped plots (with x-axis aligned to facilitate the comparison) in a single object, else will return a list of separate ggplot objects.

A ggplot object (or a list of ggplot objects if `wrap.plots = FALSE`).

**See Also**[estimate\\_R\\_cl\(\)](#)**Examples**

```
# -- THIS EXAMPLE TAKES ABOUT 30 SECONDS TO RUN --
# Estimate Rt

## Not run:
# Load SARS-CoV-2 reported cases in Quebec
# during the Summer 2021
dat <- (ern::cl.data
  |> dplyr::filter(
    pt == "qc",
    dplyr::between(date, as.Date("2021-06-01"), as.Date("2021-09-01"))
  )
)
# distributions
dist.repdelay = ern::def_dist(
  dist = 'gamma',
  mean = 5,
  mean_sd = 1,
  sd = 1,
  sd_sd = 0.1,
  max = 10
)
dist.repfrac = ern::def_dist(
  dist = "unif",
  min = 0.1,
  max = 0.3
)
dist.incub = ern::def_dist(
  dist = "gamma",
  mean = 3.49,
  mean_sd = 0.1477,
  shape = 8.5,
  shape_sd = 1.8945,
  max = 8
)
dist.gi = ern::def_dist(
  dist = "gamma",
  mean = 6,
  mean_sd = 0.75,
  shape = 2.4,
  shape_sd = 0.3,
  max = 10
)

# settings
prm.daily <- list(
  method = "renewal",
  popsize = 8.5e6, # Q3 (July 1) 2022 estimate for Quebec
```

```

    burn = 500,
    iter = 500,
    chains = 2,
    prior_R0_shape = 1.1, prior_R0_rate = 0.6,
    prior_alpha_shape = 1, prior_alpha_rate = 1
  )
  prm.daily.check <- list(
    agg.reldiff.tol = 10
  )
  prm.smooth <- list(
    method = "rollmean",
    align = "center",
    window = 7
  )
  prm.R <- list(
    iter = 20,
    CI = 0.95,
    window = 7,
    config.EpiEstim = NULL
  )
  x <- estimate_R_cl(
    dat,
    dist.repdelay,
    dist.repfrac,
    dist.incub,
    dist.gi,
    prm.daily,
    prm.daily.check,
    prm.smooth,
    prm.R
  )

  # Diagnostic plot for Rt estimates
  # from clinical data
  g = plot_diagnostic_cl(x)
  plot(g)

  g2 = plot_diagnostic_cl(x, caption = 'This is your caption', wrap.plots = FALSE)
  plot(g2$clinical_data)
  plot(g2$inferred_incidence)
  plot(g2$Rt)

  ## End(Not run)

```

---

plot\_diagnostic\_ww      *Diagnostic plot for R estimation from wastewater data*

---

### Description

Diagnostic plot for R estimation from wastewater data

**Usage**

```
plot_diagnostic_ww(r.estim, caption = NULL, wrap.plots = TRUE)
```

**Arguments**

r.estim	List. Output of <code>estimate_R_ww()</code> .
caption	Character. Optional plot caption.
wrap.plots	Logical. Wrap all diagnostic plots into one single ggplot object (default = TRUE).

**Value**

A ggplot object.

**See Also**

[estimate\\_R\\_ww\(\)](#) [plot\\_diagnostic\\_cl\(\)](#)

**Examples**

```
# Load data of viral concentration in wastewater
data("ww.data")

# Estimate Rt based on wastewater data
x = estimate_R_ww(
  ww.conc = ww.data,
  dist.fec = ern::def_dist(
    dist = "gamma",
    mean = 12.9,
    mean_sd = 1.13,
    shape = 1.75,
    shape_sd = 0.26,
    max = 33
  ),
  dist.gi = ern::def_dist(
    dist = "gamma",
    mean = 6.84,
    mean_sd = 0.74,
    shape = 2.39,
    shape_sd = 0.35,
    max = 15
  ),
  silent = TRUE
)

# Diagnostic plot
g = plot_diagnostic_ww(x)
plot(g)

g2 = plot_diagnostic_ww(x, wrap.plots = FALSE, caption = "This is your caption")
plot(g2$wastewater_data)
plot(g2$inferred_incidence)
```

```
plot(g2$Rt)
```

---

plot\_dist

*Plot a distribution*

---

### Description

Plot a distribution

### Usage

```
plot_dist(d)
```

### Arguments

d List that defines the distribution (as returned by `def_dist_incubation_period()` for example)

### Value

A ggplot object.

### Examples

```
# Define a `ern` distribution:
gi = ern::def_dist(
  dist = "gamma",
  mean = 6.84,
  mean_sd = 0.7486,
  shape = 2.39,
  shape_sd = 0.3573,
  max = 15
)

# Plot can be customized like any `ggplot` object:
g = plot_dist(gi) + ggplot2::labs(subtitle = 'your subtitle')
plot(g)
```

---

plot_gelman_rubin	<i>Plot the Gelman Rubin statistic for all parameters.</i>
-------------------	--

---

**Description**

Plot the Gelman Rubin statistic for all parameters.

**Usage**

```
plot_gelman_rubin(jags.obj)
```

**Arguments**

jags.obj      JAGS object as returned by code .sample()

**Value**

A ggplot plot.

---

plot_traces	<i>Plot MCMC traces</i>
-------------	-------------------------

---

**Description**

Plot MCMC traces

**Usage**

```
plot_traces(jags.obj)
```

**Arguments**

jags.obj      JAGS object as returned by code .sample()

**Value**

A ggplot plot.

---

`ww.data`*Sample of wastewater concentration*

---

**Description**

A subset of SARS-CoV-2 (N2 gene) concentration data in wastewater sampled from the Iona Island wastewater treatment plant in Vancouver between 7 July 2023 and 5 November 2023. Units are in N2 gene copies per milliliter of wastewater. Concentration was measured using RT-qPCR assays; RNA was extracted from suspended solids. See <https://health-infobase.canada.ca/covid-19/wastewater/>

**Usage**`ww.data`**Format**`ww.data:`

A data frame with 47 rows and 3 columns:

- `date`: sampling date
- `value`: mean sample concentration between multiple replicates

# Index

## \* datasets

cl.data, [4](#)

ww.data, [20](#)

agg\_to\_daily, [2](#)

cl.data, [4](#)

def\_dist, [5](#)

def\_dist(), [2](#), [6](#), [10](#)

EpiEstim::make\_config(), [8](#), [11](#)

estimate\_R\_cl, [6](#)

estimate\_R\_cl(), [5](#), [11](#), [14](#), [15](#)

estimate\_R\_ww, [10](#)

estimate\_R\_ww(), [8](#), [17](#)

extract\_mcmc\_values, [12](#)

get\_discrete\_dist, [13](#)

linear\_int\_daily, [14](#)

plot\_diagnostic\_cl, [14](#)

plot\_diagnostic\_cl(), [8](#), [17](#)

plot\_diagnostic\_ww, [16](#)

plot\_diagnostic\_ww(), [11](#)

plot\_dist, [18](#)

plot\_gelman\_rubin, [19](#)

plot\_traces, [19](#)

stats::loess(), [7](#), [10](#), [11](#)

ww.data, [20](#)