Package 'superspreading'

January 27, 2025

Title Understand Individual-Level Variation in Infectious Disease Transmission

Version 0.3.0

Description Estimate and understand individual-level variation in transmission. Implements density and cumulative compound Poisson discrete distribution functions ('Kremer et al.' (2021) <doi:10.1038/s41598-021-93578-x>), as well as functions to calculate infectious disease outbreak statistics given epidemiological parameters on individual-level transmission; including the probability of an outbreak becoming an epidemic/extinct ('Kucharski et al.' (2020) <doi:10.1016/S1473-3099(20)30144-4>), or the cluster size statistics, e.g. what proportion of cases cause X\% of transmission ('Lloyd-Smith et al.' (2005) <doi:10.1038/nature04153>).

License MIT + file LICENSE

URL https://github.com/epiverse-trace/superspreading, https://epiverse-trace.github.io/superspreading/

BugReports https://github.com/epiverse-trace/superspreading/issues

Imports checkmate, rlang, stats

Suggests dplyr, epiparameter (>= 0.4.0), fitdistrplus, ggplot2, ggtext, knitr, purrr, rmarkdown, scales, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website epiverse-trace/epiversetheme

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.2

NeedsCompilation no

Author Joshua W. Lambert [aut, cre, cph] (<https://orcid.org/0000-0001-5218-3046>),

Adam Kucharski [aut, cph] (<https://orcid.org/0000-0001-8814-9421>),
Dillon C. Adam [aut] (<https://orcid.org/0000-0002-7485-9905>),
Sebastian Funk [ctb, cph] (<https://orcid.org/0000-0002-2842-3406>,
.chain_sim uses code from bpmodels::chain_sim),
Pratik Gupte [rev] (<https://orcid.org/0000-0001-5294-7819>),
Hugo Gruson [rev] (<https://orcid.org/0000-0002-4094-1476>),
James M. Azam [rev, ctb] (<https://orcid.org/0000-0001-5782-7330>),
Chris Hartgerink [rev] (<https://orcid.org/0000-0003-1050-6809>)

Maintainer Joshua W. Lambert <joshua.lambert@lshtm.ac.uk>

Repository CRAN

Date/Publication 2025-01-27 18:40:02 UTC

Contents

`calc_network_R` *Calculate the reproduction number (R) for a (heterogeneous) network*

Description

The calculation of the reproduction number adjusting for heterogeneity in number of contacts.

Usage

```
calc_network_R(
  mean_num_contact,
  sd_num_contact,
  infect_duration,
  prob_transmission,
  age_range
)
```

Arguments

mean_num_contact	A numeric, mean (average) number of new contacts per unit time.
sd_num_contact	A numeric, standard deviation of the number of new contacts per unit time.
infect_duration	A numeric, the duration of infectiousness.
prob_transmission	A numeric probability of transmission per contact, also known as β.
age_range	A numeric vector with two elements, the lower and upper age limits of individuals in the network.

Value

A named numeric vector of length 2, the unadjusted (R) and network adjusted (R_net) estimates of R.

Examples

```
# example using NATSAL data
calc_network_R(
  mean_num_contact = 14.1,
  sd_num_contact = 69.6,
  infect_duration = 1,
  prob_transmission = 1,
  age_range = c(16, 74)
)
```

---

constants *Constants used in* **superspreading**

Description

FINITE_INF is a large finite number used to approximate Inf.

NSIM is the number of simulations run when generating random samples or branching process simulation replicates.

Usage

```
FINITE_INF

NSIM
```

Format

An object of class numeric of length 1.

An object of class numeric of length 1.

dpoislnorm *Density of the poisson-lognormal compound distribution*

Description

Density of the poisson-lognormal compound distribution

Usage

```
dpoislnorm(x, meanlog, sdlog)
```

Arguments

x	A number for the quantile of the distribution.
meanlog	A number for the mean of the distribution on the log scale.
sdlog	A number for the standard deviation of the distribution on the log scale.

Details

The function is vectorised so a vector of quantiles can be input and the output will have an equal length.

Value

A numeric vector of the density of the poisson-lognormal distribution.

Examples

```
dpoislnorm(x = 10, meanlog = 1, sdlog = 2)
dpoislnorm(x = 1:10, meanlog = 1, sdlog = 2)
```

dpoisweibull *Density of the poisson-Weibull compound distribution*

Description

Density of the poisson-Weibull compound distribution

Usage

```
dpoisweibull(x, shape, scale)
```

Arguments

x	A number for the quantile of the distribution.
shape	A number for the shape parameter of the distribution.
scale	A number for the scale parameter of the distribution.

Details

The function is vectorised so a vector of quantiles can be input and the output will have an equal length.

Value

A `numeric` vector of the density of the poisson-Weibull distribution.

Examples

```
dpoisweibull(x = 10, shape = 1, scale = 2)
dpoisweibull(x = 1:10, shape = 1, scale = 2)
```

---

ic_tbl *Helper function to create a model comparison table*

Description

This is a helper function for creating a model comparison `<data.frame>` primarily for use in the **superspreading** vignettes. It is designed specifically for handling `fitdistrplus::fitdist()` output and not a generalised function. See `bbmle::ICtab()` for a more general use function to create information criteria tables.

Usage

```
ic_tbl(..., sort_by = c("AIC", "BIC", "none"))
```

Arguments

`...`	dots One or more model fit results from `fitdistrplus::fitdist()`.
`sort_by`	A `character` string specifying which information criterion to order the table by, either `"AIC"` (default), `"BIC"`, or `"none"` (i.e. no ordering).

Value

A `<data.frame>`.

Examples

```
if (requireNamespace("fitdistrplus", quietly = TRUE)) {
  cases <- rnbinom(n = 100, mu = 5, size = 0.7)
  pois_fit <- fitdistrplus::fitdist(data = cases, distr = "pois")
  geom_fit <- fitdistrplus::fitdist(data = cases, distr = "geom")
  nbinom_fit <- fitdistrplus::fitdist(data = cases, distr = "nbinom")
  ic_tbl(pois_fit, geom_fit, nbinom_fit)
}
```

ppoislnorm *Cumulative distribution function of the poisson-lognormal compound distribution*

Description

Cumulative distribution function of the poisson-lognormal compound distribution

Usage

```
ppoislnorm(q, meanlog, sdlog)
```

Arguments

q	A number for the quantile of the distribution.
meanlog	A number for the mean of the distribution on the log scale.
sdlog	A number for the standard deviation of the distribution on the log scale.

Details

The function is vectorised so a vector of quantiles can be input and the output will have an equal length.

Value

A numeric vector of the distribution function.

Examples

```
ppoislnorm(q = 10, meanlog = 1, sdlog = 2)
ppoislnorm(q = 1:10, meanlog = 1, sdlog = 2)
```

ppoisweibull *Cumulative distribution function of the poisson-Weibull compound distribution*

Description

Cumulative distribution function of the poisson-Weibull compound distribution

Usage

```
ppoisweibull(q, shape, scale)
```

Arguments

q	A number for the quantile of the distribution.
shape	A number for the shape parameter of the distribution.
scale	A number for the scale parameter of the distribution.

Details

The function is vectorised so a vector of quantiles can be input and the output will have an equal length.

Value

A numeric vector of the distribution function.

Examples

```
ppoisweibull(q = 10, shape = 1, scale = 2)
ppoisweibull(q = 1:10, shape = 1, scale = 2)
```

---

probability_contain *Probability that an outbreak will be contained*

Description

Outbreak containment is defined as outbreak extinction when simulate = FALSE. When simulate = FALSE, probability_contain() is equivalent to calling probability_extinct().

When simulate = TRUE, outbreak containment is defined by the case_threshold (default = 100) and outbreak_time arguments. Firstly, case_threshold sets the size of the transmission chain below which the outbreak is considered contained. Secondly, outbreak_time sets the time duration from the start of the outbreak within which the outbreak is contained if there is no more onwards transmission beyond this time. When setting an outbreak_time, a generation_time is also required. case_threshold and outbreak_time can be jointly set. Overall, when simulate = TRUE, containment is defined as the size and time duration of a transmission chain not reaching the case_threshold and outbreak_time, respectively.

Usage

```
probability_contain(
  R,
  k,
  num_init_infect,
  ind_control = 0,
  pop_control = 0,
  simulate = FALSE,
  ...,
  case_threshold = 100,
```

```
  outbreak_time = Inf,
  generation_time = NULL,
  offspring_dist
)
```

Arguments

R	A number specifying the R parameter (i.e. average secondary cases per infectious individual).
k	A number specifying the k parameter (i.e. overdispersion in offspring distribution from fitted negative binomial).
num_init_infect	An integer (or at least "integerish" if stored as double) specifying the number of initial infections.
ind_control	A numeric specifying the strength of individual-level control measures. These control measures assume that infected individuals do not produce any secondary infections with probability ind_control, thus increasing the proportion of cases that do not create any subsequent infections. The control measure is between 0 (default) and 1 (maximum).
pop_control	A numeric specifying the strength of population-level control measures that reduce the transmissibility of all cases by a constant factor. Between 0 (default) and 1 (maximum).
simulate	A logical boolean determining whether the probability of containment is calculated analytically or numerically using a stochastic branching process model. Default is FALSE which calls probability_extinct(), setting to TRUE uses a branching process and enables setting the case_threshold, outbreak_time and generation_time arguments.
...	<dynamic-dots> Named elements to replace default arguments in .chain_sim(). See details.
case_threshold	A number for the threshold of the number of cases below which the epidemic is considered contained. case_threshold is only used when simulate = TRUE.
outbreak_time	A number for the time since the start of the outbreak to determine if outbreaks are contained within a given period of time. outbreak_time is only used when simulate = TRUE.
generation_time	A function to generate generation times. The function must have a single argument and return a numeric vector with generation times. See details for example. The function can be defined or anonymous. generation_time is only used when simulate = TRUE.
offspring_dist	An <epiparameter> object. An S3 class for working with epidemiological parameters/distributions, see epiparameter::epiparameter().

Details

When using simulate = TRUE, the default arguments to simulate the transmission chains with .chain_sim() are 105 replicates, a negative binomial (nbinom) offspring distribution, parameterised with R (and pop_control if > 0) and k.

When setting the outbreak_time argument, the generation_time argument is also required. The generation_time argument requires a random number generator function. For example, if we assume the generation time is lognormally distributed with meanlog = 1 and sdlog = 1.5, then we can define the function to pass to generation_time as:

```
function(x) rlnorm(x, meanlog = 1, sdlog = 1.5)
```

Value

A number for the probability of containment.

References

Lloyd-Smith, J. O., Schreiber, S. J., Kopp, P. E., & Getz, W. M. (2005) Superspreading and the effect of individual variation on disease emergence. Nature, 438(7066), 355-359. doi:10.1038/nature04153

See Also

probability_extinct()

Examples

```
# population-level control measures
probability_contain(R = 1.5, k = 0.5, num_init_infect = 1, pop_control = 0.1)

# individual-level control measures
probability_contain(R = 1.5, k = 0.5, num_init_infect = 1, ind_control = 0.1)

# both levels of control measures
probability_contain(
  R = 1.5,
  k = 0.5,
  num_init_infect = 1,
  ind_control = 0.1,
  pop_control = 0.1
)

# multi initial infections with population-level control measures
probability_contain(R = 1.5, k = 0.5, num_init_infect = 5, pop_control = 0.1)

# probability of containment within a certain amount of time
# this requires parameterising a generation time
gt <- function(n) {
  rlnorm(n, meanlog = 1, sdlog = 1.5)
}
probability_contain(
  R = 1.2,
  k = 0.5,
  num_init_infect = 1,
  simulate = TRUE,
  case_threshold = 50,
```

```
  outbreak_time = 20,
  generation_time = gt
)
```

`probability_epidemic` *Calculate the probability a disease will cause an outbreak based on R, k and initial cases*

Description

Calculates the probability a branching process will cause an epidemic (i.e. probability will fail to go extinct) based on R, k and initial cases.

Usage

```
probability_epidemic(
  R,
  k,
  num_init_infect,
  ind_control = 0,
  pop_control = 0,
  ...,
  offspring_dist
)
```

Arguments

`R`	A `number` specifying the R parameter (i.e. average secondary cases per infectious individual).
`k`	A `number` specifying the k parameter (i.e. overdispersion in offspring distribution from fitted negative binomial).
`num_init_infect`	An `integer` (or at least "integerish" if stored as double) specifying the number of initial infections.
`ind_control`	A `numeric` specifying the strength of individual-level control measures. These control measures assume that infected individuals do not produce any secondary infections with probability `ind_control`, thus increasing the proportion of cases that do not create any subsequent infections. The control measure is between `0` (default) and `1` (maximum).
`pop_control`	A `numeric` specifying the strength of population-level control measures that reduce the transmissibility of all cases by a constant factor. Between `0` (default) and `1` (maximum).
`...`	<dynamic-dots> Named elements to replace default optimisation settings. Currently only `"fit_method"` is accepted and can be either `"optim"` (default) or `"grid"` for numerical optimisation routine or grid search, respectively.
`offspring_dist`	An <epiparameter> object. An S3 class for working with epidemiological parameters/distributions, see `epiparameter::epiparameter()`.

Value

A value with the probability of a large epidemic.

References

Lloyd-Smith, J. O., Schreiber, S. J., Kopp, P. E., & Getz, W. M. (2005) Superspreading and the effect of individual variation on disease emergence. Nature, 438(7066), 355-359. doi:10.1038/nature04153

Kucharski, A. J., Russell, T. W., Diamond, C., Liu, Y., Edmunds, J., Funk, S. & Eggo, R. M. (2020). Early dynamics of transmission and control of COVID-19: a mathematical modelling study. The Lancet Infectious Diseases, 20(5), 553-558. doi:10.1016/S14733099(20)301444

See Also

`probability_extinct()`

Examples

```
probability_epidemic(R = 1.5, k = 0.1, num_init_infect = 10)
```

`probability_extinct` *Calculate the probability a branching process will go extinct based on R, k and initial cases*

Description

Calculates the probability a branching process will not causes an epidemic and will go extinct. This is the complement of the probability of a disease causing an epidemic (`probability_epidemic()`).

Usage

```
probability_extinct(
  R,
  k,
  num_init_infect,
  ind_control = 0,
  pop_control = 0,
  ...,
  offspring_dist
)
```

Arguments

`R`	A `number` specifying the R parameter (i.e. average secondary cases per infectious individual).
`k`	A `number` specifying the k parameter (i.e. overdispersion in offspring distribution from fitted negative binomial).

num_init_infect
: An integer (or at least "integerish" if stored as double) specifying the number of initial infections.

ind_control
: A numeric specifying the strength of individual-level control measures. These control measures assume that infected individuals do not produce any secondary infections with probability ind_control, thus increasing the proportion of cases that do not create any subsequent infections. The control measure is between 0 (default) and 1 (maximum).

pop_control
: A numeric specifying the strength of population-level control measures that reduce the transmissibility of all cases by a constant factor. Between 0 (default) and 1 (maximum).

...
: <dynamic-dots> Named elements to replace default optimisation settings. Currently only "fit_method" is accepted and can be either "optim" (default) or "grid" for numerical optimisation routine or grid search, respectively.

offspring_dist
: An <epiparameter> object. An S3 class for working with epidemiological parameters/distributions, see epiparameter::epiparameter().

Value

A value with the probability of going extinct.

References

Lloyd-Smith, J. O., Schreiber, S. J., Kopp, P. E., & Getz, W. M. (2005). Superspreading and the effect of individual variation on disease emergence. Nature, 438(7066), 355-359. doi:10.1038/nature04153

See Also

probability_epidemic()

Examples

```
probability_extinct(R = 1.5, k = 0.1, num_init_infect = 10)
```

---

proportion_cluster_size

Estimate what proportion of new cases originated within a transmission event of a given size

---

Description

Calculates the proportion of new cases that originated with a transmission event of a given size. It can be useful to inform backwards contact tracing efforts, i.e. how many cases are associated with large clusters. Here we define a cluster to as a transmission of a primary case to at least one secondary case.

Usage

```
proportion_cluster_size(
  R,
  k,
  cluster_size,
  ...,
  offspring_dist,
  format_prop = TRUE
)
```

Arguments

R	A number specifying the R parameter (i.e. average secondary cases per infectious individual).
k	A number specifying the k parameter (i.e. overdispersion in offspring distribution from fitted negative binomial).
cluster_size	A number for the cluster size threshold.
...	dots not used, extra arguments supplied will cause a warning.
offspring_dist	An <epiparameter> object. An S3 class for working with epidemiological parameters/distributions, see epiparameter::epiparameter().
format_prop	A logical determining whether the proportion column of the <data.frame> returned by the function is formatted as a string with a percentage sign (%), (TRUE, default), or as a numeric (FALSE).

Details

This function calculates the proportion of secondary cases that are caused by transmission events of a certain size. It does not calculate the proportion of transmission events that cause a cluster of secondary cases of a certain size. In other words it is the number of cases above a threshold divided by the total number of cases, not the number of transmission events above a certain threshold divided by the number of transmission events.

Value

A <data.frame> with the value for the proportion of new cases that are part of a transmission event above a threshold for a given value of R and k.

Examples

```
R <- 2
k <- 0.1
cluster_size <- 10
proportion_cluster_size(R = R, k = k, cluster_size = cluster_size)

# example with a vector of k
k <- c(0.1, 0.2, 0.3, 0.4, 0.5)
proportion_cluster_size(R = R, k = k, cluster_size = cluster_size)
```

```
# example with a vector of cluster sizes
cluster_size <- c(5, 10, 25)
proportion_cluster_size(R = R, k = k, cluster_size = cluster_size)
```

`proportion_transmission` *Estimate what proportion of cases cause a certain proportion of transmission*

Description

Calculates the proportion of cases that cause a certain percentage of transmission.

It is commonly estimated what proportion of cases cause 80% of transmission (i.e. secondary cases). This can be calculated using `proportion_transmission()` at varying values of R and for different values of percentage transmission.

There are two methods for calculating the proportion of transmission, p_{80} (default) and t_{20}, see `method` argument or details for more information.

Usage

```
proportion_transmission(
  R,
  k,
  percent_transmission,
  method = c("p_80", "t_20"),
  simulate = FALSE,
  ...,
  offspring_dist,
  format_prop = TRUE
)
```

Arguments

`R`	A `number` specifying the R parameter (i.e. average secondary cases per infectious individual).
`k`	A `number` specifying the k parameter (i.e. overdispersion in offspring distribution from fitted negative binomial).
`percent_transmission`	A `number` of the percentage transmission for which a proportion of cases has produced.
`method`	A `character` string defining which method is used to calculate the proportion of transmission. Options are `"p_80"` (default) or `"t_20"`. See details for more information on each of these methods.
`simulate`	A `logical` whether the calculation should be done numerically (i.e. simulate secondary contacts) or analytically. Default is `FALSE` which uses the analytical calculation.

...	dots not used, extra arguments supplied will cause a warning.
offspring_dist	An <epiparameter> object. An S3 class for working with epidemiological parameters/distributions, see epiparameter::epiparameter().
format_prop	A logical determining whether the proportion column of the <data.frame> returned by the function is formatted as a string with a percentage sign (%), (TRUE, default), or as a numeric (FALSE).

Details

Calculates the expected proportion of transmission from a given proportion of infectious cases. There are two methods to calculate this with distinct formulations, p_{80} and t_{20} these can be specified by the method argument.

method = p_80 calculates relative transmission heterogeneity from the offspring distribution of secondary cases, Z, where the upper proportion of the distribution comprise $x\%$ of total number of cases given R0 and k, where x is typically defined as 0.8 or 80%. e.g. 80% of all transmissions are generated by the upper 20% of cases, or p_80 = 0.2, per the 80/20 rule. In this formulation, changes in R can have a significant effect on the estimate of p_80 even when k is constant. Importantly, this formulation **does not** allow for true homogeneity when k = Inf i.e. $p_{80} = 0.8$.

method = t_20 calculates a similar ratio, instead in terms of the theoretical individual reproductive number and infectiousness given R0 and k. The individual reproductive number, 'v', is described in Lloyd-Smith JO et al. (2005), "as a random variable representing the expected number of secondary cases caused by a particular infected individual. Values for v are drawn from a continuous gamma probability distribution with population mean R0 and dispersion parameter k, which encodes all variation in infectious histories of individuals, including properties of the host and pathogen and environmental circumstances." The value of k corresponds to the shape parameters of the gamma distribution which encodes the variation in the gamma-poisson mixture aka the negative binomial

For method = t_20, we define the upper proportion of infectiousness, which is typically 0.2 i.e. the upper 20% most infectious cases, again per the 80/20 rule. e.g. the most infectious 20% of cases, are expected to produce 80% of all infections, or t_20 = 0.8. Unlike method = p_80, changes in R have no effect on the estimate of t_80 when k is constant, but R is still required for the underlying calculation. This formulation **does** allow for true homogeneity when k = Inf i.e. t_20 = 0.2, or t_80 = 0.8.

Multiple values of R and k can be supplied and a <data.frame> of every combination of these will be returned.

The numerical calculation for method = p_80 uses random number generation to simulate secondary contacts so the answers may minimally vary between calls. The number of simulation replicates is fixed to 105.

Value

A <data.frame> with the value for the proportion of cases for a given value of R and k.

References

The analytical calculation is from:

Endo, A., Abbott, S., Kucharski, A. J., & Funk, S. (2020) Estimating the overdispersion in COVID-19 transmission using outbreak sizes outside China. Wellcome Open Research, 5. doi:10.12688/wellcomeopenres.15842.3

The t_{20} method follows the formula defined in section 2.2.5 of the supplementary material for:

Lloyd-Smith JO, Schreiber SJ, Kopp PE, Getz WM. Superspreading and the effect of individual variation on disease emergence. Nature. 2005 Nov;438(7066):355–9. doi:10.1038/nature04153

The original code for the t_{20} method is from ongoing work originating from `https://github.com/dcadam/kt` and:

Adam D, Gostic K, Tsang T, Wu P, Lim WW, Yeung A, et al. Time-varying transmission heterogeneity of SARS and COVID-19 in Hong Kong. 2022. doi:10.21203/rs.3.rs1407962/v1

Examples

```
# example of single values of R and k
percent_transmission <- 0.8 # 80% of transmission
R <- 2
k <- 0.5
proportion_transmission(
  R = R,
  k = k,
  percent_transmission = percent_transmission
)

# example with multiple values of k
k <- c(0.1, 0.2, 0.3, 0.4, 0.5, 1)
proportion_transmission(
  R = R,
  k = k,
  percent_transmission = percent_transmission
)

# example with vectors of R and k
R <- c(1, 2, 3)
proportion_transmission(
  R = R,
  k = k,
  percent_transmission = percent_transmission
)
```

Index