

# Package: bSims (via r-universe)

September 18, 2024

**Type** Package

**Title** Agent-Based Bird Point Count Simulator

**Version** 0.3-2

**Date** 2024-05-21

**Author** Peter Solymos [aut, cre]  
(<https://orcid.org/0000-0001-7337-1740>)

**Maintainer** Peter Solymos <psolymos@gmail.com>

**Description** A highly scientific and utterly addictive bird point count simulator to test statistical assumptions, aid survey design, and have fun while doing it (Solymos 2024 [doi:10.1007/s42977-023-00183-2](https://doi.org/10.1007/s42977-023-00183-2)). The simulations follow time-removal and distance sampling models based on Matsuoka et al. (2012) [doi:10.1525/auk.2012.11190](https://doi.org/10.1525/auk.2012.11190), Solymos et al. (2013) [doi:10.1111/2041-210X.12106](https://doi.org/10.1111/2041-210X.12106), and Solymos et al. (2018) [doi:10.1650/CONDOR-18-32.1](https://doi.org/10.1650/CONDOR-18-32.1), and sound attenuation experiments by Yip et al. (2017) [doi:10.1650/CONDOR-16-93.1](https://doi.org/10.1650/CONDOR-16-93.1).

**License** GPL-2

**LazyLoad** yes

**Depends** intrval, mefa4, MASS, deldir (>= 1.0-2)

**Imports** parallel, pbapply

**URL** <https://github.com/psolymos/bSims>

**BugReports** <https://github.com/psolymos/bSims/issues>

**Suggests** knitr, rmarkdown, detect, shiny

**VignetteBuilder** knitr

**Language** en-US

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

**RemoteUrl** <https://github.com/psolymos/bSims>

**RemoteRef** HEAD

**RemoteSha** 641f07daa499b404399655feb662a6af64b7dd67

## Contents

acceptreject	2
bsims_init	3
dist_fun2	8
estimate	10
events	11
expand_list	13
get_nests	14
plot.bsims_landscape	16
rlnorm2	19
rmvn	20
run_app	21
<b>Index</b>	<b>22</b>

---

acceptreject	<i>Spatial point process simulator</i>
--------------	----------------------------------------

---

### Description

Spatial point process simulator based on accept/reject algorithm.

### Usage

```
acceptreject(n, f = NULL, x0 = 0, x1 = 1, y0 = 0, y1 = 1,
            m = 0, maxit = 100, fail = FALSE)
```

### Arguments

n	number of points to generate.
f	a function returning probability (value between 0 and 1) given distance as the first and only argument. The function generates spatially uniform Poisson point process (complete spatial randomness) when NULL.
x0, x1, y0, y1	x and y ranges (bounding box).
m	margin width for avoiding edge effects.
maxit	maximum number of iterations per point to try if no acceptance happens.
fail	logical, what to do when there is a problem. TRUE gives error, the default FALSE gives only a warning.

### Value

A matrix with n rows and 2 columns for x and y coordinates.

### Author(s)

Peter Solymos

**Examples**

```

## complete spatial randomness
plot(acceptreject(100), asp=1)

## more systematic
distance <- seq(0,1,0.01)
f <- function(d)
  (1-exp(-d^2/0.1^2) + dlnorm(d, 0.2)/dlnorm(exp(0.2-1),0.2)) / 2
op <- par(mfrow = c(1, 2))
plot(distance, f(distance), type="l")
plot(acceptreject(100, f, m=1), asp=1)
par(op)

## more clustered
f <- function(d)
  exp(-d^2/0.1^2) + 0.5*(1-exp(-d^2/0.4^2))
op <- par(mfrow = c(1, 2))
plot(distance, f(distance), type="l")
plot(acceptreject(100, f, m=1), asp=1)
par(op)

```

---

bsims\_init

*bSims simulation functions*


---

**Description**

Functions to initialize, populate, animate, detect, and transcribe simulated birds in a point count.

**Usage**

```
bsims_init(extent = 10, road = 0, edge = 0, offset = 0)
```

```
bsims_populate(x, density = 1, abund_fun = NULL, xy_fun = NULL,
  margin = 0, maxit = 100, fail = FALSE, ...)
```

```
bsims_animate(x, vocal_rate = 1, move_rate = 0, duration = 10,
  movement = 0, mixture = 1, avoid = c("none", "R", "ER"),
  initial_location=FALSE, allow_overlap=TRUE, ...)
```

```
bsims_detect(x, xy = c(0, 0), tau = 1, dist_fun = NULL,
  event_type = c("vocal", "move", "both"),
  sensitivity=1, direction=FALSE, ...)
```

```
bsims_transcribe(x, tint = NULL, rint = Inf,
  error = 0, bias = 1,
  condition=c("event1", "det1", "alldet"),
  event_type=NULL, perception=NULL, ...)
```

```

bsims_all(...)

## S3 method for class 'bsims_landscape'
print(x, ...)
## S3 method for class 'bsims_population'
print(x, ...)
## S3 method for class 'bsims_events'
print(x, ...)
## S3 method for class 'bsims_detections'
print(x, ...)
## S3 method for class 'bsims_transcript'
print(x, ...)
## S3 method for class 'bsims_all'
print(x, ...)

```

### Arguments

extent	extent of simulation area, an extent x extent square with (0,0) at the center.
road	half width of the road stratum (perpendicular to the y axis).
edge	width of edge, same width on both sides of the road stratum.
offset	offset to apply to road and edge strata relative to the center in the x direction.
x	a simulation object.
density	population density, D, recycled 3x for the 3 strata (H: habitat, E: edge, R: road).
abund_fun	function to simulate abundance, $N \sim \text{Poisson}(\lambda)$ , $\lambda = DA$ by default.
xy_fun	function used to simulate nest locations, see <a href="#">acceptreject</a> . The function should return probability (value between 0 and 1), NULL means complete spatial randomness.
margin, maxit, fail	arguments passed to <a href="#">acceptreject</a> when using xy_fun to simulate nest locations.
vocal_rate, move_rate	Vocal and movement rates (see <a href="#">events</a> ). Both of these rates can be: a single number; a vector of length <code>length(mixture)</code> (behavior based finite mixture groups); a vector of length 3 with <code>mixture=1</code> (corresponding to HER strata); or a matrix of dimension 3 x <code>length(mixture)</code> (HER strata x number of behavior based groups).
duration	total time duration to consider (in minutes), passed to <a href="#">events</a> .
movement	standard deviation for a bivariate Normal kernel to simulate locations centered at the nest location, passed to <a href="#">events</a> . Can refer to the same stratum and behavior based groups as <code>move_rate</code> .
mixture	behavior based finite mixture group proportions.
avoid	range along the x axis to avoid with respect to movement locations, passed to <a href="#">events</a> .
initial_location	logical, <code>move_rate</code> and <code>vocal_rate</code> are silently ignored if TRUE and nest locations are provided as part of the events table. This renders all individuals equally available for detection.

allow_overlap	logical, allowing overlap between neighboring nests when movement is involved. If FALSE, Voronoi tessellation is used to prevent overlap. If TRUE, the unconstrained bivariate Normal kernel is used.
xy	a vector of x and y coordinates describing the position of the observer.
tau	parameter of the distance function. Can be a single numeric value; a vector of length 2 to provide parameters for vocalization (1st value) and movement (2nd value) related events; (H: habitat, E: edge, R: road, in this order); a vector of length 3 to provide parameters for the 3 strata (H: habitat, E: edge, R: road); or a 3 x 2 matrix combining strata (rows) and vocalization/movement (columns) related parameters. Segmented sound attenuation is used when the values are different in the 3 strata (see <a href="#">dist_fun2</a> ).
dist_fun	distance function (1st argument is distance, second is tau).
event_type	type of events to access (vocal, movement, or both). Inherits value from input object when NULL.
tint	time interval break points in minutes.
rint	distance interval break points in units of 100 meter.
condition	conditioning type to define availability for each individual: "event1": the 1st event (detected or not); "det1": the 1st detection; "alldet": all detections (counting the same individual multiple times).
error	log scale standard deviation (SD) for distance estimation error, see <a href="#">rlnorm2</a> . When <code>direction=TRUE</code> , error changes based on the angle between the observer and the individual's (random) singing direction. When the bird faces the observer (0 degrees) SD is 0, when the bird is facing away (180 degrees) SD is error. In the range between 0-180 degrees the SD is changing according to the cosine of the degree: $SD * (0.5 - \cos(\text{degree} * \pi / 180) / 2)$ .
bias	nonnegative numeric, the distance estimation bias. The default value (1) means no bias, <1 indicates negative bias (perceived distance is less than true distance), >1 indicates positive bias (perceived distance is higher than true distance). This acts as a multiplier and can be combined with error. When <code>direction=TRUE</code> , bias changes based on the angle between the observer and the individual's (random) singing direction. When the bird faces the observer (0 degrees) perceived distance equals the true distance, when the bird is facing away (180 degrees) perceived distance is bias * true distance. In the range between 0-180 degrees the bias is changing according to the cosine of the degree: $1 + (\text{bias} - 1) * (0.5 - \cos(\text{degree} * \pi / 180) / 2)$ .
perception	perceived number of individuals relative to the actual number of individuals. A non-negative number (<1 values lead to under counting, >1 values lead to over counting), or NULL (observer correctly identifies all individuals).
sensitivity	non-negative numeric value indicating the sensitivity of the sensor receiving the signal. Can be of length 1 (applies to both vocal and movement events) or a named vector of length 2 (names should indicate which one is "vocal" or "move"). Sensitivity of 1 means that the process captured by tau is unaffected. Less than 1 values indicate lower sensitivity (effectively decreasing tau), larger than 1 values indicate higher sensitivity (effectively increasing tau).

`direction` logical. When TRUE, tau for vocalizations (not for movement) changes based on the angle between the observer and the individual's (random) singing direction. When the bird faces the observer (0 degrees) tau is unaffected, when the bird is facing away (180 degrees) tau is  $\text{sensitivity} * \text{tau}$ . In the range between 0-180 degrees the effect is changing according to the cosine of the degree  $(0.5 - \cos(\text{degree} * \pi / 180) / 2)$ .

... other arguments passed to underlying functions. For the `bsims_all` wrapper, it means all the arguments (except for `x`) that the underlying `bsims_*` functions have. `bsims_all` can also take a single list as its argument.

## Details

The functions capturing the simulation layers are supposed to be called in sequence, allowing to simulate multiple realities by keeping preceding layers intact. Construction allows easy piping. The `bsims_all` function is a wrapper for the `bsims_*` layer functions.

The simulations follow time-removal and distance sampling models based on Matsuoka et al. (2012) <doi:10.1525/auk.2012.11190>, Solymos et al. (2013) <doi:10.1111/2041-210X.12106>, and Solymos et al. (2018) <doi:10.1650/CONDOR-18-32.1>, and sound attenuation experiments by Yip et al. (2017) <doi:10.1650/CONDOR-16-93.1>.

## Value

`bsims_init` returns a landscape object.

`bsims_populate` returns a population object.

`bsims_animate` returns an events object.

`bsims_detect` returns a detections object.

`bsims_transcribe` returns a transcript object.

`get_table` returns the removal table.

`bsims_all` returns a closure with `$settings()`, `$new(recover = FALSE)`, and `$replicate(B, recover = FALSE, c1 = NULL)` functions. The `settings` function returns the input arguments as a list; the `new` function returns a single transcript object; the `replicate` function takes an argument for the number of replicates (`B`) and returns a list of transcript objects with `B` elements. The `c1` argument is used to parallelize the work, can be a numeric value on Unix/Linux/OSX, or a cluster object on any OS, see examples. The `'recover = TRUE'` argument allows to run simulations with error catching based on [try](#).

Note that simulated objects returned by `bsims_all` will contain different realizations and all the conditionally independent layers. Use a layered approach if former layers are meant to be kept identical across runs.

## Author(s)

Peter Solymos

## References

- Matsuoka, S. M., Bayne, E. M., Solymos, P., Fontaine, P., Cumming, S. G., Schmiegelow, F. K. A., & Song, S. A., 2012. Using binomial distance-sampling models to estimate the effective detection radius of point-counts surveys across boreal Canada. *Auk*, **129**: 268–282. <doi:10.1525/auk.2012.11190>
- Solymos, P., Matsuoka, S. M., Bayne, E. M., Lele, S. R., Fontaine, P., Cumming, S. G., Stralberg, D., Schmiegelow, F. K. A. & Song, S. J., 2013. Calibrating indices of avian density from non-standardized survey data: making the most of a messy situation. *Methods in Ecology and Evolution*, **4**: 1047–1058. <doi:10.1111/2041-210X.12106>
- Solymos, P., Matsuoka, S. M., Cumming, S. G., Stralberg, D., Fontaine, P., Schmiegelow, F. K. A., Song, S. J., and Bayne, E. M., 2018. Evaluating time-removal models for estimating availability of boreal birds during point-count surveys: sample size requirements and model complexity. *Condor*, **120**: 765–786. <doi:10.1650/CONDOR-18-32.1>
- Yip, D. A., Bayne, E. M., Solymos, P., Campbell, J., and Proppe, J. D., 2017. Sound attenuation in forested and roadside environments: implications for avian point count surveys. *Condor*, **119**: 73–84. <doi:10.1650/CONDOR-16-93.1>

## See Also

Plotting functions: [plot.bsims\\_landscape](#)

Getter functions: [get\\_nests](#), [get\\_events](#), [get\\_detections](#), [get\\_abundance](#), [get\\_density](#) [get\\_table](#)

Shiny apps: [run\\_app](#)

[acceptreject](#), [events](#), [estimate](#)

## Examples

```
phi <- 0.5
tau <- 1:3
dur <- 10
rbr <- c(0.5, 1, 1.5, Inf)
tbr <- c(3, 5, 10)
(l <- bsims_init(10, 0.5, 1))
(p <- bsims_populate(l, 1))
(a <- bsims_animate(p, vocal_rate=phi, duration=dur))
(o <- bsims_detect(a, tau=tau))
(x <- bsims_transcribe(o, tint=tbr, rint=rbr))

plot(x)
get_table(x, "removal")
get_table(x, "visits")

head(get_events(a))
plot(get_events(a))

head(get_detections(o))
plot(get_detections(o), "time")
plot(get_detections(o), "distance")

## wrapper function for all the bsims_* layers
```

```

b <- bsims_all(road=1, density=0.5, tint=tbr, rint=rbr)
## alternatively: supply a list
#settings <- list(road=1, density=0.5, tint=tbr, rint=rbr)
#b <- bsims_all(settings)
b$settings()
b$new()
bb <- b$replicate(3)
lapply(bb, get_table)

## parallel simulations
library(parallel)
b <- bsims_all(density=0.5)
B <- 4 # number of runs
nc <- 2 # number of cores
## sequential
system.time(bb <- b$replicate(B, cl=NULL))
## parallel clusters
cl <- makeCluster(nc)
## note: loading the package is optional
system.time(clusterEvalQ(cl, library(bSims)))
system.time(bb <- b$replicate(B, cl=cl))
stopCluster(cl)
## parallel forking
if (.Platform$OS.type != "windows") {
  system.time(bb <- b$replicate(B, cl=nc))
}

```

---

dist\_fun2

*Distance function with segmented attenuation*


---

### Description

Distance function with segmented attenuation crossing a number of boundaries of strata with different attenuation characteristics following results in Yip et al. (2017).

### Usage

```
dist_fun2(d, tau, dist_fun, breaks = numeric(0), ...)
```

### Arguments

d	distance from observer.
tau	a parameter passed to the the distance function. Length of tau must equal length(b) + 1 referring to distance function parameters in the different strata (a stratum is defined by an interval surrounded by 1 or 2 boundaries).
dist_fun	distance function taking two arguments: distance, and tau, see examples.



breaks distance breakpoints, must be  $\text{length}(\text{tau}) - 1$  in length. These breakpoints represent the boundaries between the intervals characterized by homogeneous attenuation characteristics.

... other arguments passed to `dist_fun`

### Value

Probability of detection given the distance, stratum specific parameters and the arrangement of breakpoints.

### Author(s)

Peter Solymos

### References

Yip, D. A., Bayne, E. M., Solymos, P., Campbell, J., and Proppe, J. D., 2017. Sound attenuation in forested and roadside environments: implications for avian point count surveys. *Condor*, **119**: 73–84. <doi:10.1650/CONDOR-16-93.1>

### Examples

```
tau <- c(1, 2, 3, 2, 1)
d <- seq(0, 4, 0.01)
dist_fun <- function(d, tau) exp(-(d/tau)^2) # half normal
#dist_fun <- function(d, tau) exp(-d/tau) # exponential
#dist_fun <- function(d, tau) 1-exp(-(d/tau)^2) # hazard rate

b <- c(0.5, 1, 1.5, 2) # boundaries

op <- par(mfrow=c(2, 1))
plot(d, dist_fun2(d, tau[1], dist_fun), type="n",
      ylab="P(detection)", xlab="Distance", axes=FALSE,
      main="Sound travels from left to right")
axis(1)
axis(2)
for (i in seq_len(length(b)+1)) {
  x1 <- c(0, b, 4)[i]
  x2 <- c(0, b, 4)[i+1]
  polygon(c(0, b, 4)[c(i, i, i+1, i+1)], c(0, 1, 1, 0),
          border=NA,
          col=c("darkolivegreen1", "burlywood1", "lightgrey",
               "burlywood1", "darkolivegreen1")[i])
}
lines(d, dist_fun2(d, tau[1], dist_fun))
lines(d, dist_fun2(d, tau[2], dist_fun))
lines(d, dist_fun2(d, tau[3], dist_fun))
lines(d, dist_fun2(d, tau, dist_fun, b), col=2, lwd=3)

plot(rev(d), dist_fun2(d, tau[1], dist_fun), type="n",
      ylab="P(detection)", xlab="Distance", axes=FALSE,
      main="Sound travels from right to left")
```

```

axis(1)
axis(2)
for (i in seq_len(length(b)+1)) {
  x1 <- c(0, b, 4)[i]
  x2 <- c(0, b, 4)[i+1]
  polygon(c(0, b, 4)[c(i, i, i+1, i+1)], c(0, 1, 1, 0),
    border=NA,
    col=c("darkolivegreen1", "burlywood1", "lightgrey",
    "burlywood1", "darkolivegreen1")[i])
}
lines(rev(d), dist_fun2(d, tau[1], dist_fun))
lines(rev(d), dist_fun2(d, tau[2], dist_fun))
lines(rev(d), dist_fun2(d, tau[3], dist_fun))
lines(rev(d), dist_fun2(d, tau, dist_fun, rev(4-b)), col=2, lwd=3)
par(op)

```

---

estimate

*Estimate basic parameters*

---

### Description

Estimate singing rates, effective distances, and density based on simulation objects using the QPAD approach (Solymos et al. 2013).

### Usage

```

estimate(object, ...)
## S3 method for class 'bsims_transcript'
estimate(object, ...)

```

### Arguments

object	simulation object.
...	other arguments passed to internal functions.

### Details

The method evaluates removal design to estimate model parameters and density using the QPAD methodology using the 'detect' package.

The function only works with multiple time and distance intervals. It returns NA otherwise.

### Value

A vector with values for singing rate ( $\phi$ ), effective detection distance ( $\tau$ ), and density.

### Author(s)

Peter Solymos

## References

Solymos, P., Matsuoka, S. M., Bayne, E. M., Lele, S. R., Fontaine, P., Cumming, S. G., Stralberg, D., Schmiegelow, F. K. A. & Song, S. J., 2013. Calibrating indices of avian density from non-standardized survey data: making the most of a messy situation. *Methods in Ecology and Evolution*, 4: 1047–1058. <doi:10.1111/2041-210X.12106>

## See Also

[bsims\\_init](#)

## Examples

```
set.seed(2)
phi <- 0.5           # singing rate
tau <- 1            # EDR by strata
dur <- 10           # simulation duration
tbr <- c(2, 4, 6, 8, 10) # time intervals
rbr <- c(0.5, 1, 1.5, Inf) # counting radii

l <- bsims_init(10, 0.5, 1) # landscape
p <- bsims_populate(l, 10) # population
e <- bsims_animate(p,      # events
  vocal_rate=phi, duration=dur)
d <- bsims_detect(e,      # detections
  tau=tau)
x <- bsims_transcribe(d,  # transcription
  tint=tbr, rint=rbr)

estimate(x)
```

---

events

*Event time simulator*

---

## Description

`timetoevent` turns exponential wait times to time-to-event data within a desired duration, it handles 0 and infinite rates in a robust manner. `events` simulates event times based on an exponential time-to-event distribution.

## Usage

```
timetoevent(rate, duration)

events(vocal_rate = 1, move_rate = 1, duration = 10,
  movement = 0, avoid = c(0, 0))
```

**Arguments**

rate	rate for the exponential distribution ( <a href="#">rexp</a> ).
duration	total time duration to consider (in minutes).
vocal_rate	vocal rate for exponential distribution ( <a href="#">rexp</a> ), how often a vocal event happens per minute.
move_rate	movement rate for exponential distribution ( <a href="#">rexp</a> ), how often a movement event happens per minute.
movement	standard deviation for a bivariate Normal kernel to simulate locations centered at the nest location.
avoid	range along the x axis to avoid with respect to movement locations, i.e. location for a movement event within this interval will be rejected and a new location drawn.

**Value**

An events object data frame with coordinates (x, y; centered at 0 that is nest location), event times (t) and indicator for vocal events (v).

**Author(s)**

Peter Solymos

**See Also**

[rexp](#)

**Examples**

```

timetoevent(0, 10)
timetoevent(Inf, 10)

rr <- 1
tt <- timetoevent(rr, 10)
op <- par(mfrow=c(1,2))
plot(ecdf(tt))
curve(1-exp(-rr*x), add=TRUE, col=2) # cdf

plot(stepfun(sort(tt), 0:length(tt)/length(tt)), ylab="F(x)")
curve(1-exp(-rr*x), add=TRUE, col=2) # cdf
par(op)

e <- events(movement=1, duration=60)
mx <- max(abs(e[,1:2]))
plot(e[,1:2], col="grey", type="l", asp=1,
     xlim=2*c(-mx, mx), ylim=2*c(-mx, mx))
points(e[,1:2], col=e$v+1)
abline(h=0, v=0, lty=2)
legend("topright", pch=21, col=1:2, horiz=TRUE,
       legend=c("movement", "vocalization"))

```

---

expand_list	<i>Create a list from all combinations of arguments</i>
-------------	---------------------------------------------------------

---

**Description**

Create a list from all combinations of the supplied vectors or lists.

**Usage**

```
expand_list(...)
```

**Arguments**

... vectors or lists. All arguments must be named.

**Value**

A list containing one element for each combination of the supplied vectors and lists. The first factors vary fastest. The nested elements are labeled by the factors.

The function allows list elements to be vectors, functions, or NULL. If a vector element is supposed to be kept as a vector, use `list()`.

**Author(s)**

Peter Solymos

**See Also**

[expand.grid](#)

**Examples**

```
b <- expand_list(
  movement = c(0, 1, 2),
  rint = list(c(0.5, 1, 1.5, Inf)), # in a list to keep as one
  xy_fun = list(NULL, function(z) z))
b[[1]]
str(b)
```

---

`get_nests`*Access nests, events, detections, and totals*

---

### Description

Access nests, events, detections, abundance, and density from simulation objects.

### Usage

```
get_nests(x, ...)
## S3 method for class 'bsims_population'
get_nests(x, ...)

get_events(x, ...)
## S3 method for class 'bsims_events'
get_events(x, ...)

get_detections(x, ...)
## S3 method for class 'bsims_detections'
get_detections(x, ...)

get_abundance(x, ...)
## S3 method for class 'bsims_population'
get_abundance(x, ...)

get_density(x, ...)
## S3 method for class 'bsims_population'
get_density(x, ...)

get_table(x, ...)
## S3 method for class 'bsims_transcript'
get_table(x,
  type = c("removal", "visits"), ...)
```

### Arguments

<code>x</code>	simulation object.
<code>type</code>	character, the type of table to return: "removal" includes only new individuals as time progresses, "visits" counts individuals in each time interval independent of each other.
<code>...</code>	other arguments passed to internal functions.

### Details

`get_nests` extracts the nest locations.

`get_events` extracts the events.

get\_detections extracts the detections.

get\_abundance gets the realized total abundance (N), get\_density gets the realized average density (abundance/area: N/A).

get\_table returns the removal or visits table.

## Value

get\_abundance and get\_density returns a non-negative numeric value.

get\_nests returns a data frame with the following columns: i individual identifier, s spatial stratum (H: habitat, E: edge, R: road) x and y are coordinates of the nest locations, g is behavioral (mixture) group or NA.

get\_events returns a data frame with the following columns: x and y are locations of the individual at the time of the event, t time of the event within the duration interval, v indicator variable for vocal (1) vs. movement (0) event, a direction for vocalization events (NA for movement) in degrees clockwise relative to north, i individual identifier.

get\_detections returns a data frame with the following columns: x and y are locations of the individual at the time of the event, t time of the event within the duration interval, v indicator variable for vocal (1) vs. movement (0) event, a direction for vocalization events (NA for movement) in degrees clockwise relative to north, d distance from observer when detected (otherwise NA). f indicates the angle between the bird's vocalization direction (column a) relative to the observer (the value is 0 for movement events by default), i individual identifier, j perceived individual identifier.

get\_table returns a matrix with distance bands as rows and time intervals as columns. The cell values are counts if the individuals detected in a removal fashion (only new individuals counter over the time periods) or in a multiple-visits fashion (counting of individuals restarts in every time interval).

## Author(s)

Peter Solymos

## See Also

[bsims\\_init](#)

## Examples

```
phi <- 0.5           # singing rate
tau <- 1:3           # EDR by strata
dur <- 10            # simulation duration
tbr <- c(3, 5, 10)  # time intervals
rbr <- c(0.5, 1, 1.5, Inf) # counting radii

l <- bsims_init(10, 0.5, 1) # landscape
p <- bsims_populate(l, 1) # population
e <- bsims_animate(p,      # events
  vocal_rate=phi, duration=dur)
d <- bsims_detect(e,      # detections
  tau=tau)
x <- bsims_transcribe(d,  # transcription
```

```

    tint=tbr, rint=rbr)

## next locations
head(get_nests(p))
head(get_nests(e))
head(get_nests(d))
head(get_nests(x))

## abundance
get_abundance(p)
get_abundance(e)
get_abundance(d)
get_abundance(x)

## density
get_density(p)
get_density(e)
get_density(d)
get_density(x)

## events
head(get_events(e))
head(get_events(d))
head(get_events(x))

## detections
head(get_detections(d))
head(get_detections(x))

get_table(x, "removal")
get_table(x, "visits")

```

---

plot.bsims\_landscape *Plot methods*

---

## Description

Plot methods for different bSims objects.

## Usage

```

## S3 method for class 'bsims_landscape'
plot(x,
     col_H, col_E, col_R,
     xlim = NULL, ylim = NULL, ...)

## S3 method for class 'bsims_population'
plot(x,
     pch_nest, col_nest, cex_nest, ...)

```



```
## S3 method for class 'bsims_events'
plot(x,
     event_type=c("vocal", "move", "both"), tlim = NULL,
     pch_nest, col_nest, cex_nest,
     pch_vocal, col_vocal, cex_vocal,
     lty_move, col_move, lwd_move, ...)

## S3 method for class 'bsims_detections'
plot(x,
     event_type=NULL, tlim = NULL,
     pch_nest, col_nest, cex_nest,
     pch_vocal, col_vocal, cex_vocal,
     lty_move, col_move, lwd_move,
     lty_det_vocal, col_det_vocal, lwd_det_vocal,
     lty_det_move, col_det_move, lwd_det_move,
     condition = "event1", ...)

## S3 method for class 'bsims_transcript'
plot(x,
     pch_nest, col_nest, cex_nest,
     pch_vocal, col_vocal, cex_vocal,
     lty_move, col_move, lwd_move,
     lty_det_vocal, col_det_vocal, lwd_det_vocal,
     lty_det_move, col_det_move, lwd_det_move,
     show_tint=TRUE, show_rint=TRUE,
     col_tint, col_rint, ...)

## S3 method for class 'bsims_events'
lines(x, tlim = NULL, ...)
## S3 method for class 'bsims_detections'
lines(x,
     event_type=NULL, tlim=NULL, condition="event1", ...)
## S3 method for class 'bsims_transcript'
lines(x,
     event_type=NULL, tlim=NULL, ...)

## S3 method for class 'bsims_population'
points(x, ...)
## S3 method for class 'bsims_events'
points(x,
     event_type=c("vocal", "move", "both"), tlim = NULL, ...)
## S3 method for class 'bsims_detections'
points(x,
     event_type=NULL, tlim=NULL, condition="event1", ...)

col2hex(col, alpha = FALSE)

## S3 method for class 'bsims_events_table'
```

```

plot(x,
     xlab, ylab, xlim, ylim, col_det_vocal, col_det_move, ...)
## S3 method for class 'bsims_detections_table'
plot(x,
     type=c("time", "distance"), xlab, ylab, xlim, ylim,
     col_det_vocal, col_det_move, ...)

```

### Arguments

<code>x</code>	simulation object.
<code>col</code>	color values.
<code>col_H, col_E, col_R</code>	color values for the Habitat, Edge, and Road strata.
<code>event_type</code>	type of events to access. The value is inferred from the input object when NULL.
<code>xlim, ylim, tlim</code>	x, y, time intervals.
<code>xlab, ylab</code>	x and y axis labels.
<code>pch_nest, col_nest, cex_nest</code>	visual characteristics of nest locations.
<code>pch_vocal, col_vocal, cex_vocal</code>	visual characteristics of vocalization events.
<code>lty_move, col_move, lwd_move</code>	visual characteristics of movement events.
<code>lty_det_vocal, col_det_vocal, lwd_det_vocal</code>	visual characteristics of detection events related to vocalizations.
<code>lty_det_move, col_det_move, lwd_det_move</code>	visual characteristics of detection events related to movements.
<code>alpha</code>	alpha channel for colors.
<code>show_tint, show_rint</code>	whether time and distance intervals should be displayed.
<code>col_tint, col_rint</code>	colors for time and distance intervals.
<code>condition</code>	conditioning type to define availability for each individual, see <a href="#">bsims_detect</a> .
<code>type</code>	what the x axis should be: time or distance.
<code>...</code>	other graphical arguments.

### Details

The main plotting functions use a theme defined in the option `getOption("bsims_theme")`. Overriding these default settings allows customization.

### Value

These plotting functions are called for their side effects and silently return the input object.

`col2hex` is modeled after [col2rgb](#) and returns a character vector giving hexadecimal color codes with or without alpha channel values.

**Author(s)**

Peter Solymos

**See Also**[bsims\\_init](#), [col2rgb](#)**Examples**

```

b <- bsims_all(road=1, edge=2, move_rate=1, movement=0.2)$new()
o <- getOption("bsims_theme")
str(o)
n <- o
n$col_H <- "gold"
n$col_E <- "magenta"
n$col_R <- "black"
op <- par(mfrow=c(1, 2))
plot(b)
options("bsims_theme" = n) # apply new theme
plot(b)
par(op)
options("bsims_theme" = o) # reset old theme

col2hex(c(blu = "royalblue", reddish = "tomato"), alpha = FALSE)
col2hex(c(blu = "royalblue", reddish = "tomato"), alpha = TRUE)

```

rlnorm2

*Reparametrized lognormal distribution***Description**

A lognormal distribution parametrized as mean ( $ybar$ ) and  $SDlog$ .

**Usage**

```
rlnorm2(n, mean = exp(0.5), sdlog = 1)
```

**Arguments**

n	number of random numbers desired.
mean	mean.
sdlog	log scale standard deviation.

**Details**

Log scale mean is  $\log(\text{mean}) - \text{sdlog}^2/2$ .

**Value**

Vector of random numbers.

**Author(s)**

Peter Solymos

**See Also**

`link{rlnorm}`

**Examples**

```
summary(rlnorm2(10^6, 1.3, 0.5)) # mean ~ 1.3
exp(log(1.3) - 0.5^2/2) # ~ median
```

---

rmvn

*Multivariate normal distribution*

---

**Description**

A shim of `mvrnorm` to return matrix when  $n < 2$ .

**Usage**

```
rmvn(n = 1L, mu, Sigma, ...)
```

**Arguments**

<code>n</code>	number of random vectors desired (nonnegative integer, can be 0).
<code>mu</code>	mean vector.
<code>Sigma</code>	variance-covariance matrix.
<code>...</code>	other arguments passed to <code>mvrnorm</code> .

**Value**

A matrix with  $n$  rows and `length(mu)` columns.

**Author(s)**

Peter Solymos

**See Also**

`mvrnorm`

**Examples**

```
rmvn(0, c(a=0, b=0), diag(1, 2, 2))
rmvn(1, c(a=0, b=0), diag(1, 2, 2))
rmvn(2, c(a=0, b=0), diag(1, 2, 2))

sapply(0:10, function(n) dim(rmvn(n, c(a=0, b=0), diag(1, 2, 2))))
```

---

run\_app

*Run Shiny apps*

---

**Description**

Run the Shiny apps that are included in the bSims package.

**Usage**

```
run_app(app = c("bsimsH", "bsimsHER", "distfunH", "distfunHER"))
```

**Arguments**

app                    character, which app to run.

**Details**

"bsimsH": explore simulation settings in a single stratum.

"bsimsHER": explore simulation settings in multiple strata.

"distfunH": explore distance functions through a single stratum.

"distfunHER": explore distance functions through multiple strata with segmented sound attenuation (see [dist\\_fun2](#)).

**See Also**

[bsims\\_init](#)

# Index

- \* **aplot**
  - plot.bsims\_landscape, 16
- \* **datagen**
  - acceptreject, 2
  - bsims\_init, 3
  - events, 11
  - rlnorm2, 19
  - rmvn, 20
- \* **distribution**
  - acceptreject, 2
  - events, 11
  - rlnorm2, 19
  - rmvn, 20
- \* **hplot**
  - plot.bsims\_landscape, 16
- \* **manip**
  - expand\_list, 13
  - get\_nests, 14
- \* **math**
  - dist\_fun2, 8
- \* **misc**
  - run\_app, 21
- \* **models**
  - estimate, 10
- \* **spatial**
  - acceptreject, 2

acceptreject, 2, 4, 7

bSims (bsims\_init), 3

bSims-package (bsims\_init), 3

bsims\_all (bsims\_init), 3

bsims\_animate (bsims\_init), 3

bsims\_detect, 18

bsims\_detect (bsims\_init), 3

bsims\_init, 3, 11, 15, 19, 21

bsims\_populate (bsims\_init), 3

bsims\_transcribe (bsims\_init), 3

col2hex (plot.bsims\_landscape), 16

col2rgb, 18, 19

dist\_fun2, 5, 8, 21

estimate, 7, 10

events, 4, 7, 11

expand.grid, 13

expand\_list, 13

get\_abundance, 7

get\_abundance (get\_nests), 14

get\_density, 7

get\_density (get\_nests), 14

get\_detections, 7

get\_detections (get\_nests), 14

get\_events, 7

get\_events (get\_nests), 14

get\_nests, 7, 14

get\_table, 7

get\_table (get\_nests), 14

lines.bsims\_detections  
(plot.bsims\_landscape), 16

lines.bsims\_events  
(plot.bsims\_landscape), 16

lines.bsims\_transcript  
(plot.bsims\_landscape), 16

mvrnorm, 20

plot.bsims\_detections  
(plot.bsims\_landscape), 16

plot.bsims\_detections\_table  
(plot.bsims\_landscape), 16

plot.bsims\_events  
(plot.bsims\_landscape), 16

plot.bsims\_events\_table  
(plot.bsims\_landscape), 16

plot.bsims\_landscape, 7, 16

plot.bsims\_population  
(plot.bsims\_landscape), 16

plot.bsims\_transcript  
    (plot.bsims\_landscape), 16  
points.bsims\_detections  
    (plot.bsims\_landscape), 16  
points.bsims\_events  
    (plot.bsims\_landscape), 16  
points.bsims\_population  
    (plot.bsims\_landscape), 16  
print.bsims\_all (bsims\_init), 3  
print.bsims\_detections (bsims\_init), 3  
print.bsims\_events (bsims\_init), 3  
print.bsims\_landscape (bsims\_init), 3  
print.bsims\_population (bsims\_init), 3  
print.bsims\_transcript (bsims\_init), 3  
  
rexp, 12  
rlnorm2, 5, 19  
rmvn, 20  
run\_app, 7, 21  
  
timetoevent (events), 11  
try, 6