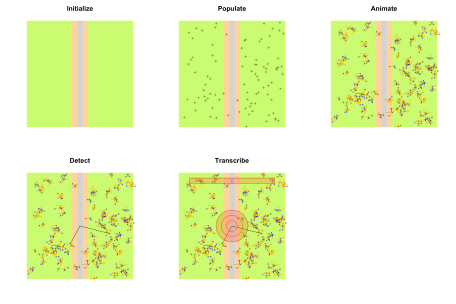
The **bSims** R package is  
a *highly scientific* and *utterly addictive* bird point count  
simulator. Highly scientific, because it implements a spatially explicit  
mechanistic simulation that is based on statistical models widely used  
in bird point count analysis (i.e. removal models, distance sampling),  
and utterly addictive because the implementation is designed to allow  
rapid interactive exploration (via **shiny** apps) and efficient  
simulation (supporting various parallel backends), thus elevating the  
user experience.



The goals of the package are to:

1. allow easy *testing of statistical assumptions* and explore effects  
   of violating these assumptions,
2. *aid survey design* by comparing different options,
3. and most importantly, to *have fun* while doing it via an intuitive  
   and interactive user interface.

The simulation interface was designed with the following principles in  
mind:

1. *isolation*: the spatial scale is small (local point count scale) so  
   that we can treat individual landscapes as more or less homogeneous  
   units (but see below how certain stratified designs and edge effects  
   can be incorporated) and independent independent in space and time;
2. *realism*: the implementation of biological mechanisms and  
   observation processes are realistic, defaults are chosen to reflect  
   common practice and assumptions;
3. *efficiency*: implementation is computationally efficient utilizing  
   parallel computing backends when available;
4. *extensibility*: the package functionality is well documented and  
   easily extensible.

This documents outlines the major functionality of the package. First we  
describe the motivation for the simulation and the details of the  
layers. Then we outline an interactive workflow to design simulation  
studies and describe how to run efficient simulation experiments.  
Finally we present some of the current limitations of the framework and  
how to extend the existing functionality of the package to incorporate  
more of the biological realism into the simulations.

**Simulation layers**

Introductory stats books begin with the coin flip to introduce the  
binomial distribution. In R we can easily simulate an outcome from such  
a random variable (Y \sim Binomial(1, p)) doing something like this:

p <- 0.5

Y <- rbinom(1, size = 1, prob = p)

But a coin flip in reality is a lot more complicated: we might consider  
the initial force, the height of the toss, the spin, and the weight of  
the coin.

Bird behavior combined with the observation process presents a more  
complicated system, that is often treated as a mixture of a count  
distribution and a detection/nondetection process, e.g.:

D <- 2 # individuals / unit area

A <- 1 # area

p <- 0.8 # probability of availability given presence

q <- 0.5 # probability of detection given availability

N <- rpois(1, lambda = A \* D)

Y <- rbinom(1, size = N, prob = p \* q)

This looks not too complicated, corresponding to the true abundance  
being a random variables (N \sim Poisson(DA)), while the observed  
count being (Y \sim Binomial(N, pq)). This is the exact simulation  
that we need when we want to make sure that an *estimator* is capable of  
estimating the *model* parameters (lambda and prob here). But such  
probabilistic simulations are not very useful when we are interested how  
well the *model* captures important aspects of *reality*.

Going back to the Poisson–Binomial example, N would be a result of all  
the factors influencing bird abundance, such as geographical location,  
season, habitat suitability, number of conspecifics, competitors, or  
predators. Y however would largely depend on how the birds behave  
depending on timing, or how an observer might detect or miss the  
different individuals, or count the same individual twice, etc.

Therefore the package has layers, that by default are *conditionally  
independent* of each other. This design decision is meant to facilitate  
the comparison of certain settings while keeping all the underlying  
realizations identical, thus helping to pinpoint effects without the  
extra variability introduced by all the other effects.

The conditionally independent *layers* of a **bSims** realization are  
the following, with the corresponding function:

1. landscape (bsims\_init),
2. population (bsims\_populate),
3. behavior with movement and vocalization events (bsims\_animate),
4. the physical side of the observation process (bsims\_detect), and
5. the human aspect of the observation process (bsims\_transcribe).

This example is a sneak peek go to the package vebsite where the  
vignette describes all the arguments.

library(bSims)

## Loading required package: intrval

## Loading required package: mefa4

## Loading required package: Matrix

## mefa4 0.3-6 2019-06-20

## Loading required package: MASS

## Loading required package: deldir

## deldir 0.1-23

## bSims 0.2-1 2019-12-16 chik-chik

phi <- 0.5 # singing rate

tau <- 1:3 # detection distances by strata

tbr <- c(3, 5, 10) # time intervals

rbr <- c(0.5, 1, 1.5) # count radii

l <- bsims\_init(10, # landscape

road=0.25, edge=0.5)

p <- bsims\_populate(l, # population

density=c(1, 1, 0))

e <- bsims\_animate(p, # events

vocal\_rate=phi,

move\_rate=1, movement=0.2)

d <- bsims\_detect(e, # detections

tau=tau)

x <- bsims\_transcribe(d, # transcription

tint=tbr, rint=rbr)

get\_table(x) # removal table

## 0-3min 3-5min 5-10min

## 0-50m 0 0 0

## 50-100m 1 0 0

## 100-150m 1 0 0

op <- par(mfrow=c(2,3), cex.main=2)

plot(l, main="Initialize")

plot(p, main="Populate")

plot(e, main="Animate")

plot(d, main="Detect")

plot(x, main="Transcribe")

par(op)

**Statistical validity of the simulations**

We can test the validity of the simulations when all of the assumptions  
are met (that is the default) in the homogeneous habitat case. We set  
singing rate (phi), detection distance (tau), and density (Den)  
for the simulations. Density is in this case unrealistically high,  
because we are not using replication only a single landscape. This will  
help with the estimation.

phi <- 0.5 # singing rate

tau <- 2 # detection distance

Den <- 10 # density

set.seed(1)

l <- bsims\_init()

a <- bsims\_populate(l, density=Den)

b <- bsims\_animate(a, vocal\_rate=phi)

o <- bsims\_detect(b, tau=tau)

tint <- c(1, 2, 3, 4, 5)

rint <- c(0.5, 1, 1.5, 2) # truncated at 200 m

(x <- bsims\_transcribe(o, tint=tint, rint=rint))

## bSims transcript

## 1 km x 1 km

## stratification: H

## total abundance: 1014

## duration: 10 min

## detected: 259 heard

## 1st event detected by breaks:

## [0, 1, 2, 3, 4, 5 min]

## [0, 50, 100, 150, 200 m]

(y <- get\_table(x, "removal")) # binned new individuals

## 0-1min 1-2min 2-3min 3-4min 4-5min

## 0-50m 1 3 1 2 0

## 50-100m 7 3 5 1 1

## 100-150m 12 2 2 1 2

## 150-200m 13 8 2 1 1

colSums(y)

## 0-1min 1-2min 2-3min 3-4min 4-5min

## 33 16 10 5 4

rowSums(y)

## 0-50m 50-100m 100-150m 150-200m

## 7 17 19 25

We use the **detect** package to fit removal model and distance sampling  
model to the simulated output. This is handily implemented in the  
estimate method for the transcription objects. First we estimate  
singing rate, effective detection distance, and density based on  
truncated distance counts:

library(detect)

## Loading required package: Formula

## Loading required package: stats4

## Loading required package: pbapply

## detect 0.4-2 2018-08-29

cbind(true = c(phi=phi, tau=tau, D=Den),

estimate = estimate(x))

## true estimate

## phi 0.5 0.5768794

## tau 2.0 2.2733052

## D 10.0 8.2330714

Next we estimate singing rate, effective detection distance, and density  
based on unlimited distance counts:

rint <- c(0.5, 1, 1.5, 2, Inf) # unlimited

(x <- bsims\_transcribe(o, tint=tint, rint=rint))

## bSims transcript

## 1 km x 1 km

## stratification: H

## total abundance: 1014

## duration: 10 min

## detected: 259 heard

## 1st event detected by breaks:

## [0, 1, 2, 3, 4, 5 min]

## [0, 50, 100, 150, 200, Inf m]

(y <- get\_table(x, "removal")) # binned new individuals

## 0-1min 1-2min 2-3min 3-4min 4-5min

## 0-50m 1 3 1 2 0

## 50-100m 7 3 5 1 1

## 100-150m 12 2 2 1 2

## 150-200m 13 8 2 1 1

## 200+m 15 9 6 6 2

colSums(y)

## 0-1min 1-2min 2-3min 3-4min 4-5min

## 48 25 16 11 6

rowSums(y)

## 0-50m 50-100m 100-150m 150-200m 200+m

## 7 17 19 25 38

cbind(true = c(phi=phi, tau=tau, D=Den),

estimate = estimate(x))

## true estimate

## phi 0.5 0.5128359

## tau 2.0 1.9928785

## D 10.0 9.2041636

**Simulation workflow**

Deviations from the assumptions and bias in density estimation can be  
explored systematically by evaluating the simulations settings. We  
recommend exploring the simulation settings interactively in the  
**shiny** apps using run\_app("distfunH") app for the homogeneous  
habitat case and the run\_app("distfunHER") app for the stratified  
habitat case. The apps represent the simulation layers as tabs, the last  
tab presenting the settings that can be copied onto the clipboard and  
pasted into the R session or code. In simple situations, comparing  
results from a few different settings might be enough.

Let us consider the following simple comparison: we want to see how much  
of an effect does roads have when the only effect is that the road  
stratum is unsuitable. Otherwise there are no behavioral or  
detectability effects of the road.

tint <- c(2, 4, 6, 8, 10)

rint <- c(0.5, 1, 1.5, 2, Inf) # unlimited

## no road

b1 <- bsims\_all(

road = 0,

density = c(1, 1, 0),

tint = tint,

rint = rint)

## road

b2 <- bsims\_all(

road = 0.5,

density = c(1, 1, 0),

tint = tint,

rint = rint)

b1

## bSims wrapper object with settings:

## road : 0

## density: 1, 1, 0

## tint : 2, 4, 6, 8, 10

## rint : 0.5, 1, 1.5, 2, Inf

b2

## bSims wrapper object with settings:

## road : 0.5

## density: 1, 1, 0

## tint : 2, 4, 6, 8, 10

## rint : 0.5, 1, 1.5, 2, Inf

The bsims\_all function accepts all the arguments we discussed before  
for the simulation layers. Unspecified arguments will be taken to be the  
default value. However, bsims\_all does not evaluate these arguments,  
but it creates a closure with the settings. Realizations can be drawn  
as:

b1$new()

## bSims transcript

## 1 km x 1 km

## stratification: H

## total abundance: 75

## duration: 10 min

## detected: 12 heard

## 1st event detected by breaks:

## [0, 2, 4, 6, 8, 10 min]

## [0, 50, 100, 150, 200, Inf m]

b2$new()

## bSims transcript

## 1 km x 1 km

## stratification: HR

## total abundance: 95

## duration: 10 min

## detected: 4 heard

## 1st event detected by breaks:

## [0, 2, 4, 6, 8, 10 min]

## [0, 50, 100, 150, 200, Inf m]

Run multiple realizations is done as:

B <- 25 # number of runs

bb1 <- b1$replicate(B)

bb2 <- b2$replicate(B)

The replicate function takes an argument for the number of replicates  
(B) and returns a list of transcript objects with (B) elements. The  
cl argument can be used to parallelize the work, it can be a numeric  
value on Unix/Linux/OSX, or a cluster object on any OS. The recover =  
TRUE argument allows to run simulations with error catching.

Simulated objects returned by bsims\_all will contain different  
realizations and all the conditionally independent layers. Use a  
customized layered approach if former layers are meant to be kept  
identical across runs.

In more complex situations the **shiny** apps will help identifying  
corner cases that are used to define a gradient of settings for single  
or multiple simulation options. Let us consider the following scenario:  
we would like to evaluate how the estimates are changing with increasing  
road width. We will use the expand\_list function which creates a list  
from all combinations of the supplied inputs. Note that we need to wrap  
vectors inside list() to avoid interpreting those as values to iterate  
over.

s <- expand\_list(

road = c(0, 0.5, 1),

density = list(c(1, 1, 0)),

tint = list(tint),

rint = list(rint))

str(s)

## List of 3

## $ :List of 4

## ..$ road : num 0

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

## $ :List of 4

## ..$ road : num 0.5

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

## $ :List of 4

## ..$ road : num 1

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

We now can use this list of settings to run simulations for each. The  
following illustrates the use of multiple cores:

b <- lapply(s, bsims\_all)

nc <- 4 # number of cores to use

library(parallel)

cl <- makeCluster(nc)

bb <- lapply(b, function(z) z$replicate(B, cl=cl))

stopCluster(cl)

In some cases, we want to evaluate crossed effects of multiple settings.  
For example road width and spatial pattern (random vs. clustered):

s <- expand\_list(

road = c(0, 0.5),

xy\_fun = list(

NULL,

function(d) exp(-d^2/1^2) + 0.5\*(1-exp(-d^2/4^2))),

density = list(c(1, 1, 0)),

tint = list(tint),

rint = list(rint))

str(s)

## List of 4

## $ :List of 5

## ..$ road : num 0

## ..$ xy\_fun : NULL

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

## $ :List of 5

## ..$ road : num 0.5

## ..$ xy\_fun : NULL

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

## $ :List of 5

## ..$ road : num 0

## ..$ xy\_fun :function (d)

## .. ..- attr(\*, "srcref")= 'srcref' int [1:8] 5 5 5 53 5 53 5 5

## .. .. ..- attr(\*, "srcfile")=Classes 'srcfilecopy', 'srcfile'

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

## $ :List of 5

## ..$ road : num 0.5

## ..$ xy\_fun :function (d)

## .. ..- attr(\*, "srcref")= 'srcref' int [1:8] 5 5 5 53 5 53 5 5

## .. .. ..- attr(\*, "srcfile")=Classes 'srcfilecopy', 'srcfile'

## ..$ density: num [1:3] 1 1 0

## ..$ tint : num [1:5] 2 4 6 8 10

## ..$ rint : num [1:5] 0.5 1 1.5 2 Inf

The package considers simulations as independent in space and time. When  
larger landscapes need to be simulated, there might be several options:  
(1) simulate a larger extent and put multiple independent observers into  
the landscape; or (2) simulate independent landscapes in isolation. The  
latter approach can also address spatial and temporal heterogeneity in  
density, behavior, etc. E.g. if singing rate is changing as a function  
of time of day, one can define the vocal\_rate values as a function of  
time, and simulate independent animation layers. When the density varies  
in space, one can simulate independent population layers.