

## Introductory Practical

### WiSP Introductory example

Objectives of practical:

1. to build familiarity with WiSP structure and commands;

**Hot tip:** *As a rule, write all your R commands in a text editor window first, then copy and paste them into R. This way you have an easily accessible record of what you did, and it is easy to make changes without having to retype it all. For this exercise, just copy and paste commands from this document into R.*

A good way to familiarize yourself with R and the library WiSP is to start using them. In this tutorial, we take you through the use of plot sampling to estimate abundance in WiSP.

Command lines in this document (those lines in Courier New font) can be copied and pasted across to the R Console window<sup>1</sup>.

### Getting R to your bidding

Once you have opened the R Console window<sup>2</sup>, you need to load the WiSP library using the command:

```
library(wisp)
```

You need to have the command window active before you can enter commands. You can switch between R windows (e.g. between Console and Graphics windows) by clicking on the window you want on top, or by clicking the Windows menu item, then window you want in the sub-menu.

These stages involved in simulating and estimating abundance using WiSP can be summarized as follows:

- To generate a population, you need to:
  - (a) define the survey region,
  - (b) define a "state model" (the statistical model determining the distribution of animals in the region, their size and their visibility/catchability), and
  - (c) place the animals in the survey region, given the region and the state model.
- To survey the population, you need to:
  - (a) specify a survey design,
  - (b) simulate the survey, given the design, region and population (this generates survey data), and
  - (c) compute point and interval estimates of abundance from the survey data.

The example below takes you through the process for a plot sampling survey. The process is much the same for other methods, except that in stages 3. and 4., the appropriate functions for the method are used in place of plot sampling functions. For example, if you were using line transect methods, you would use functions ending in ".lt" (for line transect) instead of ".pl" for (plot survey).

<sup>1</sup> Where commands run over more than one line R will automatically add the '+' symbol. Also be aware that you may have to press Return at the end of a command line to execute it before pasting a new command line.

<sup>2</sup> For instructions on installing the WiSP library, see the WiSP Manual

# 1 A plot sampling example

This example shows you how to generate the animal population similar to that shown in Figure 3.4 of Borchers *et al.* (2002), how to survey it, and how to estimate abundance using plot sampling methods.

Sometimes you want to be able to reproduce a population, design or survey exactly like one you had before. You can only do this if you can reproduce exactly the same random number(s) used in the original generation. The command `set.seed()` is very useful here. It sets the first random number to be used the next time a random number sequence is generated. So if you set it before using a command that involves randomness, you can get exactly the same result later by using exactly the same number in `set.seed()` immediately before next invoking the command.

## 1.1 The survey region

The first step is to generate a survey region. This is done using the command `generate.region()`. To generate the same region as that used in Figure 3.4 of the book, type:

```
myreg <- generate.region(x.length=100, y.width=50)
```

Note that you can set the size of the survey region to whatever you want by using appropriate values for `x.length` (the length of the region) and `y.width` (the width of the region).

You can look at it by typing

```
plot(myreg)
```

## 1.2 The population

The next step is to create a population. This is done in two stages:

1. Generate a surface that specifies the population density in the region;
2. Generate a population of specified size, using this underlying density.

To generate a simple density plane called `mydens` within the survey region called `myreg` you can use the following command:

```
mydens <- generate.density(myreg, southwest=0,southeast=0,  
northwest=10,nint.x=25, nint.y=10)
```

The density plane is defined by its height at the southwest, southeast, and northwest corners of the region. To view the density surface as a 3-D plot type

```
plot(mydens)
```

To view a multi-coloured plot of the density, type

```
plot(mydens,method="image")
```

The red areas (darker areas if you are looking at a grayscale representation) represent the higher density areas and the yellow areas (lighter areas if you are looking at a grayscale representation) represent the lower density areas.

The x- and y- resolution is pretty low in the above plot, which is why there appear to be “steps” in density. The arguments `nint.x` and `nint.y` of the command `generate.density()` determine the x- and y-resolution. Making them both larger gives a higher resolution, for example:

```
mydens <- generate.density(myreg, nint.x=250,nint.y=100,
  southwest=0,southeast=0, northwest=10)
plot(mydens,method="image")
```

Be careful of making `nint.x` and `nint.y` too large for this reason and two others: you will run out of memory, and things will take longer.

Now we can move on to creating a population with this underlying density. First set and save (in `mypop.pars`) the population parameters using the following command:

```
mypop.pars<-setpars.population(mydens, number.groups=250,
  size.method="poisson", size.min=1, size.max=5, size.mean=1,
  exposure.method="beta", exposure.min=2, exposure.max=10, exposure.mean=6,
  exposure.shape=1, type.values = c("Male","Female"),
  type.prob = c(0.48,0.52))
```

Don't worry about the meaning of all the arguments in red at the moment. Now we generate a population called `mypop`, controlling the random number seed through the argument `seed` below – this allows us to generate exactly the same population later (by using the same seed).

```
mypop <- generate.population(mypop.pars, seed=12345)
```

Don't worry about what kind of population you have generated for the moment; this will become clearer as you work through more examples.

To view the population, type `plot(mypop, type='locations')`

Each dot is a group of animals; larger dots correspond to bigger groups of animals. Lighter dots correspond to less detectable animals. If you did not set the random number seed, or used a different version of R, or used a different seed, your population would look slightly different – because generating a population is a random process. In all cases, the average density from enough generations would be exactly `mydens`.

To see the composition of your population, type `plot(mypop)` This produces the following plot:

and `summary(mypop)` produces the following summary:

```
POPULATION SUMMARY
```

```
-----
```

```
Region (length x width): 100 x 50
```

```
Number of groups : 250
```

```
Number of individuals: 420
```

Group sizes : 1 2 3 4 5

Mean group size : 1.68

Exposure boundaries : [2,10]

Mean exposure : 5.9751

Types : Female Male

Numbers of each type : 132 118

The groups of animals in WiSP populations can have three sorts of characteristic (in addition to their location) These are

1. Group size The plot on the bottom left shows the distribution of group sizes in the population; the sizes in the population are shown in the summary below the plot (there are groups of sizes 1, 2, 3, 4 and 5).
2. Exposure Animals with higher exposure are more likely to be detected or caught; the distribution of exposure is shown in the top right hand plot above; the maximum and minimum exposure levels are shown in the summary below it (exposures are between 2 and 10 here). The effect of exposure on detection or capture probability can be set when specifying the survey parameters.
3. Type This can be anything you like; typically it might be Male and Female, as in the example above. The number of animals of each type is shown in the plot at bottom right.

For more details on how to specify population parameters, type  
`?setpars.population` or `help(setpars.population)`

### 1.3 The survey

Now that we have a population to survey, we need to design and conduct the survey

Suppose we have enough survey effort to cover 20% of the survey region, and that we decide to use rectangular plots with length a tenth of the survey region length, and height a tenth of the survey region height, and we locate them at random in the survey region. The following command sets the design parameters to do this, and saves them in the object `mydes.pars`:

```
mydes.pars<-setpars.design.pl(myreg, n.interval.x=10, n.interval.y=10,
method="random",area.covered = 0.2)
```

(Note the `.pl` in the function name – because this is a function for the `plot` survey method.)

Now that we have defined the survey we can create the survey design called `mydes` using the command below. (We again set the random number seed – to allow you to reproduce the same design we used.)

```
mydes <- generate.design.pl(mydes.pars, seed=1212)
```

We can also plot the design by typing

```
plot(mydes)
```

The function `generate.sample.pl()` is used to generate plot sampling survey results from a population and a design. To store the survey data in the object `mysamp`:

```
mysamp <- generate.sample.pl(mypop,mydes)
```

To get a summary of your sample, type `summary(mysamp)` which gives this:

```
SAMPLE SUMMARY (PLOT METHOD)
```

```
-----
```

```
Number of plots : 20
Individual plot size : 50
Number of detected groups : 37
Mean detections per plot : 1.85
Survey area size : 5000
Percentage covered : 20%
```

To see which groups were detected type `plot(mysamp)`. In the case of our survey, this gives:

If you want to see the undetected groups as well, type  
`plot(mysamp, whole.population=T)`

## 1.4 Estimate abundance

### 1.4.1 Point estimate

Having done the survey, you can now estimate abundance. To do this type:

```
my.point.est<-point.est.pl(mysamp)
```

This puts the results of applying the estimation function `estimate.pl` into the object `my.point.est`. You can look at the contents of this object<sup>3</sup> by typing

```
summary(my.point.est)
```

This gives

```
POINT ESTIMATE SUMMARY (PLOT METHOD)
```

```
-----
```

```
(Maximum likelihood estimator used)
Number of groups detected : 37
Coverage probability : 0.2
Estimated number of groups : 185
Estimated number of individuals: 310
Estimated mean group size : 1.6757
```

The maximum likelihood estimate (MLE) of group abundance ( $N$ ) is 185 here, the individual abundance estimate is 310, and the mean group size is 1.6757 animals per group.

<sup>3</sup> If you want to see what is actually inside the object, type `names(my.point.est)`. By then typing `my.point.est$name` (where *name* is one of the names printed when you typed `names(my.point.est)`), e.g. `my.point.est.$Nhat.grp`, you can see each element of this object. You probably don't want to do this unless you're familiar with R objects.

### 1.4.2 Interval estimate

You can get a 95% confidence interval by

```
my.ci.est<-int.est.pl(mysamp)
```

This produces a plot showing the confidence interval bounds:

The default confidence interval (CI) method is a nonparametric bootstrap (which is what has been used above). Alternatives for the plot sampling method are CIs based on asymptotic normality and parametric bootstrap. Not all interval estimation functions in WiSP have these options.

Typing `summary(my.ci.est)` gives

```
INTERVAL ESTIMATE SUMMARY
```

```
-----
```

```
(Maximum likelihood estimator used)
```

```
(Nonparametric bootstrap confidence interval estimator used)
```

```
Confidence limit percentiles : (0.025; 0.975)
```

```
Group abundance bootstrap mean : 184.83
```

```
Group abundance Standard error : 32.266
```

```
Group abundance Coefficient of variance : 0.17457
```

```
Group abundance confidence limit values : (120; 245)
```

```
Individual abundance confidence limit values: (not yet implemented)
```

```
Mean group size confidence limit values : (not yet implemented)
```

From which we see that the lower and upper confidence level limits for group abundance are (120 and 245 in this case - you might get a different confidence interval, it is a random thing). (Confidence interval estimates for individual abundance, group abundance and mean group size have not yet been implemented.)

In ordinary English these results translate to "Given the survey data, the most likely abundance of the population is 185, and we are 95% sure that the true abundance is between 120 and 245."

## 1.5 Simulation

The results above don't tell us whether our estimator is biased (i.e. whether the average of very many such surveys would be the true abundance). The estimate of 185 might seem a bit low to you, given that there are 250 groups in the population, although the CI does include 250.

We can investigate bias using the function `point.sim.pl()` as follows:

```
my.sim<-point.sim.pl(pop.spec=mypop.pars,design.spec=mydes.pars, B=999,
show=TRUE, seed=54321)
```

This does 999 simulated surveys, randomising both the population (although always using 250 groups) and the design (i.e. where the covered region falls). If you passed `mypop` instead of `mypop.pars` as the first argument the population would not be randomised and if you passed `mydes` instead of `mydes.pars` as the second argument the location of the covered region would not be randomised. The

seed=54321 argument just specifies where the random number generator starts so that you can reproduce the same set of simulations if you want.

You can see the results of the simulations by typing `summary(my.sim)` and `plot(my.sim)`. These give you something like this:

POINT SIMULATION SUMMARY

-----

random number seed used: 54321  
 Design randomization? : TRUE  
 Population randomization?: TRUE  
 Number of simulations : 999  
 (Maximum likelihood estimator used)

Variable name Summary statistics

-----

Group Abundance : True value = 250  
 Mean = 251.81  
 s.e. of mean= 1.3809 (%CV of mean =0.54838)

Animal Abundance : True value = 394  
 Mean = 396.76  
 s.e. of mean= 2.3285 (%CV of mean =0.58686)

Mean Group size : True value = 1.5777  
 Mean = 1.576  
 s.e. of mean= 0.0036379 (%CV of mean =0.23084)

The mean group abundance estimate over all simulations is 252, with a s.e. of 1.4. Because the interval<sup>4</sup> (252-2\*1.4; 252+2\*1.4) includes the true abundance (250), we conclude that the estimation method is unbiased.

Other methods

The estimation process is much the same for methods other than plot sampling, although in other cases it is a bit more complicated because all methods other than plot sampling involve uncertain detection in the covered region – so detection functions need to be specified and estimated.

**Reference:**

Borchers, D.L., Buckland, S.T. and Zucchini, W. 2002. *Estimating animal abundance: closed populations*. Springer. London. 314pp.

---

<sup>4</sup> We are assuming approximate normality here, and using 2 instead of 1.96 as the z-value, to give us an approximate test of the null hypothesis that the mean of the simulations is equal to the true abundance.