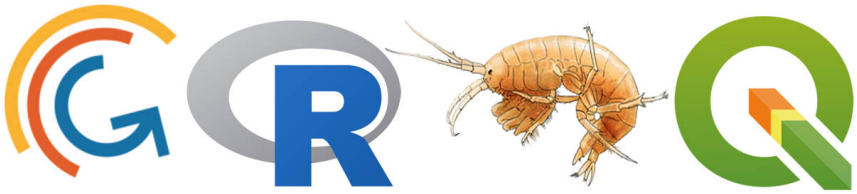
# Multi-agent simulation with Gama and R

## Practical Training Manual

Patrick Giraudoux and Nicolas Marilleau

Version of Monday 5th November, 2018, 14:39



**Prerequisite**: to have acquired basic notions in R (e.g. workspace, package, data.frame, list, for loop, function, etc.) and GIS (e.g. shapefile, attribute table, etc.) and corresponding skills.

# Contents

# Chapter 1

# Model presentation

## 1.1 Ecological bases

This model has been inspired by earlier works carried out by GDRI EHEDE researchers (Li et al., 2015; Clauzel et al., 2015; Li et al., 2017).

In brief, after Li et al. (2015), the black-and-white snub-nosed monkey *Rhinopithecus bieti* is endemic to Yunnan and Tibet, China, and categorized as "endangered" in the IUCN Red List. Surveys have shown that the monkeys live in 15 isolated groups, 12 of them in Yunnan, in a narrow range of the Three Parallel Rivers region, one of the most ecologically important areas of China, the rugged terrain of which makes it difficult to carry out surveys. The species is threatened by habitat alteration, poaching and economic activities such as farming and collection of timber. The areas between populations are damaged by logging, grazing, mining, agriculture and firewood collection. Because of this habitat fragmentation the monkeys may incur a high energy cost if they travel long distances between habitat patches. Fragmentation may subsequently prevent genetic exchange between populations, making the species more vulnerable to extinction. Where required, reserve managers need to establish habitat corridors to facilitate exchange between populations, identifying priority areas for restoration to increase landscape connectivity.

"Snubbies" is a nickname given to the species.

## 1.2 Geography and data

Tab. 1.1 describes the 5 habitat types defined by Li et al. (2017) and fig. 1.1 shows the study area and the location of the monkey groups with their ID number.

- The shapefile `source2P` corresponds to this map, and the predefined style file `source2P.qml` can be used in QGIS for a better display. The attribute `DN` gives the habitat types of Tab. 1.1.

- The shapefile `source2P_envconv` is the concave polygon of `source2P`
- The shapefile `groups` gives the geographical limit of each monkey group. The group ID number, the name of the area and a population size estimate are given as attributes.

Table 1.1: Habitat quality and composition after Li et al. (2017)

| ID | habitat type | Land cover | Altitude | Cost value |
|---|---|---|---|---|
| 1 | Optimal | Armand pine and hemlock, fir-spruce forest, coniferous broad-leaved mixed forest | 2250-4730 | 1 |
| 2 | Suboptimal | Sclerophyllous evergreen broad-leaved forest, shrub-dominated land | 1220-5240 | 10 |
| 3 | Suitable | Cold coniferous forest, sub-alpine meadow, broad-leaved forest | 1310-4950 | 70 |
| 4 | Unfavourable | Warm coniferous forest (Yunnan pine forest), hot dry savanna, sparse shrub grass | 1200-5490 | 90 |
| 5 | Highly unfavourable | Cropland, settlements, water body, barren land | 1215-5410 | 100 |

## 1.3 Model purpose

Here we want to build a multi-agent model to simulate and explore the effect of habitat and life-history trait parameters (real values known or unknown) on snubby dispersion.

## 1.4 Model parameters

### 1.4.1 Input parameters

**step** 1 hour (the iteration time resolution)

**maximum speed** the maximum speed (km/day) that can sustain a snubby in a favourable habitat

**maximum survival** [0,1] the maximum survival in a favourable habitat. The end user give it per year ($s_y$), but this should be converted to the scale of the step (here hour), using the following formula: $s_h = s_y/365 \times 24$ (actually the correct formula should

be $s_h = e^{\frac{\ln s_y}{365 \times 24}}$; we have still to find out why this makes a problem in the current model)

**disperser proportion** the proportion of snubbies who evade their group and wander outside. The end user give it per year ($p_y$), but this should be converted to the scale of the step (hour): $p_h = p_y/365 \times 24$ (actually the correct formula should be $p_h = e^{\frac{\ln p_y}{365 \times 24}}$)

**habitat viscosity** [0,1] the resistance of a given habitat. Multiplies the maximum speed to give the speed really sustained in this habitat.

**habitat security** [0,1] the security of a given habitat. Multiplies the maximum survival to give the actual survival in this habitat.

**habitat dislike** [0,0.5] the probability for a snubby to move into a habitat of lesser quality.

**habitat preference** [0.5,1] the probability for a snubby to move into a habitat of better quality.

### 1.4.2 During simulation

When one snubby dies, one snubby is created in its aboriginal group.

### 1.4.3 Output parameters

**Bitmap** Bitmap of snubby distribution (framerate: 6 months).

**Shapefile** Shapefile of snubby distribution with their individual attributes (framerate: 1 year).

**group composition and wanderers** Snubby number in each group and outside, split by group origin.

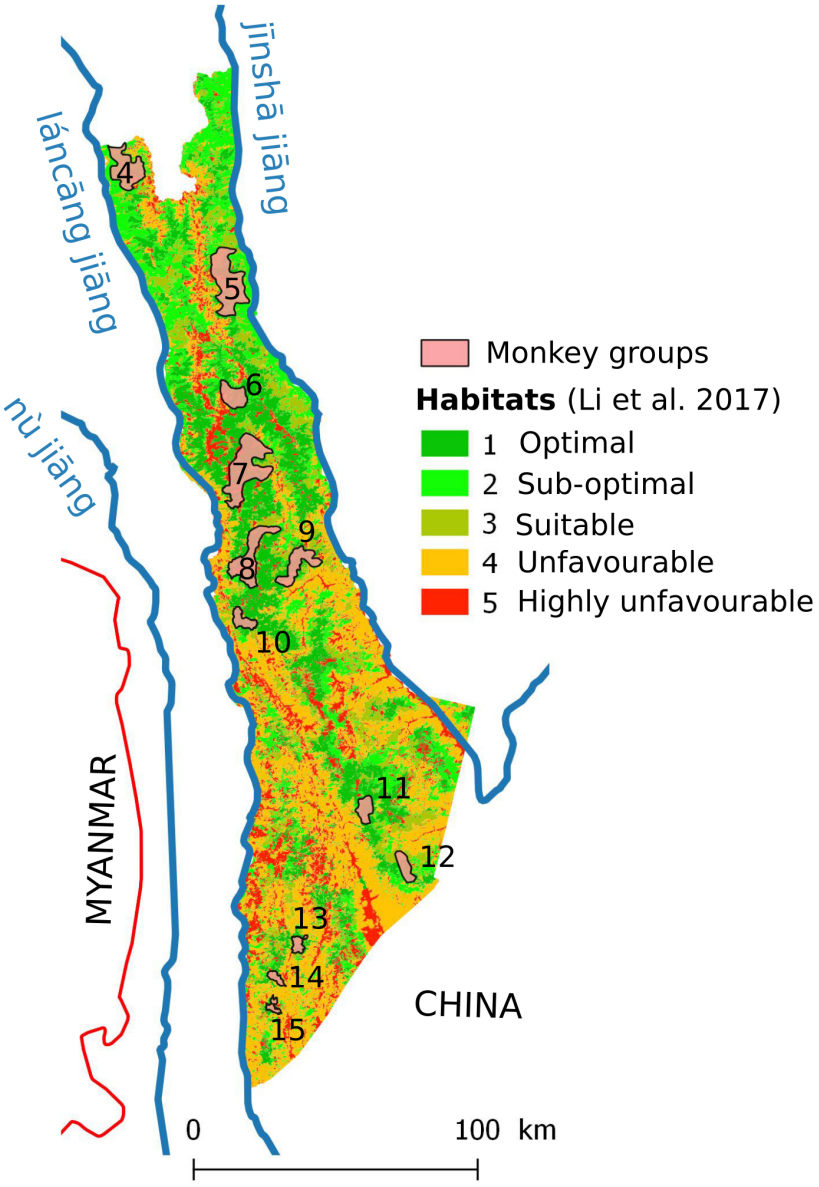



Figure 1.1: Study area. Numbers on the map are the group IDs.

# Chapter 2

# Gama model

## 2.1 Principle

See:

- Grignard et al. (2013)
- `https://gama-platform.github.io`
- `https://github.com/gama-platform/gama/wiki/Overview`
- `https://github.com/gama-platform/gama/wiki`
- `https://github.com/gama-platform/gama/wiki/Download`

## 2.2 Gama in practice

See practical training on November 6-7, 2018.

# Chapter 3

# Preparing experiments and analyzing results in R

Most likely, for full size research, you will carry out simulations on intensive computing systems with hundreds of cores available in parallel, to run thousands of experiments. Intensive computing systems generally use UNIX and simulations can be run in shell scripts using gama-headless.bat as an entry point. R can be run on such a platform as well and be used for simulations. Here, for teaching purpose, this training manual presents how to do it in R on Windows using a personal computer. Your computer has likely no more than 2-4 CPU, and it is advisable not to allocate more than two to the job (we will see later how to do it). In this context, running a large number of simulations or to simulate on a large number of time steps can be time consuming.

## 3.1 Installing gamar

`gamar` is a R package available at `https://github.com/choisy/gamar`.

With the `library(devtools)`, if not already installed, it can be installed on line as following:

```
1 library(devtools)
2 devtools::install_github("choisy/gamar")
3 library(gamar)
```

First, one declares the path to the program Gama.exe. Here below an example, but the path you are going to write must be adapted to your own computer:

```
1 defpath("C:/Program Files/GAMA1.7RC2.Windows.64bits")
```

## 3.2 Preparing experiments

### 3.2.1 Reading the model

Now, one must read the model (here `snubbies.gaml`) and extract informations that will be used by R. The path to the model must be declared explicitely (here `snubbies.gaml` is in the folder `"U:/Users/pgiraudo2/gama_workspace/snubbies/models"`). Again this to adapt to your own computer.

```
experiment1 <- getmodelparameter("U:/Users/pgiraudo2/gama_
    workspace/snubbies/models/snubbies.gaml","run")
experiment1
```

For convenience, we have prepared some home-made functions to ease readings and parameter handling. They are in the file `HomeFunctions.r`. One sources it, and get a data.frame with attributes:

```
source("Homefunctions.r")
ex1<-expe2df(experiment1)
ex1
attributes(ex1)
```

### 3.2.2 Preparing the simulations

The duration of each simulation is stored in a variable, here named `simulation_duration` and the number of experiments to run in the variable `B`

```
simulation_duration <- 2 * 365 * 24 # two years (step is one
    hour)
B<-2 # number of simulations
```

Now, a data.frame must be prepared with each row the parameter values of each experiment. Note that parameters can be explored randomly allocating values between limits. See for instance the function `runif` in R (but many others are available).

```
?runif
```

Values can be defined quite flexibly and the value(s) of a given parameter can be used as a function of another. See example below.

```
simuls<-data.frame(
v_max=0.3472222,
s_max=0.5,
explorer_snubbies=runif(B,0.1,0.5),
viscosity_factor_habitat_1=1,
viscosity_factor_habitat_2=0.9,
viscosity_factor_habitat_3=0.5,
viscosity_factor_habitat_4=0.1,
viscosit_factor_habitat_5=0,
security_factor_habitat_1=1,
security_factor_habitat_2=0.9,
security_factor_habitat_3=0.5,
security_factor_habitat_4=0.1,
security_factor_habitat_5=0
)

simuls
```

Then, experiments can be prepared. This will end up by the creation of one variable `experimentplan` and as many XML files as experiments (e.g. the number of row of `simuls`). Pay attention about the unusual way to specify the path with `setmodelpath`, with a slash "/U:/..." at the beginning. This is a provisory way to get around a bug of gama-headless running on Windows.

```
for (i in 1:nrow(simuls))
{
local_experiment <- experiment1
local_experiment <- setparametervalue(local_experiment,"
   simulation_id",i)
local_experiment <-  setparametervalue(local_experiment,"max_
   snubby_survival_init",simuls[i,"s_max"])
local_experiment <-  setparametervalue(local_experiment,"max_
   snubby_speed",simuls[i,"v_max"])
local_experiment <-  setparametervalue(local_experiment,"max_
   snubby_survival_init",simuls[i,"s_max"])
local_experiment <-  setparametervalue(local_experiment,"
   explorer_snubbies_init",simuls[i,"explorer_snubbies"])
local_experiment <-  setparametervalue(local_experiment,"
   viscosity_init_habitat_1",simuls[i,"viscosity_factor_
   habitat_1"])
```

```
10 local_experiment <-  setparametervalue(local_experiment,"
     viscosity_init_habitat_2",simuls[i,"viscosity_factor_
     habitat_2"])
11 local_experiment <-  setparametervalue(local_experiment,"
     viscosity_init_habitat_3",simuls[i,"viscosity_factor_
     habitat_3"])
12 local_experiment <-  setparametervalue(local_experiment,"
     viscosity_init_habitat_4",simuls[i,"viscosity_factor_
     habitat_4"])
13 local_experiment <-  setparametervalue(local_experiment,"
     viscosity_init_habitat_5",simuls[i,"viscosity_factor_
     habitat_5"])
14 local_experiment <-  setparametervalue(local_experiment,"
     security_init_habitat_1",simuls[i,"security_factor_habitat_
     1"])
15 local_experiment <-  setparametervalue(local_experiment,"
     security_init_habitat_2",simuls[i,"security_factor_habitat_
     2"])
16 local_experiment <-  setparametervalue(local_experiment,"
     security_init_habitat_3",simuls[i,"security_factor_habitat_
     3"])
17 local_experiment <-  setparametervalue(local_experiment,"
     security_init_habitat_4",simuls[i,"security_factor_habitat_
     4"])
18 local_experiment <-  setparametervalue(local_experiment,"
     security_init_habitat_5",simuls[i,"security_factor_habitat_
     5"])
19
20 local_experiment <- setfinalstep(local_experiment,simulation_
     duration)
21 local_experiment <- setoutputframerate(local_experiment,"
     simulation_id",24)
22 local_experiment <- setoutputframerate(local_experiment,"
     number_of_dispersers",24)
23 local_experiment <- setoutputframerate(local_experiment,"
     reading_map",30*24) # save the corresponding bit map every
     30 days
24 local_experiment <- setsimulationid(local_experiment,i)
25 local_experiment <- setmodelpath(local_experiment,"/U:/Users/
     pgiraudo2/gama_workspace/snubbies/models/snubbies.gaml")
```

```
26
27 local_experiment <- setseed(local_experiment,1)
28
29 if(i<2)
30 {
31
32 experimentplan <- addtoexperimentplan(simulation = local_
   experiment)
33 }
34 else
35 {
36 experimentplan <- addtoexperimentplan(simulation = local_
   experiment,experimentplan = experimentplan)
37 }
38 outFile <- paste0("./input_",formatC(i,width=nchar(nrow(simuls
   )),flag="0"),".xml")
39
40 writemodelparameterfile(experimentplan,outFile)
41 }
```

The variable of interest is `experimentplan`. However here, additionally, XML files are saved on the disk with `writemodelparameterfile`. This can be useful if simulations are run not from R on intensive computing systems and UNIX platforms. There, running simulations with a UNIX shell script is system dependent and will use those XML files (in this case, see with the IT manager how to proceed). Here, with this R script, XML files have been written in the workspace. You can see them either using Windows Explorer or from R with the following command:

```
1 dir()
```

## 3.3 Running simulations

Simulations are run very simply then. We add some code lines to get the duration of the simulations at the end.

```
1 t0<-Sys.time() # store the start time
2 runexpplan(experimentplan,hpc=2) # run the simulations on 2
   CPU
3 t1<-Sys.time() # store the end time
```

```
t1-t0
```

## 3.4 Reading results

### 3.4.1 XML

Simulations results are stored in the folder `./workgamar`. There one finds:

- as many files as experiments, named `run_1.xml`, `run_2.xml`, etc. Those files just describe the values of the input parameters. No point to read them, all useful information is in the data.frame `simuls` (see above 3.2.2).
- a folder named `out_run_1` (or `out_run_2`, `out_run_3`, etc. depending on the number of runs executed before). One finds the following inside:
  - a folder named `snapshot` containing bitmaps. Note the file name gives the number of the experiment and the step. E.g. `reading_map0-10.png` means experiment #1 (the experiment numbers are incremented from 0), step 10.
  - text files `console-outputs-0.txt`, `console-outputs-1.txt`, etc. corresponding to the console outputs of experiment #1, #2, etc. They are not of much interest when everything goes smoothly, but useful to identify where a simulation has crashed, should it happen.
  - xml files `simulation-outputs0.xml`, `simulation-outputs01.xml`, etc. corresponding to the outputs of experiment #1, #2, etc.

One needs some R code to read the XML file. Here we use a home-made function `readNamesXMLOutputs` to read variable names (it as been sourced before see 3.2.1)

```
library(XML)

path<-"./workgamar/out_run_1/" # the path to the XML files
outs<-dir(path,pattern="*.xml") # one reads them
outs<-outs[order(as.numeric(substr(outs,19,nchar(outs)-4)))] #
    to get the files in the right order whatever the number of
    digit (to avoid this order: 1,11,12,13,...,2, 20, 21,...)

outs<-paste0(path,outs) # now the full path with file names
outs

```

```
mycolnames<-readNamesXMLOutputs(outs[1]) # one uses a home
   made function to extract the variables names from one of
   the XML files

results<-rep(list(NA),length(outs)) # one prepares an empty
   list to store the readings
for(i in 1:length(outs)) {
  results[[i]]<-xmlToDataFrame(outs[i])  # we read the XML
     file into a data.frame
  names(results[[i]])<-mycolnames # one gives names to columns
}

names(results)<-outs # one gives a name to each experiments

head(results[[1]])
```

### 3.4.2 CSV

In the folder `outputs` one finds CSV files whose names look like that: `transfer_2019_7200.csv`. They are the files which store monkey distribution in the groups at each selected time step.

```
myfiles<-dir("./outputs",full.names = TRUE)
trans<-rep(list(NA),length(myfiles))
for(i in 1:length(myfiles)) trans[[i]]<-read.csv(myfiles[i],
   header=F)
```

Each file is in a list.
First, let us extract the first simulation of the 10th saving.

```
tabt<-trans[[10]][1:13,]
colnames(tabt)<-paste0("G",tabt[1,])
rownames(tabt)<-paste0("G",tabt[1,])
tabt<-tabt[-1,-1]
tabt

> tabt
     G6  G5 G15 G13 G14 G4 G12 G11  G9  G8  G7 G10
G6   48   0   0   0   0  0   0   0   0   0   0   0
G5    1 300   0   0   0  0   0   0   1   0   0   0
G15   0   0  98   2   1  0   0   0   0   0   0   0
```

```
12 G13   0    0    0 116    0   0    0    2    1    0    0    0
13 G14   0    0    2   2   98   0    0    0    0    0    0    0
14 G4    0    0    0   0    0  80    0    0    0    0    0    0
15 G12   0    0    0   0    0   0   50    3    1    0    0    0
16 G11   0    0    0   0    1   0    0  243    1    1    0    0
17 G9    0    0    0   0    0   0    0    2  885    0    0    0
18 G8    0    0    0   0    0   0    0    0    5  197    0    1
19 G7    1    0    0   0    0   0    0    0    2    2  100    0
20 G10   0    0    0   0    0   0    0    0    4    0    0   29
```

The first column gives the current distribution of individuals of the G6 group. 48 stayed in their group, 1 left for G5 and 1 for G7.

The sum of the first row, gives the current total number of individuals in a group. G6 has lost 2 individuals, so the sum of the row is 48. But G5 has gained 2 individuals, one from G6, the other from G9, but did not lose any, hence the size of the group is 1+300+1=302 (the sum of the row). The total number of individual in each group is easy to obtain:

```
1 rowSums(tabt)
```

### 3.4.3 Shapefile

In R one can use the package `rgdal`.

```
1 library(rgdal)
2 snub<-readOGR("U:/Users/pgiraudo2/gama_workspace/snubbies/
    outputs","snubby_2019_4392")
3 head(snub@data)
4 plot(snub)
5 plot(snub[snub@data$origin!=snub@data$current,],col="red",fg="
    red",add=TRUE)
6
7 groups<-readOGR("U:/Users/pgiraudo2/gama_workspace/snubbies/
    includes/groups","groups")
8 plot(groups)
9 plot(snub[snub@data$origin!=snub@data$current,],col="red",fg="
    red",add=TRUE)
```

The shapefile can be read from any GIS as well.

In both cases monkey location is given as a circle polygon (of class `SpatialPolygonsDataFrame` in R), which might not be optimal for display.

In R one can easily extract the centroid of each polygon:

```
snubpts<-getSpatialPolygonsLabelPoints(snub)
plot(snubpts,pch=1,cex=0.5)
plot(snubpts[snub@data$origin!=snub@data$current,],pch=19,col=
   "red",fg="red",add=TRUE)
```

One can extract the geographical coordinates alone into a matrix as well:

```
snubmat<-coordinates(snub)
```

## 3.5 Analysing results

To come soon...

# Bibliography

Clauzel, C., Deng, X. Q., Wu, G. S., Giraudoux, P., & Li, L. (2015). Assessing the impact of road developments on connectivity across multiple scales: Application to Yunnan snub-nosed monkey conservation. *Biological Conservation*, *192*, 207–217.

Grignard, A., Taillandier, P., Gaudou, B., Vo, D. A., Huynh, N. Q., & Drogoul, A. (2013). GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. In G. Boella, E. Elkind, B. T. R. Savarimuthu, F. Dignum, & M. K. Purvis (Eds.) *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, Lecture Notes in Computer Science, (pp. 117–131). Springer Berlin Heidelberg.

Li, L., Xue, Y., Wu, G. S., Li, D. Q., & Giraudoux, P. (2015). Potential habitat corridors and restoration areas for the black-and-white snub-nosed monkey Rhinopithecus bieti in Yunnan, China. *Oryx*, *49*(4), 719–726.

Li, W. W., Clauzel, C., Dai, Y. C., Wu, G. S., Giraudoux, P., & Li, L. (2017). Improving landscape connectivity for the Yunnan snub-nosedmonkey through cropland reforestation using graph theory. *Journal for Nature Conservation*, *38*, 46–55.