**Modeling Notebook notes for TRACE – BLT Model**

This is not a TRACE document, but a TRACE notebook. General guidelines for TRACE are found on [Grimm et al 2014](http://dx.doi.org/10.1016/j.ecolmodel.2014.01.018) (structure on Table 1).

Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente com confiança média

|  |  |
| --- | --- |
| **TRACE element/MN entry tag** | **MN keyword** |
| 1. Problem formulation | Model purpose; Research questions |
| 2. Model description | Model development; Design decisions |
| 3. Data evaluation | Parameterization; Patterns |
| 4. Conceptual model evaluation | Conceptual design decisions |
| 5. Implementation verification | Debugging  Software verification/Testing  Usability tools design |
| 6. Model output verification | Output verification/Goodness- of-fit Calibration; Tests on environmental drivers |
| 7. Model analysis and application | Sensitivity analysis; Uncertainty analysis Robustness analysis; Simulation experiment |
| 8. Model output corroboration | Output corroboration/Validation |

Implementation 1 refers to Step length + turning angles

implementation 2 refers to Resource visitation rules

implementation 3 refers to Territoriality

**Examples of topics to be included each day and a brief description:**

/purpose

To validate a simulation model of BLT dispersing seeds

/data

Check .xlsx

/verification

/calibration

/calibration/direct parametrization

Obtaining parameter values directly from the literature or experts

/calibration/inverse parametrization

Obtaining parameter values inversely by calibrating the model to observa- tions

/sensitivity analysis

/sensitivity analysis/submodels/energy

/sensitivity analysis/local

Varying one parameter at a time

/sensitivity analysis/global

Varying several or all parameters over their whole ranges

/alternatives

\_\_\_\_\_\_\_

\_\_\_\_\_\_\_

**2. Model description**

**February 04th 2022**

Discussion with Ronald about implementing turning angles and step lengths. We realized we need to check the empirical data a little bit further in order to implement a rule based on sl and ta in the model. I decided to check in [a published ABM](https://doi.org/10.1111/oik.07431) how it was done. Turning angles was implemented with von Mises distribution base on rvm-function from {CircStats} package, however I didn’t have a code for step length. Other two papers did similar implementations (Raghunathan et al 2020 and Gazagne et al 2020) with an HMM model but used also a gamma distribution to step lengths. With this I’ve checked the “Check if the impl. 1 needs model parameters” task. The answer is: Yes, it does. Gamma distribution for step lengths and von Mises distribution for turning angles. **For DPL and Path Twisting: don't know yet**.

After meeting with Ronald at 13:30 we decided to first have the nlrx workflow (to check results after running experiments) ready before making an implementation. Most importantly, we discussed more about step length/turning angles relation, I showed him the results from the ATBC and runned emptyspace analysis, explaining again the idea of taking time windows of resources and clumping measurements to make the home range emerge together with a DPL and Path Twisting based on a random point process specifying the tree aggregation. For last, I explained to him why I didn’t believe tamarins were going after resources and cited the mechanistic home range/central place forager hypothesis.

**February 09th 2022**

*Keywords: Model development; Design decisions*

From Feb 07th to now I’ve been troubleshooting Mayara’s code to match with the nlrx workflow. Two problems have risen:

1. Model output with nlrx to set workflow with R: even though it works with the Wolf-Sheep model, when I try with the BLT model, the px and py positions of the simulations don’t come out with **unnest\_simoutput()** function. Alternatively, [the nlrx vignette (see Step 3) explaining how to take manual output](https://docs.ropensci.org/nlrx/articles/manual-output.html) and relate to the nlrx object requires a reporter in NetLogo code, but when I insert it as a global, NetLogo gives me an error saying I should define it.
2. to-sleeping-trees procedure does not work when using the code for empirical sleeping trees. Thus I’ve created a Chooser **“sleeping-trees-scenario”** of whether simulated sleeping and resting trees are used or not (but only the code for simulated sleeping/resting trees work -> check **to-sleeping-trees** and **search-sleeping** procedure and sleeping-trees-here object is unused)

I also did the following:

* settled a **“tree-scenario”** Chooser for selecting which .shp with feeding trees was being used as input;
* settled an /runtime directory in model interface.

**February 15th 2022**

On the Feb 09th I started receiving this “Exception in thread "main" java.lang.IllegalStateException: unknown NetLogo version: NetLogo 6.2.0”. Ronald and I tried setting a stop condition because sometimes the model would run without stopping. Then I had to leave, and Ronald tried debugging the nlrx code. He was able to run the Mayara’s (the version he had) model, but mine was giving the following error together with same thread:

Exception in thread "main" Expected a literal value. at position 0 in

at org.nlogo.core.Fail$.exception(Fail.scala:27)

at org.nlogo.core.Fail$.exception(Fail.scala:25)

at org.nlogo.core.Fail$.exception(Fail.scala:23)

at org.nlogo.parse.LiteralParser.readLiteralPrefix(LiteralParser.scala:83)

at org.nlogo.parse.LiteralParser.getLiteralValue(LiteralParser.scala:33)

at org.nlogo.parse.CompilerUtilities$.$anonfun$readFromString$3(CompilerUtilities.scala:22)

at org.nlogo.parse.CompilerUtilities$.$anonfun$numberOrElse$1(CompilerUtilities.scala:37)

at scala.util.Either$RightProjection.getOrElse(Either.scala:665)

at org.nlogo.parse.CompilerUtilities$.numberOrElse(CompilerUtilities.scala:36)

at org.nlogo.parse.CompilerUtilities$.readFromString(CompilerUtilities.scala:22)

at org.nlogo.fileformat.LabLoader.$anonfun$readProtocolElement$4(LabLoader.scala:70)

at scala.collection.TraversableLike$WithFilter.$anonfun$map$2(TraversableLike.scala:874)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at scala.collection.AbstractIterator.foreach(Iterator.scala:1431)

at scala.collection.IterableLike.foreach(IterableLike.scala:74)

at scala.collection.IterableLike.foreach$(IterableLike.scala:73)

at scala.collection.AbstractIterable.foreach(Iterable.scala:56)

at scala.collection.TraversableLike$WithFilter.map(TraversableLike.scala:873)

at org.nlogo.fileformat.LabLoader.readEnumeratedValueSetElement$1(LabLoader.scala:66)

at org.nlogo.fileformat.LabLoader.$anonfun$readProtocolElement$5(LabLoader.scala:77)

at scala.collection.immutable.List.flatMap(List.scala:366)

at org.nlogo.fileformat.LabLoader.valueSets$1(LabLoader.scala:74)

at org.nlogo.fileformat.LabLoader.readProtocolElement(LabLoader.scala:94)

at org.nlogo.fileformat.LabLoader.$anonfun$apply$1(LabLoader.scala:45)

at scala.collection.immutable.List.map(List.scala:293)

at org.nlogo.fileformat.LabLoader.apply(LabLoader.scala:45)

at org.nlogo.fileformat.LabLoader.apply(LabLoader.scala:30)

at org.nlogo.fileformat.LabFormat.load(NLogoLabFormat.scala:37)

at org.nlogo.fileformat.LabFormat.load$(NLogoLabFormat.scala:35)

at org.nlogo.fileformat.NLogoLabFormat.load(NLogoLabFormat.scala:40)

at org.nlogo.headless.BehaviorSpaceCoordinator$.$anonfun$selectProtocol$1(BehaviorSpaceCoordinator.scala:34)

at scala.Option.map(Option.scala:230)

at org.nlogo.headless.BehaviorSpaceCoordinator$.selectProtocol(BehaviorSpaceCoordinator.scala:32)

at org.nlogo.headless.Main$.runExperiment(Main.scala:23)

at org.nlogo.headless.Main$.$anonfun$main$1(Main.scala:12)

at org.nlogo.headless.Main$.$anonfun$main$1$adapted(Main.scala:12)

at scala.Option.foreach(Option.scala:407)

at org.nlogo.headless.Main$.main(Main.scala:12)

at org.nlogo.headless.Main.main(Main.scala)

**Error in util\_gather\_results(nl, outfile, seed, siminputrow) :**

**Output file is empty - simulation aborted due to a runtime error!**

**Make sure that parameter value definitions of the experiment are valid and the model code is running properly!**

So I went to the model (BLT\_model\_developm.nlogo) which I did the last implementations and realized that the model was breaking (running eternally) sometimes it was initiated. Then I checked the go, move-monkeys, run\_days and next\_day procedure. I did the following:

* go: nothing
* move-monkeys: moved in the “timestep > simulation-time” condition from the to go procedure and broke the multiple ifelses with the ‘if timestep < simulation-time’. I also took out from the setup the ‘if timestep = simulation-time’ condiditon (it was before the move-monkeys) and included it in the move-monkeys procedure
* run\_days: blocked the output code
* next\_day: added a stop condition: if day > no\_days [ stop ]
* commented out all the output procedures

The model was not breaking anymore but it still didn’t work out in the nlrx.

After one or two hours trying to debug it (BLT\_model\_developm-debug.nlogo), I assessed [this issue in nlrx](https://github.com/ropensci/nlrx/issues/27) and realized that the categorical variables should have a double quote like this:

constants = list( 'tree-scenario' = "\"trees\_all\_2\"",  
 'sleeping-trees-scenario' = "\"simulated\"" )

Now the problem is: I still can’t take pxcor and pycox coordinates. Maybe because I was doing one run with only a simple design? But how, if the tamarin run for days and 108 steps per day in different pxcor and pycor? The solution I found was setting up the number of runs in nlrx (and also in the model interface) to 1, and then specifying more runs in “repetitions” on the @experiment design

Milles et al 2020 model had an export-view function linked to an interrupter which allowed him to export the view of the run when it ended. I implemented it in a separate model (BLT\_model\_2022-developm\_issue-print-final-landscape.nlogo) but I was not able to see the tracks tamarins did. I let it like that for now.

Fiquei de fazer o cabeçalho dos arquivos output (que nem em Milles et al 2020) e de lembrar que diabos é aquele número no início de cada arquivo (o Ronald tinha descoberto já).

By the start of the night, I reassessed the move-monkeys code related to routine (the one I changed the ifelse condition in timestep < simulation time) and when I used the same code as Mayara I had a model break (infinite running). I could not understand what it was, but I think there lacks a stop condition.

I have also noticed that the tamarin was moving much less than I was used to see in the model, but it might be because I’m restricting the no\_days. -> Nope. It was because I had commented out the ‘let n random 100’ argument from the ‘search-sleeping-tree’ procedure. This is basically what moves tamarins.

Left to fix/decide:

* (SOLVED) Why the output of seeds and sleeping, resting and feeding trees text files contain information only for the last ran day even though the Mayara model works perfectly.

**February 16th**

I have realized that Mayara’s model does not output seed and trees location by day. The problem was that I had copied the code ‘file-write day’ and ‘file-write timestep’ into the procedures ‘write-seeds’, ‘write-sleep’, ‘write-rest’ and ‘write-trees’.

Thus, I proceeded to add a header on each of those files and I also changed it to .csv

While running Mayara’s model I also realized that the simulations are faster when the locations of each day are written in a different file

I was succeed in extracting the pxcor and pycor from each agent by unnest\_simoutput() and proceeded with the spatial plot vignette. With this plot (one run of I don’t know how many days) it was possible to visualize that the rule of creating sleeping sites is not working properly (too many sleeping sites, a lot of them in the matrix):

Tela de jogo de vídeo game

Descrição gerada automaticamente com confiança média

Left to fix/decide:

* Check if using different files for each day of monkey locations increase velocity of the model run
* go/run\_days/next\_day procedures have two stop conditions. I think this is breaking the execution from nlrx
* include day in the nlrx simoutput
* gif animation and sequential plots not working: try to include an informative unique variable for each run (seed-number is not informative at all) -> patches explodindo
* Make the model run with the empirical sleeping sites

**5. Implementation verification [?]**

/extensions/gis

Got the code of nlrx running for Mayara’s model. Problem: It is generating one file for each day, thus it dumps everything in github

Files and code: [path](file:///D:\Data\Documentos\github\BLT_IBM-Model\Model_development)

Script: 00\_start-nlrx.R

Left to fix/decide:

* If I’m keeping model output as external files

/extensions/gis

**Feb 2nd 2022**

Debugged shapefiles of the three areas (Guareí, Suzano and Taquara) to initialize with a chooser and and if condition.

Files and code: [path](file:///D:\Data\Documentos\github\BLT_IBM-Model\Model_development\gis-extension)

Included a procedure to check if turtles are inside the shapefile: ‘check-agent-in-fragment’

Tried looking how to scale the different locations. Didn’t find anything.

Left to fix/decide:

* Better .shp files to NetLogo (for Suzano and Taquara, too big)

**6. Model output verification/calibration**

**Day 1 – January 14th**

Six steps for making a good calibration (Railsback & Grimm 2012 Chap 20):

1. Identify a few good parameters

Important, uncertain and independent parameters. To know convincingly which parameters are important, we need sensitivity analysis. It is prudent to reconsider model calibration after this sensitivity analysis. A parameter that is highly uncertain but has little effect on results should not be used for calibration.

Decision for the BLT model: activity budget (e.g. time spent travelling?)

1. Choose Categorical vs. Best-Fit Calibration

Categorial gives a range of values (e.g. mean number of agents between 120-150), while Best-Fit Calibration gives single values (e.g. mean number of agents = 127).

Decision for the BLT model: Categorical seems the best option

1. Decide Whether and How to Use Time-Series Calibration

“If our model's purpose includes representing how results change over time (e.g., how long does it take the system to recover from some perturbation? How is the system affected by changes in its environment over time?), then it usually does make sense to use time-series calibration. But some ABMs (e.g., the woodhoopoe model of section 19.4.3) are intended to explain long-term average conditions, so they intentionally do not contain all the processes that cause the real system to change over time and use no input data to represent how the agents’ environment changes over time. In such cases, time-series calibration may not be useful or necessary.”

Decision for the BLT model: As I’ll only be focusing on the mean and sd numbers at the end of the runs, I believe I don’t need Time-Series Calibration

1. Identify Calibration Criteria

* Calibrate the model against all interested patterns
* Of apples and oranges: the observations need to match the same time and space scale
* There are kinds and kinds of variation: Calibrating variability measures (CV, sd) should be done carefully
* Often our data is inaccurate and we need to know if it is inaccurate by 10, 20 or 50%. Using uncertain data is OK and unavoidable: when the calibration patterns are more uncertain, we don't worry so much about matching them exactly, and we recognize that calibrated parameter values are less certain. But we need to have at least some idea how accurate or certain the observations are, so we know how much information they really contain
* We must specify how we will compare the observed patterns to model results. Of particular concern is how to calibrate several different kinds of model results at once: if we want to calibrate a model to reproduce the number, size, and wealth of agents, how do we decide between a set of parameters that reproduces number and wealth well but not size, and a parameter set that reproduces size and wealth well but not the number of agents?
* At the end of this step of defining calibration criteria, we should have a specific algorithm for quantifying how well a set of model results reproduces the selected observations

Decision for the BLT Model:

* Calibrate against SDD, DPL, Home range and Activity Budget
* (In order of importance) prioritize SDD, DPL, Home range and Activity Budget
* Algorithm: the mean (min and max too?) SDD, the Home range and Activity Budget evaluated after 30 running days and the DPL of each of these days compared to data of Guareí (PEMD and Suzano to be independent observations serving as validation?)

1. Design and Conduct Simulation Experiments

This calibration experiment executes the model many times, using combinations of values over the feasible range of all parameters. The results of this experiment will tell us what ranges of parameter values produce results that meet the calibration criteria. Steps:

* Select values for the non-calibration parameters and input data (if any) that represent the conditions (the same time period, environment, etc.) under which calibration patterns were observed.
* Define *parameter space* (It is usually good to include values that bound the range of feasibility) (see Figure 20.3)
* If the model is stochastic, analyze means seems right, but plot model results against parameter values [? p. 357]

Decision for the BLT model:

* Select all non-calibration parameters an input data from Guareí (using values that bound the range of feasibility, e.g. energy-from-fruit 0)

1. Analyze Calibration Experiment Results

* If your ABM does meet all the calibration criteria for one or more combinations of parameter values, then you can in fact complete this step and move on to the kinds of analysis we address in part IV
* What should you do if you cannot meet all the calibration criteria at once?

1. Screen for errors (code, submodels, etc)
2. If there isn’t any error, your model is too simple or too simple in the wrong ways, to reproduce the observed patterns you chose as calibration criteria. You could consider going back to the theory development stage and seeing if you can reproduce the observed patterns you chose as calibration criteria. You could consider going back to the theory development stage and seeing if you can improve the traits for agent behavior, and you could consider adding processes or behaviors that were left out the first time. But keep in mind that there are costs to adding complexity to your model, especially if it is not very clear what change needs to be made.
3. [Alternatively] It is very common for good modelers to keep their model simple instead of adding stuff to it until the model can reproduce all the calibration criteria. Keep in mind the overfitting issue: it can be risky to try too hard to make the model reproduce a limited set of observations. It may make sense to revise your calibration criteria so they are not as restrictive. If you choose not to revise the model to make it fit more of the calibration criteria, simply document your calibration results and the extent to which the model does not meet some criteria under your “best” parameter values, and your decision not to revise the model. Then, when you use the model to solve problems, keep in mind which results are less certain as indicated by the calibration experiment.

Decision for the BLT Model: I can’t make any decision right now, but the comment 3) seems very important. I’ll probably proceed if I find a set of values that meet the first 2 variables (SDD, DPL), and then Home Range size (not core home range as it is mostly not stable throughout time)

Left to fix/decide: