User Manual of Zambia ABM Framework

[1. Introduction 2](#_Toc414303303)

[2. Software Dependencies 2](#_Toc414303304)

[3. Building the Framework 2](#_Toc414303305)

[4. Framework Configuration 3](#_Toc414303306)

[4.1. Environment variables 3](#_Toc414303307)

[4.2. Configuring Properties and Custom Parameters 3](#_Toc414303308)

[5. Running Simulations 5](#_Toc414303309)

[5.1. Input Data 5](#_Toc414303310)

[5.2. Start the Simulation 5](#_Toc414303311)

# Introduction

This framework provides automatic data preparation, model execution and data post-processing for Zambia Resilience project. It handles the large scale input data by dividing it into smaller sets following geographical boundaries and running multiple instances of NetLogo in different threads. It also allows a custom feature space in its configuration file to support parameter sweeping. The model execution has two phases: ‘world’ creation and crop production. To reuse the results from the first phase, the framework exports the ‘world’ environment from the first phase to a file on disk, and then imports it at the beginning of second phases. Besides, both phases can be repeatedly executed using different random seed (Montel Carlo Style).

# Software Dependencies

This framework has been built with the following software packages on which it has a dependency. Some of the packages are included in the library:

Opencsv v2.3: a simple CSV parser for Java under a commercial-friendly Apache 2.0 license.

While others need to be installed separately:

1. Apache ANT v1.6 or higher (for building the tool from source)

<http://ant.apache.org>

1. Java Development Kit (JDK) v5 or v6

<http://java.sun.com>

1. NetLogo v5.0.2 or higher

<http://ccl.northwestern.edu/netlogo/>

1. R v2.5 or higher

<http://www.r-project.org/>

1. R package ‘classInt’: Choose univariate class intervals

<http://cran.r-project.org/web/packages/classInt/index.html>

1. rJava - Low-level R to Java interface

<http://www.rforge.net/rJava/>

# Building the Framework

Take the following steps to build the framework from source:

1. Unzip the ZambiaABM.tgz, which contains all the source code and sample input and model, as:

|  |
| --- |
| tar xvzf ZambiaABM.tgz |

1. Setup the environment properties for ANT in file ‘**build.properties**’:

|  |  |
| --- | --- |
| Environment Variable | Explanation |
| R\_HOME | Home directory where R is installed |
| JRI\_LIB | Directory where jri libraries are |

1. Save the modified ‘build.properties’ file and build with ANT command:

|  |
| --- |
| ant |

Note: if you would like to start over, you may use the following command.

|  |
| --- |
| ant clean |

# Framework Configuration

# Environment variables

This framework depends on a number of software packages for a successful execution, and you need to specify, in script **‘createWorlds.sh’** and **‘run.sh’,** the correct environment variables that point to these software packages. Below is the detailed explanation of these environment variables:

|  |  |
| --- | --- |
| Environment Variable | Explanation |
| NETLOGO | Home directory where NetLogo is installed |
| R\_HOME | Home directory where R is installed |
| JRI\_LIB | Directory where jri libraries are |

A list of example environment variables is included in default script ‘createWorlds.sh’ and ‘run.sh’:

|  |
| --- |
| export NETLOGO=/u/chenpeng/netlogo-5.0.2  export R\_HOME=/usr/lib64/R  export JRI\_LIB=~/R/R-2.15.2/library/rJava/jri |

# Configuring Properties and Custom Parameters

The distribution package contains a properties file, **`simulator.properties`**, which has a number of properties and custom parameters to be specified. We also give a detailed explanation for each property below:

|  |  |
| --- | --- |
| Property or Parameter | Explanation |
| model.path | Path to the NetLogo model file used in simulation. |
| raster.width  raster.height | Dimension of the input raster file. |
| wards\_raster.path | Path to the input ward ID raster file. |
| landcover\_raster.path | Path to the input landcover raster file. |
| summary\_file.path | Path to a summary file that lists all the wards and their dimensions. |
| ward\_POP | Population for each ward ordered by ward id. |
| ward\_colName | Name of the column that describes the ward id in ‘HH attribute’ file. |
| HH\_size\_colName | Name of the column that describes the HH size in ‘HH attribute’ file. |
| HH\_attribute\_csv.path | Path to the ‘HH attribute’ file. |
| HH\_attribute.numeric | Names of numeric columns in ‘HH attribute’ file to be used in creating bins. |
| HH\_attribute.toBeRounded | Indicates whether the value of each column specified in property ‘HH\_attribute.numeric’ needs to be rounded to an integer. |
| HH\_attribute.nominal | Name of nominal column in the ‘HH attribute’ file to be used in creating bins. |
| HH\_attribute.  separateBinningBaseline | Baseline number of samples for creating bins: creating separate bins for wards that have number of HHs larger than baseline, otherwise creating bins using the entire sample |
| HH\_attribute.numOfBreaks | Number of Jenks breaks in creating bins. |
| executor.number\_of\_threads | Number of threads that run NetLogo instances in parallel. |
| executor.num\_HH\_allocation\_repetition | Number of repetitions of HHs’ allocation to create different ‘worlds’. |
| model.parameter.  num\_seed\_HH | List of different seed HH numbers used for different runs. |
| model.parameter.  required\_adult\_perHa | List of different ‘required adult equivalent’ values used for different runs. |
| model.parameter.alpha | List of different ‘alpha’ values used for different runs. |
| model.parameter.  search\_scope | List of different ‘search scope’ (100m per unit) values used for different runs. |

# Running Simulations

# Input Data

|  |  |
| --- | --- |
| Input File | Explanation |
| ‘ward\_raster.txt’ | 100m\*100m raster of Monze District, each cell is assigned with a ward ID; |
| ‘landcover\_raster.txt’ | Landcover 100m\*100m raster of Monze District; [default cell values: -9999 for NULL value; 3 for farm land] |
| ‘distance\_raster.txt’ | 100m\*100m raster of Monze District, the value in each cell represents its distance to the nearest road. |
| ‘summary.txt’ | (X-Y) dimensions (in 100m) for each ward; |
| ‘HH.csv’ | Sample HHs’ attribute file; [default attributes: household adult equivalent; farm land area in hectare; maize type] |

You can specify the path to each input file in ‘simulator.properties.txt’ as described in Section 4.

# Start the Simulation

The Zambia simulation has two separated phases:

1. Data preparation:
   1. The input raster is split into small pieces by ward ID;
   2. The number of HHs for each ward is calculated by dividing the total ward population with the average adult equivalent per HH;
   3. Creating HHs for each ward based on the 5\*5\*5 bins built from HH samples;

You can see an overview of the model at:

[Pseduo Code](https://drive.google.com/open?id=0B7cw0KEWujzdTXFjRnRsVmYzSlk&authuser=0)

To run the first phase, use Linux script ‘prepareData.sh’:

|  |
| --- |
| ./prepareData.sh |

Note that, if the scripts couldn’t be run and report error message like:

|  |
| --- |
| -bash: ./createWards.sh: Permission denied |

This is likely because the script does not have the execute permission. To fix it, you can use command (replace ‘scriptname.sh’ with the name of script file, for example ‘createWards.sh’):

|  |
| --- |
| chmod +x scriptname.sh |