8. BUILDING THE SIMULATION SYSTEM

8.1 Introduction

The core of the HomininSpace modelling and simulation system is an agent-based model. To turn that into a running simulation the model has to be programmed and executed in a computer simulation environment. Many such environments exist, and Repast Simphony has been selected for implementation of the model (Section 8.2). A development environment generally consists of more than just a compiler for the source code in which the model is implemented, and can include additional third party tools as well (for an overview see Section 8.3). The model is then executed within the structure of the HomininSpace simulation system (Section 8.4), using the input files that define the case study with topography, climate data, information about the archaeology, and the model (Section 8.5 and 8.6) and creating output data during the simulations (Section 8.7).

Parts of this chapter have been extracted from the "HomininSpace – user manual" document that is included in the Supplementary Materials which includes further details, formal descriptions and data file examples. HomininSpace, including source code, data files, and examples was accepted into the OpenABM model repository (Janssen *et al.* 2008). After an anonymous peer review it was included in the Computational Model Library of the Computational Modeling in Social and Ecological Sciences Network (CoMSES Net) under model number 5294[31].

8.2 Agent-based modelling tools

Many agent-based modelling toolkits for building simulations are available (e.g., Abar *et al.* 2017; Gilbert and Bankes 2002; Nikolai and Madey 2009; North *et al.* 2006; North *et al.* 2013; Railsback *et al.* 2006; Tobias and Hofmann 2004). Nikolai and Madey (2009) alone review 53 toolkits and developing environments. Out of the existing tool sets REcursive Porous Agent Simulation Toolkit (Repast Simphony or in short Repast) was selected for implementing HomininSpace (Crooks 2007; Macal and North 2009). Repast is a turn based simulation system for developing and executing agent-based models (Tatara *et al.* 2006). It is freely available as open source software (North *et al.* 2013).

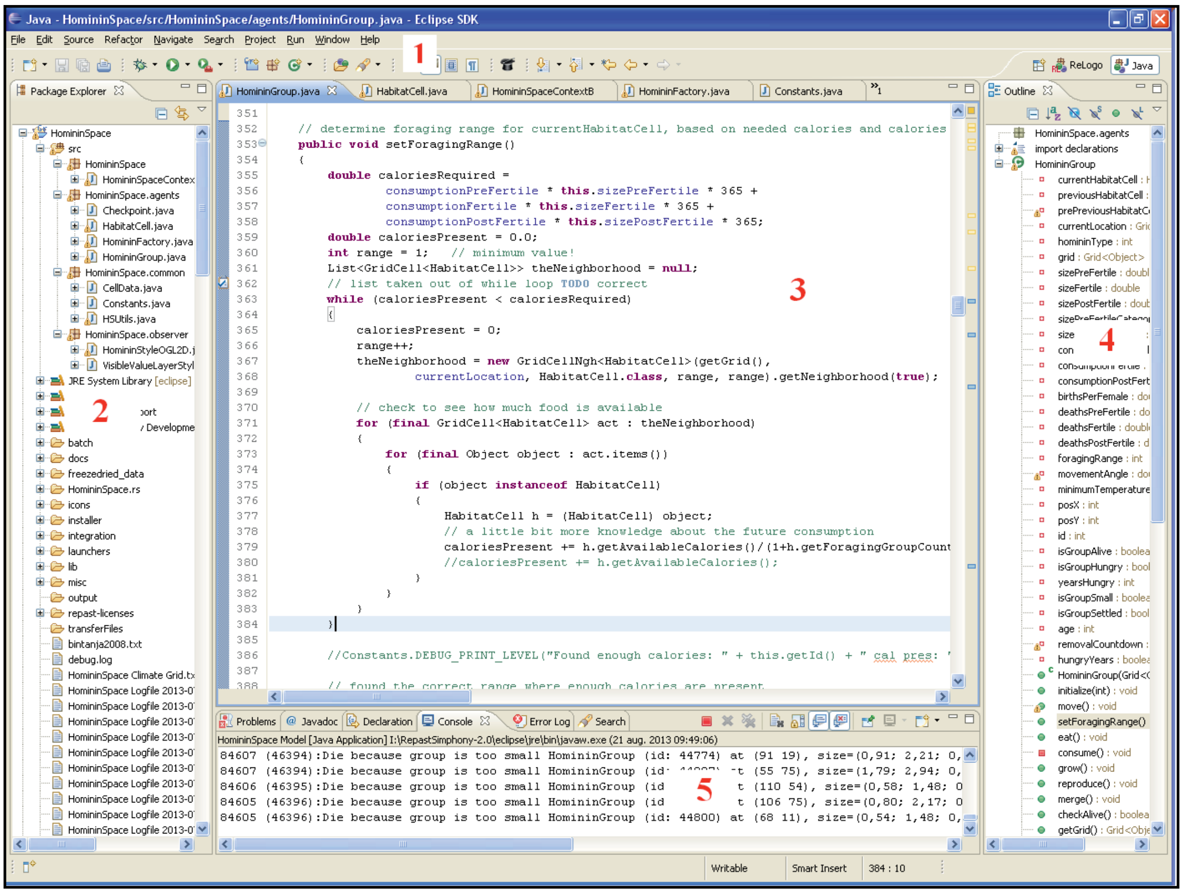
[31] https://www.comses.net/codebases/5294/releases/1.0.0/, accessed 8 March 2018·

Repast has been selected for the Google Summer of Code 2011 project, ensuring publicity, additional example programs and a larger user base (see http://www.cscs.umich.edu/~rlr/SoC/index.php?n=Main.SoC2011, accessed 28 April 2011, or http://www.google-melange.com/gsoc/homepage/google/gsoc2011, accessed 22 March 2013). Repast has for instance been used to analyse urban segregation and urban growth (Mathur 2007), to model crime and identify burglary risks (Malleson 2006), to model pedestrians and evacuation routes (Castle and Crooks 2006; Usman *et al.* 2017), to model the spread of diseases (Pais *et al.* 2017), to build FamilyNet2 (White 2013) which is part of the development of the ForagerNet2 model, a complex model that encompasses representations of space, social learning, and social networks (White 2012), the modelling of noise in the genes of growing bacteria populations (Stiegelmeyer and Giddings 2013), and for large scale emergency response simulations (Hawe *et al.* 2012).

Repast was selected to implement HomininSpace because it was positively reviewed, has an active user base, has good quality documentation, there are many example programs, it is described and used in the literature, and has built-in support for many standard libraries. The model can be programmed in Java, a multi-purpose and platform independent programming language and runtime environment that this author is proficient in. In addition, the predecessor of HomininSpace, the agent based modelling system SteppingIn (Scherjon 2012) was successfully implemented by this author using a prior version of Repast.

8.2.1 Repast Simphony version 2.2

Repast Simphony consists of two major components: a development and a runtime or execution environment. The runtime part will execute model runs and visualize the simulation results. The development module is used to maintain the source code of the simulation and to manage the program structure. Figure 31 identifies the main elements of the user interface. The Repast environment includes point-and-click model configuration and operation (execution, enquiries), integrated two or three dimensional GIS and other model views, connection to standard data sources including database support, and automated connections to external programs for statistical analysis and model visualisations (Tatara *et al.* 2006).



```
    // determine foraging range for currentHabitatCell, based on needed calories and calories
    public void setForagingRange()
    {
        double caloriesRequired =
                consumptionPreFertile * this.sizePreFertile * 365 +
                consumptionFertile * this.sizeFertile * 365 +
                consumptionPostFertile * this.sizePostFertile * 365;
        double caloriesPresent = 0.0;
        int range = 1;   // minimum value!
        List<GridCell<HabitatCell>> theNeighborhood = null;
        // list taken out of while loop TODO correct
        while (caloriesPresent < caloriesRequired)
        {
            caloriesPresent = 0;
            range++;
            theNeighborhood = new GridCellNgh<HabitatCell>(getGrid(),
                    currentLocation, HabitatCell.class, range, range).getNeighborhood(true);

            // check to see how much food is available
            for (final GridCell<HabitatCell> act : theNeighborhood)
            {
                for (final Object object : act.items())
                {
                    if (object instanceof HabitatCell)
                    {
                        HabitatCell h = (HabitatCell) object;
                        // a little bit more knowledge about the future consumption
                        caloriesPresent += h.getAvailableCalories()/(1+h.getForagingGroupCount
                        //caloriesPresent += h.getAvailableCalories();
                    }
                }
            }
        }

        //Constants.DEBUG_PRINT_LEVEL("Found enough calories: " + this.getId() + " cal pres: "

        // found the correct range where enough calories are present
```

```
HomininSpace Model [Java Application] I:\RepastSimphony-2.0\eclipse\jre\bin\javaw.exe (21 aug. 2013 09:49:06)
84607 (46394):Die because group is too small HomininGroup (id: 44774) at (91 19), size=(0,91; 2,21; 0,
84607 (46394):Die because group is too small HomininGroup (id[illegible]t (55 75), size=(1,79; 2,94; 0,
84606 (46395):Die because group is too small HomininGroup (id[illegible]t (110 54), size=(0,58; 1,48; 0
84605 (46396):Die because group is too small HomininGroup (id[illegible]t (106 75), size=(0,80; 2,17; 0
84605 (46396):Die because group is too small HomininGroup (id: 44800) at (68 11), size=(0,54; 1,48; 0,
```

Figure 31: The Repast Simphony development environment. 1 = main menu; 2 = project directory structure; 3 = source editor; 4 = file structure; 5 = log information window.

Software development in Repast Simphony can be done in the Object Oriented Programming (OOP) language Java (Gosling *et al.* 2005). The advantage of OOP in agent based modelling is that objects or classes in the programming language naturally translate into agents in the model. Within OOP, methods and attributes (data) for one object are kept together, in one entity (the object as instantiation of its class). For agents to be autonomous and individual, one needs the individual characteristics and processes together in one single instance. Although other types of development environments can and have been used, agents and OOP fit together intuitively, even as new initiatives to build more socially tailored development environments are under way (Borrill and Tesfatsion 2010, 5).

The Java Runtime Environment (JRE) which executes the java code contains next to libraries and core classes also the Java Virtual Machine (JVM). Java classes are compiled into bytecode that runs in any JVM. There are JVMs for almost any environment, from computers to telephones, making the sales pitch ‘write once, run everywhere’ literally true (Gosling and McGilton 1996). This allows upscaling of the computing environment up to certain super computer configurations if such computing power is needed. It currently is the most popular programming language (TIOBE Index for November 2017, https://www.tiobe.com/tiobe-index//, retrieved 13 November 2017), and as such a good choice for any software development effort.

Repast itself has predefined classes for building agent-based simulations and to access the simulation infrastructure during runtime. The central object in a Repast simulation is the `Context` class, which stores agents and maintains the relationships between agents (called `projections`). The `ContextBuilder` stores the information on the agents in the Context. RepastS uses configuration files (“model.score” and “scenario.xml”) to specify the roles of the classes in a simulation model, to identify the `Contextbuilder` class and to define the default context (Collier and North 2010). Repast uses the `geotools` library to display geographical information (www.geotools.org, accessed march 2011). This open source java library is used to manipulate geospatial data and to display the `Projections` objects and thus the agents with a geospatial location (Railsback *et al.* 2006).

8.3 The development environment

In the Repast Java development environment the modeller writes source code that creates model pieces (agents, grid) in the form of Java objects which are passed to the runtime system using a declarative configuration (North *et al.* 2005). The runtime system must be told how to instantiate and connect the model components. Models are managed based on interactive user input (graphical interface) and declarative and imperative requests from the components themselves.

8.3.1 The source code

When programming in an object oriented language like Java all entities are objects, with associated data and methods described by their class definition. In repast all model components are plain Java objects. There are several classes included in the HomininSpace source code. The context creator class which constructs the main context, implements the main loop and returns it to the Repast run environment is the `HomininSpaceContextBuilder` class. Part of the source code for this class is shown as an illustrative example in Figure 32. The required agents and projections (grid) are added to this context.

```
/**
 * Custom @link ContextBuilder implementation for the HomininSpace
modelling environment.
 *
 */
public class HomininSpaceContextBuilder extends
DefaultContext<Object> implements   ContextBuilder<Object> {

      /*
       * Constructor, build simulation environment
       */
      @Override
      public Context<Object> build(final Context<Object> aContext)
      {
            // schedule start and end of simulation
            ISchedule schedule =
                  RunEnvironment.getInstance().getCurrentSchedule();
            // schedule last routines to be executed
            ScheduleParameters endParams =
                  ScheduleParameters.createAtEnd(ScheduleParameters.
            LAST_PRIORITY);
            schedule.schedule(endParams, this, "end");
            // schedule start
            ScheduleParameters startParams =
                  ScheduleParameters.createOneTime(1);
            schedule.schedule(startParams, this, "start");
            // schedule end
            RunEnvironment.getInstance().endAt(tickCount);

            // schedule main loop
            ScheduleParameters loopParams =
                  ScheduleParameters.createRepeating(1, 1,
                  ScheduleParameters.LAST_PRIORITY);
            schedule.schedule(loopParams, this, "activateAgents");

            return theContext;
      }
}
```

Figure 32: Source code excerpt from the “HomininSpaceContextBuilder.java” class.

The grid based environment in which the other agents reside is a collection of objects accessible through the `Grid` object. The grid matrix is constructed from `HabitatCell` objects, each containing a `CellData` object. The interacting agents in the modelling system are the `HomininGroup`, `HomininFactory` and `Checkpoint` classes. There are several support classes, data definitions, and interface specifications, including the sources `HSUtils` and `Constants` (see Figure 33 which lists all the Java source files included in this project). Verbatim code and interface specifications are included in the Supplementary Materials.

```
/HomininSpace/src
   HomininSpace
      /HomininSpace/src/HomininSpace/HomininSpaceContextBuilder.java
   HomininSpace.agents
      /HomininSpace/src/HomininSpace/agents/Checkpoint.java
      /HomininSpace/src/HomininSpace/agents/HabitatCell.java
      /HomininSpace/src/HomininSpace/agents/HomininFactory.java
      /HomininSpace/src/HomininSpace/agents/HomininGroup.java
   HomininSpace.common
      /HomininSpace/src/HomininSpace/common/CellData.java
      /HomininSpace/src/HomininSpace/common/Constants.java
      /HomininSpace/src/HomininSpace/common/HSUtils.java
   HomininSpace.observer
      /HomininSpace/src/HomininSpace/observer/CheckpointStyleOGL2D.java
      /HomininSpace/src/HomininSpace/observer/HomininStyleOGL2D.java
      /HomininSpace/src/HomininSpace/observer/VisibleValueLayerStyleOGL.java
```

Figure 33: The Java source files in the HomininSpace development environment.

8.3.2 Software development tools

For the development of HomininSpace and execution of the model and its components several tools have been used, and some of these are mentioned throughout this thesis. These tools are listed below, with in bold the abbreviation that is used to reference that tool where appropriate:

- REcursive Porous Agent Simulation Toolkit (**Repast Simphony** or in short **Repast**) (Repast version 2.2, https://repast.github.io/download.html, downloaded 15 September 2014). This is the main simulation development environment. HomininSpace has been programmed in the Java computer coding language environment.
- The **Yourkit** Java profiler tool to identify bottle necks in execution time (https://www.yourkit.com/java/profiler/, verified 28 November 2017).
    - Identified several hotspots where much execution time is spend, with the implemented solution:
        - For `Factories`, no longer calculate all available resources, just search one valid spot for the new group;
        - For `SetForaging` range, limit the calculations to ranges < `maxForagingRange` (was endless!) Added `maxForagingRange` to the local variables for groups;
        - Some debug statements (that are not printed but do require string operations) are commented out.
        - For performance reasons groups that become extinct are removed from the simulation but are not physically removed from memory but stored in a separate list, ready to be re-initialized when needed. In some simulations more than 60,000 groups are simulated, activating and

deactivating these from a list consumes considerably less memory and processing power than instantiating and destroying them as objects.

- Some scripts that are used in HomininSpace have been written in **R**, a system for statistical computation and graphics (R Core Team 2017). Scripts were developed in the R Studio environment (RStudio 2017). Downloaded from (https://www.rstudio.com/products/rstudio/download/). R can be found at https://cran.rstudio.com/, both addresses verified 28 November 2017.
- For exploratory statistical analysis the Statistical Package for the Social Sciences (**SPSS,** today part of IBM Business Analytics (IBM Corp. 2015)) was used. https://www.ibm.com/analytics/nl/nl/technology/spss/, verified 28 November 2017; However, all numbered statistical output was generated with R scripts, to allow proper replication of these results (Marwick 2017).
- Converter for degrees to decimal http://andrew.hedges.name/experiments/convert_lat_long/ (possibility to indicate W/E), accessed 2 August 2016. Via Excel sheet: http://support.microsoft.com/kb/213449, accessed 28 May 2013.
- Spatial Analysis in Macroecology, version 4.0 (**SAM**) (http://www.ecoevol.ufg.br/sam/, accessed 14 February 2012 (Rangel *et al.* 2006; Rangel *et al.* 2010) is used to select grid cells and to distribute GIS data sets (for instance the WorldClim datasets) over these grid cells.

8.4 The runtime environment

Repast offers simulation execution and facilities for agent storage, display, interaction, data analysis and data presentation (North *et al.* 2005). Common tasks like scheduling, object activation, simulation termination, data acquisition and storage, object communication, etc. are supplied by the Repast system. There is a strict separation between model, data storage and visualisation.

Simulations can be started in the Repast runtime user interface (Figure 34). A user interface is no more than a representation of internal data structures, so it is also possible to execute simulations without user interface. Running simulations with the user interface requires installation of the Repast system on the target computer. For the execution of batches the runtime system is prepackaged in an archive file. Practical details including a list of Repast system files are given in the HomininSpace User Manual, included in the Supplementary Materials.

At the start of a HomininSpace simulation the system is initialized by reading the configuration and data files, including grid data, climate data for modern day and the past, and archaeological data with the checkpoints. The simulation grid data structure is created and filled with the environment information. Factories if desired are created to implement core areas. Data for the reconstructed global sea level and the yearly mean temperatures for the simulation period are read from file databases and stored in memory.

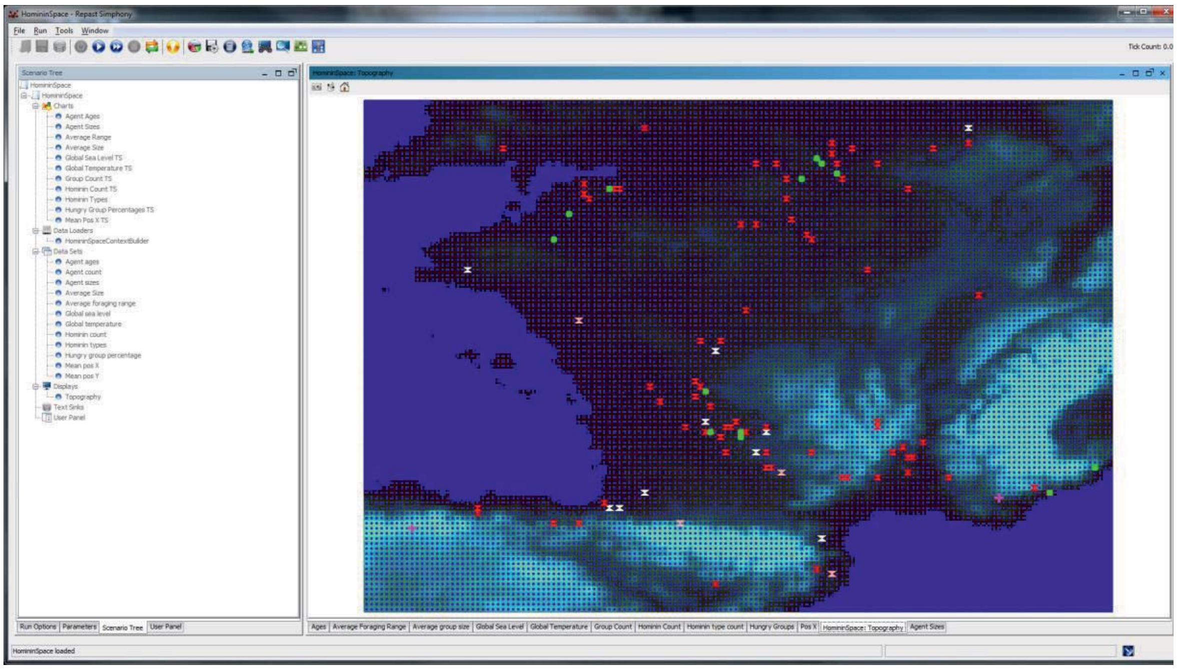
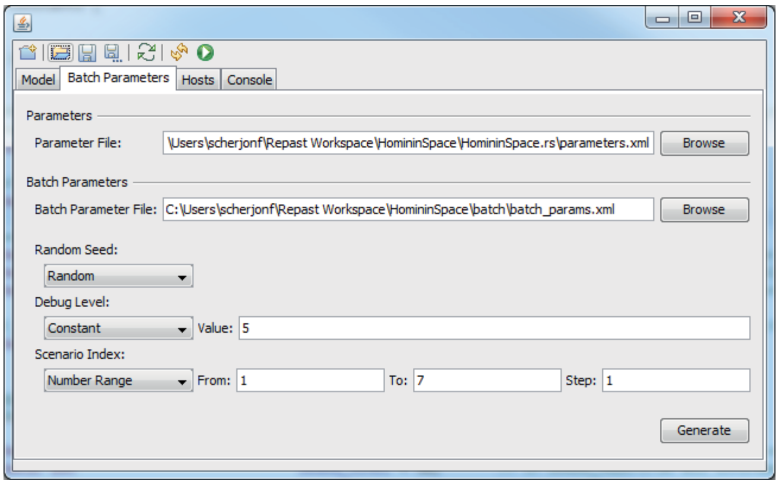


Figure 34: User interface of HomininSpace after initialization. On the right the Topography layer for the simulation area, on the left the scenario tree.

8.4.1 Batch execution of simulations

In batch mode simulations are executed without user interface. Multiple simulations can be run in parallel and all results are collected and combined after a batch is completed. An interactive user interface is not needed and not present with batch execution. It is possible to set model parameter values for each run. For batch processing the batch variant of the Repast launch scripts must be started. A dialogue window will ask for a file with batch parameters for each simulation in the batch (Figure 35). The user must also specify the number of processors of the target computer (this defines the number of parallel simulation executions).



Parameters

Parameter File: \Users\scherjonf\Repast Workspace\HomininSpace\HomininSpace.rs\parameters.xml

Batch Parameters

Batch Parameter File: C:\Users\scherjonf\Repast Workspace\HomininSpace\batch\batch_params.xml

Random Seed: Random

Debug Level: Constant Value: 5

Scenario Index: Number Range From: 1 To: 7 Step: 1

Figure 35: Values for the batch parameters provided in the Batch Parameters tab.

To facilitate the execution of large numbers of simulations it is possible to start batch simulations outside of the development environment. To prepare this the complete model has to be available in a single archive batch file named "complete_model.jar". This file is created by executing the batch Model command in the development environment. Configuration of the batch can be provided as required. The following steps describe standalone batch execution in HomininSpace:

1. Create a directory for execution: copy the created "complete_model.jar" file into this directory and create the files "local_batch_run.properties" and "batch_unrolled_params.txt";
2. Start execution with: "`java -cp "./lib/*" repast.simphony.batch.LocalDriver local_batch_run.properties`" which reads the appropriate parameter values from the file "HomininSpace Parameters.txt" and the settings for this simulation from "HomininSpace Settings.txt";
3. Collect results via: "`java -Xmx512m -cp "lib/*" repast.simphony.batch.ClusterOutputCombiner . combined_data`";
4. Combine results with previous batch executions: merge data from each "combined_data\Batchresults.txt";
5. Ensure that the input file is adjusted: remove those simulations that were executed from the parameter values file "HomininSpace Parameters.txt" and return to step 2 for further batch processing. If the Genetic Algorithm is used it will update the parameter values file with new models and then return to step 2.

The content of an example parameter value file is presented verbatim below. The first line contains the headers for each column, separated by commas. Each subsequent line presents a model with associated parameter values, starting with the unique simulation number (see also Table 21).

```
ScenarioNumber,RandomSeed,BirthRate,DeathRate_PreFertileCohort,DeathRate_FertileCoho
rt,DeathRate_PostFertileCohort,Subsistence_PreFertileCohort,Subsistence_FertileCohor
t,Subsistence_PostFertileCohort,Years_Before_Group_Maturity,GroupSize_BeforeMerge,Gr
oupSizeFertile_BeforeMerge,GroupSize_BeforeSplit,Temperature_Tolerance,CohortSize_Pr
eFertile,CohortSize_Fertile,Calories_Per_Kg_Meat,Max_ForagingRange,Parent_Scenario_1
,Parent_Scenario_2,ViabilityIndex
2306,7355,50,3,1,2,5000,2500,3250,7,4,0,26,-21,9,38,3350,10,1774,2039,92976
2307,6971,43,3,1,3,3375,2200,3000,1,4,4,16,-29,8,38,3150,10,1993,2266,45951
2308,5432,43,1,1,3,2500,2750,5000,1,4,0,16,-28,8,40,3465,10,2220,2255,92541
2309,5432,43,3,1,4,3750,2500,4250,1,4,1,16,-29,9,38,3150,13,2209,2193,31274
2310,8477,43,3,1,12,3575,4750,3250,1,5,2,16,-29,9,40,3150,14,2143,0,32354
2311,8477,43,2,1,2,2750,4750,4250,1,6,1,16,-29,8,38,3150,10,2259,0,63218
2312,7355,48,1,1,12,3250,2250,3250,1,4,1,16,-32,8,40,2900,13,2208,0,201902
```

8.5 Data files defining a case study

The HomininSpace system is designed as a general hominin simulation model, with explicitly no hominin species specific elements in the source code or the model structure. For instance, the birth-rate which could be different between species is a model parameter, not a fixed value. The simulated environment and time frame are all taken from input files, as well as the archaeology that simulation results are compared against. That means that any hominin species from any region from any time frame can be analysed with this tool by simply providing the appropriate input files.

During initialisation of a simulation the input files for the model elements are read from several input files. These files are located in the model directory of the HomininSpace system which is defined in the `Constants` object in the source code. HomininSpace has been designed with flexibility in mind and all specifics for a certain case study are read from file. Replacing input files creates a different case study. Files are provided for the topography and climate data, for past sea levels and global temperatures, and with archaeological data in the form of checkpoints.

8.5.1 Simulation grid, topography and climate data

The simulation area is divided into grid cells with the Spatial Analysis in Macroecology tool (SAM). SAM is used to distribute topographical and climate parameters onto these grid cells. Next to the topographical data (elevation) these parameters include 19 bio-values from http://www.worldclim.org (both for present and past data sets) and the bed-rock values from the ETOPO1 dataset. The values for the global mean temperature in the past are used as an index for a linear interpolation between the extremes of modern day climate and LGM climate data. After processing by SAM a selected group of variables are stored in the climate data file called “HomininSpace Climate Grid.txt”. See Appendix 1 for a complete list of all variables. It contains one line for comments and 14948 data lines. Each line contains the bioblimatic values for one grid cell. The structure of the file is described in Table 15. The file is read by the following routine in “HSUtils.java”:

```
public static List<CellData> readClimateDataFile(final String
cellDataFileName)
```

The file is read into the HomininSpace model at startup. Interpolation of climate data for submerged land is done by HomininSpace. On reading the data a list of `CellData` objects is returned, each containg the relevant climate data for one grid cell.

Table 15: Structure of the climate data file after preprocessing by SAM.

Field	***Description***
1	Longitude (X Centroid)
2	Latitude (Y Centroid)
3	n_alt, the number of altitude points in this cell
4	Mean_alt, the mean altitude (current day)
5	Mean_bio1, annual mean temperature (current day)
6	Mean_bio5, max temperature of the warmest month (current day)
7	Mean_bio6, min temperature of the coldest month (current day)
8	Mean_bio12, annual precipitation (current day)
9	Mean_bio13, precipitation of the wettest month (current day)
10	Mean_bio14, precipitation of the driest month (current day)
11	Mean_wc_2_5m_CCSM_21k_bio_14, CCSM value for bio14 (LGM)
12	Mean_wc_2_5m_CCSM_21k_bio_1, CCSM value for bio1 (LGM)
13	Mean_wc_2_5m_CCSM_21k_bio_5, CCSM value for bio5 (LGM)
14	Mean_wc_2_5m_CCSM_21k_bio_6, CCSM value for bio6 (LGM)
15	Mean_wc_2_5m_CCSM_21k_bio_12, CCSM value for bio12 (LGM)
16	Mean_wc_2_5m_CCSM_21k_bio_13, CCSM value for bio13 (LGM)
17	Max_etopo1_bedrock, the maximum bedrock value (minimum depth)

8.5.2 Reconstructed global mean temperature and sea level

Reconstructed temperatures and sea levels for the simulation period are read by the HomininSpace system from the input file “Bintanja2008.txt”, taken verbatim from Bintanja and van de Wal (2008). A description of the file is given in Table 16. From this file only the data for fields one (timestamp), five (surface temperature) and nine (global sea level) are used. The routine that reads the file is provided by the "HSUtils.java" class and has the following signature:

```
public static void readBintanjeDataFile(final String cellDataFileName)
```

Table 16: Structure of input file "Bintanja2008.txt". Timestamp (1), reconstructed temperatures (5) and sea levels (9) are used by HomininSpace.

Field	*Description*
1	Time (years ago)
2	Modeled marine oxygen isotope value, relative to present (o/oo)
3	Ice sheet contribution to the marine isotope signal, relative to present (o/oo)
4	Deep-ocean temperature contribution to the marine isotope signal, relative to present (o/oo)
5	Atmospheric surface air temperature relative to present (degC)
6	Deep-ocean temperature relative to present (degC)
7	Eurasian ice volume relative to present (m sea level equivalent)
8	North American ice volume relative to present (m sea level equivalent)
9	Global sea level relative to present (m)

On reading the data three private array variables in the `HSUtils` class are constructed: `timeStamps, seaLevels,` and `temperatures`. Values from these variables are available for each tick (year) through the routines `getSeaLevel(tick)` and `getTemperature(tick)`. When there is no value for a given year, the closest value is returned. The values of the Neanderthal case study are depicted in Figure 36 (global mean temperatures) and Figure 37 (global sea level).

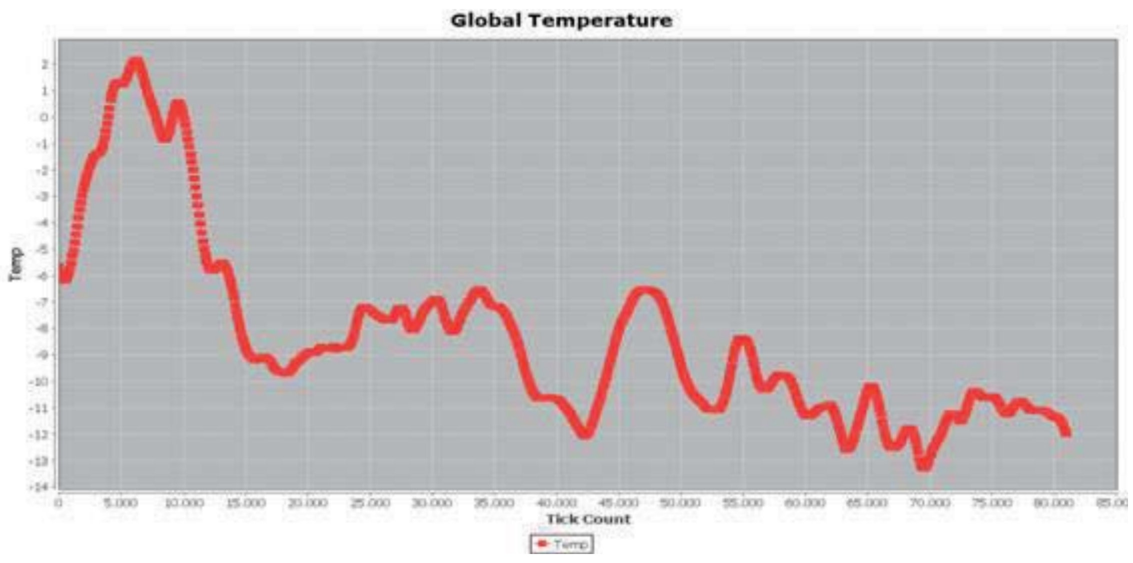


Figure 36: Reconstructed global mean temperature relative to modern day temperatures. Tick 0 is the start of the simulations (131 ka).

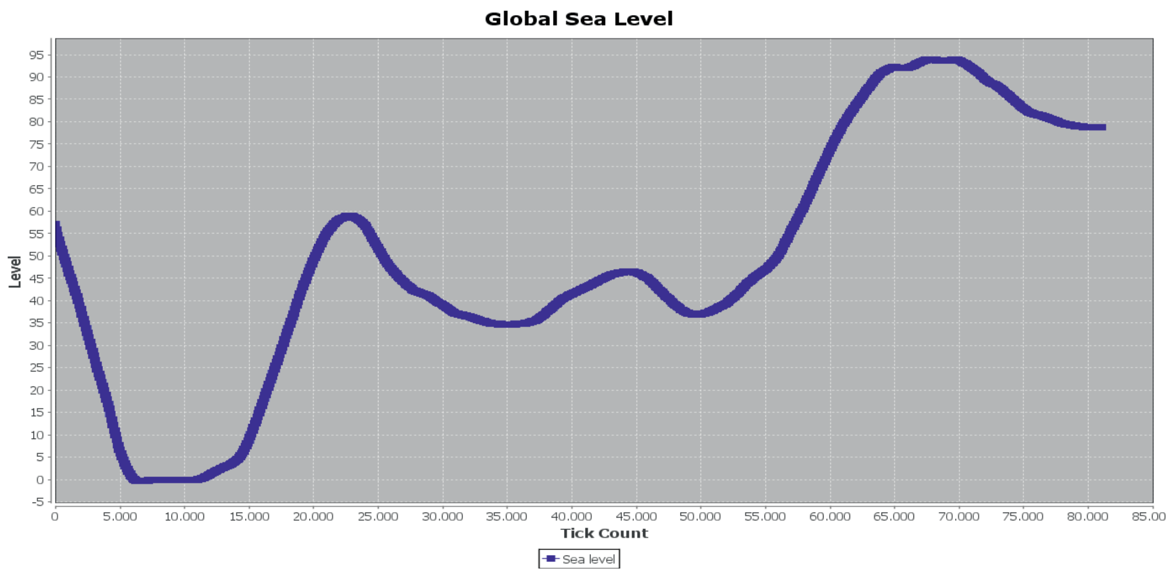


Figure 37: Global sea level relative to the modern day sea levels Values in meters *below* modern day level. Tick 0 is the starting point of the simulations (131 ka) with a global sea level of almost 60 meters below the level of today (0).

8.5.3 Checkpoints

Checkpoints are objects located in a grid cell with a specific task. There are four different types of checkpoints in a simulation: regular Checkpoints in Space and Time (CSTs), initial starting locations for agents, monitoring checkpoints, and climate monitoring checkpoints. Checkpoints all have a geographical location and are stored as checkpoints with a specific type in one single input file named "HomininSpace Checkpoint List.txt". The routine that reads this file is provided in the "HSUtils.java" source file and has the following signature:

```
public static List<Checkpoint> readCheckpointDataFile(final String fileName)
```

After successfully reading the file a list with `Checkpoints` is returned. More than one checkpoint can be in the same location (each grid cell is roughly ten square kilometers, and some archaeological sites are located very close together).

The checkpoint file starts with a line that can contain comments, followed by any number of checkpoints, one per line. The four checkpoint types are indicated by the value of the first field in each line. A value of one (1) indicates a regular checkpoint, a value of two (2) is a monitoring checkpoint, three (3) is a starting location for local groups that are present when the simulation starts, and four (4) indicates a climate monitoring checkpoint. For regular checkpoints all fields described in Table 17 must have a value, for the others only latitude and longitude needs to be provided. If the checkpoint location falls outside the grid, the checkpoint is ignored. Fields are separated by tabs.

For the regular CSTs each line is one interval. Intervals for the same locality (site) are stored in one CST. I have assigned a confidence level indicating the estimated reliability of the date provided. It is used in calculating scores for the visits. Preliminary analysis suggests that using only reliable dates seems to make no difference in matching archaeology when compared to using all the dates.

Table 17: Layout of the checkpoint input file. Fields are separated by tabs.

Field	*Description*	*Value type*	*Description / remarks*
1	Type checkpoint	Integer value	1 = regular checkpoint 2 = monitoring checkpoint 3 = starting location 4 = climate monitoring checkpoint
2	Locality (site name)	Character string	Any characters (but no tabs)
3	Latitude	Floating point value	Example (Artenac): 45.85, 0.3333.
4	Longitude	Floating point value	
5	Age (ka)	Floating point value	Example: Artenac 110 +9/-7
6	One sigma plus (ka)	Integer value	
7	One sigma minus (ka)	Integer value	
8	Confidence level	Integer value	1 = unreliable date 2 = modest confidence 3 = high confidence in measurement
9	Type interval	Character string	Can be: absence/presence for indicating an absence or presence interval.
10	Layer name	Character string	Name of the archaeological layer with which the date is associated.

In total, the default input file contains 4 climate monitoring points, 13 monitoring points, 13 starting points, and 83 checkpoints with 470 dated intervals. When two or more intervals refer to the same archaeological layer they are joined together to form a new interval with as starting point in time the minimum of all starting points, and as the end the maximum of all ending points. After reading the checkpoints and joining any overlapping intervals a total of 198 intervals are included in the simulations. For easy reference a list of all checkpoints and joined intervals is included in the Supplementary Materials.

Initial population starting points

When a simulation is started from the Last Interglacial onwards (all simulated scenarios), a number of hominin groups is assumed to be already present in the simulation area. This is the initial population. This is modelled as groups of hominins located at known penultimate glaciation archaeological presence sites (Bocquet-Appel and Tuffreau 2009)[32]. For each group in the initial population a location is provided in the checkpoint input file as a starting location. During initialization of the simulation at each of these starting points

[32] From the sites by Bocquet-Appel and Tuffreau (2009) three are not in France: Pietraszyn, Ripiceni Izvor and Külna.

a group of 25 individuals is created[33]. An overview for these points is given in Table 18. For each site the decimal version of the location is given. An geographical view of where these starting points are is given in Figure 38.

Table 18: Default starting locations for groups of the initial population.

#	*Name*	*Latitude (dec)*	*Longitude (dec)*
1	**Abri Suard (La Chaise)**	45.66667	0.46667
2	**Bapaume les Oisiers**	50.14600	2.83700
3	**Barbas I**	44.84990	0.56670
4	**Beaumetz-les-Loges**	50.24238	2.65206
5	**Combe-Grenal**	44.80840	1.22300
6*	**Mont-de-l' Evangile**	49.85463	2.42174
7	**Le Lazaret**	43.69028	7.29500
8*	**Longavesnes**	49.97088	3.05989
9	**Piégu**	48.60000	-2.55000
10[1]	**Port Pignot**	49.68330	-1.45000
11	**Vaufrey**	44.79997	1.20000
12[2]	**El Colombo**	44.14400	8.19090
13[3]	**La Cotte-de-St-Brelade**	49.18500	-2.20200

(*) = not in the RPDE database, location by approximation
(1) = This is also known as Fermanville-Port-Pignot
(2) = Site is called Colombo Cave
(3) = This site is located on the island of Jersey

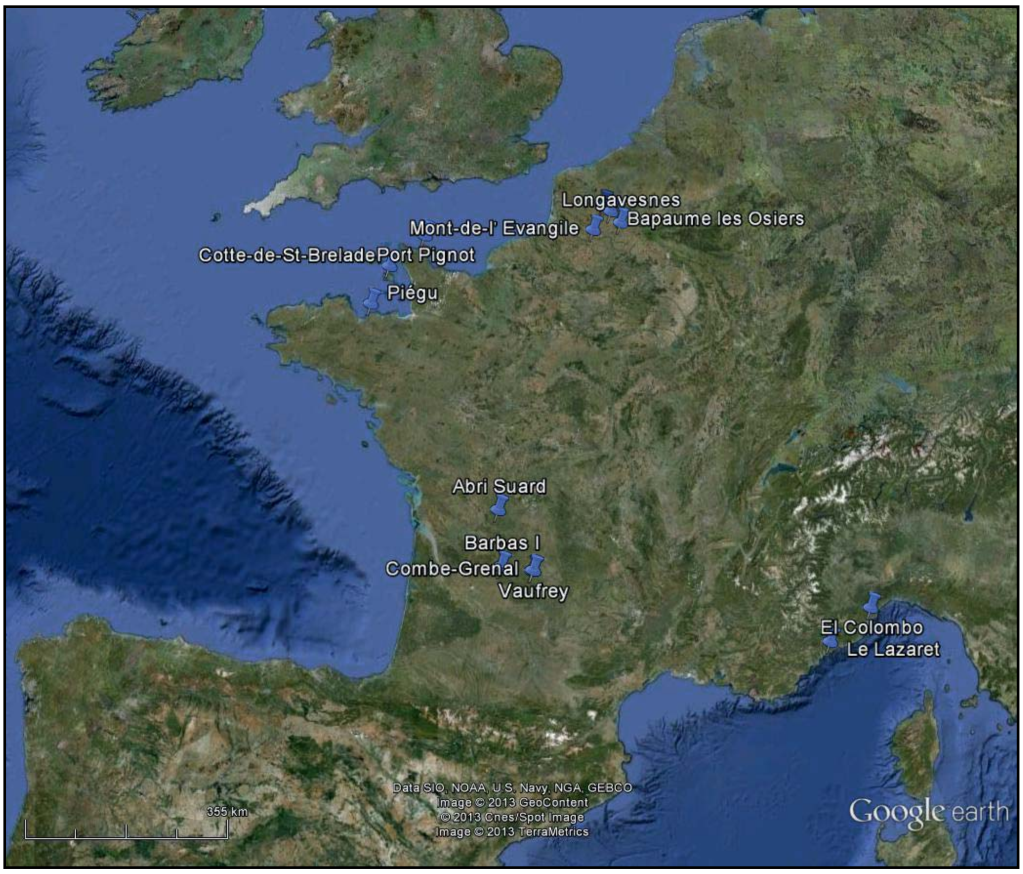


Figure 38: Google Earth view of the starting locations for the initial population.

[33] Note that at a starting point no checkpoint is created, only a group of hominins.

Monitoring checkpoints

Monitoring checkpoints have no presence interval list. These checkpoints are included to record hominin presence during the simulation and can be used for verification and validation purposes. Included are those sites with known hominin presence but where dates fall outside the simulation period, well known hominin sites without exact dates, locations that can be used to detect population expansion, etc. (). These sites include Maastricht-Belvédère, Kent's Cavern, and more.

Table 19: Monitoring checkpoints located at example positions.

#	*Name*	*Latitude (dec)*	*Longitude (dec)*
1	**Les Rochers de Villeneuve**	46.41528	0.74083
2	**Les Bosses**	44.46535	1.50190
3	**Grotte des Fieux**	44.85000	1.70000
4	**Petit Bost**	45.08130	0.46083
5	**La Micoque**	44.93300	1.01667
6	**Maastricht-Belvédère**	50.85000	5.68330
7	**Boxgrove**	50.86901	-0.69028
8	**Kent's Cavern**	50.46987	-3.53000
9	**Grotte de Saint-Marcel**	44.33080	4.54200
10	**La Grotte du Pape**	43.63000	-0.70000
11	**El Castillo**	43.28960	-3.96499
12	**Isturitz**	43.36670	-1.19611
13	**Abri Olha**	43.36206	-1.38676
14	**Montou-la-Joliette**	42.78328	2.83330
15	**Kervouster**	48.05000	-4.23330

Many known Middle and Late Pleistocene sites have not been included in this list. Included monitoring checkpoints are just to illustrate the use of such sites and for system tests. If any area is to be monitored in greater detail, more checkpoints surrounding that area can be included.

Climate recording checkpoints

These points register every time step for the grid cell they are located in the two most important climate parameters: precipitation and temperature. They also store the type of reconstructed habitat, the type of topography (which can change due to fluctuating sea levels) and the amount of available energy through time. Climate recording checkpoints are provided for verification and validation purposes[34]. They have been used for instance to create a graph of local habitat type through time for comparison with reconstructed past

[34] Note that for performance reasons these checkpoints are not installed by default. If the user wants to use this functionality it can be activated by setting in the source code the boolean `USE_CLIMATE_RECORDING_CST` to true.

climate graphs from other sources. This can also be used to verify that the grid cell does become a water cell when the sea level rises. These checkpoints do not store absence and presence data for hominin groups. Table 20 lists the climate recording checkpoints that are installed by default. The user can add more as needed. The names of the checkpoints here are indicative of their geographical location (Figure 39). Locations have been selected for their informative value.

Table 20: The climate monitoring checkpoints with their locations.

#	*Name*	*Latitude (dec)*	*Longitude (dec)*
1	Climate Monitor 1 Atlantic	47.0	-2.0
2	Climate Monitor 2 Occitanie	44.4	1.3
3	Climate Monitor 3 Mediterranean	42.0	3.2
4	Climate Monitor 4 Pyrenees	43.0	0.5
5	Climate Monitor 5 Scladina	50.5	5.0
6	Climate Monitor 6 Channel	50.0	-1.1

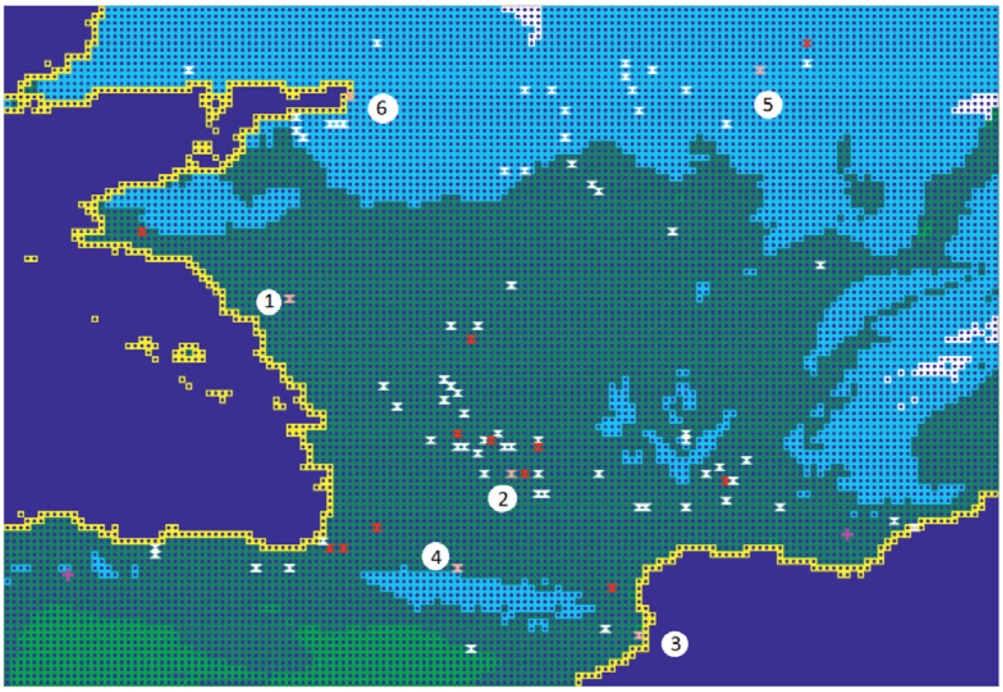


Figure 39: Climate recording checkpoint locations (pink crosses numbered 1-6).

8.6 Input files defining a simulation

There are two files defining what simulation the HomininSpace simulation system actually executes: a file with the values for the model parameters including the value for the random seed, and a file which defines what model elements participate and thus which questions are being answered (see Chapter 9).

8.6.1 Model parameter values

Parameter values for all simulations are stored in a file: “HomininSpace Parameters.txt”. This comma separated file contains a header with the parameter names and then for each line a (different) set of values. When running a single simulation from the user interface it is possible to select the line number with the values to be used. The 16 parameters have been described in Table 13. An example from the Standard parameter values list file is given in Table 21 with the header line with the names of the parameters included.

Table 21: The header and the first ten parameter value sets (comma separated) from the Standard parameter file.

ScenarioNumber,RandomSeed,BirthRate,DeathRate_PreFertileCohort,DeathRate_FertileCohort,DeathRate_PostFertileCohort,Subsistence_PreFertileCohort,Subsistence_FertileCohort,Subsistence_PostFertileCohort,Years_Before_Group_Maturity,GroupSize_BeforeMerge,GroupSizeFertile_BeforeMerge,GroupSize_BeforeSplit,Temperature_Tolerance,CohortSize_PreFertile,CohortSize_Fertile,Calories_Per_Kg_Meat,Max_ForagingRange,Parent_Scenario_1,Parent_Scenario_2,ViabilityIndex
1,30573,10,4,9,6,2750,4000,4750,6,5,5,46,-28,16,16,3300,10,0,0,0
2,31589,7,9,9,12,2500,5000,2500,1,2,2,16,-20,13,19,3350,8,0,0,0
3,6218,36,5,6,8,4000,3750,3000,3,2,4,76,-29,20,37,2850,3,0,0,30
4,16673,46,6,8,11,4000,2250,2250,8,2,2,96,-15,9,23,2600,5,0,0,103
5,2875,43,6,1,7,5000,3000,3000,8,2,5,146,-6,18,24,3250,3,0,0,548
6,20649,30,7,14,14,4750,3000,2250,10,1,1,26,-16,9,21,2850,12,0,0,0
7,8368,22,4,4,6,4000,4750,4250,3,2,1,116,-26,12,34,3250,4,0,0,34
8,20427,32,10,5,11,4250,5000,2250,3,3,3,126,-5,12,38,3150,10,0,0,18
9,9611,27,9,8,11,3000,2500,2750,6,4,5,56,-11,10,33,2750,11,0,0,1
10,23136,6,2,9,11,2000,2250,2500,8,4,4,6,-27,20,23,3200,12,0,0,0

Note that `ScenarioNumber`, `RandomSeed`, `Parent_Scenario_1`, `Parent_Scenario_2`, and `ViabilityIndex` are variables used for administrative and analytical purposes: `ScenarioNumber` gives each simulation an unique number, `RandomSeed` provides the seed for the random number generator, `Parent_Scenario_1` and `Parent_Scenario_2` are the parents for these parameter values if this set is generated by the Genetic Algorithm, and the `ViabilityIndex` is a pre-calculated measure of the effectiveness of the demographic sub-model.

8.6.2 Simulation settings

For hypotheses testing purposes and to answer specific questions some model elements can be activated or deactivated. Such model elements include the use of coastal resources, and the maximization of a foraging range. A detailed description is given in Chapter 9. A setting is a Boolean variable that defines if a certain model element is included in the simulations or not. There are ten different settings. Each setting is a Boolean variable that can have a value of either True or False, where True stands for activated. The model settings are read from a file that is located in the model data directory with the name: “HomininSpace Settings.txt”. This file contains two lines: one with a header with setting names, and one line with values for the settings.

The names of the settings, in the same order as they appear in the input file are: `ENERGY_CONTINUOUS`, `STATIC_DISPERSAL`, `USE_BEACHES`, `USE_MAXIMUM_FORAGING_RANGE`, `GROUPS_CAN_CROSS_WATER`, `USE_FACTORIES`, `DEATH_PENALTY_FOR_ABSENCE`, `ZERO_MODEL_DISABLE_WATER`, `ZERO_MODEL_DISABLE_ENERGY`, `ZERO_MODEL_RANDOM_WALK`. An example line with values is: “`false, false, true, false, false, false, false, false, false, false`” with all settings activating their false value but the coastal resources (#3) which is activated. See Table 23 (Chapter 9) for a description of what model elements can be activated for what purposes.

8.7 Simulation output

Output of a simulation run consists of a log file with logging and debug information, simulation data collected in maps, time series and charts (user interface only), and a results file with data collected during the simulation. Log information can be written at the end of each time step (optional). Visual output in maps and time series are created interactively and continuously in the user interface. When the simulation terminates, the file with results is written. Figure 40 presents the flow chart describing this process, without user interface.

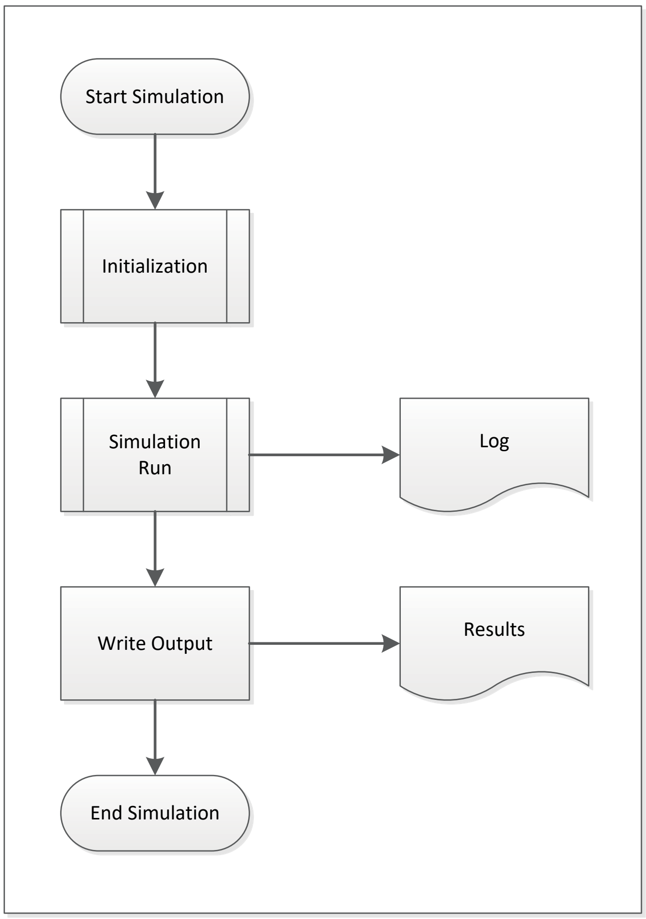


Figure 40: Flowchart - main simulation routine.

8.7.1 Logging and debug information

The log file is written in the data directory of HomininSpace. The name of this file is “HS Logfile [date][timestamp].txt”. For example the file “HS Logfile 2017-11-27 1144.txt” is the logfile written on 27 November 2017, where the simulation started at 11:44 in the morning. The content of the file is variable and is controlled by the Debug Level parameter[35]. With each logging statement in the source code a level is given, and if that level is less than the Debug Level, the information is written into the file. So if the user asks for level 0, no information is written to the file, and if the user selects 1000, all information is written (which can be costly in execution time). The default value is 5, where only information about the model construction and simulation results is written to the file. The main purpose of the debug log file is verification of the system. Debug information is also written to the debug console if the simulation is executed with user interface. Fragments of an example log file are given verbatim below, the full file is included in the Supplementary Materials (“HS Logfile 2018-11-29 0045.txt”).

[35] Either provided by the user via the user interface or in the “local_batch_run.properties” parameter file.

```
HomininSpace Simulation Thu Nov 29 00:45:24 CET 2018
HomininSpace Modelling and Simulation system, version 1.9.98
Reading run settings: HomininSpace Settings.txt
ParameterIndex: 1384
Parameters: 1395,5759,38,4,3,4,2750,5000,4000,6,1,4,16,-
26,14,17,3300,15,0,0,169
SimulationId: 1395
HomininSpace system initialized.
Scenario index: 1384
startingTimeStep: 131000
timeStepIncrement: 1
DEBUG_LEVEL: 10
tickCount: 81000
minimumRequiredTemperature: -26.0
birthsPercentage: 38
sizePreFertileGroup: 14
sizeFertileGroup: 17
…
useFactories: false
deathPenalty: false
intialIberiaGroups: 0
intialItalyGroups: 0
ZeroWater: false
ZeroEnergy: false
ZeroWalk: false
Model directory: D:\Repast Workspace\HomininSpace\
Ordering: Local, Iberia, Italy.
Habitat model (Kelly/Binford)
Random seed: 5759
Random seed: 5759
Reading Climate data file: HomininSpace Climate Grid.txt
Reading sealevel file: bintanja2008.txt
Creating grid: 148x101
Starting year = 131000 years BP; sealevel: 57.121; temperature
offset: -5.6483
Creating factories for Iberia (6,9) and Italy (125,20)
Reading checkpoints: HomininSpace Checkpoint List.txt
Added checkpoint Abauntz [x=42, y=17, 1 intervals][40000 - 54000]
Added checkpoint Abri Bourgeois-Delaunay [x=67, y=43, 6
intervals][130000 - 166000, 91000 - 116000, 111800 - 128670, 77060
- 79280, 107000 - 117000, 65000 - 121000]
```

8.7.2 Maps, bar charts and time series

Data from a running simulation in the user interface of HomininSpace can be displayed as maps, as bar charts and as time series. At the end of the simulation aggregated overviews of the data are created. These visualization aids are only available from within the user interface. Unless stated differently, all example output in the description below is created for simulation number 21, a model parameter value set with a high viability index[36]. Hominin groups are represented in maps by coloured circles, occupying one single grid cell regardless of number of individuals. Red dots are checkpoints, pink crosses are core areas. In figures with habitat reconstructed simulation results, the following colours are used for the habitats, with yellow (not shown) being used for beach areas (Table 22):

Table 22: Color coding of reconstructed habitats in simulation maps.

Tundra	**Boreal forest**	**Evergreen forest**	**Grassland**	**Deciduous forest**	**Woodland**
[white]	🟦	🟩	🟩	🟩	🟨

Maps are used to show information on the simulation grid itself: topography (height levels and sea grid cells) (Figure 41a), reconstructed environment (in two variants: energy level (Figure 41b) and habitat reconstruction (Figure 41c-d)), visit density (count of all visits to any cell in the simulation this far), and death density (count of all groups that died for each cell) (Figure 42). By default topography is shown at initialization, habitat reconstruction is visible during the simulation, and visit densities are presented at the end of the simulation. What is displayed can be configured in the code, by changing in the “Constants.java” source file the values of `VISIBLE_LAYER_WHILE_RUNNING` and `VISIBLE_LAYER_AT_END` accordingly. The different map types are used to verify simulations and to identify global patterns through time.

[36] Parameter values are: 21,1420,50,9,1,12,3500,3250,5000,7,3,5,26,-24,16,33,2700,9,0,0,1367. All settings (see Chapter 9) are set to False.

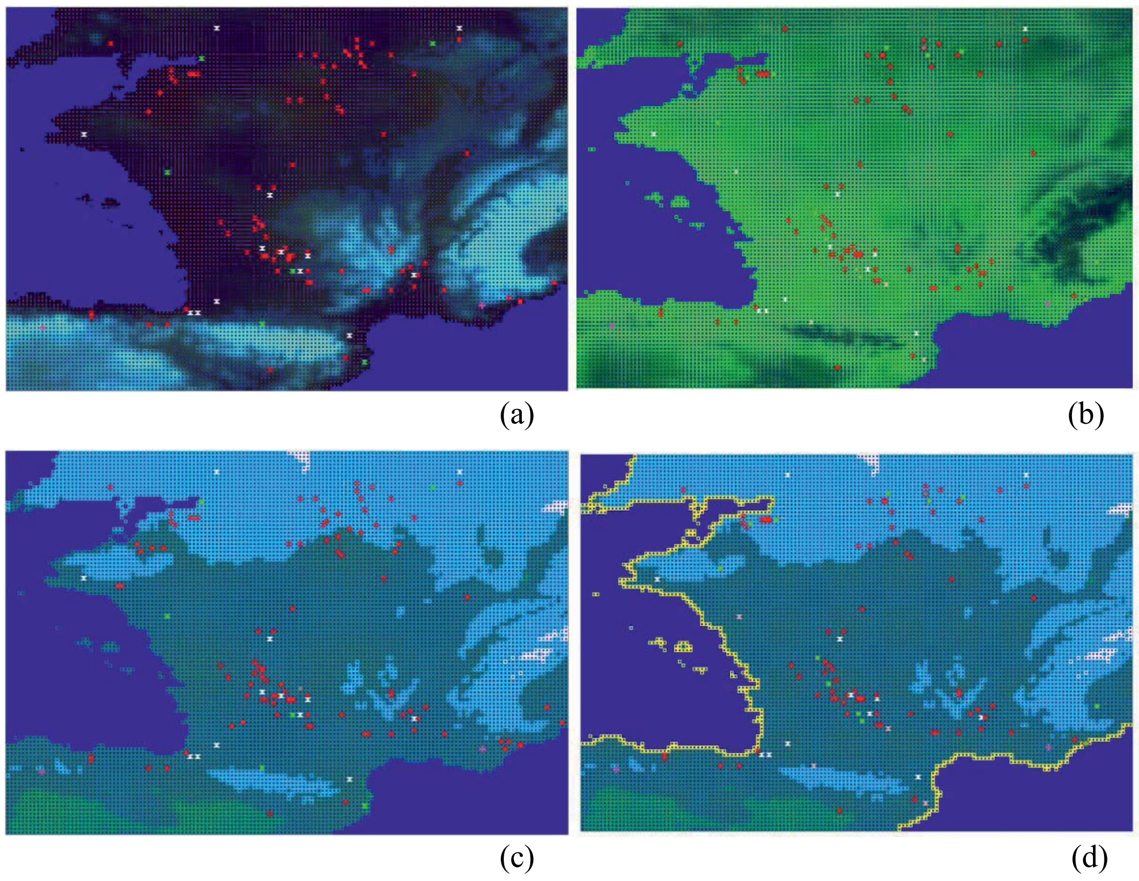


Figure 41: Different map types at the start of a simulation: topography (a), energy levels (b), habitat reconstruction (c) and habitat reconstruction with beaches (d).

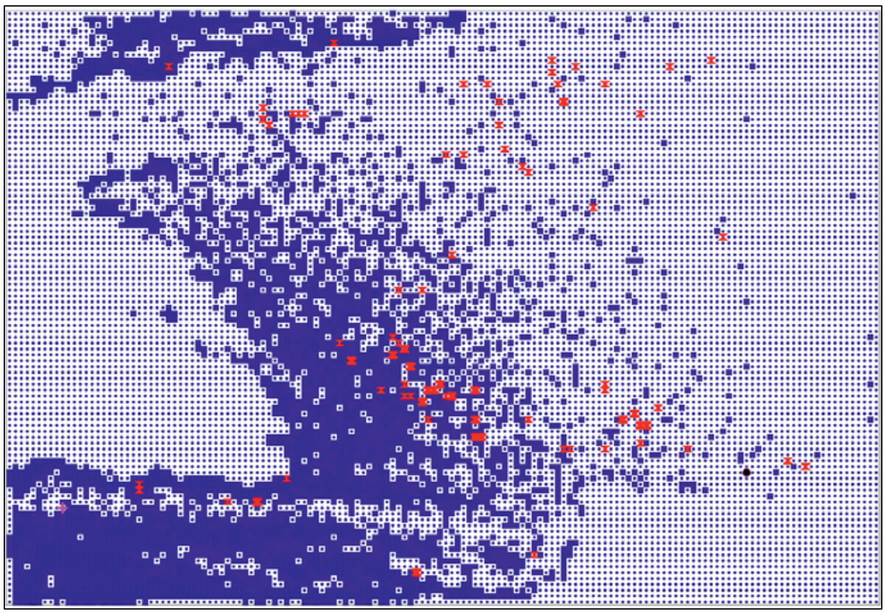


Figure 42: Death density map at the end of a simulation. Darker colors indicate more perished hominins.

Bar charts are graphical representations of data in the form of bars. In HomininSpace they present data for one point in time, updated at each time step during a simulation. Two charts are available: group ages and agent (group) sizes. Group age indicates how long a group survives. Agent size is the total group size, adding together all three age cohorts. These charts are used to display the character of the hominin population at any point in time.

Time series are graphs that can display values or statistical information like average, standard deviation, minimum, and maximum value per time step for variables while running the simulation. The following time series are implemented: global temperature, global sea level, position of all groups (the latitude), hominin count (total and per age cohort, see Figure 43), hominin type count (for three hominin types), group count (Figure 44), hungry group percentage, average foraging range, and average group size. Each of these visualization aids are fed by a data stream defined in the user interface. To enable output through time the data set must have the schedule parameter Frequency set to the value REPEAT, feeding values every time step.

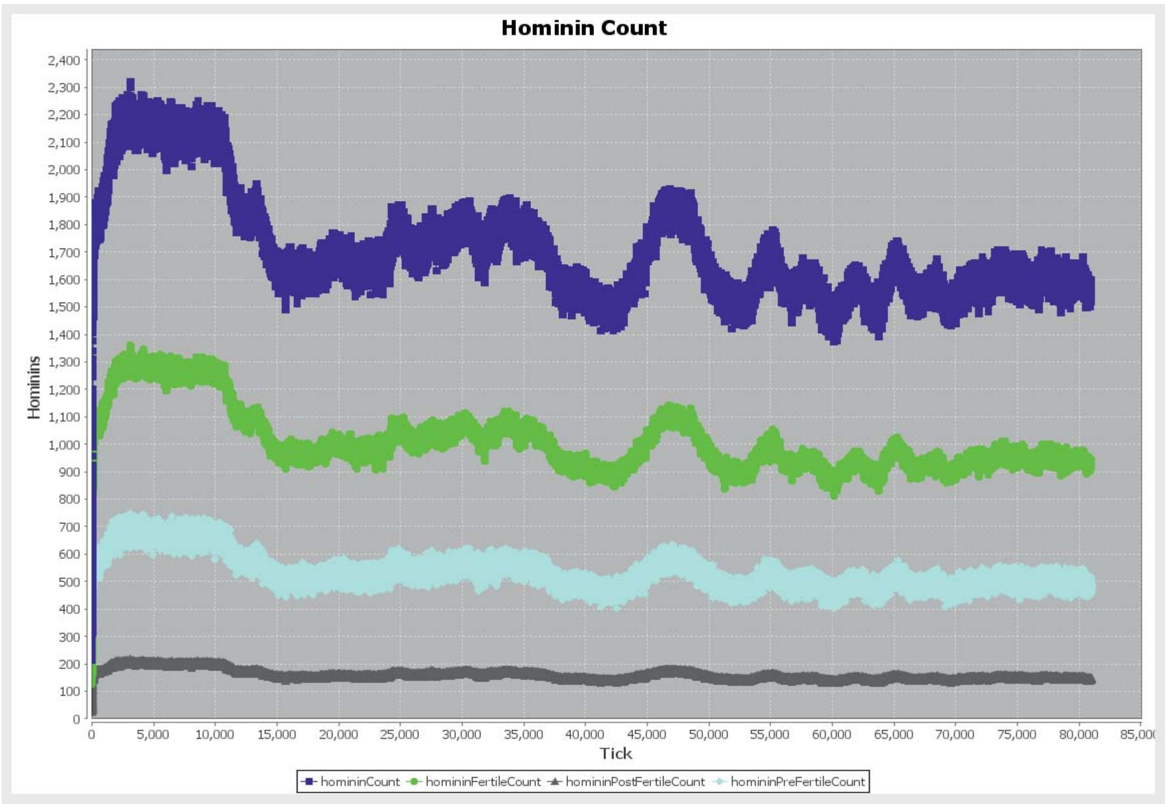


Figure 43: Example output at the end of a simulation: hominin count (totals in dark blue, different cohorts underneath) through time.

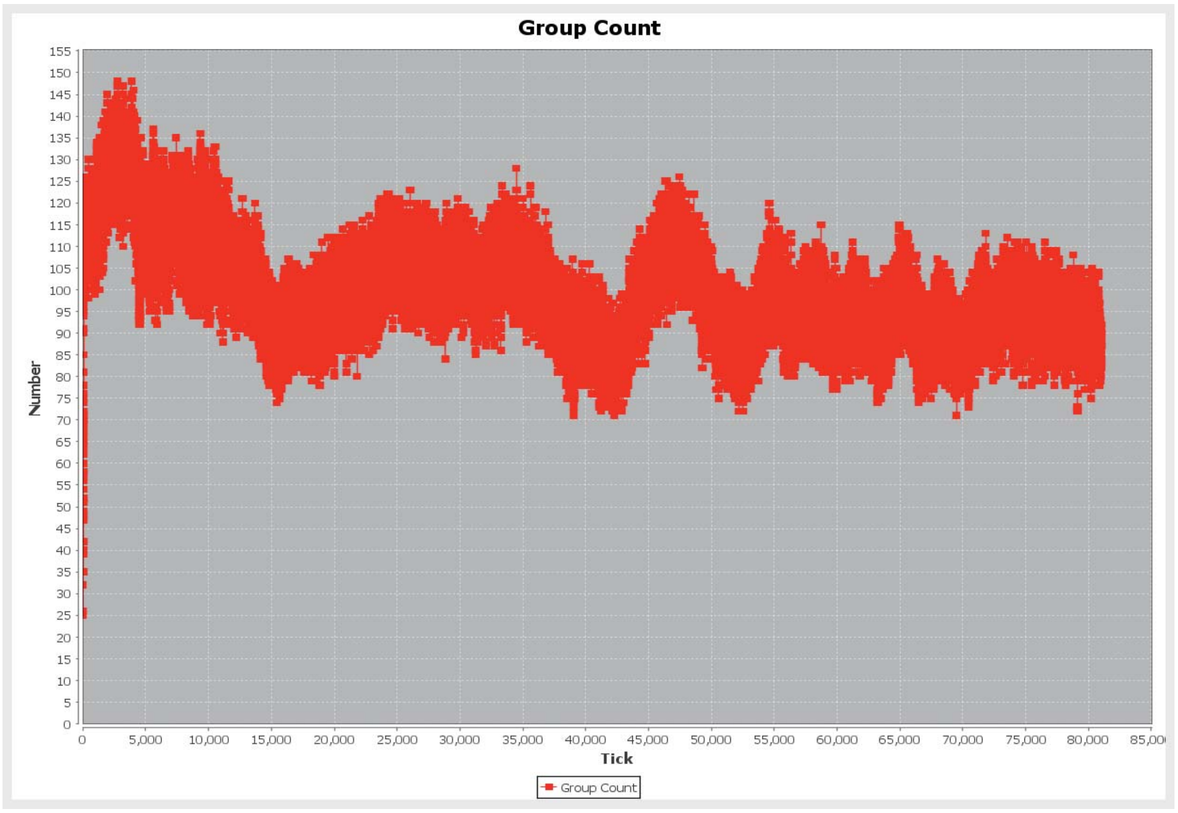


Figure 44: Example output at the end of a simulation: group count through time.

8.7.3

8.7.3 Simulation statistics

Besides a simulation score, simulation results including statistics on hominin groups and visits are written at the end of each simulation into the file “HS Results [parameter index] [date] [timestamp].txt”. For example, the file “HS Results 21 2017-11-27 1147.txt” contains the results for the simulation that used the values for parameters from (Standard) set 21 and ended on 27th November 2017 at 11:47 am. Every results file contains the following data: all parameter values and settings that were used, a number of simulation totals (explained below), the output totals for the climate monitoring checkpoints (each of them can also write an individual output file with data per time step), the output totals for the monitoring checkpoints and finally the output totals for all the regular checkpoints.

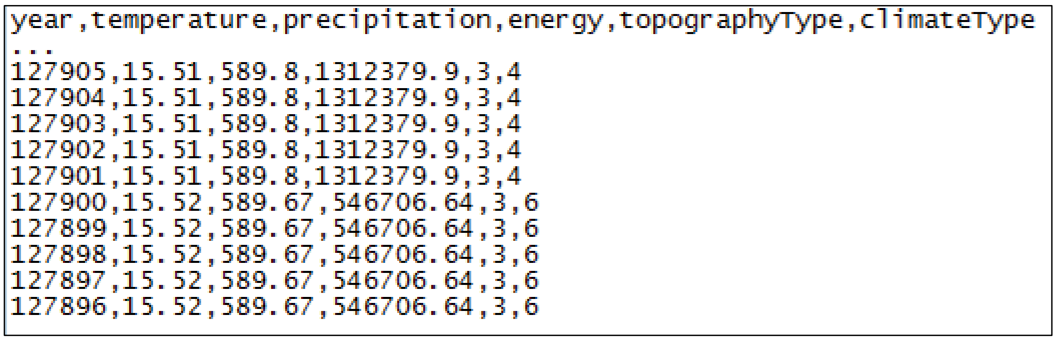
The following simulation totals are included in the results file and available for analysis:

`created_groups`, `created_hominins`, `averageGroupAge`, `totalCSTVisits`, `matchingVisits`, `matchedIntervals`, `matchedIntervalCoverage`, `totalVisitsAsHomerangeInsideInterval`, `totalVisitsAsHomerangeOutsideInterval`, `matchingVisitsWithConfidence`, `absenceVisits`, `deathsCrossing`, `deathsCrossingIberia`, `deathsCold`, `deathsFlooding`, `deathsHunger`, `deathsMerging`, `deathsTooSmall`, `Iberia`, `Italy`, and `Offspring`.

For each checkpoint the following totals are written in the results file:

> `totalCSTVisits`, `matchingVisits`, `matchedIntervals`, `matchedIntervalCoverage`, `totalVisitsAsHomerangeInsideInterval`, `totalVisitsAsHomerangeOutsideInterval`, `matchingVisitsWithConfidence`, and `absenceVisits`.

Furthermore, each climate monitoring checkpoint writes at the end of a simulation a results file with the following name: "HS Climate [name] [x=[xpos], y=[ypos], [n] intervals][simulation id][date][time].txt". For example: "HS Climate Climate monitoring station 3 [x=92, y=7, 0 intervals]21 2017-11-27 1300.txt", with information from simulation 21 for climate monitoring checkpoint "Climate monitoring station 3" with 0 associated intervals written at 27 November 2017 at 13:00. These files contain the following information for each time step: year, temperature, precipitation, energy, topographyType, and climateType. An example is given in Figure 45, illustrating the change in climateType at 127901 years ago. These files can be used to verify reconstructions, analyse local conditions and compare against other reconstruction data sets if available.



```
year,temperature,precipitation,energy,topographyType,climateType
...
127905,15.51,589.8,1312379.9,3,4
127904,15.51,589.8,1312379.9,3,4
127903,15.51,589.8,1312379.9,3,4
127902,15.51,589.8,1312379.9,3,4
127901,15.51,589.8,1312379.9,3,4
127900,15.52,589.67,546706.64,3,6
127899,15.52,589.67,546706.64,3,6
127898,15.52,589.67,546706.64,3,6
127897,15.52,589.67,546706.64,3,6
127896,15.52,589.67,546706.64,3,6
```

Figure 45: Example output from a Climate Monitor object. Taken from "HS Climate Monitor 3 Mediterranean Habitat [x=94, y=7, 0 intervals] 21 2018-03-23 1303.txt".