An Agent Based Model for Simulating Herd Dynamics

Beth Johnson
johnsel@umd.edu

Advisor:
Dr. Bill Fagan
Department of Biology
bfagan@umd.edu

Abstract

Biology and ecology currently lack a unifying theory for movement mechanisms as well as population level patterns. Most methods for analyzing both movement and population patterns require assumptions about the type of movement or expected population patterns. An agent based model, where each animal is represented as an evolutionarily trained neural network, is used to generate relocation data for a variety of landscapes without making assumptions about movement mechanisms. This data is analyzed using spatial metrics to classify the population level patterns that have emerged.

1 Project Background

There are many different mechanisms that animals may use to direct movement, for example an animal could use sensory cues (sight, smell) to move towards a resource such as food or water. Movement mechanisms fall into three broad categories: non-oriented, oriented, and spatial memory. Animals may use multiple mechanisms, for example, a random walk around a central place for part of the year followed by a migration to a new location that is guided by memory of landmarks. While there are various models for each of the different types of mechanisms, there is not a model that can accommodate all three. Based on the spatiotemporal variability of the landscape, there are three population movement patterns that animals may exhibit: sedentary ranges, migration, and nomadism. Sedentary range is when the population moves about a central place (relatively small compared to the available range for that species), migration is when a population follows a predictable cycle of movement between two disjoint locations, and nomadism is when the population moves to multiple locations in a non-predictable manner. It is hypothesized that the spatial and temporal variability of resource landscapes will determine the most efficient movement mechanism as well as the overall population pattern, as summarized in Figure 1. While there are different analysis methods for each of the possible population distributions, most rely on an initial assumption of the type of distribution the population is exhibiting [6].

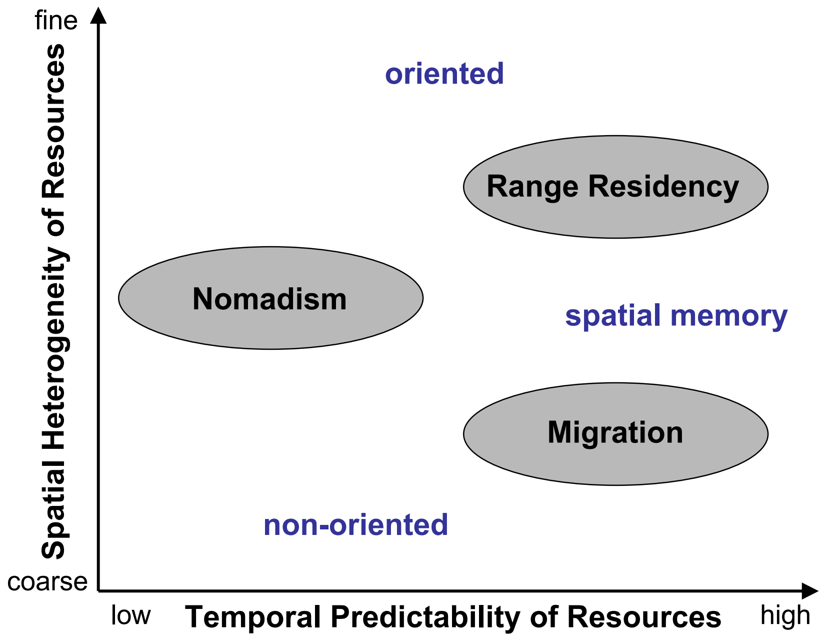


Figure 1: Hypothesized relationship between temporal predictability and spatial heterogeneity. For example, if the resources are highly predictable, spatial memory will be the most efficient mechanism. Based on the heterogeneity of the landscape, this could cause migration or range residency to occur. From [6].

1.1 Notation

The following notation will be used throughout this document.

Symbol	Meaning
N	Number of animals (or agents)
R	Number of total resources on the grid
r	Length (in cells) of side of resource patch {1, 2, 4, 8}
Pr	Predictability of landscape {0, 25%, 50%, 75%, 100%}
$\Delta x(t_i)$	Distance traveled in x direction from t_{i-1} to t_i
$\Delta y(t_i)$	Distance traveled in y direction from t_{i-1} to t_i
$P^i = \{p_1^i, p_2^i, \dots p_m^i\}$	Set of data points for animal i, where p^i is an (x, y) location
M	Total number of time steps (number of (x,y) pairs)
$S = \{s_1, s_2, \dots s_n\}$	Set of distances for which to calculate the PDI
μ_x	Global mean shift in x direction
μ_y	Global mean shift in y direction
σ	Global variance
ρ	Global correlation

2 Approach

There will be two parts to this project. The first part is an implementation of an agent based model with evolutionarily trained neural networks to generate relocation data. The second part is a suite of spatial metrics that will be used to analyze the simulation or relocation data to discern the type of population distribution.

2.1 Agent Based Model

The agent based model will consist of two stages – an evolution phase and a herd movement phase. The movement rules (as determined by the neural network) will be the same for both stages. Currently, only stage one has been implemented.

2.1.1 Details

The following gives further details about the landscape, agents, and evolution.

2.1.1.1 Landscape

The landscape will consists of 64 by 64 grid of cells with reflective boundaries. Each cell has an integer value indicating a resource count in the cell. If an agent moves to a cell with a non-zero resource count, a resource is transferred to the agent (the agent's resource count is increased by 1, the cell's resource count is decreased by one). The landscape is varied in two ways – patch size and predictability. The patch size is defined as the size of the resource squares as they are placed on the landscape, as seen in Figure 2. Predictability is defined as the percentage of the patches that remains in the same location throughout all generations, as seen in Figure 3. The simulation will be run for all 20 possible combinations of patch size and predictability. For a given generation, the starting landscape is the same for all individuals, but the landscape changes each generation according to the predictability. For the evolution stage, the maximum resource count in any cell is 1 and if a cell is depleted it will remain depleted for the current agent and generation. For the herd movement phase, the maximum resource count in any cell will be 5 and once a cell is depleted, it will regrow after 5 time steps.

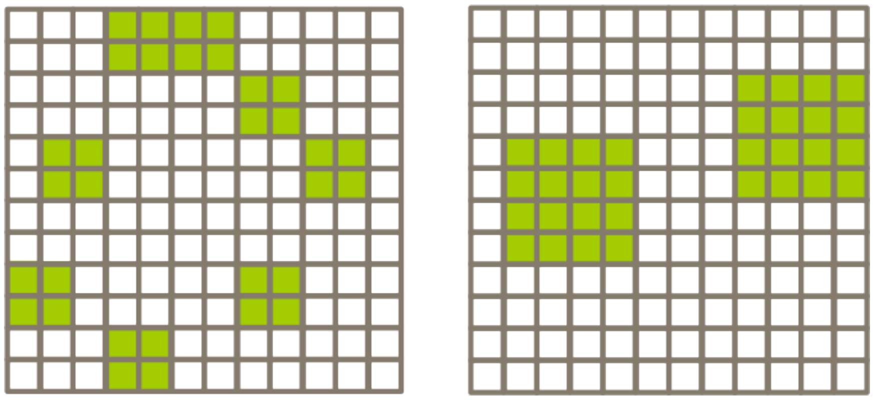


Figure 2: Patch size. The grid on the left shows a patch size of r = 2, while the right shows a patch size of r = 4. Note that the total number of resource cells remains constant. Patch size will be{1, 2, 4, 8}.

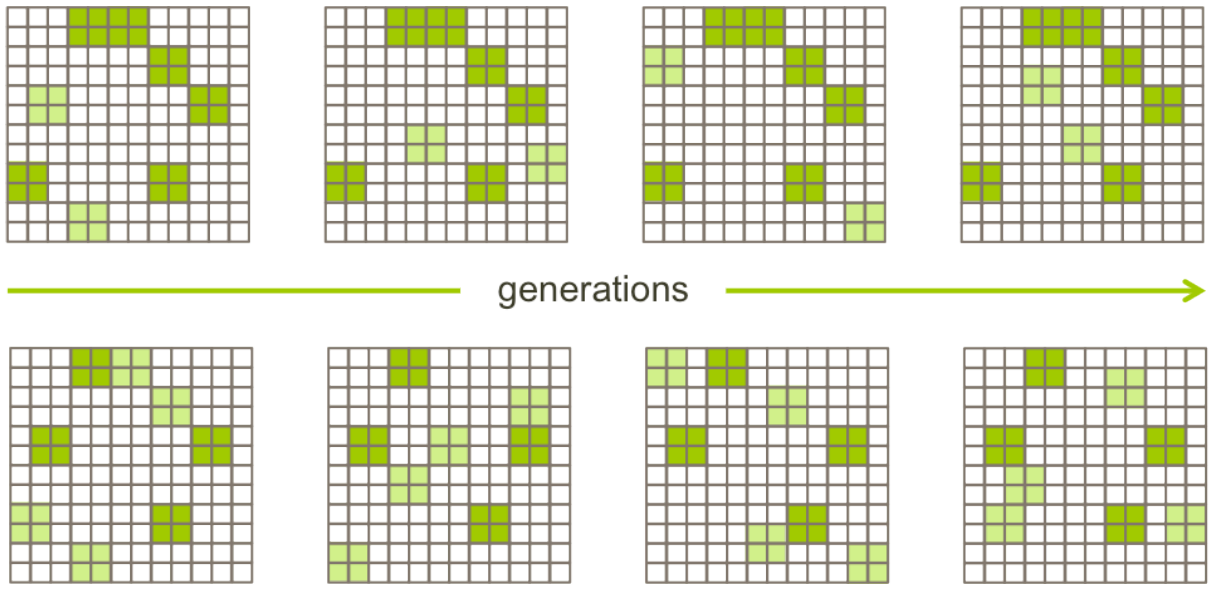


Figure 3: Landscape predictability. The top figure shows a landscape predictability of Pr = 0.75, where 6 of the 8 resource patches remain constant throughout generations. The bottom shows a landscape predictability of Pr = 0.5, where 4 of 8 resource patches remain constant throughout generations. Landscape predictability will be selected from {0, 0.25, 0.5, 0.75, 1.0}.

2.1.1.2 Agents

Each agent is represented by a fully connected feed forward artificial neural network, consisting of seven input variables (and a bias), three hidden nodes, and three output nodes as shown in Figure 4.

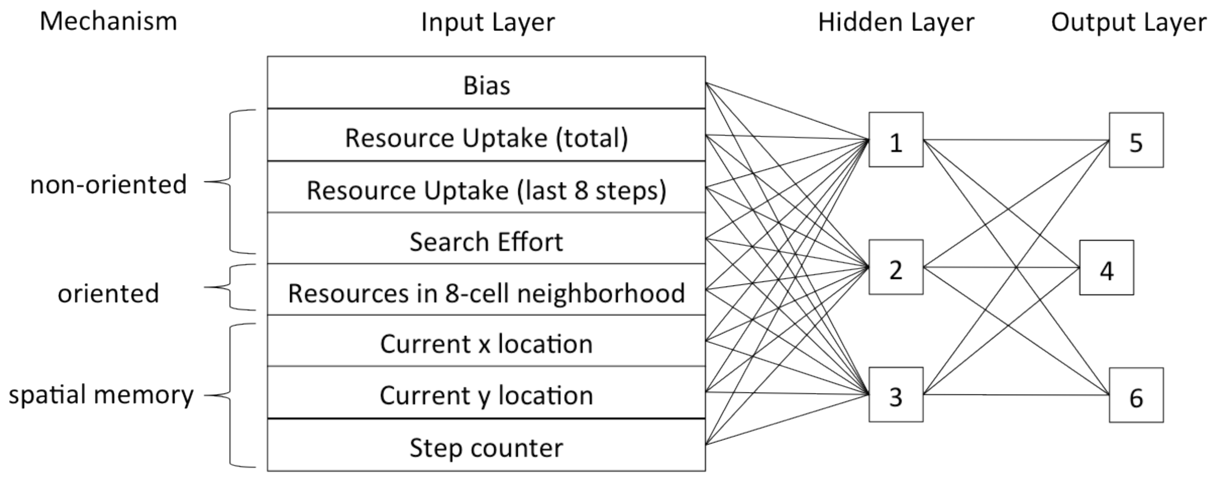


Figure 4: Agent Neural Network. There are 8 input nodes representing three different movement mechanisms. Modified from [5].

This gives 33 weights connecting the nodes, and these 33 weights are representative of the agent's "genes" and are integer values from [-5,5]. Initially, the weights are randomly selected from a uniform distribution. The inputs to the hidden layer are scaled to [-1,1], and the values of the hidden nodes and output nodes are determined by the following equations.

$$\text{HiddenNode}_j = \frac{\exp\left(\sum_{i=1}^{8} \text{Input}_\text{i} \times \text{weight}_\text{ij}\right)}{1 + \exp\left(\sum_{i=1}^{8} \text{Input}_\text{i} \times \text{weight}_\text{ij}\right)}$$

$$\text{OutputNode}_k = \frac{\exp\left(\sum_{j=1}^{3} \text{HiddenNode}_\text{j} \times \text{weight}_\text{jk}\right)}{1 + \exp\left(\sum_{j=1}^{3} \text{HiddenNode}_\text{j} \times \text{weight}_\text{jk}\right)}$$

Input variables are as follows and are updated after each movement step:

1. Current value of resource counter (starts at zero)
2. Increase in resource counter over the last eight steps (starts at zero)
3. Search effort over the last eight steps (defined as an agent's intention to deviate from current movement direction – see 2.1.2.3, starts at zero)
4. Number of cells in surrounding eight cell neighborhood that contains a resource
5. Current x-position
6. Current y position
7. Current time step

Agents may move to any cell immediately adjacent and may only move one cell at any given time step. Agents may not stay in the same cell unless they are "trapped" by the edges of the landscape and other agents.

2.1.1.3 Evolution

Each generation, agents are randomly selected without replacement in groups of six. The agent with the largest value of the resource counter is chosen to reproduce. When the agent is reproduced, the 33 weights (representing the genes of the agent) are copied six times to create six new agents. When copying the weights, there is a 20% probability of a crossover event occulting and a 2% probability at each weight for a point mutation.

2.1.2 Algorithms

The following algorithm describes how the overall model was (stage 1) or will be (stage 2) implemented. For ease of reading, the landscape initialization, landscape updates, agent direction selection, and reproduction algorithms are separated and detailed following the main algorithm.

For each execution of this overall algorithm, a patch size (r) and predictability (Pr) will be chosen.

```
Stage 0 – Initialization
    1. For i = 1:140
        a. Select 33 random integers [-5, 5] to be initial weights for the neural network for
           agent i.
    2. Initialize landscape

Stage 1 – Evolution
    1. For generation j = 1:5000
        a. For agent i = 1:240
            i. Place agent i in a random starting location.
            ii. For t = 1:150
                1. Determine direction to move.
                2. Move to new location.
                3. If resource exists at that location, add one to resource counter for
                   agent, subtract one from resource counter for cell.
            iii. Renew landscape to state at beginning of generation.
        b. If j != 5000
            i. Create 240 new agents according to reproduction algorithm.

Stage 2 – Herd Movement
    1. Choose agent from last generation with largest resource counter and create 240 copies
       with no mutation
    2. For t = 1:150
        a. For j =1:240
            i. Choose without replacement an agent to update.
            ii. If t = 1
                1. Place agent in random starting location. If agent already exists at
                   that location, select a new location.
            iii. Else (t > 1)
                1. Determine direction to move.
                2. Move to new location.
                3. If resource exists at that location, add one to resource counter for
                   agent, subtract one from resource counter for cell.
                4. Record agent location.
        b. If a cell was originally designated to hold a resource but has been depleted for 5
           time steps, reset cell resource count to 5.
```

2.1.2.1 Landscape Initialization

1. For i:R/r^2
   a. Randomly select a location for the top left corner of patch i.
   b. If the patch does not overlap another patch or the edge of the grid.
      i. Place patch i (set the value of cells in patch i to 1 for evolution stage or 10 for herd movement stage).
   c. Else, go to Step 1a.
2. Select $\frac{R}{r^2} \times Pr$ patches to remain constant throughout the generations. Let C be the set of patches that will remain constant.

2.1.2.2 Landscape Update

1. Start with blank landscape.
2. Place all patches in C.
3. For i:# of patches not in C
   a. Randomly select a location for the top left corner of patch i.
   b. If the patch does not overlap another patch or the edge of the grid
      i. Place patch i (set the value of cells in patch i to 1 if evolution stage generation counter is <5000, 20 if generation counter is 5000)
   c. Else, go to Step 3a.

2.1.2.3 Agent Direction Selection

The selection of a movement direction of an agent proceeds according to the following algorithm. If at any time a cell is unavailable due to another agent occupying that cell or due to cells not existing off the edge of the grid, the agent will move in a random direction selected from the available directions.

1. Compute hidden layer node values and output layer node values according to 2.1.1.2.
2. If Node 4 is [0, 1/3), use oriented movement. Set search effort to 0.
   a. If there is more than one resource in the surrounding cells, randomly select a cell with resource as new location.
   b. Else, if there is one resource, select that cell as new location.
   c. Else, there are no resources, randomly select cell.
3. Else, if Node 4 is [1/3,2/3), use non-oriented movement.
   a. If Node 5 is [0,1/3), move in same direction as previous time step, set search effort to 0.
   b. If output layer output is [1/3,2/3), set search effort to 1 and randomly select between
      i. Same direction as previous time step.
      ii. 45 degrees CW from previous time step.
      iii. 45 degrees CCW from previous time step.
   c. If output layer output is [2/3,1], set search effort to 9 and randomly select direction.
4. Else, Node 4 is [2/3,1], set search effort to 0 and use memory-oriented movement.
   a. If Node 5 is [0,1/8), go N.
   b. If Node 5 is [1/8,2/8) go NE.
   c. If Node 5 is [2/8, 3/8), go E.
   d. If Node 5 is [3/8, 4/8), go SE.
   e. If Node 5 is [4/8, 5/8), go S.

```
        f. If Node 5 is [5/8, 6/8), go SW.
        g. If Node 5 is [6/8, 7/8), go W.
        h. If Node 5 is [7/8, 8/8), go NW.
```

2.1.2.4 Reproduction

```
    1. For i = 1:40
        a. Randomly select six agents without replacement.
        b. Find agent with largest resource counter, call this agent A.
        c. For j = 1:6
            i. For k = 1:33
                1. Choose random number [0,1] from uniform distribution.
                2. If random number is <0.02
                    a. Choose integer from uniform distribution [-5,5] and set this value as gene k for agent j.
                3. Else, set gene k for agent j to the value of gene k for agent A.
            ii. Choose random number [0,1] from uniform distribution.
                1. If random number is <0.2
                    a. Choose random integer [1,33] from uniform distribution; this is the start of the crossover, call it c.
                    b. Choose a random agent (with replacement), call it agent B from previous generation.
                    c. Set genes c:33 for agent j to the values of genes c:33 for agent B.
```

2.2 Spatial Statistics

2.2.1 Realized Mobility Index

Realized mobility index, or RMI, describes the proportion of the annual range an animal is using. The range of the animal is defined as the minimum convex polygon covering all location points for that animal. The range of the population is defined as the minimum convex polygon covering all location points for all animals in the population. The RMI is simply the ratio of the area of the individual range to the area of the population range. An RMI is calculated for each individual in the population, and then averaged over the population [7]. The minimum convex polygon will be found using the following algorithm from Kirkpatrick and Seidel [3]. First, the upper hull will be determined, then the lower hull will be determined using the same method. If the left and right endpoints on the upper and lower hull are not the same, vertical lines can be drawn to connect them.

Algorithm for Finding Upper Hull

Note: in this algorithm, $x(p_i)$ represents the x value of the data point p, with index i.

1. Initialization – Let min and max be the indices of two points in A that form the left and right endpoints of the upper hull of S.

$$x(p_{min}) \leq x(p_i) \leq x(p_{max})$$
$$y(p_{min}) \geq y(p_i) \text{ if } x(p_{min}) = x(p_i),$$
$$y(p_{max}) \geq y(p_i) \text{ if } x(p_{max}) = x(p_i) \text{ for } i = 1, \ldots, n$$

   If min = max, print min and stop.
   Let $T := \{p_{min,}p_{max}\} \cup \{p \in S | x(p_{min}) < x(p) < x(p_{max})\}$

```
2. CONNECT(min, max, T)
```

Algorithm for CONNECT(k, m, s)

```
1. Find a real number a such that
       $x(p_i) \le a$ for $\left\lceil \frac{|P|}{2} \right\rceil$ points in P and
       $x(p_i) \ge a$ for $\left\lfloor \frac{|P|}{2} \right\rfloor$ points in P
2. Find the "bridge" over the vertical line $L = \{(x, y) | x = a\}$
       $(i, j) := BRIDGE(P, a)$
3. Let $A_{left} := \{p_i\} \cup \{p \in S | x(p) < x(p_i)$
   Let $A_{right} := \{p_j\} \cup \{p \in S | x(p) > x(p_i)$
4. If i = k then print(i)
5. Else CONNECT(k, i, $P_{left}$)
6. If j = m then print (j)
7. Else CONNECT (j, m, $P_{right}$)
```

Algorithm for BRIDGE(P,a)

```
1. $CANDIDATES := \emptyset$
2. If |P| = 2 then return ((I,J)), where $P = \{p_i, p_j\}$ and $x(p_i) \le x(p_j)$.
3. Choose $\left\lfloor \frac{|P|}{2} \right\rfloor$ disjoint sets of size 2 from P.
4. Determine the slopes of straight lines defined by the pairs.
5. Determine K, the median of $\{k(p_i, p_j) | (p_i, p_j) \in PAIRS\}$
6. Let $SMALL := \{(p_i, p_j) \in PAIRS | k(p_i, p_j) < K\}$
   Let $EQUAL := \{(p_i, p_j) \in PAIRS | k(p_i, p_j) = K\}$
   Let $LARGE := \{(p_i, p_j) \in PAIRS | k(p_i, p_j) > K\}$
7. Find the set of points A which lie on the supporting line h with slope K.
   Let MAX be the set of points $p_i \in S$, s.t. $y(p_i) - K \times x(p_i)$ is maximum.
   Let $p_k$ be the point in MAX with minimum x-coordinate.
   Let $p_m$ be the point in MAX with maximum x-coordinate.
8. Determine if h contains the bridge:
   If $x(p_k) \le$ and $x(p_m) > a$ then return ((k,m)).
9. H contains only points to the left of or on L:
   If $x(p_m) \le a$ then
   For all $(p_i, p_j) \in LARGE \cup EQUAL$, insert $p_j$ into CANDIDATES
   For all $(p_i, p_j) \in SMALL$ insert $p_i$ and $p_j$ into CANDIDATES
10. H contains points only to the right of L:
   If $x(p_k) > a$ then
   For all $(p_i, p_j) \in SMALL \cup EQUAL$ insert $p_i$ into CANDIDATES
   For all $(p_i, p_j) \in LARGE$ insert $p_i$ and $p_j$ into CANDIDATES
11. Return (BRIDGE(CANDIDATES, P)).
```

2.2.2 Population Dispersion Index

Population dispersion index, or PDI, describes how clustered the population is at different spatial scales [7]. The PDI is the average of the bivariate k-function for each individual in the population. The bivariate k-function is defined as expected number of points of pattern 1 within a distance s of an arbitrary point of pattern 2, divided by overall point density in pattern 1. The algorithm for calculating the PDI is given below. For ease of reading, the algorithm for the bivariate k-function is separated and detailed following the PDI algorithm. The algorithm for computing the bivariate k -function is based on Cressie [2] with edge corrections from Lotwick and Silverman [4].

```
1. Initialize the following:
    a. MCP, the set of ordered points representing the minimum convex polygon for the entire population range.
    b. A, the area of the minimum convex polygon defined by the hull.
    c. $S = [s_1, s_2, \dots, s_{N_s}]$, a set of distances to calculate the bivariate k-function.
2. For l = 1:N
    a. $P^1 = \{all\ location\ measurements\ for\ animal\ l\}$
    b. $P^2 = \{all\ location\ measurements\ for\ all\ animals\ except\ animal\ l\}$
    c. $\left[k_1, k_2, \dots, k_{N_s}\right]_l = bivariatek(P^1, P^2, A, MCP, S)$
    d. Find mean and variance for each $k_1, k_2, \dots$ over l.
```

2.2.2.1 Algorithm for Bivariate K-function

$\left[k_1, k_2, \dots, k_{N_s}\right]_l = bivariatek(P^1, P^2, A, MCP, S)$

```
1. Initialize $K_{12}^1 = \left[K_{12,1}^1, K_{12,2}^1, \dots K_{12,N_s}^1\right] = 0$
              $K_{12}^2 = \left[K_{12,1}^2, K_{12,2}^2, \dots K_{12,N_s}^2\right] = 0$
2. For i = 1:length of $P^1$
    a. For j = 1:length of $P^2$
        i. Compute distance between $(p_i^1, p_j^2)$
        ii. If distance is less than any value of s
            1. Compute $w(p_i^1, p_j^2)$ and $w(p_j^2, p_i^1)$, where $w(a, b)$ is defined as the fraction of the circumference of a circle centered at a and crossing b that lies within the minimum convex polygon
                a. For each value of s greater than the distance
                    $K_{12,s}^1 \to K_{12,s}^1 + \frac{1}{w(p_i^1, p_j^2)}$
                    $K_{12,s}^2 \to K_{12,s}^2 + \frac{1}{w(p_j^2, p_i^1)}$
        iii. Else, distance is greater than all s values, do nothing as $a\left(p_i^1, p_j^2, s\right) = 0$ and $a\left(p_j^2, p_i^1, s\right) = 0$
3. $K_{12}^1 \to \frac{A}{m_1 m_2} K_{12}^1$ and $K_{12}^2 \to \frac{A}{m_1 m_2} K_{12}^2$
```

2.2.3 MCI

The mobility correlation index is a measure of how much the movements of the animals are correlated with each other. The MCI assumes that the movements of the animals are described by a discrete time based Gaussian position jump process with N individuals and M time steps. A global mean shift in the x-direction (μ_x), mean shift in the y-direction (μ_y), variance (σ^2), and correlation (ρ) are assumed for all individuals. The global parameters are found by following the process described by Calabrese [1]. The log-likelihood function is given by

$$\ell = -MTrlogC_l - \frac{1}{2}\left\{\sum_{i=1}^{M}[\Delta x(t) - \boldsymbol{\mu_x}]^T C_l^{-1}[\Delta x(t) - \boldsymbol{\mu_x}] + \left[\Delta y(t) - \boldsymbol{\mu_y}\right]^T C_l^{-1}\left[\Delta y(t) - \boldsymbol{\mu_y}\right]\right\}$$

Where C$_l$(N by N) , μ_x (N by 1) and μ_y (N by 1) are defined as follows

$$C_l := \begin{bmatrix} \sigma & \rho & \rho & \cdots \\ \rho & \sigma & \rho & \cdots \\ \rho & \rho & \sigma & \ddots \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix}, \boldsymbol{\mu_x} = \mu_x \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \boldsymbol{\mu_y} = \mu_y \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

To find the parameters, the log-likelihood function is maximized. Because the C$_l$ matrix is circulant, the C$_l$ matrix can be re-expressed in terms of a uniform eigenvector and eigenvalues and the likelihood function maximized with respect to the eigenvalues. There is a closed form solution to the eigenvalues that maximize the likelihood function and they are as follows

$$\lambda_{0,max} = \frac{1}{2}\langle(\phi_0^T[\Delta x(t) - \boldsymbol{\mu_x}])^2 + (\phi_0^T\left[\Delta y(t) - \boldsymbol{\mu_y}\right])^2\rangle_t$$

$$\lambda_{+,max} = \frac{1}{2(N-1)}\langle\Delta x(t)^T\left(1 - \phi_0\phi_0^\dagger\right)\Delta x(t) + \Delta y(t)^T\left(1 - \phi_0\phi_0^\dagger\right)\Delta y(t)\rangle_t$$

where $\phi_0 = \frac{1}{\sqrt{N}}[1,1,\cdots]^T$. Then, the values of the parameters can be obtained with the following relations (where the MCI is equivalent to ρ).

$$\mu_x = \phi_0\phi_0^\dagger\langle\Delta x(t)\rangle_t$$

$$\sigma = \frac{\lambda_0 + (N-1)\lambda_+}{N}$$

$$\rho = \frac{\lambda_0 - \lambda_+}{N}$$

3 Implementation

The above algorithms will all be implemented using R due to its popularity among biologists and ecologists. The code will be run on a Macbook Pro with a 2.9 GHz Intel Core i7 processor with 8 GB of RAM.

4 Databases

The agent based model will generate the data for input to the PDI, RMI, and MCI algorithms. Additionally, real relocation data from gazelles will be used as input to the PDI, RMI, and MCI algorithms. The relocation data will consist of a time series of x and y locations from 36 gazelles with GPS collars. The time series for an individual gazelle ranges from 50 days to 917 days, with 8111 data points in total. The time intervals between data points vary from 1 to 25 hours and will have gaps from satellite interference or battery saving programs. Some preprocessing of this data will likely be necessary to use with the PDI, RMI, and MCI metrics.

5 Validation

5.1 Agent Based Model

The agent based model will be validated in smaller modules as well as overall.

5.1.1 Landscape Initialization

The landscape initialization can be validated by computing the number of cells with a resource and ensuring it equals R. The landscape update can be validated by updating the landscape many times and summing the number of resources per cell over time. The cells that were selected to be stationary throughout generations should have a resource value equal to the number of updates to the landscape, and the other cells should have a resource value that is less than one but uniform across all cells that were not selected to be stationary. Visualizing this with a color map of the matrix (`image.plot` function in R) leads to little information about the uniformity of the cells not selected to be stationary as the resource value of the stationary cells is much larger than the surrounding cells. Thus, for better visualization, the stationary plots (as determined by the landscape generation function) were removed from the following plots. The lack of pattern in the non-stationary cells confirms that the landscape is being generated appropriately. There is a noticeable decrease in resource counts near the edges of the map as well as near the edges of the stationary patches. This is because the boundary conditions of the map are reflective, not periodic, so that patches may not be placed such that they hang off the edges of the map. Thus, there are fewer patch placement locations that result in a resource being on the edges of the map. A similar pattern occurs around the stationary patches, because to keep the number of resources constant, the patches are not allowed to overlap. It should also be noted that due to the definition of predictability, with a patch size of 8, there are only two resource patches, which only allows for a predictability of 0%, 50%, or 100%, so patch size of 8 is not included in the plots for 25% or 75%. The landscape validation plots can be seen in Figure 5 to Figure 9, below.

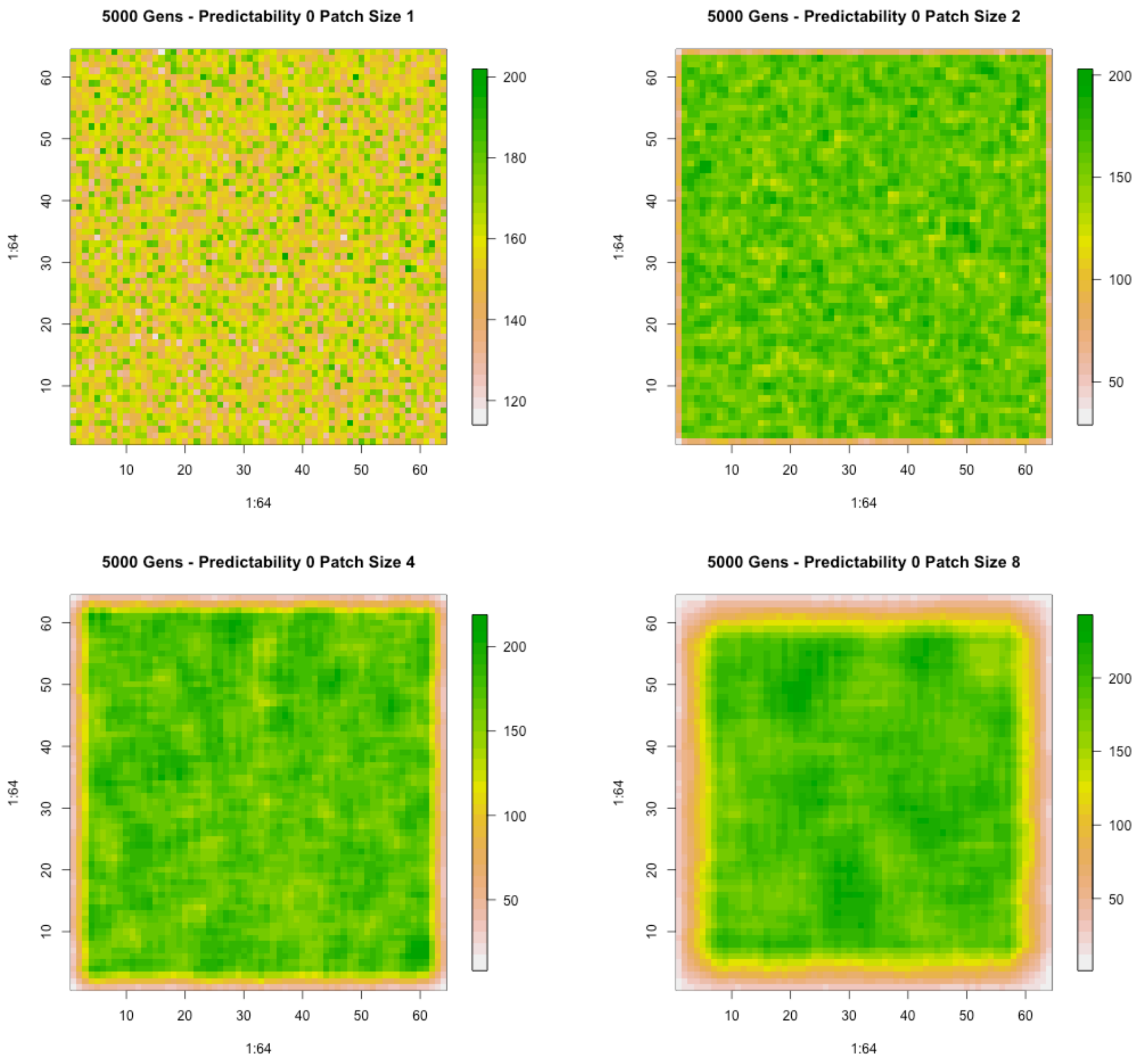


Figure 5: Landscape Validation for Predictability of 0%

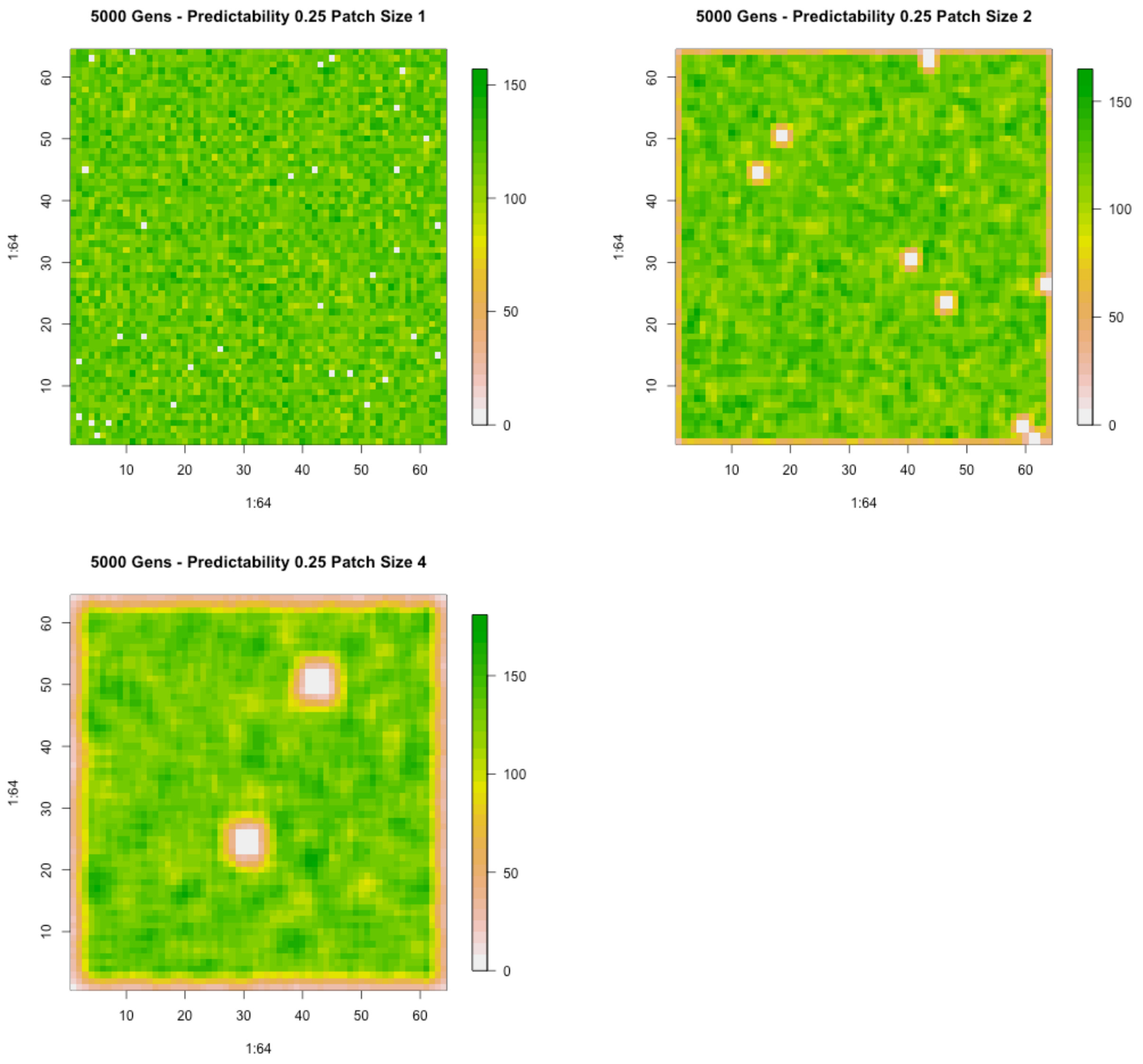


Figure 6: Landscape Validation for Predictability of 25%

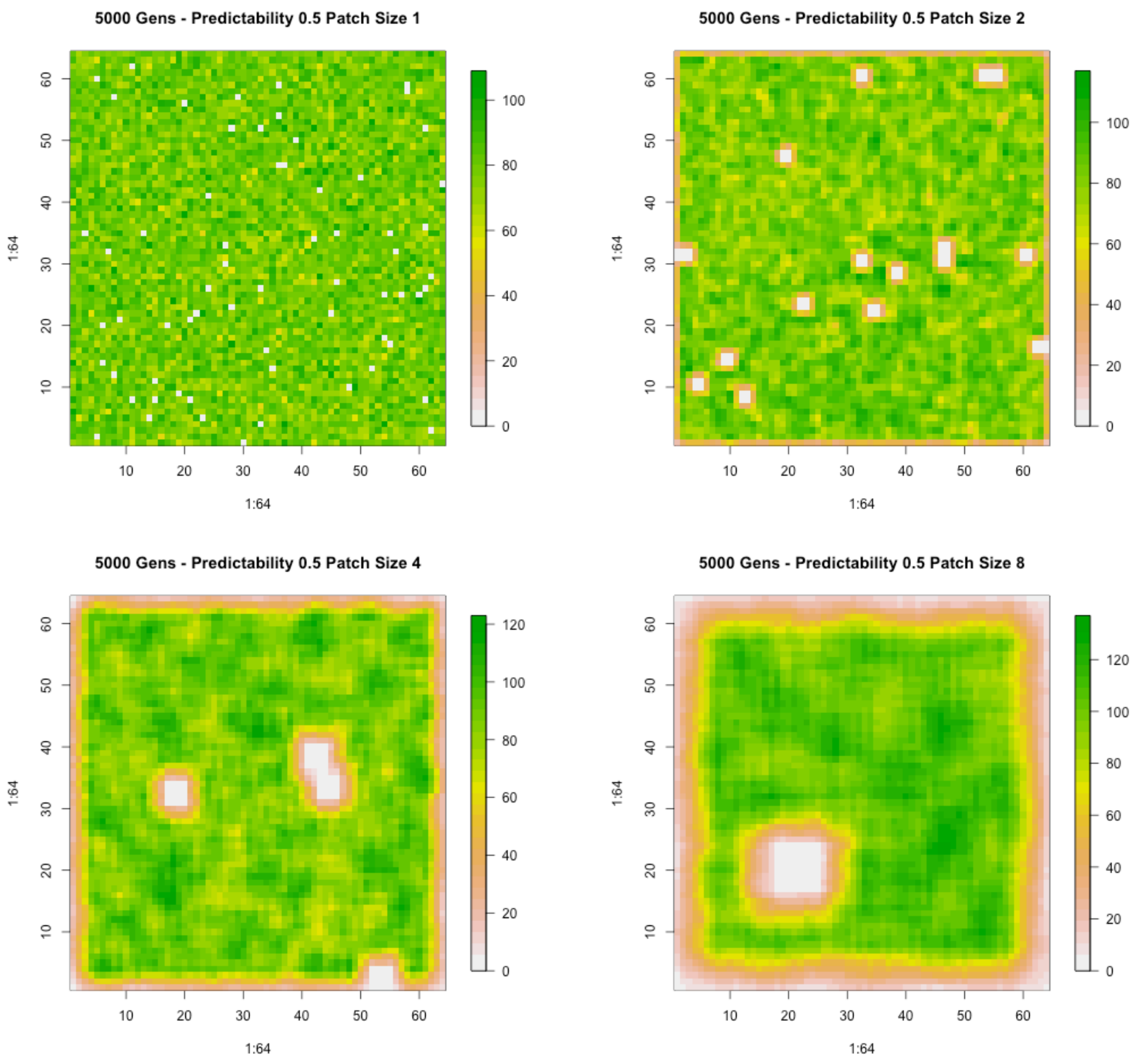


Figure 7: Landscape Validation for Predictability of 50%

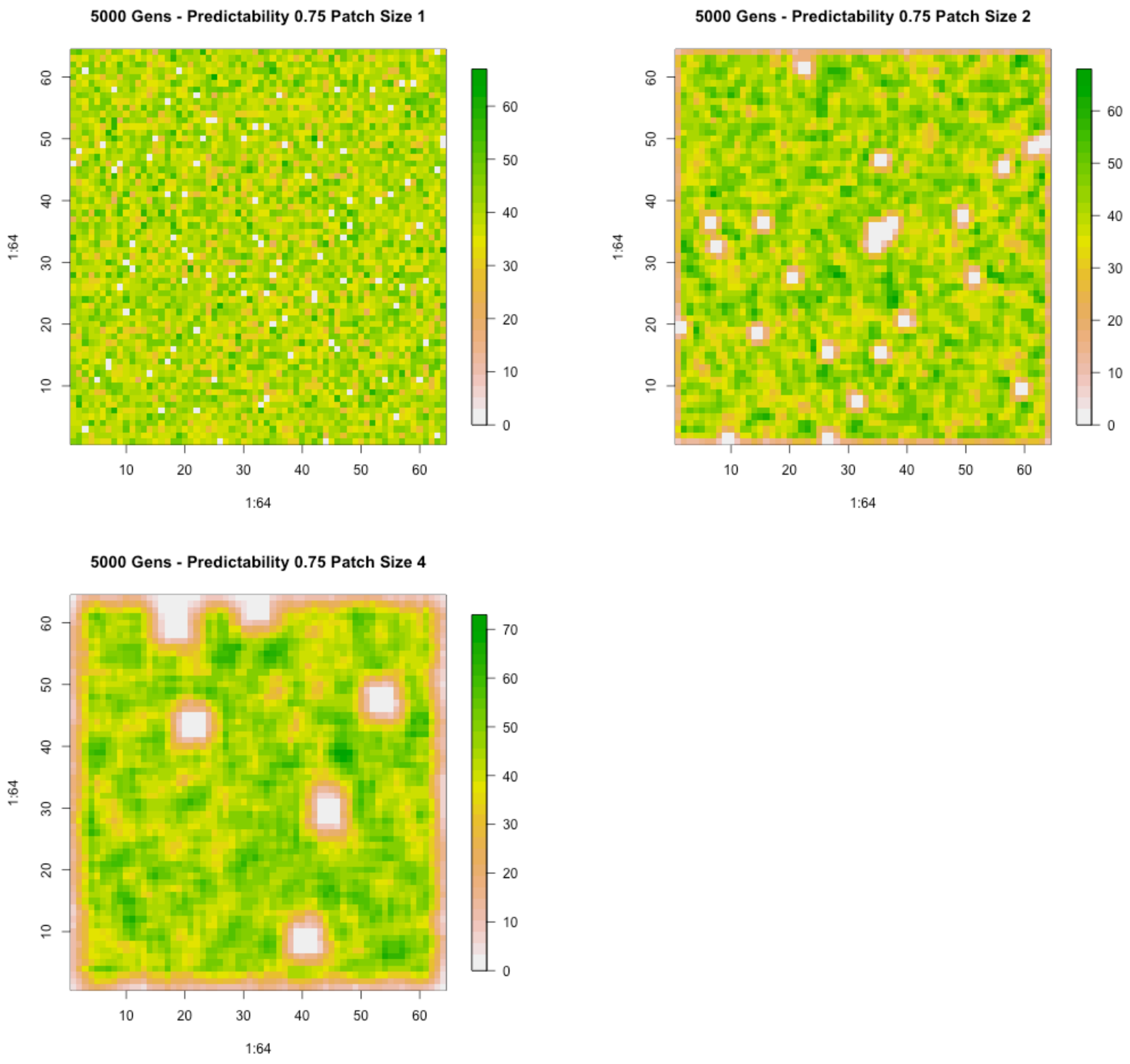


Figure 8: Landscape Validation for Predictability of 75%

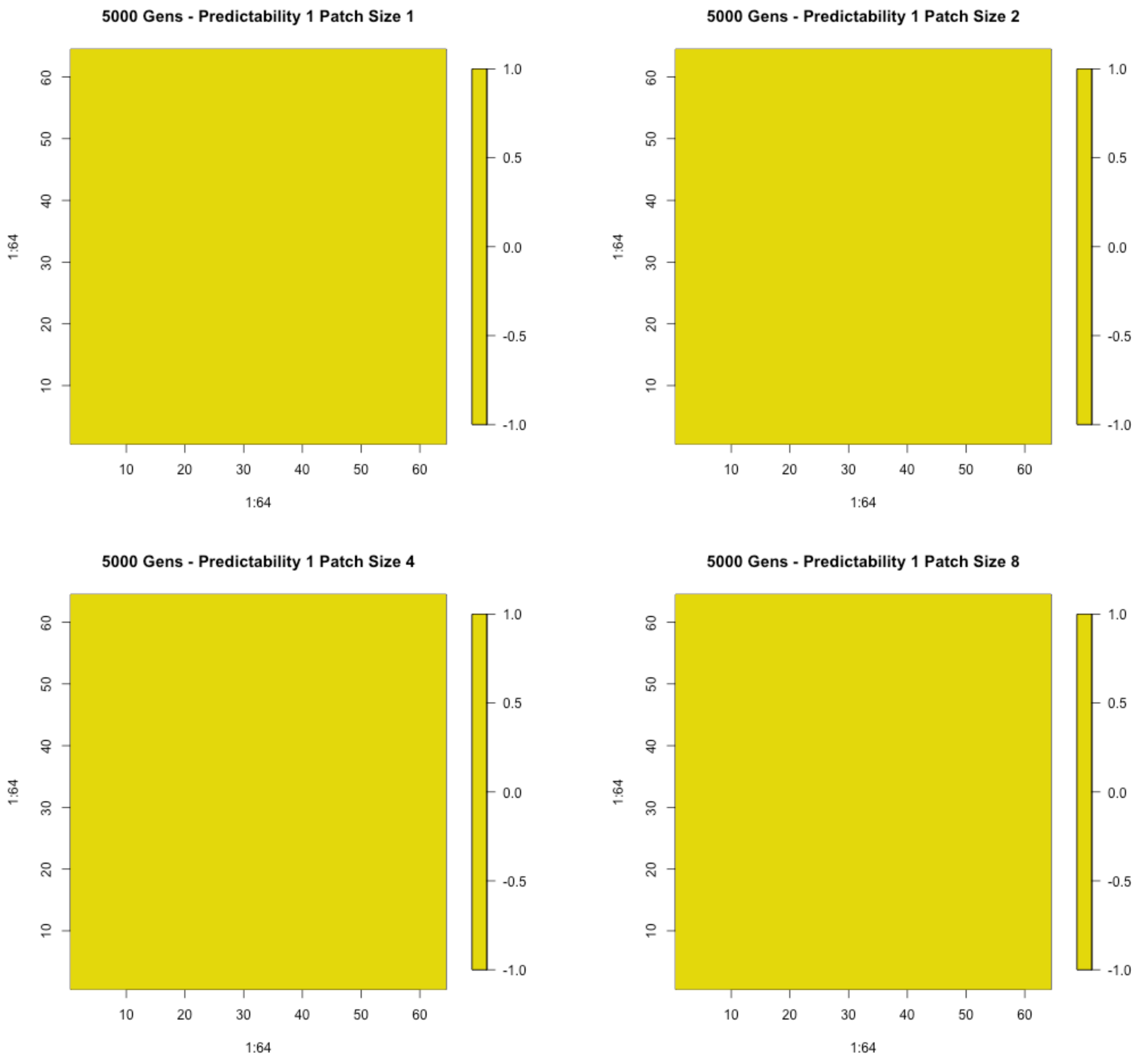


Figure 9: Landscape Validation for Predictability of 100%. Note that once stationary patches are removed, there are no resources elsewhere on the map, giving every cell a value of 0 for resources.

5.1.2 Agent Movement

The agent movement can be validated by hand computations of the values of the nodes as well as hand comparison of the node values and the resulting movement decisions. For agent movement validation, 3 agents were initialized with specific weights, input values, and previous direction. The location and visible resources were varied throughout the test cases. Focus was given to the edges of the map in selecting locations. For each test case, the values of the hidden and output nodes were calculated by hand, and then the agent direction selection algorithm was followed. For some combinations of node values, location, and resources, the algorithm specifies multiple directions from which to randomly select. The agent direction function was called 10,000 times for each test case, and a histogram was created from the distribution of directions selected. The histograms confirm that the correct direction was always selected by the agent direction function, as well as a uniform distribution between multiple directions if multiple directions were allowed for the resulting node values. In the following figures, the location of the agent, previous direction, calculated node values, and an image of the 8-cell neighborhood are given. The x represents the agent's location, the grey shaded cells represent the map does not exist in those cells, 0 represents no resource in that cell, and 1 represents a resource is in that cell.

Test Case 1

Row	1
Column	64
Previous Dir	2

▨	▨	▨
0	x	▨
0	0	▨

Node 4	0.810
Node 5	0.906
Node 6	0.004

Memory Movement, direction of 1, but unavailable, so randomly select from directions 5, 6, 7

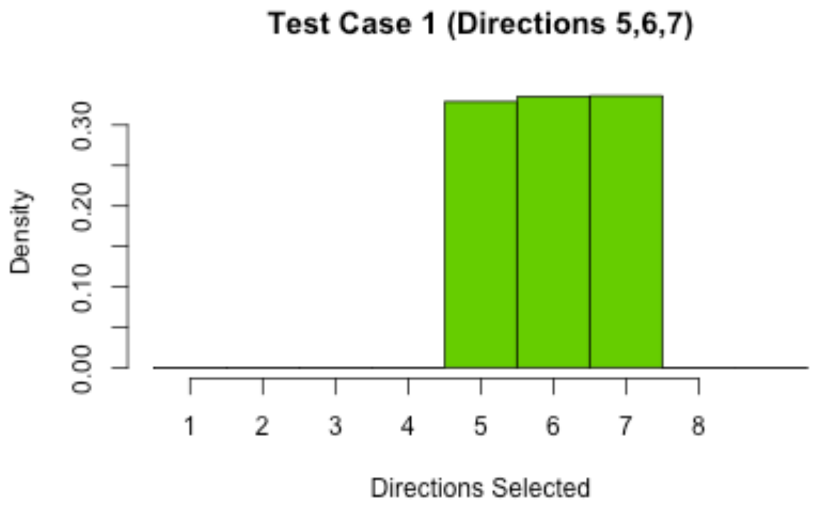


Figure 10: Agent Movement Validation Test Case 1

Test Case 2

Row	64
Column	1
Previous Dir	6

▨	0	0
▨	x	0
▨	▨	▨

Node 4	0.981
Node 5	0.003
Node 6	0.053

Memory Movement, direction of 1

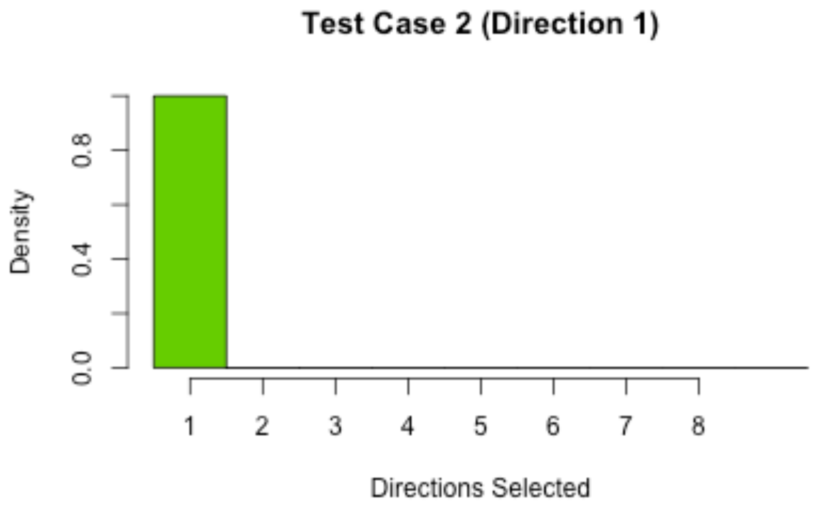


Figure 11: Agent Movement Validation Test Case 2

Test Case 3

Row	64
Column	64
Previous Dir	4

1	1	▨
1	x	▨
▨	▨	▨

Node 4	0.501
Node 5	0.501
Node 6	0.490

Oriented Movement, 3 resources visible, select from 1, 7, 8

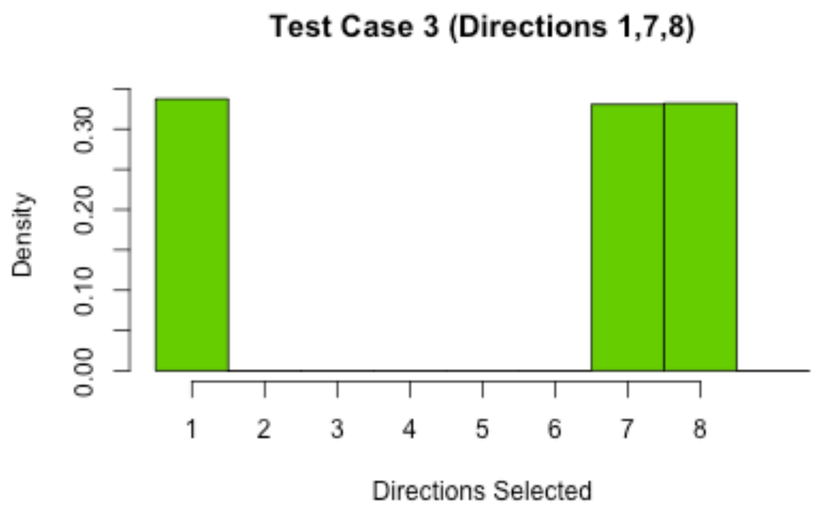


Figure 12: Agent Movement Validation Test Case 3

Test Case 4

Row	64
Column	64
Previous Dir	4

0	0	▨
1	x	▨
▨	▨	▨

Node 4	0.501
Node 5	0.501
Node 6	0.490

Oriented Movement, 1 resources visible, direction of 7

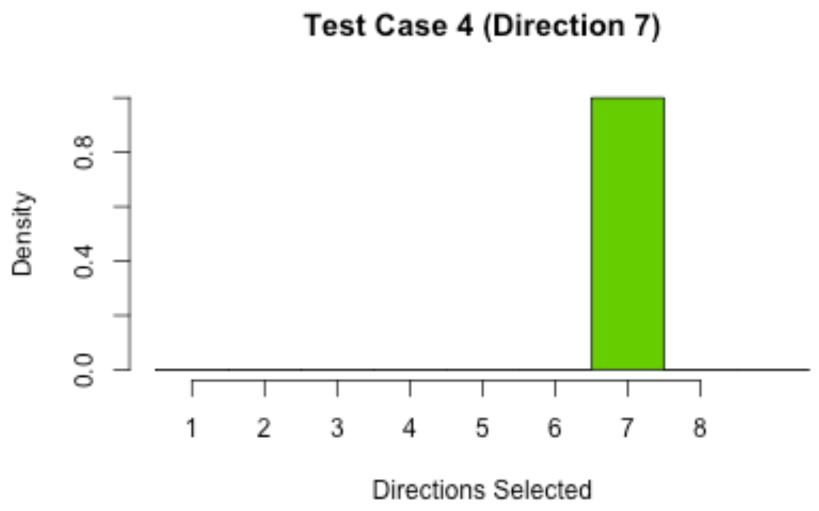


Figure 13: Agent Movement Validation Test Case 4

Test Case 5

Row	2
Column	64
Previous Dir	1

0	0	▨
0	x	▨
0	0	▨

Node 4	0.267
Node 5	0.504
Node 6	0.048

Non-Oriented Movement, previous direction was 1, 2 is unavailable, so select from 1, 8

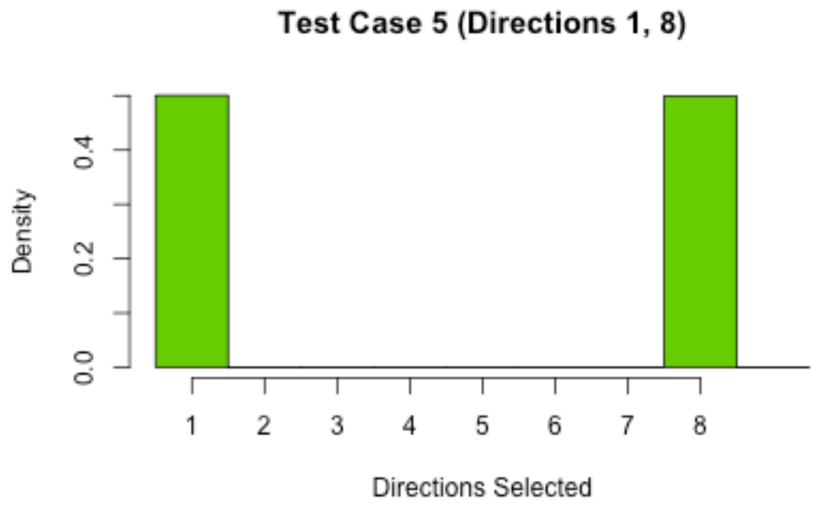


Figure 14: Agent Movement Validation Test Case 5

Test Case 6

Row	64
Column	2
Previous Dir	6

0	0	0
0	x	0
▨	▨	▨

Node 4	0.269
Node 5	0.501
Node 6	0.047

Non-Oriented Movement, previous direction was 6, 5 and 6 is unavailable, direction of 7

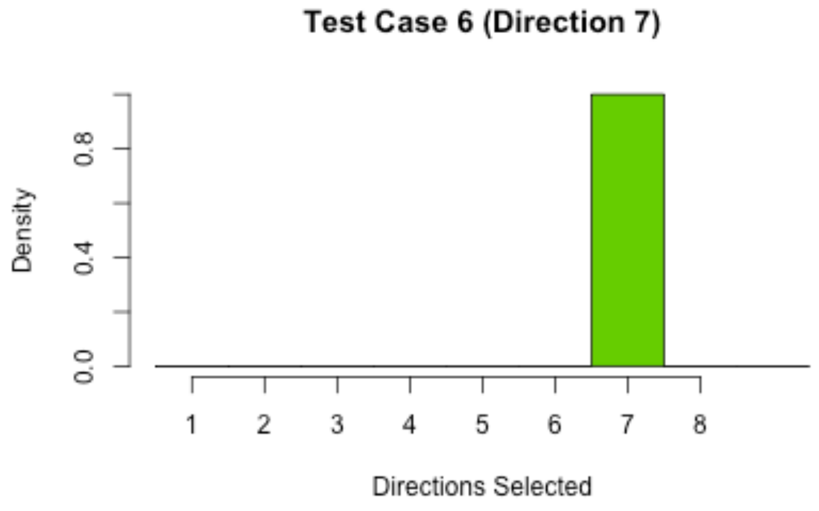


Figure 15: Agent Movement Validation Test Case 6

Test Case 7

Row	1
Column	64
Previous Dir	2

▨	▨	▨
0	x	▨
0	0	▨

Node 4	0.267
Node 5	0.504
Node 6	0.048

Non-Oriented Movement, previous direction was 2, 1, 2, 3, 4, 8 unavailable, so select from 5, 6, 7

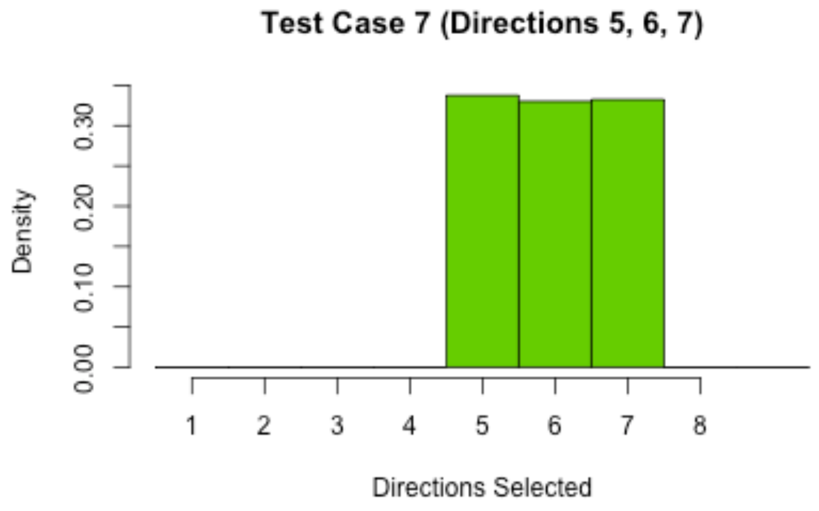


Figure 16: Agent Movement Validation Test Case 7

Test Case 8

Row	63
Column	63
Previous Dir	1

0	0	0
0	x	0
0	0	0

Node 4	0.267
Node 5	0.484
Node 6	0.055

Non-Oriented Movement, previous direction was 1, select from 1, 2, 8

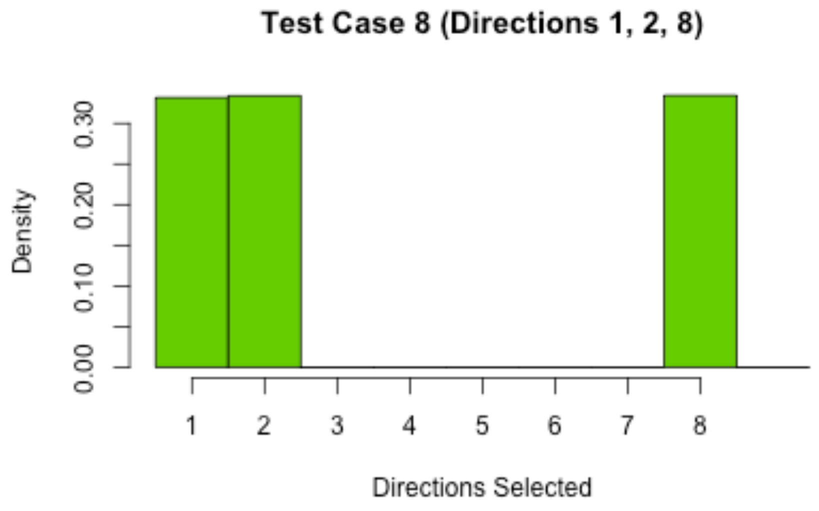


Figure 17: Agent Movement Validation Test Case 8

Test Case 9

Row	35
Column	35
Previous Dir	1

0	1	0
0	x	0
0	0	0

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 1 resource available, direction of 1

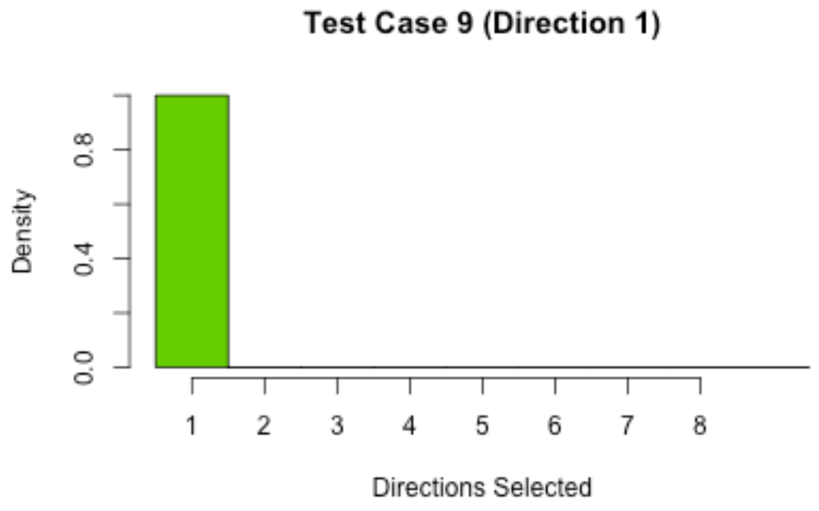


Figure 18: Agent Movement Validation Test Case 9

Test Case 10

Row	35
Column	35
Previous Dir	1

1	1	1
0	x	0
0	0	0

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 3 resources available, randomly select from direction of 1, 2, 8

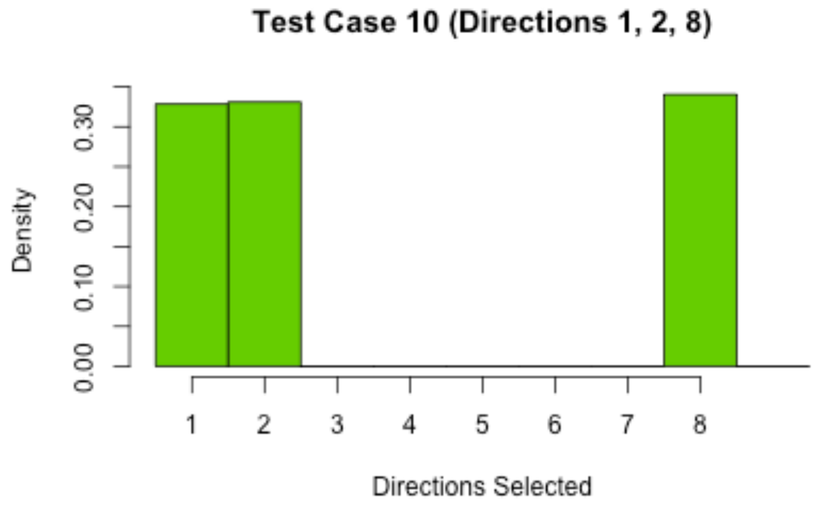


Figure 19: Agent Movement Validation Test Case 10

Test Case 11

Row	35
Column	35
Previous Dir	1

1	1	1
0	x	0
1	0	1

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 5 resources available, randomly select from direction 1, 2, 4, 6, 8

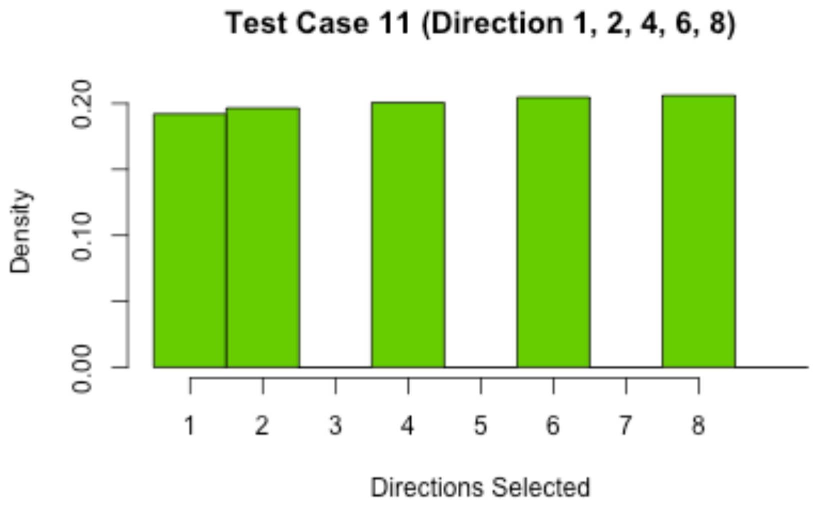


Figure 20: Agent Movement Validation Test Case 11

Test Case 12

Row	35
Column	35
Previous Dir	1

1	1	1
1	x	1
1	0	1

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 7 resources available, randomly select from directions 1, 2, 3, 4, 6, 7, 8

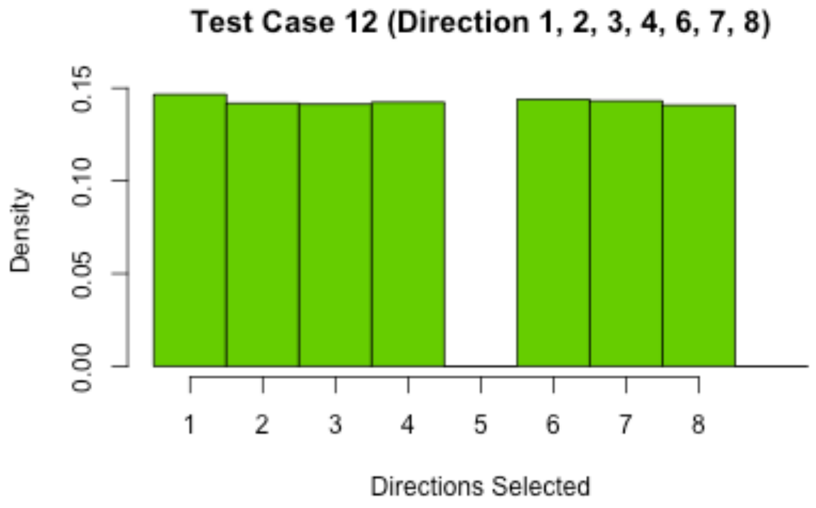
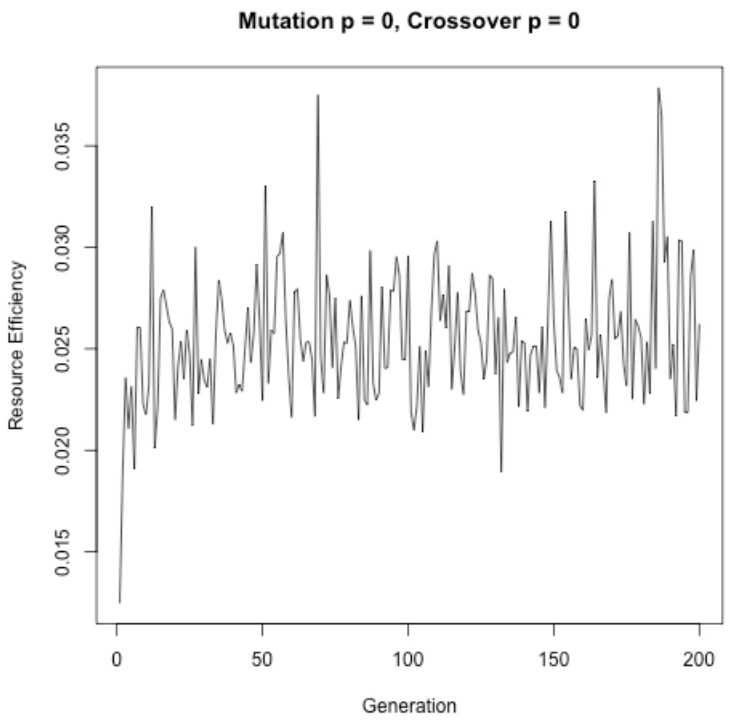


Figure 21: Agent Movement Validation Test Case 12

5.1.3 Reproduction

The reproduction can be validated with edge cases. If the mutation and crossover are set to zero, the overall efficiency, defined as the ratio of resources gathered to time steps taken, should have a sharp initial increase (where the most efficient agents are copied perfectly) and then be relatively constant throughout all generations. As seen in Figure 22, below, this is the case when mutation and crossover are both set to zero, with a constant efficiency of 2.5% after about 20 generations, where the variance is due to random placement of agents. When the crossover is set to 20% and point mutation to 2%, the efficiency is increasing, with an efficiency of about 8% after 200 generations.



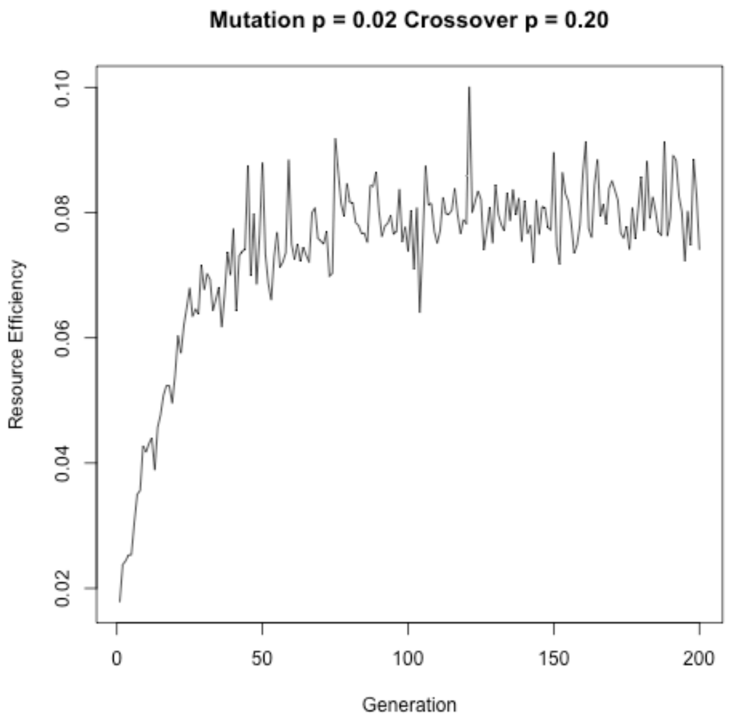


Figure 22: Reproduction Validation using edge case (top) compared to model values for mutation and crossover (bottom)

5.1.4 Overall Stage 1

The overall model for stage 0 and 1 can be compared to an existing C implementation [5].

5.1.5 Overall Stage 2

As stage 2 is only a slight modification of stage 1, stage 2 can be further validated by creating a visualization of the agent movement and checking that the agents do not occupy the same cell at the same time.

5.2 Spatial Statistics

Both the RMI and the PDI can be validated by comparing to a hand calculation of a small set of data points. Additionally, the minimum convex polygon can be compared to the results of the R function `chull`, which will compute the minimum convex polygon via a different algorithm, but should yield the same result. The bivariate k-function calculated can be compared to the values from the R function `k12hat`. The MCI can be validated by comparing to an implementation in MATLAB.

6 Testing

6.1 Run Times

Using the MATLAB profiler, the self times for each function can be obtained, as shown in Figure 23, below.

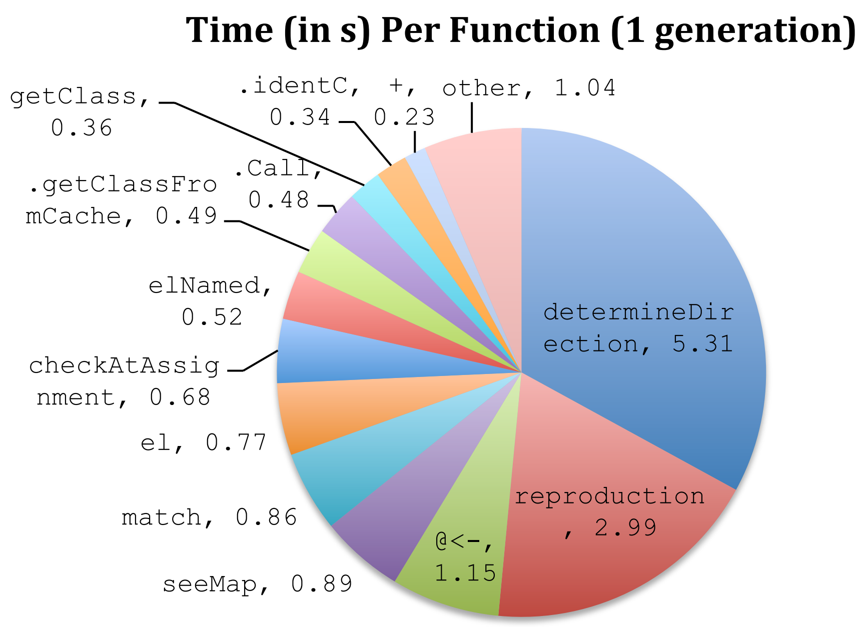


Figure 23: Self time, in seconds, per function

Determining direction takes the most amount of time per generation, but as this function is mostly if statements in order to match the agent movement algorithm, it is not likely that the time spent in this function can be reproduced. However, other functions, such as `checkAtAssignment`, `elNamed`, `el`, and match, are all internal R functions that are called when data elements in user-defined classes are accessed. It is possible that some of the time spent in these functions could be reduced by changing R defaults or no longer using a user-defined class, which reduce readability of code, but might increase speed. This will be explored in December and January before proceeding to Stage 2.

6.2 Spatial Metrics

Testing of the RMI, PDI, and MCI will be done using the gazelle data set and various simulation data sets. Additionally, sets of simulation data will be degraded by removing points based on selecting the index of points to be removed from a uniform distribution. The RMI, PDI, and MCI will be computed for both the original data set and the degraded data set, to compare performance of both metrics when the data is not perfect. The RMI, PDI, and MCI statistics for the combinations of landscape predictably and patch size will be examined to see if these statistics are sufficient for determining large-scale population patterns. If the clustering is not obvious from a per-metric basis, a linear classifier could be used. Time permitting, a parallel version will be implemented and computation times compared with the non-parallel version.

7 Project Schedule

The tentative schedule is described below, given in two-week increments. Currently, the project is on schedule and all tasks proposed have been completed.

Week	**Tasks**
October 21 – November 3	Set up data structures for agents ✓* Implement landscape initialization ✓ Implement landscape update ✓ Implement agent initialization ✓
November 4 - November 17	Implement agent direction ✓ Validate agent direction ✓
November 18 – December 1	Validate landscape initialization ✓ Implement reproduction ✓ Validate reproduction ✓
December 2 – December 16	**Connect implementations for full stage 0 + stage 1 ✓** Create mid-year report **Give mid-year presentation ✓**
January 20 – February 2	**Implement Stage 2**
February 3 – February 16	**Implement RMI**
February 17 – March 2	**Implement PDI** **Implement MCI**
March 3 – March 16	Validate PDI, RMI, and MCI
March 24 – April 6	Validate Stage 0, Stage 1, Stage 2
April 7 – April 20	Test RMI and PDI with gazelle data Test RMI and PDI with full simulation data Test RMI and PDI with degraded simulation data
April 21 – May 4	Write final report and create final presentation
May 5 – May 19	Complete final report and presentation **Give final presentation, submit final report**

Time permitting: parallel implementation of agent based model and/or spatial metrics and visualization of simulation.

8 Milestones

Milestones will coincide with the bolded elements in the project schedule.

9 Deliverables

Deliverables for this project are

1. Proposal presentation
2. Proposal document
3. Code for agent based model
4. Code for RMI, PDI, MCI
5. Data sets created by simulation (degraded and full)
6. Gazelle data sets
7. Mid-year report
8. Final report
9. Final presentation

10 Bibliography

1. Calabrese, Justin, Chris H. Fleming, Bill F. Fagan, Marin Rimmler, Petra Kaczensky, Peter Leimgruber, and Thomas Mueller. From the fish tank to the Gobi desert: A general, scalable approach to quantifying animal movement coordination. (Unpublished, 2013)
2. Cressie, Noel A. *Statistics for spatial data*. New York: Wiley, 1993
3. Kirkpatrick, David G., and Raimund Seidel. "The ultimate planar convex hull algorithm?." *SIAM journal on computing* 15.1 (1986): 287-299.
4. Lotwick, H. W., and B. W. Silverman. "Methods for analysing spatial processes of several types of points." *Journal of the Royal Statistical Society. Series B (Methodological)* (1982): 406-413.
5. Mueller, Thomas, William F. Fagan, and Volker Grimm. "Integrating individual search and navigation behaviors in mechanistic movement models."*Theoretical Ecology* 4.3 (2011): 341-355.
6. Mueller, Thomas, and William F. Fagan. "Search and navigation in dynamic environments–from individual behaviors to population distributions." *Oikos*117.5 (2008): 654-664.
7. Mueller, Thomas, et al. "How landscape dynamics link individual‐to population‐level movement patterns: a multispecies comparison of ungulate relocation data." Global Ecology and Biogeography 20.5 (2011): 683-694.