PATHWALKER: USER GUIDE

Siddharth Unnithan Kumar, Żaneta Kaszta and Samuel A. Cushman

siddharth.unnithankumar@gmail.com

1. INTRODUCTION

Pathwalker [1] is an individual-based model, designed to simulate movement paths across heterogeneous landscapes (parameterised by a resistance surface [2,3]), and to provide predictions of movement density and landscape connectivity. The model runs according to a set of parameters which govern the walker's movement (see section 3 for details), which include a variety of responses to landscape features at multiple spatial scales, and different types of directionality bias. Other than Pathwalker, there is at present no other model available for the ecological community which incorporates these mechanisms for movement. See [1] for full details on this model.

By producing simulated movement paths according to the specified parameters for movement, Pathwalker provides predictions of individual movement through a complex landscape. Moreover, the model can combine individual movement paths to produce a density of movement surface: this is a grid in the same format as the resistance surface, and which gives the density of movement through the landscape. As such, Pathwalker's density surface provides a prediction of landscape connectivity in the same format as other widely used toolks in connectivity analysis, such as resistant kernels [4] and circuit theory [5]. However, in contrast with these other connectivity models, Pathwalker's density surface is explicitly process-based, meaning that it provides a prediction of landscape connectivity based on actual mechanisms for movement, rather than proxies for movement such as least-cost paths and circuit theory.

2. HOW PATHWALKER WORKS

Pathwalker works by taking as input a resistance surface [2,3] in ascii format (the same format as raster layers, as used in ArcGIS), together with a collection of source coordinates (and optionally, destination coordinates), and performs a

random walk on the resistance surface starting at a source location. At each step, nine possibilities for movement are available: the neighbouring eight tiles together with a choice to remain stationary on the present tile. The movement of the walk is biased in response to the heterogeneous resistance surface, with the three mechanisms of energy, attraction and risk governing the basic movement, along with the possibility of incorporating further rules for movement. We outline these movement aspects below, but see [1] for more details if required.

With the energy, the random walk is unbiased, but the cumulative cost of movement across the resistance surface is measured, and the walk ends when a specified total cost has been reached. With attraction, the walk responds probabilistically to the resistance surface values, with movement biased towards areas of lower resistance. With risk, the walk is unbiased but has a chance of ending at each tile, with a higher chance of ending at tiles with higher risk values (which are either derived from the resistance surface or provided by a separate risk surface). These three mechanisms can be combined in any manner to produce more complex modes of movement.

The futher aspects of movement incorporated into Pathwalker are: (1) autocorrelation of movement, (2) response to resistance values at multiple spatial scales, and (3) bias towards target locations. For (1), the user chooses a parameter between 0 and 1 which determines the tendency for the walker to continue along its present direction. For (2), the resistance values of available tiles become functions of the resistance values of a square neighbourhood of the tiles; thus, the user specifies the neighbourhood size, together with the 'focal function' (which computes either the mean, maximum or minimum of the neighbourhood resistance values). For (3), the user specifies a collection of destination coordinates, together with a parameter between 0 and 1 which determines the tendency of the walker to move towards this target location. The sum of the correlation parameter and the destination bias parameter should be at most equal to 1.

3. SETTING UP PATHWALKER

Pathwalker is run from the command line, and requires Python 3, together with the packages *numpy* and *matplotlib*. Some computer platforms (such as Linux) come with Python 3 already installed, but others (such as Windows) will require

Python 3 to be installed by the user. There are plenty of online tutorials for how to install Python 3 and the packages *numpy* and *matplotlib*.

Once these have been installed, the Pathwalker program can be downloaded from *https://github.com/siddharth-unnithankumar/pathwalker*. Extract the *pathwalker* folder from the *pathwalker.zip* file to your preferred directory. The *pathwalker* folder contains the executable program *pathwalker.py* together with this user guide, and also a folder called *demo* which contains the data for a demo run of Pathwalker. You are now ready to configure the input data and then run Pathwalker!

4. INPUT AND OUTPUT

Create a folder inside the *pathwalker* directory, and name it according to your session name (just like the demo session is contained in a folder called *demo*, which is in the *pathwalker* directory). In this folder, three files are required: (1) a text file called *settings.txt* (2) a resistance surface in ascii format, and (3) a text file specifying the source point coordinates, under the same projection as the resistance surface. If you would like the model to run with destination points, and if these are to be different from the source points, then a fourth file is required: (4) a text file specifying the destination point coordinates, in the same projection as the resistance surface and source points. See the *demo* folder for an example of the correct file formats.

To set the model parameters, configure the settings file according to the options in the table below, in the same format as the *demo* settings file. If you have named your session *xx*, then a folder called *xx_results* will be created and will contain the outputs.

With all inputs configured, you are now ready to run Pathwalker. Open the command line, navigate to the *pathwalker* directory, and enter the text python pathwalker.py. You will then be prompted to enter your session name. Once you do this, the model will run according to the specified input data. Along with the outputs contained in the *_results* folder, the terminal will display the time taken for each stage of the program to run.

session_name	a chosen name for the session, the same as the title of the folder containing the settings.txt file

mechanism	1 = energy 2 = attraction 3 = risk 4 = energy + attraction 5 = energy + risk 6 = attraction + risk 7 = energy + attraction + risk
scale	an integer $n > 0$, where n-by-n is the window size for the scaling neighbourhood. the default is 1 (i.e. no spatial scaling)
scaling_function	1 = focal mean scaling (default) 2 = focal max scaling 3 = focal min scaling
correlation	a number between 0 (default) and 1 the sum of the correlation and the destination_bias should be at most equal to 1
destination_bias	a number between 0 (default) and 1 the sum of the correlation and the destination_bias should be at most equal to 1
steps	an integer greater than 0 (default 1000)
energy	an integer greater than 0 (default 1000)
resistance_surface	the name of the resistance surface file, including the file extension. this file should be placed in your folder titled with the session name. for example, in the *demo* folder we have 'res_surface.asc'
risk_surface	the choice of risk surface. if either no risk surface is required, or you would like to use the default risk surface (which is the resistance surface first rescaled to take values between 1 and 100, and then divided by 1000), then the input is the number 0. otherwise, the input is the name of the risk surface (with file extension), which should be in the same format as the resistance surface, and placed in the session folder
source_points	the name (with file extension) of the file containing the source coordinates, which must be a .txt file, and placed in the session directory. see the file 'source_coords.txt' in the *demo* folder for an example of the correct format for the coordinates
destination_points	if no destination points are required, this input is 0 (default). otherwise, the input is the name (with file extension) of the file containing the destination coordinates, which must be a .txt file, and placed in the session directory. see the file

	'dest_coords.txt' in the *demo* folder for an example of the correct format for the coordinates
source_destination_pairing	0 = no destination points are provided in the input (default) 1 = paths for all source-destination pairs will be computed. so if there are *k* source points and *m* destination points, there will be *km* paths 2 = if there are the same number of source and destination points, then this will pair up the source and destination points (in the order given in the source and destination .txt files), and compute the path for each pair. so if there are *k* source points and *k* destination points, there will be *k* paths 3 = if the destination points are the same as the source points, then this computes paths for all source-destination pairs, but excludes the pairing of a point with itself. so if there are *k* locations, used as both source and destination points, there will be *k* (*k* - 1) paths
runs	the number of times to run the simulation for each source point, default is 1. if destination points are included, then this will be the number of times the simulation is run for each source-destination pair
output	0 = no output data provided 1 = a matrix .csv file, with each pair of columns giving the x- and y-coordinates for each path. thus, if there are *k* total paths, this will be a matrix with 2*k* columns. See below the table for how these columns are ordered 2 = a matrix .csv file, with each column providing the 'additional information' for each path. See below this table for details of the additional information 3 = a point density surface, and its log transform in a separate file, calculated from a cumulative sum of the paths from all runs. both are provided in the same ascii format as the original resistance surface. the pixel values of the log density surface is given by ln(x+1), where x is a pixel value on the original density surface 12 = both 1 and 2, so that for each run, the output is a .csv file containing the xy coordinates of all paths and also a .csv file containing the additional information for each path 13 = both 1 and 3, so that the output is a .csv file containing the xy coordinates of all paths, and also the two density surfaces in ascii format

	23 = both 2 and 3, so that the output is a .csv file with the additional information for each path, and also the two density surfaces in ascii format 123 = all three together: coordinates, information, and the two density surfaces
figures	0 = no figures provided 1 = figures provided for each run 2 = figure provided of the two density surfaces 12 = figures provided of each run, together with figures of the two density surfaces figure size and colour can be altered by modifying the code for the output figures in pathwalker.py

In the output data, the paths are ordered by source point first, then destination point, then run number. For example, the 2nd run computed from the 4th source point and 1st destination point will be labeled *path_4_1_2*. Thus if you have chosen to output the x- and y-coordinates for the individual paths, then the coordinates for this path will be given by the columns with headers *path_4_1_2_x* and *path_4_1_2_y*.

If you have chosen to output additional information for each path, then you will obtain a file with up to three rows of output data. From top to bottom, these rows provide: (1) the total number of steps taken in the walk, (2) the total energy accumulated from the walk, and (3) the total risk accumulated from the walk. The first row (total number of steps) is always provided, and the remaining two rows will be provided depending on the choice of mechanism specified in the *settings.txt* file. For example, if mechanism 6 (corresponding to 'attraction' and 'risk') is chosen, then 'energy' does not feature in this setup and so the output will contain two rows of data, the first corresponding to the total number of steps and the second to the total accumulated risk. The *demo* session is configured to run mechanism 7 ('energy', 'attraction' and 'risk' together), and so there are three rows of output data, in the order specified above.

5. TROUBLESHOOTING

- Make sure you have installed the *numpy* and *matplotlib* packages. If you are using a virtual environment manager (such as *anaconda* or *miniconda*), make sure to activate the environment which has these packages installed.

- Be sure to have the coordinates of your source (and, if included, destination) points within the bounds of the region covered by the resistance surface, else you may encounter the error '*index ... is out of bounds ...*'
- There are two important considerations when increasing the window size for spatial scaling. First, the computation will slow down dramatically as the window size is increased. Second, when the computation is running, the model creates a boundary of infinite resistance around the edge of the resistance surface - thus, for example, if a source point is located 1 pixel away from the boundary of the resistance surface, and a window size of 11-by-11 is used with either the focal mean or focal max functions, then the walk ends immediately since all 9 possible tiles will have infinite resistance value.
- If you encounter the error *File exists: 'xx_results'*, this is because you already have a folder titled *xx_results*, where *xx* is your session name.

6. ACKNOWLEDGEMENTS

7. REFERENCES

1. Unnithan Kumar et al 2021
2. Zeller, Katherine A., Kevin McGarigal, and Andrew R. Whiteley. "Estimating landscape resistance to movement: a review." *Landscape ecology* 27.6 (2012): 777-797.
3. Cushman, Samuel A., et al. "Biological corridors and connectivity [Chapter 21]." *In: Macdonald, DW; Willis, KJ, eds. Key Topics in Conservation Biology 2. Hoboken, NJ: Wiley-Blackwell. p. 384-404.* (2013): 384-404.
4. Compton, Bradley W., et al. "A resistant-kernel model of connectivity for amphibians that breed in vernal pools." *Conservation Biology* 21.3 (2007): 788-799.
5. McRae, Brad H., et al. "Using circuit theory to model connectivity in ecology, evolution, and conservation." *Ecology* 89.10 (2008): 2712-2724.

8. FIGURES

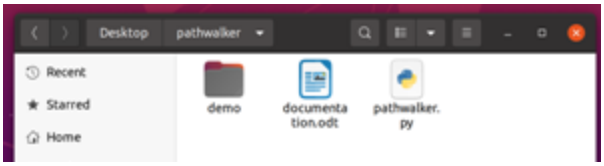

Figure 1: The contents of the pathwalker directory.

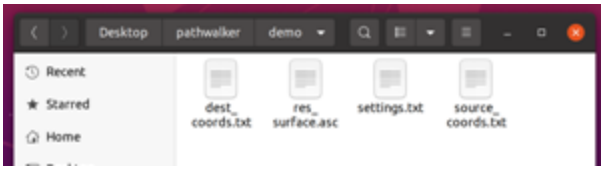

Figure 2: The contents of the test folder 'demo': (1) the settings file, (2) the resistance surface, (3) source coordinates, (4, optionally) destination coordinates.

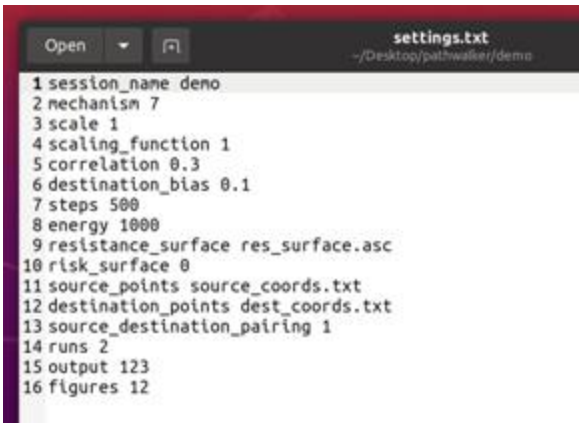

Figure 3: The format of the example settings file.

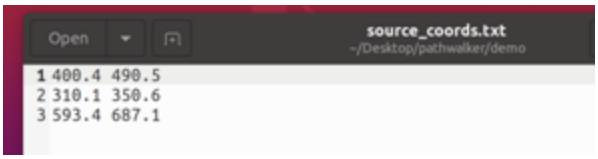

Figure 4: The format of the example source coordinates file.

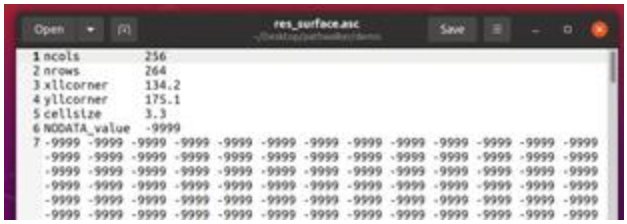

Figure 5: The format of the example resistance surface.

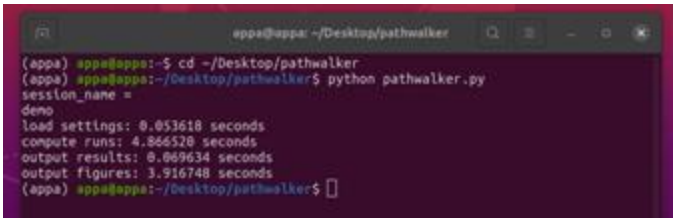

Figure 6: Running pathwalker in the terminal.

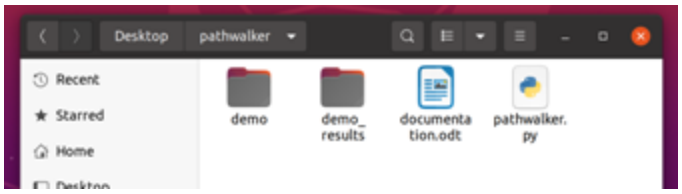

Figure 7: The results folder, 'demo_results', has been created.

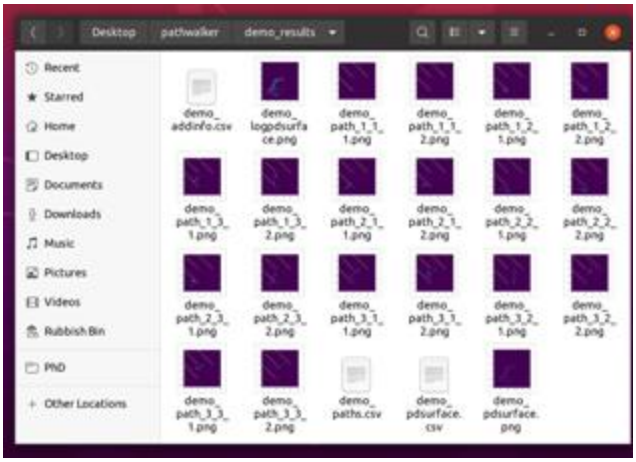

Figure 8: The contents of the results folder, with the choice of outputs have been specified in the settings file.