# UNICOR

# USER MANUAL

2011

Version: 1.0

Authors:

**E. L. Landguth[1], B. K. Hand[1], J. M. Glassy[1,2], S. A. Cushman[3] and R. T. Carlson[1]**

1 - University of Montana, Division of Biological Sciences, Missoula, MT, 59812, USA.

2 - Lupine Logic Inc, Missoula, MT, 59802, USA.

3 - U.S. Forest Service, Rocky Mountain Research Station, 2500 S. Pine Knoll Dr., Flagstaff, AZ 86001, USA

**Table of Contents**

# 1 Introduction

Habitat loss and its effects on populations of vulnerable species is among the most urgent problems in conservation ecology. It is critical that managers and scientists have effective tools to evaluate the effects of landuse and climate change on the extent and connectivity of populations. To address this need, we introduce UNIversal CORridor network simulator (UNICOR), a species connectivity and corridor identification tool. UNICOR applies Dijkstra's shortest path algorithm to individual-based simulations. Outputs can be used to designate movement corridors, identify isolated populations, quantify effects of climate and management changes on population connectivity and prioritize conservation plans to maintain population connectivity. The key features include a driver-module framework, connectivity mapping with thresholding and buffering, and calculation of graph theory metrics. Through parallel-processing, computational efficiency is greatly improved, allowing analyses of large extents (grid dimensions of thousands) and large populations (individuals in the thousands).

## 1.1 What can UNICOR do

UNICOR is intended for use by land managers as well as the research community and will be a valuable tool in applied conservation biology. It provides new functionality to increase understanding of species connectivity in current and future landscapes. This, in turn, provides invaluable ability to quantitatively compare spatially explicit conservation and restoration scenarios and prioritize actions to have the largest cumulative effects on population connectivity. The results can be used to designate sites as potential source or sink populations, and identify corridors and barriers. Simulations could address prioritizing areas of greatest concern, effects of climate change on wildlife populations, or habitat fragmentation under future climate or landuse change.

## 1.2 How does UNICOR work

UNICOR simulator uses a modified Dijkstra's algorithm (Dijkstra 1959) to solve the single source shortest path problem from every specified species location on a landscape to every other specified species location. Figure 1 provides a step-by-step conceptual workflow. UNICOR requires two input files as the first step: 1) a landscape resistance surface and 2) point locations for each population or individual's location (see section 3 for program input). Prior to running UNICOR, users must create a resistance surface where each cell value (pixel) represents the unit cost of crossing each location. Pixels are given weights or 'resistance values' reflecting the presumed influence of each variable to movement or connectivity of the species in question (e.g., Dunning *et al*. 1992; Cushman *et al.* 2006; Spear *et al.* 2010). Resistance surfaces could be parameterized to reflect different costs to movement associated with vegetation types,

elevation, slope, or other landscape features.

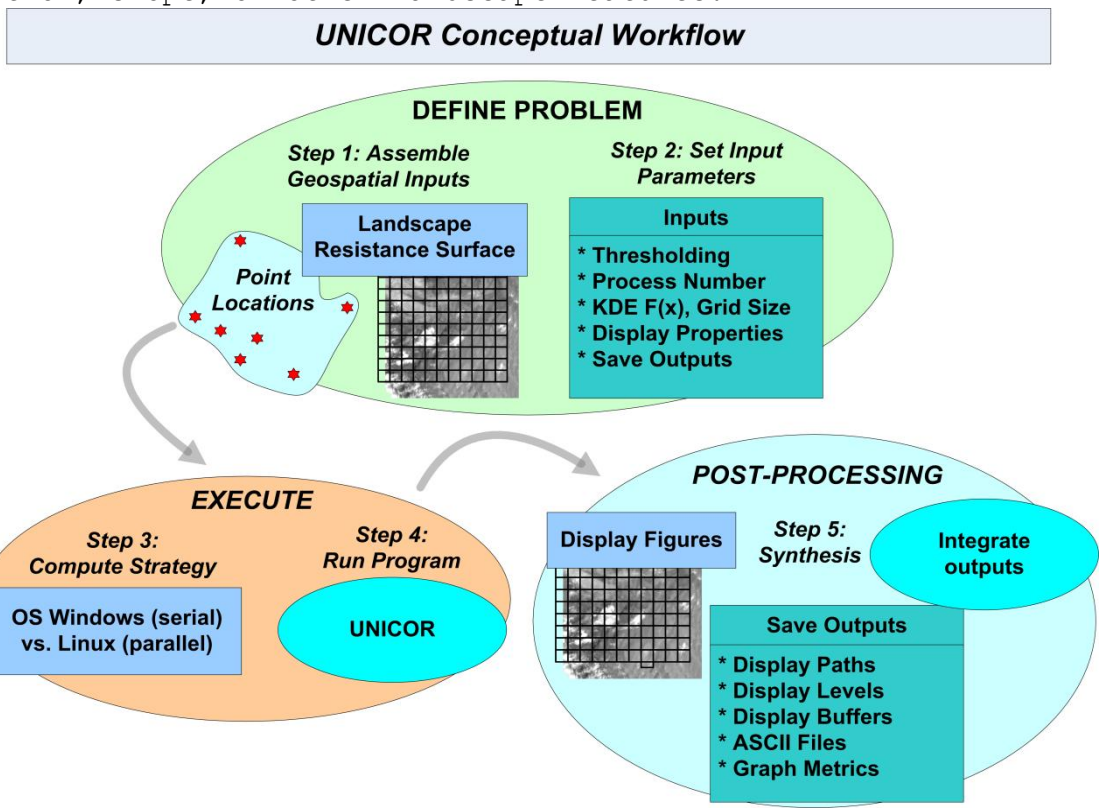


Figure 1: UNICOR conceptual workflow diagram. Steps 1 and 2 define the inputs and problem. Steps 3 and 4 execute the program. Step 5 provides synthesis and post-processing.

Point locations define starting and ending nodes of paths connecting pairs of individuals. The points must be referenced on the landscape resistance surface, with any user specified placement pattern (e.g., uniform, random, or placement in habitat suitability) and density. From graph theory and network analysis, we can then represent the landscape resistance surface as a graph with nodes and edges (Diestel 2010). Every pixel is considered to be a node. The graph edges, which represent possible movement paths between each node, are weighted by the resistance value of the cell times the distance to the next pixel center, times the distance to the next pixel center, which gives the total edge length in terms of raster cell units (cost distance). Dijkstra's algorithm is modified in the UNICOR code to find all shortest paths to all destination nodes associated with the same starting node. This provides a substantial boost in computational efficiency where all pairwise combinations are found for the same starting node before clearing the search space from memory. All paths found are optimal paths of movement computed for every paired combination of starting and ending nodes. The combination of these shortest paths create a path density map which is also a connectivity graph.

In essence, this approach becomes a large graph problem for the

applied landscape connectivity assessments. In analyses involving large numbers of individuals across a large and fine-grained environment computational processing time becomes intractable. However, parallel processing allows for efficient use of increasingly ubiquitous, modern multi-core processors. Dijkstra's breadth-first search algorithm is ideal for running in parallel for sets of source and destination points because pairwise distances can be calculated independently. We have implemented parallel processing in UNICOR using the multiprocessing module from Python version 2.6. Parallel processing in UNICOR is currently only implemented under the Linux operating system.

To reflect species-specific differences in dispersal abilities, users can specify connectivity thresholds. These connectivity thresholds are expressed as the maximum path length for a species given its dispersal ability. This enables UNICOR to realistically reflect the biological dispersal abilities of a particular species. Users can specify the maximum dispersal distance based either on cumulative cost distance or Euclidean distance.

Dijkstra's base algorithm assumes the optimal is followed by all individuals. However, this is unlikely to realistically represent the behavior of organisms. Thus, it is beneficial to consider either multiple low-cost paths, or to smooth output paths using a probability density function such as a Gaussian bell curve (Cushman et al. 2008; Pinto and Keitt 2008). UNICOR implements the latter and allows for the application of a variety of smoothing functions referred to as kernel density functions: Gaussian, Epanechnikov, uniform, triangle, biweight, triweight, and cosine functions can be used for the kernel density estimations (Li & Racine 2007). The outputs that are produced by the program show the cumulative density of optimal paths buffered by kernel density estimations (see Silverman 1986; Scott 1992) following a distribution around frequency of common connections.

Through batch capability, users may specify alternative connectivity thresholds to assess how scale dependency of dispersal ability will be affected by landscape change and fragmentation under a range of scenarios (e.g. Cushman *et al*. 2010a; Watts *et al*. 2010). Outputs include paths among habitat patches that can be used to display expected species movement routes and can provide managers with visual guidance on identifying corridors that are likely critical for maintaining network connectivity. Quantification of changes to habitat fragmentation, and corridor connectivity is enabled through outputs of graph theory metrics (e.g., density, number of nodes, radius, etc.) (Hagberg *et al.* 2008) and connectivity outputs that can directly input into popular landscape pattern analysis programs (e.g., FRAGSTATS (McGarigal *et al.* 2002)).

The program is written in Python 2.6. UNICOR is built on a driver-module, plug-in, docking architecture that allows for ease of future modular development. The program's input parameters are organized as name-value-pairs in a stanza oriented, text file format. The inputs

are parsed using the RipMgr package, a flexible symbol table manager for science models that includes special parsing capabilities (Glassy, 2010). UNICOR has been debugged as carefully as possible by testing all combination of simulation options. The program is freeware and can be downloaded at http://cel.dbs.umt.edu/software/UNICOR/.

# 2 Getting started

## 2.1 Dependencies

### 2.1.1 Baseline Requirements

UNICOR requires the Python2.6.x interpreter, NumPy package, and SciPy package. Several optional Python module packages, if enabled, facilitate additional UNICOR functionality. Remember that Python modules usually require particular Python interpreters, so be sure the version ID for any external Python module or package (e.g. NumPy or others) matches the version of your Python interpreter (normally v2.6.x).

### 2.1.2 Python on Non-Windows Platforms

Some common computer platforms come with Python installed. These include MAC OS X and most Linux distributions. To determine which Python a MAC or Linux workstation has installed, start a terminal console and enter "python." You'll see the version number on the top line (enter Control-D to exit). Replacing an older Python interpreter (pre v2.4) with a newer one (v.2.6.x) on a Linux or MAC OS X machine can be tricky, so ask a System Administrator for help if you're not sure which packages depend on the current Python installed.

### 2.1.3 Python on Windows

Windows (7, XP, 2000, Server) does not come with Python installed, so follow the instructions below to obtain and install Python on a computer running the Windows operating system. Get a windows installation of the base Python installation (current v.2.6.x) at: http://www.python.org/download/releases/.

### 2.1.4 Obtaining NumPy and SciPy

We recommend using the superpack Windows installer available from the SourceForge website: http://sourceforge.net/project/. Note that more complete information for NumPy is available at www.scipy.org, where the SciPy module is also presented. Another source is http://www.enthought.com/products/epd.php for a free academic and educational usage in a single downloadable installer that has everything and then some (Numpy, Scipy, Matplotlib, and 70+ modules for python).

## 2.2 Installation

### 2.2.1 Install Python, NumPy, and SciPy

Make sure that Python and NumPy are installed, and available to you. You can test this by typing "python" at a command window. If python is available you'll get the python prompt ">>>". If it is not a

recognized command, it means either that python is installed but is not in your command shell's paths, or that python is not installed. In the first case ask an administrator to add it to your command paths. If your shell locates and loads python, type, "import numpy". Similarly, type, "import scipy". If python does not complain that there are no such modules, all is well.

The following instructions assume Python, NumPy, and SciPy are not yet available on your computer; if they are, skip to section 2.2.2.

* First run the Python executable installer you've chosen (either from www.python.org or ActiveState, accepting defaults for the installation directory. On Windows this will typically place the executables and libraries in c:/Python2.6/bin and the "site-packages" package tree for user installed Python modules in c:/Python2.6/lib/site-packages. If you are installing it on a network on which you do not have administrative privileges, you may need to ask a system administrator to install python and the NumPy and SciPy packages in their default locations.

* Next install NumPy and SciPy using the supplied executable (superpack) installer or visiting http://www.scipy.org/Download. This will install NumPy and SciPy in your Python ./site-packages directory.

### 2.2.2 Unpack the UNICOR Archive

Navigate to the directory on your PC where you wish to install UNICOR, and unpack the supplied zip archive file using a free archive tool like 7Zip (7z.exe), Pkunzip, Unzip, or an equivalent. Seven-Zip (7Z.exe) is highly recommended since it can handle all common formats on Windows, MAC OS X and Linux. On Windows, it is best to setup a project specific modeling subdirectory to perform your modeling outside of any folder that has spaces in its name (like "My Documents").

### 2.2.3 Install UNICOR

Next, install the UNICOR software itself by unpacking the zip archive supplied. At this point you should be able to execute the supplied test inputs.

### 2.2.4 Optional Python Extension Modules

As UNICOR is supplied in the archive, it does not require any additional contributed Python modules to run. However, several additional Python modules are needed if you want the following functionality:

NetworkX is required for graph theory metrics and can be obtained from http://networkx.lanl.gov/.

wxPython is required to run the GUI and can be obtained from http://www.wxpython.org/.

## 2.3 Example run

### 2.3.1 Command line run

The example run is for 10 points representing individuals on a Euclidean distance resistance surface. To run the following example, follow these steps:

1. Double check the UNICOR source, UNICOR additional packages, and UNICOR example files are in the same directory.

2. The included .rip file specifies the parameters that can be changed and used in the sample UNICOR run. Open example.rip in your editor of choice (e.g., notepad or wordpad).

3. This file is the stanza format following RipMgr documentation. All '#' signs are comments followed by variable names with a tab to the parameter specified. The parameter can be changed for running UNICOR, but downloaded parameters will run as is. See section 3 for more details on each parameter along with its dependency.

4. Start the program with a graphical interface or at the command line: For example, if you downloaded Python 2.6.x from www.python.org, then you are provided with a graphical interface, IDLE. In Windows you can find IDLE from your Start menu > All Programs > Python 2.6 > IDLE (Python GUI). Alternatively, if you use Python from the command line, then open a terminal window and change your shell directory to the UNICOR home directory.

5. Run the program: There are a number of ways to run this program. For example, if you are using a command shell you can run the program by typing "python UNICOR.py example.rip".

6. Check for successful simulation run completion: The program will provide a log file in your UNICOR home directory. Once completed, output files will be created in UNICOR home directory.

### 2.3.2 GUI Run

The following are instructions for a simulation run with an optional graphical user interface (GUI). Note that this GUI has a dependency on python library, WX python. Go to http://wxpython.org/download.php and download your OS's version of WX python.

1. Navigate to UNICOR folder and double click unicorGUI.py.

2. Enter in values for each variable. To find out more information on a specific variable, click the radio button to the left of the variable label.

3. After all values are entered, click the Submit button. If a value is in the wrong format, you will be notified at the bottom.

4. The program is running successfully if the command prompt opens up and displays text related to running UNICOR.

# 3 Input

See Table 1 for UNICOR inputs and outputs.

## 3.1 Resistance grid

Prior to running UNICOR, users must create a resistance surface where each cell value (pixel) represents the unit cost of crossing each location. Pixels are given weights or 'resistance values' reflecting the presumed influence of each variable to movement or connectivity of the species in question. Resistance surfaces could be parameterized to reflect different costs to movement associated with vegetation types, elevation, slope, or other landscape. The filename for the resistance surface must be in an ascii format with header file (any file extension is acceptable, must be space delimited). The example simulation runs are small_test.rsg and medium_test.rsg.

## 3.2 XY locations

Point locations define starting and ending nodes of paths connecting pairs of individuals. The points must be referenced on the landscape resistance surface, with any user specified placement pattern (e.g., uniform, random, or placement in habitat suitability) and density. The filename for the individuals with (x,y) locations can have any file extension, but must be comma delimited. The example simulation runs are small_test_10pts.xy and medium_test_100pts.xy.

## 3.3 Thresholding

To reflect species-specific differences in dispersal abilities, users can specify connectivity thresholds. These connectivity thresholds are expressed as the maximum path length for a species given its dispersal ability. This enables UNICOR to realistically reflect the biological dispersal abilities of a particular species. Users can specify the maximum dispersal distance based either on cumulative cost distance or Euclidean distance. To get all path lengths set the 'Edge Distance' to 'max'.

## 3.4 Number of processors

In essence, this approach becomes a large graph problem for the conservation biology problems faced today. In analyses involving large numbers of individuals across a large and fine-grained environment computational time becomes intractable. However, parallel processing allows for efficient use of increasingly ubiquitous, modern multi-core processors. Dijkstra's breadth-first search algorithm is ideal for running in parallel for sets of source and destination points because pairwise distances can be calculated independently. We have implemented parallel processing in UNICOR using the multiprocessing module from Python version 2.6, and is currently only available in the Linux operating system. For parallel computing, specify the number of processors that are used in a simulation.

## 3.5 Kernel density estimation

Dijkstra's base algorithm assumes the optimal is followed by all

individuals. However, this is unlikely to realistically represent the behavior of organisms. Thus, it is beneficial to consider either multiple low-cost paths, or to smooth output paths using a probability density function such as a Gaussian bell. UNICOR implements the latter and allows for the application of a variety of smoothing functions referred to as kernel density functions: Gaussian, Epanechnikov, uniform, triangle, biweight, triweight, and cosine functions can be used for the kernel density. The outputs that are produced by the program show the cumulative density of optimal paths buffered by kernel density estimations following a distribution around frequency of common connections.

## 4 Output

Outputs include paths among habitat patches that can be used to display expected species movement routes and can provide managers with visual guidance on identifying corridors that are likely critical for maintaining network connectivity. Quantification of changes to habitat area, fragmentation, and corridor connectivity is enabled through outputs of graph theory metrics (e.g., density, number of nodes, radius, etc.) and connectivity outputs that can directly input into popular landscape pattern analysis programs (e.g., FRAGSTATS (McGarigal *et al.* 2002)).

Table 1: UNICOR inputs and outputs with description and dependencies.

| Input Name | Default/ Example Input | Description | Dependency |
|---|---|---|---|
| Grid Filename | small_test.rsg | The filename for the resistance surface in ascii format with header file (any file extension is acceptable, must be space delimited). | |
| XY Filename | small_test_10pts.xy | The filename for the individuals with (x,y) locations (any file extension is acceptable, must be comma delimited). | |
| Use ED Threshold | False | Option for using Euclidean distance thresholding. | |
| ED Distance | 50000 | If Use ED threshold is True, then the Euclidean distance in map units to apply to the (x.y) point locations | |
| Edge Distance | max | The resistance distance threshold in terms of edge distance to apply to the path lengths. | |
| Number of Processors | 8 | For parallel computing, the number of processors that are used in a simulation. | Linux |
| KDE Function | Gaussian | The probability distribution used to calculate the kernel density buffer [Gaussian, Epanechnikov, Uniform, Triangle, Biweight, Triweight, Cosine]. | SciPy |
| KDE GridSize | 2 | The kernel buffer window used to calculate the buffered maps. | SciPy |
| Number of Levels | 3 | The number of categories used to display the kernel density buffer map. | SciPy |

| Output | Default/ Example Input | Description | Dependency |
|---|---|---|---|
| Save Path Output | TRUE | The surface of paths in ascii format with header file (space delimited – .addpaths). | |
| Save Individual Paths Output | TRUE | The list of individual path values and length from every point to every other point (comma delimited - .paths). | |
| Save Cost Distance Matrix Output | TRUE | The resistance distance matrix of all the source-destination connection lengths (comma delimited - .cdmatrix). | |
| Save KDE Output | TRUE | The surface of kernel buffered paths in ascii format with header file (space delimited – .kdepaths). | |
| Save Levels Output | TRUE | The categorical surface created from the KDE output in ascii format with header file (space delimited - .levels) | |
| Save Graph Metrics Output | FALSE | Path graph theory metrics (comma delimited - .graphmetrics) | NetworkX |

# 5 General issues

## 5.1 How to obtain UNICOR

The program is freeware and can be downloaded at http://cel.dbs.umt.edu/software/UNICOR/ with information for users, including manual instructions, FAQ, publications, ongoing research, and developer involvement.

## 5.2 Debugging and troubleshooting

For help with installation problems please check first for postings at our web site. Otherwise, please report problems including any bugs, to me at erin.landguth@mso.umt.edu.

## 5.3 UNICOR limitations

The following is a list of the current (as we know of) limitations with UNICOR:

1. The resistance surface is in ASCII format: header file with 6 lines of information and space delimited.
2. The point locations have a header row and file is comma delimited.
3. Point locations must fall inside the resistance grid extent. The code will run when points lie outside of grid, but no paths will be calculated.

## 5.4 How to cite UNICOR

This program was developed by Erin Landguth, Brian Hand, and Joe Glassy. GUI development was done by Mike Jacobi. Ross Carlson assisted with graphics, data set, and website creation. The reference to cite is:

Landguth EL, Hand BK, Glassy JM, Cushman SA, Carlson RT (2011) UNICOR: A species connectivity and corridor network simulator. Ecography. Submitted.

### 5.5 Disclaimer

The software is in the public domain, and the recipient may not assert any proprietary rights thereto nor represent it to anyone as other than a University of Montana-produced program (version 1.x). UNICOR is provided "as is" without warranty of any kind, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The user assumes all responsibility for the the accuracy and suitability of this program for a specific application. In no event will the authors or the University be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages arising from the use of or the inability to use this program.

We strongly urge you to read the entire documentation before ever running UNICOR. We wish to remind users that we are not in the commercial software marketing business. We are scientists who recognized the need for a tool like UNICOR to assist us in our research on landscape ecology issues. Therefore, we do not wish to spend a great deal of time consulting on trivial matters concerning the use of UNICOR. However, we do recognize an obligation to provide some level of information support. Of course, we welcome and encourage your criticisms and suggestions about the program at all times. We will welcome questions about how to run UNICOR or interpret the output only after you have read the entire documentation. This is only fair and will eliminate many trivial questions. Finally, we are always interested in learning about how others have applied UNICOR in ecological investigation and management application. Therefore, we encourage you to contact us and describe your application after using UNICOR.

*We hope that UNICOR is of great assistance in your work and we look forward to hearing about your applications.*

## 6 References

Bunn,A.G., Urban,D.L. and Keitt T.H. (2000) Landscape connectivity: A conservation application of graph theory. *Journal of Environmental Management,* 59, 265-278.

Compton,B., et al. (2007) A resistant kernel model of connectivity for vernal pool breeding amphibians*. Conservation Biology*, 21, 788–799*.*

Cushman,S.A., et al. (2006) Gene flow in complex landscapes: testing multiple hypotheses with casual modeling. *The American Naturalist*, 168, 486-499.

Cushman,SA, McKelvey,K.S. and Schwartz,M.K. (2009) Use of empirically derived source-destination models to map regional conservation corridors. *Conservation Biology*, 23, 368-376.

Cushman,S.A., Chase,M.J. and Griffin,C. (2010a) Mapping landscape resistance to identify corridors and barriers for elephant movement in southern Africa. In Cushman,S.A. and Huettman,F. (eds). *Spatial complexity, informatics and wildlife conservation*, Springer, Tokyo, pp. 349-368.

Cushman,S.A., Compton,B.W. and McGarigal,K. (2010b) Habitat fragmentation effects depend on complex interactions between population size and dispersal ability: Modeling influences of roads, agriculture and residential development across a range of life-history characteristics. In Cushman,S.A. and Huettman,F. (eds). *Spatial complexity, informatics and wildlife conservation*, Springer, Tokyo, pp. 369-387.

Dale,V.H., et al. (2001) Climate change and forest disturbances. *BioScience*, 51, 723-734.

Diestel,R. (2010) *Graph Theory*, Springer-Verlag, Heidelberg, Fourth Edition.

Dijkstra,E.W. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269–271.

Dunning,J.B., Danielson,B.J. and Pulliam,H.R. (1992) Ecological processes that affect populations in complex landscapes. *OIKOS*, 65, 169 -175.

Fall,A., et al. (2007) Spatial graphs: principles and applications for habitat connectivity. *Ecosystems*, 10, 448-461.

FAO (2006) Global Forest Resources Assessment 2005, Main report. *Progress towards sustainable forest management*, FAO Forestry Paper 147, Rome, p 320.

Hagberg,AA., et al. (2008) Exploring network structure, dynamics, and function using NetworkX, In Varoquaux,G., et al. (eds) *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, pp. 11-15.

McGarigal,K., et al. (2002) FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. Available at the following web site: http://www.umass.edu/landeco/research/fragstats/fragstats.html

McRae,B.H. and Beier,P. (2007) Circuit theory predicts gene flow in plant and animal populations. *Proceedings of the National Academy of Science USA*, 104, 19885-19890.

McRae,B.H., et al. (2008) A multi-model framework for simulating wildlife population response to land-use and climate change. *Ecological Modelling*, 219, 77-91.

Li,Q. and Racine,J.S. (2007) *Nonparametric Econometrics: Theory and Practice*. Princeton University Press.

Opdam,P. and Wascher,D. (2003) Climate change meets habitat fragmentation: linking landscape and biogeographical scale levels in research and conservation. *Biological Conservation*, 117, 285-297.

Pinto,N. and Keitt,T.H. (2009) Beyond the least cost path: evaluating corridor robustness using a graph-theoretic approach. *Landscape Ecology*, 24, 253-266.

Riitters,K. et al. (2000) Global scale patterns of forest fragmentation. *Conservation Ecology*, 4, [online] URL:

http://www.consecol.org/vol4/iss2/art3/

Sawyer,H., et al. (2009) Identifying and prioritizing ungulate migration routes for landscape-level conservation. *Ecological Applictions*, 19, 2016-2025.

Scott,D.W. (1992) Chapter 6. In: *Multivariate Density Estimation; Theory, Practice, and Visualization*. John Wiley and Sons, New York.

Schwartz,M.K., et al. (2009) Wolverine gene flow across a narrow climatic niche. *Ecology*, 90, 3222-3232.

Silverman,B.W. (1986) Chapter 3. In: *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, New York.

Spear,S.F., et al. (2010) Use of resistance surfaces for landscape genetic studies: Considerations for parameterization and analysis. *Molecular Ecology*, in press.

Urban,D. and Keitt,T. (2001) Landscape connectivity: A graph-theoretic perspective. *Ecology*, 82, 1205-1218.

Watts,K., et al. (2010) Targeting and evaluating biodiversity conservation action within fragmented landscapes: an approach based on generic focal species and least-cost networks. *Landscape Ecology*, 25, 1305-1318.

## 7 Acknowledgements

This research was supported in part by funds provided by the Rocky Mountain Research Station, Forest Service, U.S. Department of Agriculture and by the National Science Foundation grant #DGE-0504628.