**CDPOP**

**USER MANUAL**

2016

Version: 1.2.31

Last Updated: 2016.05.25ell

Contributors:

**E. L. Landguth1,**

**J. M. Glassy1,2,S. A. Cushman3, M. Jacobi1, T. J. Julian1, B. K. Hand1, Allen Warren1, Sam Banks**

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

4 – National Australian University, Canberra, AU.

**Table of Contents**

1. Introduction.................................................. 3

1.1 Changes from CDPOP v0.7............................... 3

* 1. What can CDPOP do..................................... 3
  2. How does CDPOP work................................... 4

1. Getting started............................................... 7

2.1 Dependencies..........................................

2.1.1 Baseline requirements............................

2.1.2 Python on non-windows platforms..................

2.1.3 Python on windows................................

2.1.4 Obtaining NumPy and SciPy........................

* 1. Installation..........................................

2.2.1 Installing Python, NumPy, and SciPy..............

2.2.2 Installing CDPOP.................................

2.2.3 Description of CDPOP files......................

* 1. Example run...........................................

2.3.1 Command line run.................................

1. Input......................................................... 12
2. Output........................................................ 28
3. General issues................................................ 31

5.1 How to obtain CDPOP...................................

* 1. Debugging and troubleshooting.........................
  2. How to cite CDPOP.....................................
  3. Disclaimer............................................

1. References.................................................... 32
2. Acknowledgements.............................................. 32
3. **Introduction**

The goal of this user manual is to explain the technical aspects of the current release of the CDPOP program. CDPOP v1.2 is a major extension of the CDPOP program (Landguth and Cushman 2010). CDPOP is an individual-based program that simulates the influences of landscape structure on emergence of spatial patterns in population genetic data as functions of individual-based movement, breeding, and dispersal.

* 1. **Changes from CDPOP v0.7**

There are major innovations in v1.2 which were not included in the previously published v0.70. We list below the new functionalities of CDPOP v1.2:

* **Natural selection** is implemented through differential offspring viability as functions of fitness landscapes.
* Gene flow and natural selection can now be simulated in **dynamic landscapes**.
* A **graphical user interface** provides a user friendly platform that enables users to explore, analyze, and model the effects of life-history and differential models of complex landscapes on the genetic structure of populations.
* **Demographics** allows for fluctuating population sizes.
* Sex-specific dispersal.
* Changes of some internal software components have allowed an overall speed increase and to improve program stability.
* Additional movement function option: negative exponential movement.
* Inclusion of both a mating landscape and a dispersal landscape.
* Vertical transmission of an infection with giving probability.
* mtDNA option.
* Output genotype option in multiple formats: genalex, structure, genepop, general.
* Generation can be specified to change landscape surface for both mating and dispersal, e.g., climate change scenarios.
* Subpopulation differential mortality. An individual can be given an increased mortality if it disperses to another subpopulation.
* Philopatry option added for both male and females.
* Multiple paternity option added, i.e., females can have litters from multiple males but birth rate is same for each female.
* Additional mutation models added: step-wise forward and backward.
* Age-structured mortality added for overlapping generations.
* Burn-in procedure options
  1. **What can CDPOP do**

CDPOP’s realistic representation of the spatial environment and population genetic processes provide a powerful framework to investigate the impact of ecological factors on the genetic structure of populations. This approach has already advanced knowledge of the patterns of genetic variation in spatially-explicit contexts (Landguth et al 2010a; Landguth et al 2010b; Cushman and Landguth 2010; Landguth and Balkenhol 2012). Example simulations have included:

* Quantifying the time to detect barriers.
* Correlating migration rates and landscape resistance barriers.
* Testing for the effects of population sample size and number of markers.
* Assessing relative influence of adaptive versus neutral markers in detection of population genetic differentiation.
  1. **How does CDPOP work**

CDPOP v1.2 models genetic exchange for a given resistance surface and *n – (x, y)* located individuals as functions of individual-based movement through mating and dispersal, vital dynamics, and mutation. A user must specify the input parameters through a graphical user interface or input script file. As the model simulates stochastic processes, most applications will quantify mean and variability of genetic structure across many runs. Thus, a Monte Carlo option is provided for the user to choose the number of runs to simulate given a single set of input parameters. In addition, a user may also frequently wish to launch several runs with different parameter values simultaneously (i.e., sensitivity analysis). This functionality is provided through batch capability.

The simulation program assumes constant population density over time. Individuals are assumed to occupy a fixed grid on the landscape that is user defined by the *n – (x,y)* located individuals. The genotype of each locus for each individual can be initialized by randomly choosing from a file containing allele frequencies for each locus, or by reading in a file containing the initial multi-locus genotypes of all the individuals. The initial age structure of the population is specified by an input file specifying initial age frequency. The sex of each initial individual is randomly assigned.

There are five movement functions that define how individuals choose a mate and disperse on the landscape as a function of cost distance: linear, inverse square, negative exponential, nearest-neighbor, and random mixing. With the nearest-neighbor movement function, an individual moves to the available grid location nearest its initial location. Random mixing moves an individual to a grid location that is randomly chosen from the *n* grids in the population. In linear, inverse-square, and negative exponential movement functions, individuals move a distance from their initial location based on a draw from a probability distribution inversely proportional to a linear, inverse-square, or negative exponential function. The user specifies the maximum dispersal distance (in cost units) an individual can travel on the landscape. The probability is one at no distance from the original location and goes to zero at the maximum dispersal distance.

Reproduction is defined by the user as either hermaphroditic or heterosexual. With hermaphroditic mating, there are no distinct sexes, but individuals mate with other individuals according to the movement function choice, exchanging genes in Mendelian reproduction. In heterosexual reproduction, mated pairs are one male to possibly many or no females, and the end of the mating process occurs when all females have mated. Each mated pair can have a number of offspring that is a bounded random draw based on a uniform probability distribution, a Poisson draw with specified mean, or a constant number. Mendelian inheritance with *k*-allele mutation (rate chosen by the user) is used to generate the offspring’s genotype and the sex assignment is random.

Dispersal of offspring occurs from the mother’s *(x,y)* location according to the selected movement function and the sex of the individual. The vital rates (birth and death) define whether or not the population will have emigrants or immigrants.

**Simulating natural selection.** Past versions of CDPOP modeled three sources of genetic variation: gene flow, genetic drift, and mutation. These versions assumed that different genotypes have an equal probability of surviving and passing on their alleles to future generations and thus, natural selection was not operating. CDPOP V 1.0 implements natural selection analogously to the adaptive or fitness landscape of allele frequencies (Wright 1932). This new functionality enables extension of landscape genetic analyses to explicitly investigate the links between gene flow and selection in complex landscapes at an individual’s level. The user specifies fitness landscape surfaces (or values at given x,y location) for each genotype of a single diallelic locus that is under selection. For example, three relative fitness surfaces must be specified for the three genotypes, AA, Aa, and aa, from the two alleles, A and a. Selection is implemented through differential survival of dispersing individuals as a function of the relative fitness at the location on that surface where the dispersing individual settles. CDPOP v1.2 reads and extracts genotype and location specific fitness values for each *n – (x, y)* individual in the pre-processing step. The program will continue all other processes the same as CDPOP, with an additional step implement selection during the dispersal process.

**Simulating dynamic landscapes.** The potential impacts of climate change on the connectivity of populations have become an area of concern among scientists and land managers. Current needs include quantitative and spatially-explicit predictions of current and potential future patterns of fragmentation under a range of climate change scenarios (Opdam & Wascher 2003). To address this need, CDPOP v1.2 allows users to input a new landscape surface at a given generation time through new cost distance matrices for both mating and dispersal.

**Simulating demographics.** CDPOP v1.2 now allows for fluctuating population sizes controlled by birth rate, death rate, carrying capacity, and ability of offspring to disperse given a specified resistance surface.

The program is written in Python 2.7 and provided with installation instructions for most platforms, along with sample input files. CDPOP v1.2 is built on a driver-module, plug-in, docking architecture that allows for ease of future modular development. CDPOP v1.2 has been debugged as carefully as possible by testing all combinations of simulation options. Information for users, including user manual, FAQ, publications, ongoing research, developer involvement, and downloads can be found at http://cel.dbs.umt.edu/software/CDPOP/.

1. **Getting started**
   1. **Dependencies**

**2.1.2 Baseline Requirements**

CDPOP requires the Python2.7.x interpreter, NumPy package, and SciPy package. 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 v2.7.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.2.7.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.2.7.x) at:

<http://www.python.org/download/releases/>.

**2.1.4 Obtaining NumPy and SciPy**

A good suggestion for all-in-one installations is through 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). We highly recommend this version for easy installs.

* 1. **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](http://www.python.org), ActiveState, or EPD, accepting defaults for the installation directory. On Windows this will typically place the executables and libraries in c:/Python2.7/bin and the “site-packages” package tree for user installed Python modules in c:/Python2.7/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. Note that if you choose EPD that you do not need to additionally install NumPy or SciPy.

**2.2.2 Install CDPOP**

Next, install the CDPOP software itself by unpacking the zip archive supplied. Navigate to the directory on your PC where you wish to install CDPOP, 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 simulations outside of any folder that has spaces in its name (like “My Documents”). At this point you should be able to execute the supplied test inputs.

**2.2.3 Description of CDPOP files**

3 directories will be installed in your directory. Here is a description of each:

1. src – CDPOP source code
2. doc –
   * README.txt – a quick how to run CDPOP instructions
   * CDPOP\_user\_manual.pdf – this file
   * CDPOP\_history.txt – Notes on history and version changes.
3. data – Example input files

* Edcdmatrix16.csv – example Euclidean distance cost distance matrix file
* xyED16.csv – example n-(x,y) file for individuals
* xyED16\_NAs.csv – example n-(x,y) file with sparse individuals
* xyED16\_known.csv – example n-(x,y) file with known genetic data
* Agevars.csv, AgevarsA.csv, AgevarsB.csv – examples of age class file
* Agevars\_climate.csv – an example age class file to with temporal variable parameters
* allelefrequencyA.csv, allelfrequencyB.csv – examples of allele frequency distribution file
* fitvals0.txt and fitvals50.txt – example fitness landscapes for natural selection (uniform spatial selection files with differential mortality corresponding to 0% and 50%).
* gradient4x4\_N23\_S66.txt, gradient4x4\_N50\_S50.txt, and gradient4x4\_N80\_S20.txt – example fitness landscape for natural selection (gradient spatial selection files with differential mortality from North to South at 23-66%, 50-50%, and 80-20%, respectively)
* inputvars.csv – run parameters corresponding to the example files
  1. **Example run**

**2.3.1 Command line run**

The example run is for 16-points representing individuals with a cost distance matrix calculated with Euclidean distance. To run the following example, follow these steps:

1. Double check that the 3 directories provided in the archive are in the same directory.
2. The included file inputvaribles16pnts.csv in the data directory specifies the parameters that can be changed and used in a sample CDPOP run. Open inputvaribles16pnts.csv in your editor of choice. A spreadsheet program like Microsoft Excel, allows for easy editing of the tabular values.
3. There will be 3 lines of information in inputvariables.csv: a header line and 2 lines of information corresponding to 2 separate CDPOP runs (batch process). See the user\_manual.pdf that contains a breakdown for each column header and the parameters that can be changed. The ‘Input’ in the table listed is for the first row in the file. Make sure you save this file in the same format – a comma delimited file – when you make changes to the parameters. Select ‘Yes’ or ‘OK’ for any Excel questions about saving in this format.
4. Start the program: For example, if you use python from the command line, then open a terminal window and change your shell directory to the CDPOP src home directory (i.e., > cd C:\"homedirectorylocation"\src).
5. Run the program: There are a number of ways to run this program. If you are using a command shell you can run the program by typing “python CDPOP.py C:/"homedirectorylocation"/data inputvariables16pts.csv output\_test”. Note that there are 5 arguments here that must be included with spaces in between:

* "python" starts python, for example from the command line. Note that other python environments may have different calls here. In PyLab (the IDE distributed with EPD), the call is “run”.
* "CDPOP.py" runs CDPOP program.
* "C:/"homedirectorylocation"/data" is the directory location of the input test files. You can point this directory to other project files, for example. We suggest not having any spaces in your directory names.
* "inputvariables16pts.csv" is the parameter file.
* "output\_test" is the name of the directory that will be created with CDPOP output in the directory specified by the third argument above.

1. Check for successful model run completion: The program will provide step-by-step output in the Shell window. Once completed, a simulation time will be printed out and folders batchrun0mcrun0, batchrun0mcrun1, batchrun0mcrun2, batchrun0mcrun3, batchrun0mcrun4, and batchrun1mcrun0 will be created in your CDPOP home directory to store output from the separate batch and/or Monte-Carlo runs. These folders are located in the data folder specified in above step. The output folder will have a unique date/time stamp after the name of the output folder in case you want to run multiple CDPOP runs in this same directory. The program will also provide a log file with program steps in your specified output directory. If parameters are such that population becomes extinct before specified generation time, then program will end.
2. **Input**

The following are the general input parameters and files used in CDPOP. See examples provided for formatting. The file headers listed are for the GUI and the first row in the inputvariables.csv describing each file or parameter. The example provided is for the first line in the inputvariables.csv file.

|  |  |  |  |
| --- | --- | --- | --- |
| **File Header in GUI** | File Header in .csv | Example | Description |
| XY Filename | xyfilename | ‘xyED16’ – example supplied for 16 individuals.  \*xyED16\_NAs.csv gives an example of how to specify your initial starting population size. This example is 16 possible habitat locations that can be filled with only 11 initialized with individuals. You must specify NA values in the ‘ID’ and ‘sex’ columns, but still provide the xy locations that are ‘OPEN’and a ‘Supopulation’ identifier field.  \*xyED16\_cdclimate.csv give an example of how to use the fitness values with systematic change through the cdclimate module. | The *n-(x,y)* grid location values. This is a comma delimited file with *17* column headings:  (Subpopulation)- a unique identifier for each individual corresponding to a unique subpopulation. This is an optional tracker for individuals that may be located in designated subpopulations. If individuals are just continuously distributed, then fill with arbitrary value, like ‘1’. If these field is used, then subpopulations must be in sequence, e.g., 1, 2, 3, …  (XCOORD)-x-coordinate location,  (YCOORD)-y-coordinate location (YCOORD),  (ID)-a string label identifier, and  (sex)-an initial sex assignment (use 0/1 or F/M). See xyED16.csv for an example xyfilename. The column order is necessary and header file included.\*See below for specifying constant versus non-constant population sizes.  (3 Fitness values for 1-locus model)-see below.  (9 Fitness values for 2-locus model)-see below. |
| *The following are the fitness values for when CDEVOLVE Answer is 1. This corresponds to 1 locus that is under selection. The x,y location of an offspring is matched up with the closest x,y fitness value for the offspring’s corresponding fitness surface defined by the genotype that the offspring has. That value then becomes the individual offspring mortality percentage. An offspring becomes more or less fit relative to the other offspring at that generation as a function of its genotype and where it occurs on a surface. Extract values to x,y location using a GIS and values represent percent mortality [0 - 100]. See xyED16\_cdclimate.csv for an example fitness values and format.* | | | |
| Fitness\_AA, Fitness\_Aa, Fitness\_aa | 50, 0, 100, … | When CDEVOLVE Answer is 1, then this is the offspring viability selection value for given xy location for AA, Aa, and aa, respectively. | |
| *The following are the fitness values for when CDEVOLVE Answer is 2. This corresponds to 2 loci that are under selection. The x,y location of an offspring is matched up with the closest x,y fitness value for the offspring’s corresponding fitness surface defined by the genotype that the offspring has. (Extract values to points using a GIS). That value then becomes the individual offspring mortality percentage. An offspring becomes more or less fit relative to the other offspring at that generation as a function of its genotype and where it occurs on a surface. Extract values to x,y location using a GIS and values represent percent mortality [0 - 100]. See xyED16\_cdclimate.csv for an example fitness values and format.* | | | |
| Fitness\_AABBFitness\_AaBB  Fitness\_aaBB  Fitness\_AABb  Fitness\_AaBb  Fitness\_aaBb  Fitness\_AAbb  Fitness\_Aabb  Fitness\_aabb | [0,100] | When CDEVOLVE Answer is 2, then this is the offspring viability selection value for the 9 genotypes in the 2-locus selection model. If offspring has AABB, then this mortality fitness value is used at given xy location. | |
| ***Age level controls*** | | | |
| Age Distribution Filename | agefilename | ‘Agevars’ – example age distribution file | The distribution that is used to initialize each individual’s age, and parameters associated with age classes.  Filename - Supply a file in the “../data/” folder (for example ‘Agevars.csv’ would be entered for the example provided). See the ‘Agevars.csv’ file for formatting and must be comma delimited with the following headings.  ‘Age class’ – Number of age classes sequential (start initial age class at 0, assume Age 0’s are the egg/clutch/litter produced from 1+ ages). The first row of information corresponds to egg/litter and tracking numbers are reported for 1+ age classes.  ‘Distribution’ – numbers here specify the distribution within each class - do not have to add up to total N0. Initialize with 1+, enter 0 for age class 0.  ‘Male Mortality’ – the age specific mortality percentages [0-100]. Used for the ‘exp’ population model described below and ignored if density dependence is operating.  ‘Female Mortality’ – the age specific mortality percentages [0-100]. Used for the ‘exp’ population model described below and ignored if density dependence is operating.  ‘Fecundity (mean birth)’ – the lambda or mean births per age class (Poisson, random, constant, or normal draws defined in inputvars.csv file – see below).  ‘Fecundity (sigma)’ – standard deviation for each mean birth per age class with the normal fecundity option only (see below).  ‘Male Maturation’ – the age probability a male becomes a reproducing individual.  ‘Female Maturation’ – the age probability a female becomes a reproducing individual.  (option i) – If CDClimate module is initiated, then there is an option of entering in multiple mortality and fecundity values separated by a ‘|’, see Agevars\_climate.csv  (Option ii) – multiple age files for each subpopulation. A separate age distribution file can be entered for each subpopulation given. E.g., in the XY file, if 4 subpopulations were designated in the first column and separate age distributions are wished for each of these populations, then enter in the file name for each subpopulation separated by a ‘|’, e.g., ‘AgevarsA.csv|AgevarsB|AgevarsC.csv|AgevarsD’.  Note: If non-overlapping generations are desired, then enter in two rows of data with Age class 0 and 1, specifying 100% mortality for age class 1 and 0 distribution for age class 0. |
| ***Run parameters and output*** | | | |
| Monte Carlo Replicates | mcruns | ‘5’ – 5 replicate runs denoted in folders labeled with ‘mcrun0’, ‘mcrun1’,… | The repeated number of simulations to be conducted for the Monte Carlo method (i.e., the number of replicates for 1 batch of parameters). |
| Generation/Time | Looptime | ‘10’ – 10 generations | Simulation run time [generation or year]. File output indexed from 0 – (looptime-1). For example grid0.csv, grid1.csv, grid2.csv, grid3.csv, and grid4.csv would be output for a looptime of 5. |
| Generations of Saved Genotypes Choice | Output\_years | ‘0’, ‘1’, ‘0|3|4’ – the sequence value or list of generations to save. | The specified simulation time steps [year/generation] to write to file, summarize year metrics, or to calculate genetic distance matrices.   * Enter a single number to produce a sequence of values. For example ‘1’ with runtime = 10 would produce output for years 0, 1, …, 9. A value of ‘2’ with runtime = 10 would produce output for years 0, 2, 4, 6, 8. * Enter exact years by using a ‘|’. For example ‘0|3|4’ would produce output for years 0, 3, and 4. Note that years begin counting at 0, so the last value must be one less than the runtime (e.g., runtime = 10, then 0|5|9 with 9 being the maximum value for the last year). |
| GD Matrix | Gendmatans | ‘Dps’ – proportion of shared alleles genetic distance matrix is produced for every grid.csv file. | This is the genetic distance matrix answer. The genetic distance matrix used for the *n* original grid locations for specified generation time of the simulation run is calculated.  Enter ‘braycurtis’ for the Bray-Curtis distance measure,  ‘Dps’ for the proportion of shared alleles, or  ‘Da’ for Nei’s genetic distance.  Specify ‘N’ here if you do not want to calculate these matrices.  Note that Nei’s genetic distance takes the longest to calculate and may decrease your total CDPOP simulation time. All are slightly expensive in computation time and we are working on speeding these up. |
| Genotype Output Format | gridformat | ‘genepop’ | This is the genotype output format option. The format for the genotype output is specified by entering:  ‘cdpop’ – This format is the default. The cdpop format lists the genotypes with values for each allele -> either 0, 1, or 2. Output will be labeled grid{generation}.csv  ‘general’ - for a general genotype output. The general format will follow Locus1a, Locus1b, Locus2a, Locus2b, …, LocusNa, LocusNb. Output will be labeled generalgrid{generation}.csv  ‘genalex’ – for the program GENALEX. Output will be labeled genalexgrid{generation}.csv  ‘structure’ – for the program structure or related programs. Output will be labeled structuregrid{generation}.stru  ‘genepop’ – for the program GENEPOP or related programs. Output will be labeled genepopgrid{generation}.txt |
| ***CDClimate and movement surfaces with functions***  *Parameter options to vary temporally, include effective distance matrices, movement functions within the Inputvars.csv file, and mortality and fecundity values within the Agevars.csv file. See Agevars\_climate.csv as an example formate for mortality and fecundity temporal change. The first batch run of the Inputvars.csv file then corresponds to how temporal changing values should be entered to intiated the module CDClimate.* | | | |
| CDCLIMATE  Generation | Cdclimgentime | ‘0|5|10’ or ‘0’ | To initiate the CDClimate module, this is the generation/year that the next effective distance matrix will be read in at. You can specify multiple generations by separating each generation to read in the next cost distance matrix by ‘|’. Then in the following surface columns, a separate file can be given for each generation.  Place only a ‘0’ here to start simulations with one surface and continue to use this surface throughout looptime. |
| Mate CD Matrix  Filename | matecdmat | ‘EDcdmatrix16’ – an example Euclidean distance matrix used for the mating movement. | A *[n x n]* effective distance matrix for mating movement, where *n* is the number of individuals on the landscape. This is a comma delimited file. See the example given for formatting this file. Also note that this file can be calculated from any program you choose (e.g., PATHMATRIX, CIRCUITSCAPE, UNICOR, COSTDISTANCE, and with R gdistance functions).  If CDClimate module was initiated with multiple ‘cdclimgentime’, then the same number of surfaces must be given here separated by a ‘|’, e.g., ‘EDcdmatrix16| EDcdmatrix16| EDcdmatrix16’ could correspond to the example given for reading in a new surfaces at generations ‘0|5|10’. |
| Dispersal CD Matrix Filename | dispcdmat | ‘EDcdmatrix16’ – an example Euclidean distance matrix used for the dispersal movement. | Used for the movement/dispersal of individuals. Same description as ‘matecdmat’. It also can be the same file as the matecdmat. |
| Movement  Functions | Matemoveno,  Fdispmoveo, Mdispmoveno, | ‘1’ – linear probability function of mating cost distance. | Movement function answer for mating probability.  1 = Linear (1 – (1/Threshold) \* Cost Distance)  2 = Inverse Square (1 / (Cost Distance^2)). This function gets rescaled to min and threshold of the inverse square cost distance.  3 = Nearest Neighbor (Use threshold to specify Moore neighborhood  4 = Random Mixing and will consider the cost distance threshold. Use the maximum cost distance in the threshold field if you want to consider the entire population as random movement.  5 = Negative Exponential (parA \* 10^(-parB \* Cost Distance)). This function gets rescaled to min and threshold of the negative exponential cost distance.  6 = Subpopulation: Given the subpopulation j, then movement will occur using the cost distance matrix of individuals within as (1 / (Cost Distance\_j^2))  7 = Gaussian function: A \* exp ( - (Cost Distance - B)^2 / (2\*C^2)). This function gets rescaled to min and threshold of the Gaussian function cost distance. Note that parameter C should be scaled appropriately for your cost distance matrix range.  8 = Use the cost distance matrix, rescaled to the min and threshold.  Note that individual cost distance values are stored in each grid{}.csv file, but that option 5,7,and 8 show probability values. These can be converted given function parameters and threshold values entered for each option below.  If CDClimate module is initiated with multiple read in matrices, then this option must also match the generations to run new surface and function separated by a ‘|’. All probabilities are scaled between 0 and 1. Some functions below are naturally between 0-1, while others use the minimum, maximum, and threshold values of the effective distance matrix to rescale.  If ‘F’ or ‘M’ is before this field name, then separate options can be entered for female or male movement. |
| Movement Parameter A | matemoveparA, FdispmoveparA, MdispmoveparA, | ‘0.0005’ | This is the A parameter used for the function in movement answer ‘5’, ‘7’. |
| Movement  Parameter B | matemoveparB, FdispmoveparB, MdispmoveparB, | ‘0.01’ | This is the B parameter used for the function in movement answer ‘5’, ‘7’. |
| Movement  Parameter C | matemoveparC | ‘0.01’ | This is the C parameter used for the function in movement answer ‘7’. |
| Movement  Threshold | Matemovethresh, Fdispmovethresh, Mdispmovethresh, | ‘max’ | A threshold option (in effective distance units) for how far an individual can search for a mate, migrate, or disperse to a new location, equivalent to the effective distance kernel. You can specify ‘max’ to consider all individuals for mating movement.  You can also place an integer value in front of ‘max’ to consider a percent cost distance movement for mating. For example ‘10max’ would consider all mating individuals that are within 10 percent of the maximum cost distance on the surface. Caution using this option when comparing across landscape surfaces.  You can also just specify a specific cost distance value. |
| ***Reproduction options*** | | | |
| Reproduction  Answer | sexans | ‘Y’ – sexual reproduction is choosen | ‘Y’ for sexual reproduction. In sexual reproduction, mated pairs consider male and females with or without replacement.  ‘N’ for asexual reproduction. With asexual reproduction, all *n* individuals mate and bear offspring, with mates selected according to the movement function choice and without regard to any gender or mating type. It is important to note that this “asexual” reproduction is functionally the sexual paring of hermaphroditic individuals; thus it is asexual in the sense that there are no distinct sexes, but is sexual in the sense that individuals mate with other individuals, exchanging genes in Mendelian reproduction. |
| Female Replacement | Freplace | ‘N’ – females mate without replacement | If you want females to mate with replacement, then specify ‘Y’.  If you want females to mate without replacement, then specify ‘N’. |
| Male  Replacement | Mreplace | ‘Y’ – males mate with replacement. | If you want males to mate with replacement, then specify ‘Y’.  If you want males to mate without replacement, then specify ‘N’. |
| Multiple Paternity Answer | multiple\_paternity | ‘Y’ – females can have a litter from multiple males. | If you want multiple paternity with birth rates applied the same for each female with litter, then specify ‘Y’.  If you want multiple paternity with birth rates applied unequally for each female with litter, then specify ‘N’.  This answer is only functionly when Freplace = ‘Y’ and Mreplace = ‘Y’ or multiple paternity option. The difference is subtle. For example if the birth rate is set at a constant litter size (offno = 3) of 2 (lambda = 2) and multiple\_paternity = ‘Y’, then if a female mates with 2 males the female would have 2 offspring (1 from each male). However, if multiple\_paternity = ‘N’, then if a female mates with 2 males, then that female would have 4 offspring. |
| Selfing  Answer | Selfans | ‘N’ – selfing is turned off. | If you want to allowing selfing (i.e., individuals mate with themselves), then specify ‘Y’.  If you do not want to allow for selfing, then specify ‘N’. |
| Philopatry | Philopatry | ‘N’ – philopatry turned off | This is the behavior of remaining at the individual’s birthplace. The options are:  N – turned off.  F – Female philopatry  M – Male philopatry  For example, if this is specified for female philopatry (‘F’), then females will be able to remain in their birth locations or given female dispersal threshold. If this is turned off (‘N’), then males have the chance of coming into a female territory and taking it over. When (‘N’) is specified, then the generation will produce equal female and male occupied locations.  Care should be taken when running this philopatry option. In order to prevent unequal sex ratio bias due to bias dispersal capabilities, this module will sort and place the respective sex under philopatry first. For example, if female philopatry (‘F’) is specified, then the female offspring will first have the opportunity to disperse to a new location and then male offspring.  Use a movement dispersal threshold of ‘0’ for strict philopatry (e.g., female offspring takes over the mother’s locations). The use of AtBirth sex ratio and even litter sizes is recommended in order to prevent unequal sex ratio bias. |
| ***Litter options*** | | | |
| Offspring  Choice | Offno | ‘2’ – Poisson distribution | This is the number of offspring each mate pair can have.  1 - for a random draw between 0 and ‘mean fecundity’ given in Agevars.csv file,  2 - for Poisson draw around ‘mean fecundity’ given in the Agevars.csv file.  3 - for a constant number of offspring of size ‘mean fecundity’ given in the Agevars.csv file.  4 – for an equal clutch size for each female that would be equal to the ‘mean fecundity’ given in the Agevars.csv file.  5 – for a normal draw, with mean and sigma for fecundity given in the Agevars.csv file.  6 – special case for stable age distribution: females reproduce with given fecundity values, however, an additional mortality can happen to correct the number of offspring produced in each age class. I.e., the survival of age 0s from given age class = Total N in that age class \* (1-mortliaty of age class 0) / Females in that age class. Note that this option should be used for special cases and some parameters will not be considered: e.g., female and male age0 mortality values will be averaged.  Note that option 1 – 3 will assign a clutch size to each mate event. For example, if a female mates with 4 males and offno is set to 3 with a ‘mean fecundity’ of 2, then she will have 8 offspring. Option 4 will assign a clutch size to each female that mated. For example, if a female mates with 4 males and offno is set to 4 with lambda of 2, then she will have 2 offspring total and the father(s) is randomly selected from the 4 mating events.  (Option) – If CDClimate module is initiated with multiple ‘cdclimgentime’, then multiple birth functions can also be specified here separated by a ‘|’. E.g., ‘1|1|1’ with a combination of a ‘Lmbda’ value below, e.g., ‘2|3|2’. |
| Female offspring | Femalepercent | ‘50’ – 50% random female assignment. | Percent number of female born in each litter. This is a random assignment from given percentage, i.e., even if you set this to 50% some generations could have 499 female births and 501 male births, for example. |
| Equal Sex  Ratio for Offspring | EqualsexratioBirth | ‘N’ | This ensures an exact equal sex ratio for the following options:  ‘WrightFisher’ - The answer to have every generation start with equal sex ratios. CAREFUL, this parameter is not realistic for non panmictic populations and should only be used to match Wright-Fisher assumptions on equal sex ratios.  ‘AtBirth’ – This will ensure that each litter is equal sex ratio, but not necessary ensure that the generation will result in equal sex ratio after dispersal do to unbiased dispersal parameters and stochastic dispersal (i.e., random offspring chosen for dispersal).  N – This option is not used and offspring sex is assigned using the Female-percent parameter above. |
| ***Mortality options***  *Mortality can be controlled through a constant value (exponential growth) or through density dependence applied to survival. Depending on the choice of the population model, some parameters here will be ignored.* | | | |
| Population Model | Popmodel | ‘exp’ ‘logistic’, ‘richards’, ‘rickers’ | The choice of population growth models.  ‘exp’ – or exponential growth where n(t+1) = birth-rate \* n(t) – death-rate \* n(t). Population numbers can reach the set carrying capacity of the individuals in the XY file, but not exceed this number. The use of Leslie matrices and age structured growth can be done through the Agevars.csv file with survival and fecundity numbers. Thus the ‘Mortality’ values in the Agevars.csv file are used. Note that if an age/stage structure model is used, set up Leslie matrices as census before birth pulse.  ‘logistic’ – n(t+1) = n(t) + r\*n(t) (1 – n(t) / K\_environment) where r is the intrinsic growth rate and K\_environment can be a temporal fluctuating value (See below).  \*Note that the average male and female age specific survival (i.e., mortality values entered in the AgeVars.csv file) will be used. |
| Density Dependent growth rate | r | ‘1.0’ | The growth rate used in the density dependent functions above (‘logistic’, ‘richards’, and ‘rickers’).  (option) – If the cdclimate module is specified then multiple values can be entered for the time intervals creating a temporally varying density dependent system. |
| Environmental carrying capacity | K\_env | ‘14’ | This is the carrying capacity used in the density dependent equations above (‘logistic’, ‘richards’, and ‘rickers’).  Note that this K\_env can be less than or equal to the total carrying capacity specified in the XY file.  (option) – If the cdclimate module is specified then multiple values can be entered for the time intervals creating a temporally varying density dependent system. |
| Subpopulation Mortality | Subpopmortperc | ‘0|0|0|40’ | This parameter is the percent mortality for a dispersing offspring into another subpopulation. Each subpopulation gets separated by a ‘|’ and has it’s own percent mortality. For example, if an offspring was born in the second subpopulation and disperses to the fourth subpopulation it has a 60% chance of surviving there. If it stays in its own subpopulation, then it would have no differential mortality consequences.  Mate movement and mate selection is also considered here with these mortality percentages. For example, if a female from the second population selects a mate from them 4th population, then there would be a 60% chance that this mate pairing would occur.  This parameter is very similar to cdevolve offspring viability, but applied to subpopulations instead of individuals spatial locations. |
| ***Genetic options*** | | | |
| Mutation  Rate | muterate | ‘0.0005’ | The mutation rate. |
| Mutation Model | mutationtype | ‘random’ – the KAM model. | The type of mutation model:  ‘random’ – This is the kth-allele mutation model.  ‘forward’ – This is a step-wise mutation in which an allele can mutate forwards only (i.e., to the right).  ‘backward’ – This is a step-wise mutation in which an allele can mutate backwards only (i.e., to the left).  ‘forwardbackward’ – This is a step-wise mutation in which an allele can mutate forward or backwards only (i.e., to the left or right with equal probability).  ‘forwardAbackwardBrandomN’ – This is a special case for the 2-loci selection model. The first locus under selection can only go forward (A -> a) and the second locus under selection can only go backward (b -> B). The rest of the neutral loci are random mutations. |
| Loci | loci | ‘10’ | The number of loci (microsatellites). |
| Initialize Genotypes | intgenesans | ‘random’ | The choice for how to initialize the genotype for each *n-(x,y)* individuals.  If ‘random’ is entered, then the genotypes get a random assignment and the population is at a maximum genetic diversity.  If ‘file’ is entered, then the genetics get drawn from the allele frequency distribution file (specified in next column, allefreqfilename).  If ‘known’ is entered, then the genotypes are directly read from a given known file. This file is very similar to the initial xyfilename and example xyED16\_known.csv is supplied with test data files. |
| Allele Frequency File | Allefreqfilename | ‘N’ – allele frequency file not used. | The allele frequency distribution for each locus, used to initialize the model’s *n* individual’s genotype. If you want to use a frequency distribution file, you must set Initialize Genes Answer to equal ‘file’ and then enter in the filename in this field. See allelefrequency.csv example file for formatting this file. It is basically a column of allele frequencies and make sure the length of the column equals your starting loci \* starting alleles.  (Option) – multiple allele frequency distributions for each subpopulation. Make sure ‘intgenesans’ equals ‘file’. Then a separate allele frequency file can be entered for each subpopulation given. E.g., in the XY file, if 4 subpopulations were designated in the first column and separate allele frequencies are wished for each of these populations, then enter in the file name for each subpopulation separated by a ‘|’, e.g., ‘allelefrequencyA.csv|random|allelefrequencyB.csv|random’. Note here, that a file or random starting can be used. |
| Alleles | alleles | ‘5’ | The number of starting alleles per locus. |
| Start Gene Swap | startGenes | ‘20’ | The generation/time at which genetic exchange will occur. Values for genotypes in grid{}.csv files will be ‘NA’ until this time unit occurs. Then when startGenes time loop begins, the genotypes will be initialized randomly or begin with ‘known’ genetics.  Note that ‘known’ initialization option for genotypes will not work for a startGenes greater than 0 – due to potentially fluctuating populations. |
| mtDNA | Mtdna | ‘N’ | If ‘Y’, then last locus becomes mtDNA and every offspring inherits this locus from its mother only. If ‘N’, then regular Mendal inheritance occurs for this last locus. |
| ***Spatial selection options*** | | | |
| CDEVOLVE  Answer | cdevolveans | N | This is the answer for how many loci are under selection.  Use ‘N’ to turn off CDEVOLVE.  Use ‘1’ for natural selection with 1 locus.  Use ‘2’ for selection with 2 loci. Alleles must be 2 if ‘Y’ is entered.  A special case for considering spatial selection for mature individuals only can be considered. Enter ‘3’. |
| Tart  Burn-in Generation | Burningen | 10 | This is the generation or year that the selection surface will begin operating on the locus or loci under selection, specified in previous field (cdevolveans). If ‘N’ is specified for ‘cdevolveans’, then this field is ignored. |
|  | | | |
| Fitness  AA |  |  |  |
| Fitness  Aa |  |  |  |
| Fitness  aa |  |  |  |
| ***Infection options*** | | | |
| CDINFECT | cdinfect | ‘N’ – turned off | This is the infection parameter answer. This tracks vertical transmission in the population. A column in grid.csv denotes the infection status at each generation for every individual.  If ‘Y’, then a random status infection (0 or 1) is created and initialized for each individual.  If ‘N’, then the status 0 is created for all individuals and initialized. |
| Transmission Probability | Transmissionprob | ‘0.5’ | This is the transmission probability for if a parent has the infection the chance that the infection will be passed along to the offspring. |

1. **Output Files**

Folders will be created in your project directory labeled with a unique time stamp (dos convention), e.g., 1332964297batchrun0mcrun0. Monte Carlo runs will be uniquely labeled mcrun0, mcrun1,… and each batch run will be uniquely labeled batchrun0, batchrun1, … . In each folder you will see grid{generation}.csv files that list each individual’s genotype, spatial locations, unique ID(Time of birth, Mother and Father ID, and Pop it dispersed from), age, sex, infection status, and cost distance moved\*\*. If you specified a grid format option, then you will additionally see the format followed by grid{generation}.csv.

\*\* Note that functions 5, 7, and 8 will display as probability and not in cost units. This value can be converted back to cost units given user specification for function parameters and threshold values.

In addition, an output.csv is automatically created for each batch and Monte Carlo run. These are population based metrics calculated at each generation. The following is a summary of each calculation:

* Year – This is the generation time or year if using overlapping generations.
* Population – The total population in each generation. If you specify subpopulations in the first column of your xy.csv file, then this field will be separated by ‘|’. The first value is the total population size and each additional value corresponds to the subpopulation sizes in order.
* Population\_Age – The total population count for each age class 1+. Situations in which age classes become greater than age classes specified are assumed to be lumped into the last age class.
* GrowthRate – Approximate growth rate (lambda) or Nt+1/Nt.
* ToTFemales – The total number of females in each generation. If you specify subpopulations in the first column of your xy.csv file, then this field will be separated by ‘|’. The first value is the total female size and each additional value corresponds to the subpopulation sizes in order.
* ToTMales – The total number of males in each generation. If you specify subpopulations in the first column of your xy.csv file, then this field will be separated by ‘|’. The first value is the total male size and each additional value corresponds to the subpopulation sizes in order.
* BreedFemales – The total number of breeding age females in each generation. If you specify subpopulations in the first column of your xy.csv file, then this field will be separated by ‘|’. The first value is the total breeding female size and each additional value corresponds to the subpopulation sizes in order.
* BreedFemales\_Age – The total number of breeding age females in each age class 1+ separated by a ‘|’ for each generation.
* BreedMales – The total number of breeding age females in each generation. If you specify subpopulations in the first column of your xy.csv file, then this field will be separated by ‘|’. The first value is the total breeding male size and each additional value corresponds to the subpopulation sizes in order.
* Female\_BreedEvents – This is the number of breeding events for females. This number times the birth rate will give the total number of births in that generation.
* Females\_NoMate – This is the number of females that did not have a mate.
* Migrants – The number of dispersers in each generation that make it to the next generation.
* DisperseDeaths – The number of dispersers in each generation that do not make it to the next generation (due to selection).
* Births – The number of offspring born at that generation.
* Age0Deaths – The number of deaths at the age0 class split up for males and females.
* Deaths – The number of deaths of the adult population split up for males and females, not the offspring born that year/generation. The ‘|’ separates the age class deaths for overlapping generations for the ordered age class in that generation. Note that some years may not have all age classes and not reported.
* Alleles – This is the total number of unique alleles at each generation. This value can be calculated automatically for specified subpopulations if different subpopulations were designated in the initial xyfilename. If there are ‘|’, then the first value corresponds to the total alleles in the population and subpopulation values follow after.
* He - This is the expected heterozygosity value at each generation. This value can be calculated automatically for specified subpopulations if different subpopulations were designated in the initial xyfilename. If there are ‘|’, then the first value corresponds to the total He in the population and subpopulation values follow after.
* Ho - This is the observed heterozygosity value at each generation. This value can be calculated automatically for specified subpopulations if different subpopulations were designated in the initial xyfilename. If there are ‘|’, then the first value corresponds to the total Ho in the population and subpopulation values follow after.
* Mutations – The total number of mutations at each generation.
* MateDistED – The average Euclidean distance individuals travel to mate.
* DispDistED – The average Euclidean distance individual offspring disperse from their natal location separated into female and male movement.
* MateDistCD – The average cost distance (as a probability) individuals travel to mate. This can be converted back to cost distance if using a linear function.
* AllMateDistances – For the given generation/year, all of the mating pairs and corresponding cost distances moved to mate are reported in this column and row, separated by a ‘|’.
* DispDistCD – The average cost distance (as a probability) individual offspring disperse from their natal location separated into female and male movement. This can be converted back to cost distance if using a linear function.
* MateDiststd – The standard deviation Euclidean distance individuals travel to mate.
* DispDiststd – The standard deviation Euclidean distance individual offspring disperse from their natal location (for both female and males).
* MateDiststd – The standard deviation cost distance (as a probability) individuals travel to mate.
* DispDiststd – The standard deviation cost distance (as a probability) individual offspring disperse from their natal location (for both females and males).
* P1 – This is the allele frequency of A (used for CDEVOLVE).
* P2 – This is the allele frequency of a (used for CDEVOLVE).
* q1 – This is the allele frequency of B (used for CDEVOLVE).
* q2 – This is the allele frequency of b (used for CDEVOLVE).
* SubpopImmigration – The number of individuals that immigrate to the ordered subpopulation number. Separated by ‘|’.
* SubpopEmigration – The number of individuals that emigrate from the ordered subpopulation number. Separated by ‘|’.
* SubpopNoMate – The number of times a mate selection was attempted and did not succeed from another subpopulation. Used with subpopmortperc parameter.
* Infected – The number of individuals that are infected at each generation (used for CDINFECT) with given transmission probility (vertical infection).
* FemalesMeanMate – The mean number of females matings in a generation.
* MalesMeanMate – The mean number of male matings in a generation.
* FemalesSDMate – The standard deviation in the number of females matings in a generation. (variance in reproductive success)
* MalesSDMate – The standard deviation in the number of male matings in a generation. (variance in reproductive success)
* OpenLocations – The number of habitat available locations that did not get filled during dispersal.
* CouldNotDisperse – The number of individuals that could not disperse due to high cost to moving to an open locations.

1. **General issues**
   1. **How to obtain CDPOP**

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

* 1. **Debugging, troubleshooting, and general problems/solutions**

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](mailto:erin.landguth@mso.umt.edu).

1. **Is your population crashing?**

There are 2 main reasons why you are not getting a stable population. (1) Your birth rate < death rate. Try increasing the mean number of offspring / mature female.

If this is still producing a crash in population then (2) open grid files in the output folders. Notice if the same XY locations are always going extinct. This is what I call a ‘isolated grid’. Depending on how your resistance surface was set up with how you initialized the XY locations, you could have a cluster of XY points that are separated from the rest of the population. The cost to reach this cluster could be too large and random extinction can occur. Try increasing your thresholding to maximum to see if this fixes the problem or increase the density of XY points.

1. **Your simulated data are not producing a signal?**

First, double check that your XY order matches the cost distance matrix order!

Second, I would recommend plotting your cost distance matrix and converting to probability based on the function used. Compare your function to a linear conversion. Does it convex or concave from the linear line? For example, inverse square most always will produce a signal even with a maximum movement distance. So, double check your function used. Third, decrease your thresholding until the desired signal is reached. T

The combination of function and thresholding are important parameters when considering the strength of signal.

1. **You can’t get the example CDPOP to run?**

You have followed all of the installation instructions for python and attempted to run the example steps, but get an error like: “CDPOP\_Modules not found”. First check that you have installed the correct python version that matches your computer (e.g., 64 bit vs 32 bit).

Second, what version of python did you install? Open a command prompt and type ‘python’, what version was displayed? Currently if it says anything other than 2.7.something, then I recommend reinstalling a 2.7 vesion.

Third, do you have numpy or scipy installed? In the command prompt, after python is open, you will see >>> and then type ‘import numpy’ and ‘import scipy’. If you did not get an error after these statements, then good. However, there still could be issues with what version of numpy and scipy you are using. Because of this, I highly recommend installing a superpack installer that takes care of everything for you. EPD or Canopy is a good choice. Anaconda too.

If you are running simulations on a cluster, make sure your administrator knows about the above issues.

If you still are getting the same error, then please email me.

* 1. **How to cite CDPOP**

This program was developed by Erin Landguth with help from Brian Hand, Joe Glassy, Sam Cushman, and Tyler Julian. GUI development was done by Mike Jacobi. The reference to cite is as follows, substituting the version number:

Landguth EL, Cushman SA, Jacobi M (2010) CDPOP: A spatially-explicit

cost distance population genetics program. Molecular Ecology Resources. 10:156-161 “Version X”.

Landguth EL, Cushman SA, Johnson NJ (2011) Simulating natural

selection in landscape genetics. Molecular Ecology Resources. doi: 10.1111/j.1755-0998.2011.03075.x.

* 1. **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). CDPOP 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 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 CDPOP. 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 CDPOP 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 CDPOP. 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 CDPOP 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 CDPOP in ecological investigation and management application. Therefore, we encourage you to contact us and describe your application after using CDPOP.

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

1. **References**

Allendorf,F.W. and Luikart,G. (2007) Conservation and the genetics of

populations. Blackwell, Malden, MA.

Bowcock,A.M. *et al.* (1994) High resolution of human evolutionary trees with polymorphic micorsatellites. *Nature*. **368**, 455-457.

Cushman,S.A. *et al.* (2006) Gene Flow in Complex Landscapes: Testing Multiple Hypotheses with Casual Modeling. *The American Naturalist* **168**, 486-499.

Cushman,S.A. and Landguth,E.L. (2010) Spurious correlations and

inferences in landscape genetics. *Molecular Ecology*, **19**, 3592-3602.

Holderegger,R. and Wagner,H.H. (2006) A brief guide to Landscape Genetics. *Landscape Ecology* **21**, 793-796.

Landguth,E.L. and Cushman,S.A. (2010) CDPOP: A spatially-explicit cost

distance population genetics program, *Molecular Ecology Resources*, **10**, 156-161.

Landguth,E.L. *et al.* (2010a) Quantifying the lag time to detect

barriers in landscape genetics. *Molecular Ecology*, **19**, 4179-4191.

Landguth,E.L. *et al.* (2010b) Relationships between migration rates and

landscape resistance assessed using individual-based simulations. *Molecular Ecology Resources*, **10**, 854-862.

Legendre,P. and Legendre,L. (1998) Numerical ecology. 2nd English ed.

Elsevier,Amsterdam.

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.

Nei,M. *et al.* (1983) Accuracy of estimated phylogenetic trees from molecular data. *Journal of Molecular Evolution* **19**,153–170.

Ray,N. (2005) PATHMATRIX: a GIS tool to compute effective distances among samples. *Molecular Ecology Notes* **5**, 177-180.

Storfer,A. *et al.* (2010) Landscape genetics: where are we now?

*Molecular Ecology*, **19**,3496–3514.

Wright,S. (1932) The roles of mutation, inbreeding, crossbreeding, and

selection in evolution, *Proceedings XI International Congress of Genetics*, **1**, 356-366.

1. **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.