**CDMetaPOP**

**USER MANUAL**

Version: 2.60

Last Updated: 2023.02.22

Authors:

**Erin L. Landguth1, Casey C Day1, Andrew Bearlin2, Jason Dunham4, Ryan Simmons2, Kaeli Davenport5, Travis Seaborn6, Brenna Forester7**

1 – University of Montana, Computational Ecology Lab, Missoula, MT, 59812, USA. erin.landguth@mso.umt.edu.

2 – Seattle City Light, Environmental Affairs Division, Seattle, WA, USA. andrew.bearlin@seattle.gov

3 – University of Montana, Computational Ecology Lab, Missoula, MT, 59812, USA. [caseycday@gmail.com](mailto:caseycday@gmail.com)

4 – U.S. Geological Survey, Forest and Rangeland Ecosystem Science Center, Corvallis, Oregon, 97331, USA.

5 – University of Montana, Wildlife Biology Program, Missoula, MT, 59812, USA.

6 – North Dakota State University, School of Natural Resource Sciences. Fargo, ND. travis.seaborn@ndsu.edu.

7 - Department of Biology, Colorado State University, Fort Collins, CO, 80523, USA. Email: Brenna.Forester@colostate.edu

How to cite:

The reference to cite is as follows, substituting the version number:

Landguth EL, Bearlin A, Day C, Dunham J (2017) CDMetaPOP: an individual-based, eco-evolutionary model for spatially explicit simulation of landscape demogenetics. Methods in Ecology and Evolution 8(1):4-11. v2.54.

**Table of Contents**

[**1.** **Introduction** 5](#_Toc108193966)

[**1.1** **What can CDMetaPOP do?** 5](#_Toc108193967)

[**1.1.1.** **Major changes / extensions** 5](#_Toc108193968)

[**1.1.2.** **Potential applications** 8](#_Toc108193969)

[**1.2** **How does CDMetaPOP work** 8](#_Toc108193970)

[1.2.1 Main Processes 9](#_Toc108193971)

[1.2.2 Module details 27](#_Toc108193972)

[**2** **Getting Started** 39](#_Toc108193973)

[**2.1** **Dependencies** 39](#_Toc108193974)

[2.1.1 Baseline requirements 39](#_Toc108193975)

[2.1.2 Python on non-windows platforms 39](#_Toc108193976)

[2.1.3 Python on windows 39](#_Toc108193977)

[2.1.4 Obtaining NumPy and SciPy 39](#_Toc108193978)

[**2.2** **Installation** 39](#_Toc108193979)

[2.2.1 Installing Python, NumPy, and SciPy 40](#_Toc108193980)

[2.2.2 Installing CDMetaPOP 40](#_Toc108193981)

[2.2.3 Description of CDMetaPOP files 41](#_Toc108193982)

[**2.3** **Example run** 41](#_Toc108193983)

[2.3.1 Command line run 42](#_Toc108193984)

[**3** **Input** 44](#_Toc108193985)

[**3.1** **RunVars.csv file – run parameters apply to all species** 45](#_Toc108193986)

[**3.2** **Run parameters and output – PopVars.csv file** 49](#_Toc108193987)

[3.2.1 CDClimate and movement surfaces with functions 51](#_Toc108193988)

[3.2.2 Reproduction options 57](#_Toc108193989)

[3.2.3 Offspring/litter/egg/sex ratio options 60](#_Toc108193990)

[3.2.4 Genetic options 62](#_Toc108193991)

[3.2.5 Spatial selection options 63](#_Toc108193992)

[3.2.6 Phenotypic Plasticity options 68](#_Toc108193993)

[3.2.7 Infection options 69](#_Toc108193994)

[3.2.8 Growth options (see Section 1.2.1.6) 69](#_Toc108193995)

[**3.3** **Patch level controls – PatchVars.csv file** 70](#_Toc108193996)

[***3.4*** ***Class level controls – ClassVars.csv file*** 76](#_Toc108193997)

[**4** **Output** 80](#_Toc108193998)

[**4.1** **Individual files** 80](#_Toc108193999)

[4.1.1 ind.csv or indSample.csv 80](#_Toc108194000)

[4.1.2 Genetics format option 83](#_Toc108194001)

[**4.2** **Summary measures** 83](#_Toc108194002)

[4.2.1 Time versus class 83](#_Toc108194003)

[4.2.2 Class by given time 84](#_Toc108194004)

[4.2.3 Time versus patch 84](#_Toc108194005)

[4.2.4 Patch by given time 88](#_Toc108194006)

[**5** **Special Cases** 88](#_Toc108194007)

[**5.1** **Non-overlapping generations** 88](#_Toc108194008)

[**5.2** **Wright-Fisher assumptions** 88](#_Toc108194009)

[**5.3** **Deterministic and stochastic variability** 90](#_Toc108194010)

[5.3.1 Deterministic example runs 93](#_Toc108194011)

[**5.4** **Terrestrial Example** 99](#_Toc108194012)

[**6** **General Issues** 105](#_Toc108194013)

[**6.1** **How to obtain CDMetaPOP** 105](#_Toc108194014)

[**6.2** **Debugging and troubleshooting** 105](#_Toc108194015)

[**6.3** **How to cite CDMetaPOP** 105](#_Toc108194016)

[**6.4** **Disclaimer** 106](#_Toc108194017)

[**7** **References** 106](#_Toc108194018)

[**8** **Acknowledgements** 111](#_Toc108194019)

# **Introduction**

The goal of this user manual is to explain the technical aspects of the current release of the CDMetaPOP program. CDMetaPOP is a major extension and branch of the programs CDFISH (Landguth et al. 2012a) and CDPOP (Landguth and Cushman 2010; Landguth et al. 2012b). CDMetaPOP models genetic exchange for individuals across given resistance surfaces and *n – (x, y)* located subpopulations (i.e., patches) as a function of individual-based movement through mating, migration and dispersal, vital dynamics (birth and death), density-dependent mortality, selection, and mutation. CDMetaPOP was developed for use in both aquatic and terrestrial and can be used for both plant and animal applications.

Throughout this user manual a number of example models will be discussed. During development of the CDMetaPOP program, the primary species of interest were Westslope Cutthroat Trout (*Oncorhynchus clarkii*; WCT), Bull Trout (*Salvelinus confluentus*; BT), Lahontan Cutthroat Trout (*Oncorhynchus clarkii henshawi*; LCT), and Eastern Brook Trout (*Salvelinus fontinalis*; EBT). While the primary development was geared for riverine examples, we have also used the program with Whitebark Pine (*Pinus albauculis*) and Mountain Pine Beetle (*Dendroctonus ponderosae*), and thus, can be used for terrestrial systems as well.

## **What can CDMetaPOP do?**

We created CDMetaPOP to simulate 100s of thousands of individuals for more complex biological processes that are not included in CDPOP, CDFISH, or any other program to our knowledge. Therefore, major updates, extensions and additions to modules and functions within CDPOP/CDFISH were added that created a branch in these programs with modifications necessary for significant performance time improvements. We list below the new functionalities for this program.

* + 1. **Major changes / extensions**
* Metapopulation framework:
  + 100s of thousands of individuals can be tracked using a metapopulation framework where individuals are in ‘patches’ on the landscape.
  + Patches can be designed to have a single individual (i.e., operating truly at the individual-based level) or a set carrying capacity with multiple individuals (i.e., operating at the deme-based level). All individuals in a patch share a common environment.
* Class structure: Age or size classes initialized by user and can control various processes in model, including
  + Maturation rate
  + Growth rate
  + Dispersal, migration and straying probabilities
  + Fecundity
  + Capture probability
* Density dependence: Individuals are allocated space in ‘patches’ based on size and density dependent carrying capacity using a ‘packing algorithm’ or age-structured logistic growth.
* Movement rules:
  + Individuals move between patches for mating, dispersal, migration (to and from mating grounds), and straying.
  + Migration occurs at two separate time steps: 1) movement out of natal patches to an overwintering/foraging/rearing location and 2) movement back to natal patches from the overwintering/foraging/rearing location.
  + Individuals can fail to return back to natal patches and then be considered a mortality event or can stray.
  + Straying can be modeled as a separate process during movement back to natal populations.
  + Genetically coded behavioral plasticity can be used to simulate temperature or habitat quality avoidance dependent on life experience and genotype.
* All movement (mating, dispersal, migration and straying) can be modeled separately, spatially, and temporally as functions of landscape/riverscape surfaces. Sex-based processes: processes are split up for sexes (XX, XY, and YY chromosomes), including:
  + Maturation rate
  + Growth rate
  + Movement distances and probabilities
  + Sex ratios can now include a third and fourth sex for trojan individuals and sex chromosomes now specify the sex of each egg: Females (XX), males (XY), trojan males (YY), and Trojan females (YY).
* Growth: Individuals may grow based on von Bertalanffy equations (Bertalanffy 1983) and depending on where an individual settles during migration stages, as a function of temperature or other environmental surfaces and possibly admixture coefficient.
* Size- and age-based processes: some processes may operate as a function of individual size or age:
  + Maturation
  + Fecundity
  + Mortality
  + Movement probability
  + Capture probability
* Temporal systematic, stochastic, and demographic variability: processes and parameters can vary through time and linked spatially to environmental or climate variables at the patch level (see Table 2):
  + Carrying capacity
  + Spatial selection surfaces
  + Environmental movement surfaces and rules
  + Species specific movement strategies
  + Harvesting and survival
  + Migration and straying probabilities
  + Temperature surfaces controlling growth rates and growing days.
  + N0, class variables, and gene files: new individuals can be introduced in the simulations at a specified time. This feature could be used assisted migration or stocking scenarios.
* Assortative mating option using models options of self-preference, dominance-preference, or linear-preference with the first locus.
  + Hybrid index and amount of backcross is also calculated based on this locus.
  + Assortative mating can be implemented following M’Gonigle and FitzJohn (2009).
* Multiple species can be simulated in parallel across the same landscape(s).
  + Lotka-Volterra competition for multi-species applications
  + Alternatively, multiple species and hybridization may be considered using different input files and hybrid index (Allendorf et al. 2001).
* Natural selection: Spatial selection operates at different time steps (i.e., summer and winter) as migrating individuals settle into foraging and/or mating grounds. Selection can operate on survival at the genotype-environmental level or through genotype-maturation, genotype-growth or genotype-stray options. Additionally, selection can operate on survival based on individual homozygosity or the hybrid index (Allendorf et al. 2001).
* Vertical transmission (infected parents to offspring) is included with giving probability and tracking of infected individuals.
* Hybridization between species may be considered using different input files and hybrid index (Allendorf et al. 2001).
* Output genotype option in multiple formats: GENALEX (Peakall and Smouse 2006, 2012), STRUCTURE (Pritchard et al. 2000), GENEPOP (Raymond and Rousset 1995; Rousset 2008), or a general format.
* Format from CDMetaPOP can be read into the newest version of PopGraph (Dyer 2014) in R (e.g., R Development Core Team 2012).
* Additional mutation models added: step-wise forward and backward.
* Burn-in procedure options for both population dynamics, neutral genetics, and adaptive genetics.
* Ability to simulate SNPs or microsatellites. mtDNA option included.
  + 1. **Potential applications**

Our goal for this program is to provide the ability to model more complex demographic and genetic (‘demogenetic’; e.g., Labonne et al. 2008; Frank et al. 2011) processes for systems with up to hundreds of thousands of individuals and for multiple species simultaneously. CDMetaPOP is a spatially-explicit, metapopulation-based program in which individuals move between patches. The program simulates the spatial patterns in individual-based genetic data as functions of individual-, patch-, and population-based demographics and movement as a function of the underlying riverscape/landscape structure. CDMetaPOP’s representation of the spatial environment, population demography, and genetic processes provides a powerful framework to investigate the impact of ecological factors on the genetic and demographic structure of populations. This approach has already advanced knowledge of the patterns of genetic variation in spatially-explicit contexts. Example simulations have included:

* Quantifying the time to detect barriers (Landguth et al. 2010).
* Validating landscape genetics inferences and climate change applications (Castillo et al. 2014).
* Testing for the effects of population sample size and number of markers (Oyler McCance et al. 2012; Prunier et al. 2014).
* Assessing relative influence of adaptive versus neutral markers in detection of population genetic differentiation (Landguth and Balkenhol 2012).
* Comparing methods utility and approach in landscape genomics (Jones et al. 2014).
* Testing reintroduction strategies (Mims et al. 2019).
* Testing management strategies for managing invasive species, including YY male fish (Day et al. 2018, Day et al. 2020).
* Evaluating management of disease in tree systems (Landguth et al. 2017)

## **How does CDMetaPOP work**

The program is written in Python 3.8 utilizing Numpy (Oliphant 2006) and Scipy (2020) packages and provided with installation instructions for most platforms, along with sample input files. Multispecies simulations are run in parallel using the multiprocessing package and Queue data structures. CDMetaPOP is built on a driver-module, plug-in, docking architecture that allows for ease of future modular development. CDMetaPOP 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://github.com/ComputationalEcologyLab/CDMetaPOP>.

### Main Processes

#### **Dynamic landscapes**

Quantitative methods have a long history of application in the natural sciences. The use of models, ranging from simple heuristic approaches to highly theoretical and predictive tools, is widespread. However, few simulation tools frame individual-based numerical models within a spatially explicit context. This approach is critical in natural resource management since many of the actions that managers can control are implicitly and inseparably connected to the landscapes that comprise the habitats of the subject biota. For example, simulations using this model can address how increasing temperature through growth and/or movement capability will affect demogenetic population responses, how insertion/removal of barrier features may potentially decrease/increase connectivity, or how habitat improvement, or harvesting actions through time, can affect demogenetic population viability. The ability to explicitly account for systematic changes in the landscape is critical not only to many aspects of applied management of populations, but also to the development of resource monitoring and evaluation plans that are sensitive to expected changes because of intervention. Consideration of dynamic landscape processes (e.g., the interaction between discrete environmental manipulations such as habitat improvements and climate change) is especially important in this context.

#### **Neutral genetics**

CDMetaPOP uses the architecture of CDPOP (Landguth and Cushman 2010) for modeling neutral genetics with several significant additions. First, the user has several options for how to initialize their individual genotypes in the program. By specifying the number of loci and alleles to simulate, individual genotypes can be initialized randomly, by a given allele frequency file, or by actual known genotypes. Offspring then receive genotypes based on Mendelian inheritance. The user has the option to delay genetic exchange to allow population dynamics to reach equilibrium (i.e., burn-in). There are several mutational models that the user can consider (k-allele and variations on step-wise; See Table 1) with rate chosen by the user. An optional maternal-DNA marker can be used, as well as the option to simulate SNP genetic data.

#### 

#### **Natural selection (selection-driven genetics)**

Past versions of CDFISH (Landguth et al. 2012) modeled three sources of genetic variation: gene flow, genetic drift, and mutation. These past 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. CDMetaPOP now implements natural selection analogously to the adaptive or fitness landscape of allele frequencies (Wright 1932; Landguth et al. 2012). This new functionality enables extension of landscape genetic analyses to explicitly investigate the links between gene flow and selection in complex landscapes at the individual level. Three different types of selection can be specified:

1. Genotype-environment: The user specifies fitness landscape surfaces (values for each patch) for each genotype of a single diallelic locus or two diallelic loci that are under selection (one- or two-locus selection models). For example, three relative fitness surfaces must be specified for the three genotypes, AA, Aa, and aa, from the two alleles, A and a. Nine genotypes must be specified for the two-locus model. 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. The program will continue all other processes the same with an additional step implementing selection during each of the migration processes. In addition, these spatial selection surfaces can vary through time and/or can be considered for only mature individuals.
2. Genotype-maturation timing: The user specifies maturation curves (two parameters each for female and male probability of maturing) for each patch and for each genotype in the one-locus model. For example, AA could correspond to a slower maturation probability than aa and/or linked spatially to environmental effects.
3. Genotype-growth timing: The user specifies growth curves (five parameters) for each patch and for each genotype in the one-locus model. For example, BB could correspond to a larger growth response curve than bb and/or linked spatially to environmental effects.
4. Genotype-stray rate: The user specifies different stray rates for each patch and for each genotype in the one-locus model. For example, AA could correspond to a high stray rate (0.05) than aa (0.00) and/or linked spatially to environmental effects.
5. Hybridization index: The user specifies initial Hindex in the gene frequency files at the first locus by assigning values between 0-1 to each allele (e.g., AA corresponds to Hindex = 1 and aa corresponds to Hindex = 0). Future generations will be assigned an Hindex (or admixture) value between 0 and 1 based on the mean Hindex of the parents. Selection is implemented through a probability of differential survival of individuals as a function of relative fitness based on their Hindex value. For example, a hybrid index of 0.5 (offspring of an *AA* individual and an *aa* individual) may correspond to lower fitness than a hybrid index of 1 (offspring of two *AA* individuals). These processes can be implemented during different points within the simulation (e.g. summer or winter) and can be assigned to specific age groups.
6. Homozygosity: Selection is implemented through a probability of differential mortality of individuals based on their observed homozygosity. For example, offspring that have a relatively higher homozygosity as a result of mating of individuals that are related to each other can have a lower probability of survival. These processes can be implemented during different time points within the simulation (e.g. winter or summer) and can be assigned to specific age groups.
7. *Polygenic selection model:* Newer versions can implement polygenic selection using a linear additive model. This new module for CDPOP incorporates polygenic selection from linear regression models, as is traditional in quantitative genetics (Falconer et al. 1996; Wade et al. 2001). This enables extension of landscape genomics analyses to explicitly and fully investigate adaptive evolution in complex landscapes. As with the previous description of selection implementation in CDPOP, the user specifies the genotype for each individual at the initial time step (i.e., number of loci and number of starting maximum alleles per locus). Now, the user also has the option of choosing any number of loci and alleles, as well as any number of environmental variables that affect selection of the alleles. In this regression model, alleles at multiple loci associated with multiple environmental variables affect the fitness in an additive manner described by Equation 1:

*F\** = *b0* + [1]

where bijk are the effects of alleles Aijk on fitness given the value of environmental variables (Xi) for n environmental variables, a number of alleles considered at l loci, and b0 provides an option to shift the intercept of the linear model. A fitness value, F, between 0 and 1 is obtained by rescaling Equation 1 by (F\* - F\*min) / (F\*max - F\*min), where F\*max and F\*min are the absolute maximum and minimum, respectively, calculated from Equation [1]. Rescaling the lowest fitness to 0 ensures there are no negative fitness values. F\*max and F\*min are calculated before simulations begin on the hypothetical genotype-by-environmental space given the user defined bijk. Within the simulation workflow, CDPOP implements selection through differential survival (1 – F) of an offspring given the absolute fitness from Equation 1 at the location on the landscape where the dispersing individual settles. We provide a spreadsheet with the installation of CDPOP (betaFile\_General.xlsx) that allows users to investigate the fitness impact of beta values in a simple two-locus, two-allele model with one or two environments.

Users can also specify the genetic basis of local adaptation by modifying selection surfaces to reflect antagonistic pleiotropy (alternate alleles favored in different environments) or conditional neutrality (alleles favored in one environment but neutral in another; Anderson et al. 2013; Yoder and Tiffin 2018). Because values for the environmental variables are spatially-explicit and can have very different scales of variability, we require that a standardization (z-score) is performed for each environmental variable (e.g., elevation, precipitation, land-use categories, etc.).

\* Note that for options (5) and (6) these processes can be implemented together.

\* Note that for options (2) and (3) that these processes can be considered independent or associated. If independent, then locus A would control maturation timing and locus B would control growth. If associated, then locus A genotype determines the epigenetic response in locus B.

Simulations using the modules’ genotype-environmental associations can be used, for example, to warm high-elevation streams and ask how quickly alleles for thermal tolerance “invade” these areas. Similarly, simulations can ask how alleles for earlier spawning and migration tied to changes in thermal and hydrologic regimes alter demographics in headwater streams. Simulations with the maturation timing linked to genotypes and space could be used to ask how commercial harvest could lead to fisheries-induced evolution, or linking warmer temperatures to quicker maturity.

#### **Behavioral Plasticity**

CDMetaPOP can be used to simulate behavioral plasticity related to temperature avoidance through a genotype-environment interaction. This genotype by environment response occurs when an individual has the plastic region, allele “1” at the plastic locus, and is exposed to a patch with a temperature higher than a user-defined threshold. This shifts the state of the plastic allele to “2”, resulting in the individual not moving to patches above another user-defined threshold. This switch to being turned on lasts for the remainder of the individual’s lifespan. When the individual reproduces, the plastic region, but not the state, is passed on to offspring. So, each generation produces offspring with “0” or “1” at the locus, but not “2”.

There are four user options. First, users determine if temperature or habitat quality is the patch value to base the behavior on. Second, user options include selecting dominance, codominance, or recessive for the necessary “allele” response for habitat avoidance. In addition, users enter the percentage reduction to the probability of entering a patch that has the temperature or habitat value that will be avoided. Last, users can determine whether the avoidance occurs during either “Out” or “Back” stages.

#### **Demographics**

CDMetaPOP can model complex demographics through either age or size controls. The following describes how maturation, fecundity, mortality, and density-dependence operates within the program.

##### Maturation

If age control is operating, then the probability of becoming a mature, reproductive individual for both a male and a female is given in the ClassVars.csv input file.

If size control is operating, then the probability of maturation for males and females is modeled using a logistic function of size (Downs et al. 1997).

p(mature) = exp(A + B \* Length) / (1 + exp(A + B \* Length)) [2]

where parameters A (intercept) and B (slope) can be fit based on user length data. The default parameters are from Downs et al. (1997) length-maturity fit data.

##### Fecundity

Mean number of offspring per mature female can follow a Poisson deviate, normal deviate, constant draw, or random draw. Recruitment of age 0 individuals does not occur until after the ‘egg mortality’ stage which can optionally be used to model the loss of eggs prior to hatching.

If age control is operating, then a mean and standard deviation for eggs per mature female is specified in the ClassVars.csv input file.

If size control is operating, then the mean offspring (egg) number parameter is derived as a function of length with three functional forms available: 1) linear; (Downs et al. 1997), 2) exponential; (Bowerman 2013), and 3) power (Schillings et al. 2011). The user may specify the parameters of the fit based on size data to fecundity relationships, or by using literature values. For example, in the WCT model, egg mean numbers to length data were fit with the following linear equation and parameters (Downs et al. 1997)

Number of Eggs = -790.7 + 6.2 \* Length. [3]

In the BT model, an exponential fit was used (Bowerman 2013)

Number of Eggs = 9.5576 \* exp (Length \* 0.0181) [4]

In the LCT model, a power fit was used (Schill et al. 2010)

Number of Eggs = 0.0002 \* Length ^ 2.5989. [5]

Egg laying frequency, iteroparity and semelparity (or skipped spawning) can also be considered for the population as a whole.

##### Density-independent mortality

Several options are available to regulate mortality. All density-independent sources of mortality are specified with a mean and standard deviation and are truncated for the minimum/maximum interval on the desired distribution. Note that future versions will consider a recalculation on the unit interval to preserve the desired distribution (Todd and Ng 2001). Mortality variables can be specified to operate at the age class, size class, and/or patch level. These mortalities can be additive or multiplicative to simulate different rates of additional mortality operating on populations such as may be due to anthropogenic (e.g., harvest pressure) or environmental (e.g., flow or temperature; Ray 2007; Peacock and Dochtermann 2012) impacts.

At all 3 levels, two temporally independent mortality values can be implemented: mortality out (applied to individuals of a given class during overwintering or when migrants are away from spawning grounds) and, mortality back (applied to individuals of a given class when migrants are back at spawning grounds). Patch level mortalities are compounded (exclusively mutual or independent) with the class specific mortalities such that exposure to any spatially-explicit mortalities can be specified for any or all of the age or size classes specified. See Figure 1 and 2 for a schematic on how the multiplicative (i.e., independent) and additive (i.e., exclusively mutual) mortalities are used, respectively.

Finally, spatially-explicit egg mortality (applied to egg class or eggs at birth location) is specified in the PopVars and/or PatchVars file. Note, that Age 0 class variables in the ClassVars file does not correspond to the egg class, but rather the emerged class that enters the population (e.g., fry or fingerlings).



Figure 1: Patch-, size-, or age-based additive mortality option (1).



Figure 2: Patch-, size-, or age-based multiplicative mortality option (2). Note that if 0 is the response for any category of mortality, all other categories will be multiplied by 0 and result in no mortality at the relevant patch. If a response of ‘N’ is entered, then no mortality is used for that category. The combination of N \* N \* N will result in no mortality for the corresponding patch.

##### Density-dependent mortality

Individuals can move to new patches or remain in their current patch. Recruits are added into their natal patches. This means at each time step in the model, the number of individuals within each patch can potentially exceed resource capacity. The resulting level of mortality, reflecting competition for limited resources may be implemented via a simple “ceiling” function (i.e., population size will grow until patch carrying capacity (Kj) is reached using an exponential population growth model), an age-structured logistic growth model based on the Leslie matrix (Miller and Ankley 2004), or a density-dependent function with a hierarchical specification of patch- and class-structured carrying capacities (e.g., Ray and Hastings 1996).

The Leslie matrix model (Leslie 1945) is originally an exponential growth model, though we have implemented a Leslie model that has been modified for age-structured density-dependence (Miller and Ankley 2004):

[6]

where *nt­* is the vector of population age structure at time *t*, *M* is the Leslie matrix (Leslie 1945), and *Pt* is population size. Because CDMetaPOP is individual-based and not able to accommodate fractions of individuals like the standard Leslie matrix, we incorporated a stochastic process in which the leftover fraction in the projection for each age class is used for a probability draw to determine whether that last individual in the class survives.

The mechanism of density-dependence (‘packing’ option) is based on a set of logistic equations:

Nt+1 = Nt \* exp(R0 (1 - Nt / K)) [7]

with the ability to specify an ideal distribution of K among the population classes (age or length) and then by dynamically varying the “R0” parameter of the Ricker equation, together with recursive reallocation of under-utilized capacity according to the relative abundances of each class competing within the patch. This simulates length or age-based competition and partition of habitat patch resources, with the user able to specify the degree of overlap and relative influence of class capacity versus patch capacity.

When density-dependence is enabled, the number of individuals within each patch j is influenced at the overall patch level, as well as within each class i, by specifying patch carrying capacity (Kj), the distribution of carrying capacity across each class i in patch j (Kj,i), class abundance (Nj,i), and abundance of individuals within a patch (Nj). The following describes the steps taken to allocate space to individuals arriving at a given patch j:

Initially, we calculate the patch-level Ricker model moment (R0j) for each patch given total individuals arrived (Nj) and carrying capacity for that patch (Kj):

R0j = log(Kj / Nj) / (1 – Nj / Kj) [8]

with a special case if Kj = Nj, then R0j = 1.

Then, given the patch level R0j, the population of each class i within patch j (Nj,i) and the patch population (Kj), a class-specific, independent growth rate moment (R0j,i) is calculated:

R0j,i = exp(R0j \* (1 – (Nj,i / Kj))). [9]

This rate parameter is akin to the potential maximum growth rate of this class as a function of both overall patch level effects, and the intra-class competition effects (e.g., within cohort competition).

Next we describe the ideal distribution of Kj among classes, scaled to 1, as given by:

Kj,i = exp(C \* (i + 1)) / [10]

where C is the coefficient that describes the shape of the distribution of individuals among size classes (For WCT, C = - 0.6821, derived from length survey data in patch locations via otolith data (38 otoliths; SCL unpublished data) that were used to develop a von Bertalanffy age/length relationship which was then used to identify mean length for classes of interest identified in the model). Note, that this parameter, C, can be changed to fit user specific survey data.

Now, given Nj,i Kj,i, and R0j,i, the membership of each stage in each patch can be allocated a space. An interim variable (N’j,i)is used to denote the temporary membership during the recursive process described below. Starting with the largest class i in patch j, and iteratively working to the smallest class, the proportion of individuals for the given class that can fit into j is given by:

Nj,i’ = Nj,i;t0 \* exp(R0j,i \* (1 – (Nj,i;t0 / (Kj \* Kj,i))) [11]

If Nj,i;to > Nj,i’, then N’j,i individuals are randomly selected from Nj,i;t0 to survive to the next time step in that class i at patch j (Nj,i;t1) and the remainder are assumed to die. If Nj,i;t0 < N’j,i then all of the Nj,i;t0 individuals survive to Nj,i;t1. The remaining carrying capacity (N’j,i‑ Nj,i;t1) from that class is then made available to the remaining classes proportionate to the original distribution of Kj (equation 10).

Kj,i’ = Kj,i – (Nj,i;t1 \* Kj,i / N’j,i). [12]

Finally, the new available habitat Kj,i;t1 is calculated for the remaining classes as

Kj,i;t1 = exp(-C \* (i + 1)) + Kj,i’ / [13]

Alternatively, all available carrying capacity from one class can be allocated to the next smallest size class only, narrowing the breadth of competition among classes.

The recursive formulation of equation 10, beginning with the largest (or oldest) class, simulates a dominance hierarchy in allocation of space to different classes. This provides a density-dependent mortality on each patch as a function of Nj,i, Nj, and Kj. Note that there can be rare situations in which the new calculated Nj > Kj.

The ‘ceiling’ option can be implemented by turning off the ‘packing’ option and then population size will grow exponentially until Kj in each patch is reached. Extra individuals will be discarded at random with no preference for class. In addition, Kj can vary at each time step with given values as a function of a known variable, such as temperature, at each patch.

##### Competition

For multispecies applications, the user has the option to implement interspecific Lotka-Volterra competition, which allows for exploration of the effects of interspecific interactions on genetics (gene flow, population structure) and demographics. Because CDMetaPOP is individual-based, the original Lotka-Volterra equations are modified for use with an age-structured Leslie matrix model adapted for logistic growth (Miller and Ankley 2004), described in section

To implement the Lotka-Volterra competition effect, we adjusted the ratio of *Pt*/*K*to account for each competitor species when implementing density-dependent mortality, so that

[14]

where *x* represents a specific competitor species and α21 represents the competition coefficient of species 2 affecting species 1. Parameters required to implement this growth and competition model include age-structured fecundity and survival, carrying capacity, and competition coefficients for each species pair. Intrinsic growth rate is derived from the Leslie matrix.

#### **Patch versus class level controls**

The user can select for parameters that operate at the patch level, class level, or both. Patch level controls can, for example, be used to define life history strategies (e.g., resident versus migratory individuals via patch controls on migration rates) or implement environmental patch factors (e.g., mortality linked to drought) or harvesting at spatial locations along the network (by adding a patch and/or class specific mortality). To turn off class level controls, set all probability parameters within the class file to 1 and mortality parameters to 0. The additional complexity of class level controls would mimic behavioral differences between classes. Furthermore, the class level controls and specific input file (e.g., ClassVars.csv) is specified in the patch level input file (e.g., PatchVars.csv). This allows for different class level controls to operate spatially. Also, there are a few additional population level controls that operate on all patches and classes equally (e.g., environmental movement functions and thresholds).

#### **Age versus size control**

The user can choose parameters that operate based on age or size. Some parameters, such density-independent mortality, can be controlled for both age and size. An example of this usage could be when a mortality is implemented for the oldest age class, another may be the case where harvesting is differentially applied to different size classes.

#### **Growth**

Initialization of body sizes for individuals in the model is accomplished by specifying a mean and standard deviation of length in the ClassVars.csv input file. Size is allocated using a normal draw for each age class, i.

There are five growth options: (1) ‘N’ – turn off growth altogether, (2) ‘known’ – grow individuals at a known increment each year which is specified in the ClassVars input file with given age, (3) ‘vonB’ – grow individuals based on the von Bertalanffy equation [15] below, (4) ‘temperature’ – grow individuals based on spatially-explicit temperature values and von Bertalanffy equations [15-18] below, or (5) ‘temperature\_hindex’ – grow individuals based on the temperature model (Eqns 12-15) with the proportion of admixture (‘hindex’) defining the maximum potential growth (Eqn 16). Population growth rates are then specified in the PopVars.csv input file. Growth can occur at two times of the year, when they are back at their natal grounds or when they are away from their natal grounds.

##### Von Bertalanffy growth

The option ‘vonB’ will grow each individual based on the von Bertalanffy equation (Bertalanffy 1938),

L = Loo \* (1 – exp(– k \* (i + 1-t0))) [15]

where Loo and k are the theoretical maximum size and growth rate, respectively, i is the age class, and L is the new size.

##### Temperature and statistical fitting for growth

The option ‘temperature’ will use spatially-explicit temperature values at each patch location to drive growth for each individual, depending on which patch the individual is located, Xj. This option requires the user to specify three key elements: (1) von Bertalanffy parameters in Eqn. 12, (2) the temperature-growth response curve, and (3) the length of the intra-time unit (e.g., ‘season’) during which individuals are exposed to different temperatures at patches (i.e., grow days). In example models presented here, literature-derived temperature-growth curves and the von Bertalanffy model were fit to empirical data (e.g., field and hatchery) to obtain observed von Bertalanffy parameters at the observed temperatures and derive the values of X\_max and X\_CV that describe the temperature-growth curve approximated to a normal distribution. With optimal von Bertalanffy parameters, and temperature-growth relationship specified, an adjusted k (k’) (based on patch temperature) is used to calculate the seasonal increment in length (L\_inc):

k’ = k \* norm(X\_max,X\_CV\*X\_max).pdf(Xj) /

norm(X\_max,X\_CV\*X\_max).pdf(X\_max) [16]

and

L’ = Loo \* (1 – exp(k’ \* (i + 1 – t0) ) ) \*

norm(X\_max,X\_CV\*X\_max).pdf(Xj) /

norm(X\_max,X\_CV\*X\_max).pdf(X\_max). [17]

Then, at the given time step, t,

L\_inc = L’ \* exp(k \* (i +1)) \* (GD / 365) [18]

which gives the incremental growth given the environmental variable value, Xj, at patch j, adjusted for the grow days (GD) during that time period. Parameters X\_max, X\_CV, are the growth maximum and coefficient of variation that describe growth with respect to temperature. Figure 3 shows the resulting von Bertalanffy surface across a range of temperatures (2-20 degrees Celsius) for the WCT model.

Figure 3: Age-length-temperature growth data (Creston National Fish Hatchery; Piper et al. 1982; unpublished SCL data; Bear 2005)

##### Temperature and statistical fitting for growth with HIndex

The model for growth follows the ‘temperature’ model outlined previously. However, with this model, an individual’s percent admixture or ‘HIndex’ can be associated with the Loo term. If this model is chosen, then the user supplies two Loo parameters corresponding to HIndex = 0.0 and 1.0. For example, Loo\_1 and Loo\_2 would be used with the individuals HIndex to produce a unique incremental growth for each ‘sub-species’ (i.e., HIndex class). Loo in the above equation [17] is modified by

Loo\_hindex = HIndex \* (Loo\_2 – Loo\_1) + Loo\_1, [19]

Where HIndex is the individual’s proportion of *AA* genotype with values between 0 – 1 (e.g., 1.0 corresponds to *AA* and 0.0 corresponds to *aa*). Applications of this growth module could be used to test the differences in growth between known pure and hybrid species.

#### **Individual movement**

Landscape structure can govern movement of individuals between patches and this concept is incorporated through resistance (or permeability) surfaces whose values represent the stepwise cost from patch-to-patch of crossing each location (Spear et al. 2015). The model simulates movement as probabilistic functions of cumulative cost across these resistance surfaces (e.g., Landguth and Cushman 2010). Examples of types of cost distances from each patch to every other patch include least-cost path, Euclidean, or riverine distance. These distance matrices can be calculated from any program the user chooses (e.g., PATHMATRIX; Ray 2005, CIRCUITSCAPE; McRae 2006, UNICOR; Landguth et al. 2012c, ‘gdistance’; Ettan 2015).

CDMetaPOP considers five effective distance matrices (mating, migration out, migration back, straying, and dispersal) that are *j* X *j*, controlling where an individual can move as well as determining available mating partner(s). There are 11 movement functions with threshold options that determine the probability of moving to a patch as a function of effective distance. The movement functions with thresholding can be used to determine the dispersal kernel for an individual from its natal patch and also help determine the most probable mate partner. The movement functions are linear, inverse square, negative exponential, Gaussian, nearest-neighbor, random mixing, within patch random mixing, normalized distance, Pareto, and FIDIMO (Radinger and Wolter 2014). With the nearest-neighbor movement function, an individual moves to the available patch location nearest its initial location (or chooses a mate in a patch nearest to its own patch). Random mixing moves an individual to a patch location that is randomly chosen from all the patches on the landscape. Within-patch random mixing would only move individuals or choose a mate within the same patch (most useful for mating options or restricting individuals to patches, e.g., ponds). In linear, inverse-square, negative exponential, Gaussian, Pareto, and FIDIMO movement functions, individuals move a distance from their initial location based on a draw from a probability distribution proportional to inverse square, negative exponential, Pareto, FIDIMO, or Gaussian, and inversely proportional to a linear. For example,

prob(move to patch j) =

D \* 10^(-E \* effective distance to patch j) [20]

is the negative exponential that returns a probability for movement to a given patch as a function of the effective distance to that patch with user specified parameters D and E (see Table 1 for a complete list). With normalized distance, individuals are placed based on a draw from the probability distribution inversely proportional to the rescaled maximum and minimum effective distance distribution. The user specifies the maximum dispersal distance (in effective distance 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. There is an additional option in which probability values from patch to patch can be considered or a user can convert effective distance values beforehand to probability of movement.

The movement functions and matrices can be different for each sex, (e.g., controlling for sex-biased dispersal). In addition, asymmetrical effective distance matrices can be given, for example, to test hypotheses in hilly terrain or one-way versus two-way barriers. The simulation program assumes these matrices are ordered by column, then row, so it is important to format asymmetrical matrices correctly (transpose if necessary). In other words, if patch 2 is uphill of patch 4, then the cost of patch 2 to patch 4 < the cost of patch 4 to patch 2. This is reflected in the columns of the cost matrix (i.e., matrixValue[column,row]) or matrixValue[2,4] < matrixValue[4,2]. Note that if following the ‘gdistance’ vignette example, then no transpose is necessary. See Figure 4 below for how probability of migration back to natal grounds is considered.

The FIDIMO distribution function is based on a meta-analysis of fish movement studies in which the authors fit an equation to predict fish dispersal based on fish size, stream order, and aspect ratio of the caudal fin (Radinger and Wolter 2014). The function itself comprises two superimposed normal distributions, one for each of two components of the population: stationary and mobile. The function requires 3 parameters: (1) p = the share of the population belonging to the stationary component, (2) sigmastat = the mean movement distance of the stationary component, and (3) sigmamob = the mean movement distance of the mobile component. Where dispersal distances are unknown, one can use the R package ‘fishmove’ to predict the sigma values given fish size, stream order, and aspect ratio of the caudal fin. While FIDIMO is available for all movement types (e.g., mating, migration, etc.), we suggest limiting the use of this function to dispersal movement as was done in the meta-analysis.



Figure 4: Probability matrices (patch X patch that can be calculated based on effective distance matrices) are used to spatially place individuals in the landscape. Individuals can move based on 3 processes (probability matrices): migration out from natal grounds, migration back to natal grounds, straying, and dispersal. Probability matrices can also be used to choose mate partners from other patches. An individual can fail to return home either because the Prob(return home) = 0 or the random chance it did not make it home due to Prob(BA) < Prob(AB). See flow diagram (Fig. 5) on movement order of each process.

#### **Modeling capture probability**

Detection or capture probabilities can be specified at the patch or class level. To only consider class level detection probabilities, enter a value of 1 for each patch. To only consider patch level detection probabilities, enter a value of 1 for each class. The program will produce 0/1 detection for each individual as well as recapture information. The option to capture individuals can occur when they are back at natal grounds, when they are away from natal grounds, or at both time periods. This flexibility allows complex capture dynamics to be simulated, e.g., class probabilities might be used to simulate size-based gear selectivity bias, while patch probabilities might be used to represent spatially explicit patterns of capture probability related to habitat complexity.

#### **Modeling multiple species**

Two options exist for simulating multiple species: Independent (up to *n* species) or hybrids (2 species).

*Independent* multispecies applications for *n* species launch separate instances of CDMetaPOP across *n* processors. The number of species is indicated by the number of Popvars files listed in the first column of the RunVars file. All species must occupy the same landscape (although landscape attributes such as habitat suitability, movement cost, etc. may vary) but all other parameters are independent of other present species. Interaction among species may be implemented via Lotka-Volterra competition under a Leslie matrix framework modified for logistic growth. Future modules may add additional functionality such as predatory/prey or interspecific disease transmission.

*Hybrid* applications (or individuals with different genetics and demographic parameters) are run within a single instance of CDMetaPOP on a single processor, and are modeled as a single population differentiated by their respective ClassVars files and genetic inputs. Multiple allele frequency files can be given for each patch corresponding to different genotypes or species. Each allele frequency file can have a corresponding ClassVars file. Then, upon initialization in a given patch, individuals are assigned a genotype from a respective allele frequency file and tagged corresponding ClassVars parameters. The parameters from the tagged ClassVars file are then used at the individual level for corresponding class processes (e.g., class migration or stray rates, class specific mortality values). **Demographic and patch-level processes act the same for all individuals**, so it is only the class-level processes that can change based on an individual’s tagged file. For example, the resource allocation algorithm does not consider different ‘species’, so that inter-specific competition is not considered in this module. Offspring have an equal probability of receiving tagged ClassVars parameters from either their mother or father. There is also an option to weigh the probability of receiving either mother’s or father’s ClassVars file by the Hindex value. See section 3.1 and 3.2 on how to initialize the program with multiple species.

#### **Model output**

Outputs produced by CDMetaPOP consist of a series of comma delimited files with demographic and genetic information represented both temporally and spatially at the individual, class, and population levels. For example, at specified time-units; size, location, and alleles can be tracked and reported for each individual; deaths, growth, and capture are reported by class; and broad-scale demographic (e.g., births, movement rates) and genetic (e.g., heterozygosity) metrics are reported for each time unit for the entire simulated population. In addition, allelic scores can be generated and produced in file formats ready for analysis in GENALEX (Peakall and Smouse 2006, 2012), STRUCTURE (Pritchard et al. 2000), or GENEPOP (Raymond and Rousset 1995; Rousset 2008).

### Module details

CDMetaPOP’s main architecture schematic and workflow is shown in Figure 5a. Figure 5b shows an example of timing events or seasonal processes that occur during one time loop of the model. Each module (box) is explained in further detail below with example parameterization for a Westslope Cutthroat trout model (Fig. 5a) for the Sullivan Basin in northeastern Washington.

#### **PreProcess()**

This module is for pre-processing of input files supplied by the user. There are four input files to specify parameters and control for the job run, population, patch, and demographics (class). In addition, users specify effective distance matrices that guide movement of individuals. Optional genetic files can be included to initialize population genetic structure. Table 1 explains each input file and parameters in more detail.

1. Run parameters: Example RunVars.csv files are given in the ../data/ folder. These parameters apply to all species in a model run, and the first column is used to designate the Popvars.csv files for each species. As the model simulates stochastic processes, most applications will quantify mean and variability of population and 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 with each row in the file representing a different batch run.
2. Population control: Example PopVars.csv files are given in the ../data/ folder. The user must specify the population wide input parameters through an input file for each species. These are parameters that affect the population as a whole and regulate movement, selection, reproduction, growth, etc. Batch capability is also available through the PopVars file, but each species must have parameters set for the same number of rows.
3. Patch control: Example PatchVars.csv files are given in the ../data/ folder. The user must specify *j – (x, y)* located patches through an input file with information on carrying capacity (K) and initial starting numbers (N0) for each patch, *j*. The program simulates a density dependent K within each patch as a function of classes (see ‘packing algorithm’ above). Patches will populate and fill based on the packing algorithm, and the ability of individuals to move to and out of a patch is a function of the riverscape or landscape resistance surface. N0,j individuals are assumed to occupy random locations within each patch. The genotypes for each individual can be initialized by randomly choosing from a file containing allele frequencies for each locus or random assignment (i.e., maximum allelic diversity). Patch level controls on migration and straying can be implemented to help define life history strategies. Patch level mortality on all classes or eggs can be given and linked to environmental/climate variables or harvesting data. Temperature, growing days, and environmental/climate values can be given for each patch that determine growth and/or differential mortality tied to 1- or 2-locus selection models. Most all of these parameters can change through time through the module CDCLIMATE and through stochastic temporal variability.
4. Demographic control: Example ClassVars.csv files are given in the ../data/ folder. The user must specify a file containing the number of classes and distribution. Then, the age, sex, maturity, and body size of each individual is initialized in each patch based on user specified information within this file. The sex of each initial individual is randomly assigned but can be as an equal or unequal ratio. As of version 2.6, four sex classes can be considered; XX female, XY male, YY male, and YY female. Body size is drawn from a normal distribution with mean and standard deviation of body size supplied within this file. Probability of maturation for each sex may be implemented as a function of size or specified for each age class. Fecundity can similarly vary based on size or be specified by class within this file. Age or size level controls on migration, straying and mortality may also be specified here to define class movement and survival.
5. Effective distance matrices: Example effective distance files are given in the ../data/cdmats/ folder. The effective distance (i.e., cost distance) matrices are n x n and govern how individuals move between patches. Five different matrices can be considered for the five different movement processes (mating, migration out of natal patch, migration back to natal patch, dispersal, and straying). The same matrix can be used for all processes and the ability to turn off some of the process is an option through the use of movement rules (e.g., in terrestrial systems you may want to consider only migration out and/or dispersal, but not migration back as with natal homing in some aquatic systems). In addition, at any year these matrices can be replaced to simulate changing resistance surfaces in the module CDCLIMATE. The effective distance matrices can be calculated with any connectivity algorithm (e.g., least-cost path in UNICOR (Landguth et al. 2012c), ArcGIS, or the package ‘gdistance’ in R (Ettan 2015), or circuit theory (e.g., McRae 2006)).
6. Genetic files: Example allele frequency files with format needed for CDMetaPOP are given in the ../data/genes/ folder. A user can specify the genetics to be randomly assigned for each individual or provide allele frequency distribution files that can be linked spatially to different patches. Files can be used for each patch or groups of patches.

#### **GetMetrics()**

Various metrics for tracking and output are tallied and recorded in this module. See section 4 for the complete list.

#### **First Update()**

An initial time step file will be written to the output directory, giving information for each individual, including species, genotypes, sex, age, size, maturity, and infection status.

#### **Begin time loop**

After preprocessing of input data, a time loop begins. In Figure 5, examples are given for the corresponding modules/processes and timing events for spring (Fig. 5a) and fall (Fig. 5b) breeders (e.g., in aquatic systems, different seasonal spawners).

*Example*: For the WCT model, the beginning of the timing loop corresponded to the initial census data of June 1.

#### **CDClimate()**

This module controls additional variability through the ability to add in new individuals or patches, as well as update means for existing paramaters or environmental surfaces.

#### **Mate()**

Reproduction is either hermaphroditic or heterosexual. With hermaphroditic mating there are no distinct sexes, but individuals mate with other individuals according to the movement function choice (described below), exchanging genes in Mendelian reproduction. There is a selfing option that allows for a true hermaphroditic individual. In heterosexual reproduction, mated pair options include polygamy, polyandry, polygyny, and monogamy, which are specified through replacement options for males and females. For example, in polygamy mated pairs are one male (male replacement is ‘Y’) to possibly many or no females with females only mating once (female replacement is ‘N’), and the end of the mating process occurs when all females have mated. Frequency of egg laying or ‘skipped spawning’ (Rideout and Tomkiewicz 2011) may also be optionally specified to simulate iteroparous or semelparous life histories.

Assortative mating through self-preference, dominant-preference, or linear-preference (i.e., M’Gonigle and FitzJohn 2009) can be considered using the first locus as the selecting genotype. Hybrid index values are calculated by averaging the parents’ values for every individual and used to determine preferential mating. For example, a hybrid index of 1.0 corresponds to the original ‘parent species’, *AA*, a hybrid index of 0.0 corresponds to the original ‘parent species’, *aa*, and a hybrid index in between 0 and 1 refers to a hybrid individual that can be any makeup of the first locus. The equation for mating preference used from M’Gonigle and FitzJohn (2010) is:

i,j = *ci,j \* fj* [21]

where *fj* denotes the frequency of genotype *j* males (in this case, either *AA*, *Aa*, or *aa,* but using the calculated hindex [0 – 1]) in the female’s sample patch, and *i,j* is Kronecker’s Delta, which is equal to one when *i = j* and zero otherwise. i,j’s are standardized across all males. The user must specify the *c* parameter from [1-100000], where *c =* 1 corresponds to random mating and infinity would correspond to strict self-preference mating. Dominance-preference will adjust the Kronechker’s Delta to be 0 only for the hindex pairings 1.0 and 0.0. The linear-preference model follow similarly to the equation given in M’Gonigle and FitzJohn (2010), but adjusted slightly to consider similar hindex. Hindex values are rounded to 1 decimal place.

Probability of maturation can be specified for male and female age/size classes and updated at this time as a function of size with fitted parameters following Downs et al. 1997:

p(mature) = exp(intercept + slope \* Length) / (1 + exp(intercept + slope \* Length)) [22]

There are ten movement functions that define how individuals choose a mate on the landscape as a function of effective distance (Table 1). For example, with the nearest-neighbor movement function, an individual chooses a mate that is nearest its location. Random mixing chooses a mate that is randomly chosen from the total individuals in the population or patch. In the linear, inverse square, negative exponential, Gaussian, and Pareto movement functions, mate partners are chosen a distance from their initial location based on a draw from the inverse of the respective probability distribution. The random options do not use the effective distance matrix, and mate choice is an equal probability for all. In addition, there is a choice to not supply a functional form, but instead uses the rescaled cost values as the probability distribution (the choice of mate partners will then be dependent on the distribution of the effective distance matrix). The user specifies the maximum dispersal distance (in effective/cost units) an individual can travel on the landscape to find a mate. The probability is one at no distance from the original location and goes to zero at the maximum dispersal distance. There is also the option of supplying the software with the pairwise probability matrix rather than use one of the available transformations.

Individuals can be female (XX), male (XY), supermale (YY), or superfemale (YY). Males and YY males act the same, unless parameters associated with these chromosomes are specified. This is a beta version of this application and more details will be coming soon.

*Example*: In the WCT model, mating was without replacement for females and with replacement for males and occurred randomly within a maximum patch distance of 2 km (option 4). In addition, there were known patches that were not suitable spawning locations (i.e., set based on gradient, temperature and known barrier locations) and initialized with N0 = 0. Probability of maturation was based on Downs et al. (1997) size data.

#### **Offspring()**

Four options for determining the draw of mean litter size or egg numbers are available: (i) Each mated pair can have a number of offspring that is a bounded random draw based on a uniform probability distribution, (ii) a Poisson draw with specified mean, (iii) a constant number or (iv) a normal draw with specified mean and standard deviation. Mean litter size or egg number can be considered for multiple paternity, i.e., if a mean litter size was 8 and a female had 2 mate partners, then the female could either have 16 total offspring or 8 offspring split between fathers. A deterministic egg number can be specified by using a constant egg number.

Fecundity can be determined based on the mother’s class (age or size, i.e., the draw mean with standard deviation). If size is specified, then one of three functional forms (exponential, power or linear) may be used to model size-offspring relationships with parameters optionally developed from empirical data.

Sex assignment is randomly drawn from the parents’ sex chromosomes, i.e., parents can be XX, XY, or YY and thus, resulting offspring can be XX, XY, or YY.

Mendelian inheritance with *k*-allele mutation (rate chosen by the user) is used to generate the Age 0’s genotype once they are added to the population. Various mutational model options are available (see Table 1). A minimum of two loci and two alleles per locus can be used up to as many loci and alleles per locus within limitations of computational resources (recommended loci X alleles < 10^5). SNPs can be simulated using 2 alleles and X number of loci, but it is recommended to simulate with more alleles per locus and sample to a SNP equivalent (Landguth et al. 2012d).

Disease dynamics can be considered here, passed along from the mother based on a vertical transmission probability specified in the population control file (see CDINFECT module).

Mortality of offspring/eggs or survival to young of year (age 0)can be based on a draw mean mortality with standard deviation can be specified at the population level (PopVars.csv). In addition, spatially-explicit patch mortality can be considered in the patch level file (PatchVars.csv) and could be derived based on relationships for survival and environmental data, such as flow, temperature, etc.

The age class is still assumed to be 0 for the survived individuals after the module with size initialized from a normal draw with given mean and standard deviation (specified in the ClassVars.csv). However, this age class is not included in the population until the *DoEmigration()* module (or offspring to young of year are assumed to be resident individuals their first year). The age 0 individuals then enter the populationduring the overwintering period (migration away from natal grounds). These fingerlings are then considered in the packing algorithm in the *DoEmigration()* module. An ’egg delay’ option is available to delay the emergence of eggs by any number of years to address questions related to gestation or emergence timing.

*Example*: In the WCT model and following Downs et al. 1997, fecundity was modeled as a function of size, X, using a linear model with slope (Egg\_Mean\_par1) and intercept (Egg\_Mean\_par1) parameters to estimate mean egg number (*E\_mu*).

*E\_mu = Egg\_Mean\_par1 + Egg\_Mean\_par2 \* X*

*= -790.7 + 6.2 \* fork length (mm)* [23]

*Example*: In the WCT model, mean and standard deviation for egg survival (Creston National Fish Hatchery; Piper et al. 1982; unpublished SCL data; Bear 2005) were specified and a normal deviate was generated (mean probability of mortality 0.624 with standard deviation 0.1872), which was truncated at tail probabilities (i.e., values less than 0 became 0 and values greater than 100 became 100). Additional mortality was considered for temperature thresholds and flow regimes (e.g., scouring) for each natal patch.

#### **Second Update()**

A second update for individuals is conducted in which body size is incremented as specified (see *Growth* description for more details above). Age is also updated here and individuals in the last age class remain grouped with the last age class (e.g., Age 5+). The option of capturing (sampling) individuals can occur here as well. Finally, optional output files are written to an output directory giving information for each individual, including genotype, sex, age, maturity, individual ID, infection status, movement distance, and body size information (see Section 4 for more details on output files).

*Example*: As described above, in the WCT model, growth was modeled as a von Bertalanffy process where the growth rate parameter (R0) was dependent on temperature and growing days in the currently occupied patch. Hatchery growth data (Creston National Fish Hatchery), empirical data (unpublished SCL data), and published values (Piper et al. 1982; Bear 2005) were used to derive a temperature response curve for *R\_max* from which a growth increment was calculated based on temperature, grow days, age and the von Bertalanffy parameters. For a spring-spawning species, such as WCT, the growth at this time step represents non-overwintering growth, thus approximately 7 months (60%) of the year in the NE WA state study area.

For the time period in which individuals were back at their natal grounds, we used 123 grow days (June 1 – October 1) and extracted mean temperature values across this time range. The temperature model is described further below.

#### **Emigration()**

This module controls movement out of a patch (or natal population) for each individual. Multiple controls are available to specify a wide range of potential mechanisms for emigration. A patch level migration probability can be used in combination with an age/size class probability of migration. These are assumed independent events. In order to use just patch control, set all class migration probabilities to 1. In order to use just class control, set all patch migration probabilities to 1. In addition, the ‘set migration’ parameter can be used to fix migration for an individual (i.e., once an individual has migrated they continue to be an obligate migrator until death). If obligate migration is specified care must be taken to use the appropriate probability of becoming a migrant at a given age/size since rates will be cumulative across age/size classes.

Given these compounding probabilities of movement, the effective distance matrix with movement function and thresholds for emigration is used to place individuals in other patches (see the description in choosing a mate, as the same rules apply for choosing a patch location).

Following emigration, several sources of mortality may be implemented at the migration grounds. These include movement deaths (for individuals that are required to move but have no available patch), selection deaths (see section 1.2.1.3), and density-dependent mortality (see section 1.2.1.5.4).

*Example*: In the WCT model, migration out of natal patches was assumed to occur during late Summer and Fall. We assumed equal probability of migrating for every patch (turn off patch migration by setting values to 1.0), but used 0, 0.1, 0.2, 0.3, 0.4, 0.4, … for the size class migration rates. Migration probabilities are unknown for WCT and these values were assumed.

To place individuals in an overwintering patch, we used an isolation-by-riverscape resistance hypothesis that considered directional cost in the river network using elevation and barriers. We used the ‘gdistance’ package (Ettan 2015) in R to calculate the least cost path along this river network considering direction. The calculation for asymmetrical cost followed the gdistance vignette example for hiking in a hilly terrain, modified with the following equation for the ‘slope-speed’ relationship. If slope <= 0, then speed (i.e., permeability or conductance) for a fish was assumed to be 1.0. If slope > 0, then speed was assumed to be:

speed = 2 /(1 + exp(0.3 \* slope )) [24]

The Age 0 class is assumed to emerge as fry into the population with their size initialized around 31 mm with a standard deviation of 3 mm (Creston National Fish Hatchery; Piper et al. 1982; unpublished SCL data; Bear 2005).

#### **After Emigration Mortality()**

Density-independent mortality for each patch or age/size class can be applied at this stage. These constant mortality modules may be linked with environmental attributes (e.g., flow regimes) or harvesting giving additional mortality across space. A user has the option to apply mutually exclusive or independent mortality events here (Figures 1 and 2).

#### **Third Update()**

A module for a third update for individuals in which body size is incremented as previously described (see section 1.2.1.8), with the exception that the complement of the “growing season” parameter defined for each patch is used to calculate the growth increment appropriate at this time and patch location. Reproductive maturity is also simulated here (see section 1.2.1.5.1). Optional output files are written to the output directory giving information for each individual, including genotype, sex, age, maturity, individual ID, infection status, movement distance, and body size information (see Section 4 for more details on output files). This second optional output to file could be used to assess sampling at different times during a year. Note, age is not incremented here, as it was incremented in the previous (second) DoUpdate(). The option of capturing (sampling) individuals can also be implemented here.

*Example*: In the WCT model, this time period occurs over winter for a spring spawner (but note that this time period could be over summer for fall spawners). We used 242 grow days for the period Oct 2 – May 31 and extracted the mean temperature values from the daily temperature model (described above).

#### **Immigration()**

This module returns individuals back to original patch (i.e., home), strays individuals to a new patch (only applies to migrants; see Fig. 4), and/or disperses individuals locally. Again, both patch and age/size class specific probabilities for straying are available to the user and assumed to be independent events. In order to use just patch control, set all class straying probabilities to 1.0. In order to use just class control, set all patch straying probabilities to 1.0. If a successful straying probability event occurs, then these migrants move to any possible patch that the stray effective distance matrix with stray movement functions and thresholds allow. If an individual is not chosen for straying, then it is assumed to migrate back to its natal population. However, movement back to natal populations is also based upon a dispersal back effective distance matrix with movement function and thresholding, allowing asymmetrical cost/probabilities of movement to be considered. Here, directional effective distance matrices could prevent individuals that traveled too far from returning back (e.g., one-way barriers). If Prob(Emigrated Patch to Natal Patch) = 0.0, then an individual cannot return to its original patch location. If Prob(Emigrated Patch to Natal Patch) = Prob(Natal Patch to Emigrated Patch), then an individual is assumed to return back to its original patch. Considering asymmetrical probabilities and directionality, if Prob(Emigrated Patch to Natal Patch) < Prob(Natal Patch to Emigrated Patch), then a random draw is taken to decide if an individual will return back based on the relative difference or 1 – Prob(Natal Patch to Emigrated Patch) – Prob(Emigrated Patch to Natal Patch) / Prob(Natal Patch to Emigrated Patch). If an individual fails to return back, then an option for a mortality event or additional straying chance can occur.

By specifying matrices for both movement out (Emigration() process), and movement back (Immigration() process), simulating directional movement is possible. Movement back can also restrict individuals to stay in their emigrated patches, which allows this movement option to not be considered.

Following straying, dispersal may be implemented using an independent resistance surface and movement distribution to simulate annual movement by residents and migrants among local patches. Age/size-based dispersal probability is available.

Once individuals stray, migrate back to natal populations, and/or disperse, they are subject to selection mortality (see section 1.2.1.3) and density-dependent mortality via any of the demographic model options (exponential, logistic, packing; see section 1.2.1.5.4).

*Example*: In the WCT model, straying and migration back occurs during spring and summer for spring spawners. We assumed equal probability of straying for every patch (turn off patch straying by setting values to 1.0), but used 0.05 straying probability for each size class. For individuals that were successful under 0.05 probability of straying, we placed them into a new patch randomly, but only to patches that were suitable for spawning.

For individuals that did not stray, they were assumed to migrate back to their natal patches. We considered the same isolation-by-riverscape resistance hypothesis for migration out that considered directional cost in the river network using elevation and barriers (described above).

#### **After Immigration Mortality()**

Density-independent mortality for each patch or age/size class can again be applied at this stage. These constant mortality modules may be linked with environmental attributes (e.g., flow regimes) or harvesting to simulate additional mortality at this time step. A user has the option to apply mutually exclusive or independent mortality events here (Figures 1 and 2).

*Example*:In the WCT model, we applied a mortality catch of 75% to the last age class (Age 10) to simulate appropriate senescence of populations in the simulations. No additional size specific mortality or patch specific mortality was considered.

#### **GetMetrics()**

A second call to gather and track summary metrics within the time loop occurs here. A complete list is given in section 4.

#### **End time loop**

Repeat above processes until the end of simulation time (or generations) is reached.

#### **PostProcess()**

Once generations or years are completed, tracking metrics through the GetMetrics() are summarized and written to summary files. Options are included due to space constraints for summarizing patch metrics at given times.

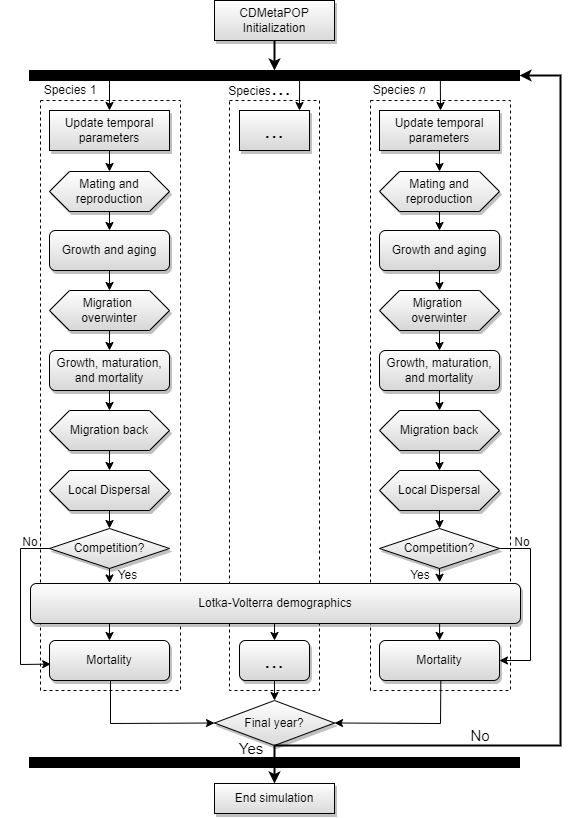


Figure 5: CDMetaPOP main work flow schematic with multiple species and an option for Lotka-Volterra competition.

# **Getting Started**

## **Dependencies**

### Baseline requirements

CDMetaPOP requires the Python3.8.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 v3.8.x).

### 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 v2.7) with a newer one (v.3.8.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.

### Python on windows

Windows (10, etc) 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.3.8.x) at:

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

### Obtaining NumPy and SciPy

We highly recommend using Anaconda for an all-in-one installation: <https://www.anaconda.com/products/distribution> is free for 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 the easiest installs if you are new to Python.

## **Installation**

### Installing Python, NumPy, and SciPy

Make sure that Python 3.8+, SciPy, and NumPy are installed, and available to you. You can test this by typing “python” at a command window. If python is available, you will 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, EPD/Canopy, Anaconda, etc., accepting defaults for the installation directory. On Windows, this will typically place the executables and libraries in c:/Python3.8/bin and the “site-packages” package tree for user installed Python modules in C:/Python38/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 an Anaconda or similar ‘all-in-one’ installation, then you can skip this step.

\* Note as of March 2018, Anaconda3 provides a nice all-in-one installation package with front ends very similar to RStudio (i.e., Spyder). You can select your Python version from this site: https://repo.continuum.io/archive/.

### Installing CDMetaPOP

The CDMetaPOP software requires no installation other than unpacking the zip archive supplied or downloaded from GITHUB (www.github.com/ComputationalEcologyLab/CDMetaPOP). Navigate to the directory on your PC where you wish to install CDMetaPOP and unpack the supplied zip archive file using a free archive tool like 7Zip (7z.exe), Pkunzip, Unzip, or equivalent. 7-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.

### Description of CDMetaPOP files

Three directories will be installed in the parent CDMetaPOP directory. A description of each follows:

1. src – CDMetaPOP source code
2. doc – documents
   * README.txt – a quick how to run CDMetaPOP instructions
   * CDMetaPOP\_user\_manual.pdf – this file
   * CDMetaPOP\_history.txt – Notes on history and version changes.
3. data – Example input files – see table 1 for further description of example files.

* cdmats/
  + Patch7\_Cdmatrix.csv – example effective distance matrix file for 7 patches calculated using UNICOR (Landguth et al. 2012) for riverine distance.
* genes/
  + allelefrequencyA.csv and allelfrequencyB.csv – example format for allele frequency distribution files for 10 loci and 5 alleles.
* classvars/
  + ClassVarsAS1.csv and ClassVarsAS2.csv – example demographic files with parameters/variables for age/size classes
* RunVars.csv, RunVars\_multispecies.csv – Contains parameters that apply to all species in a run, with options for 1-, 2-, and 3- species runs.
* popvars/
  + PopVarsS1.csv – population parameters corresponding to the example files for 1-, 2-, and 3-species runs.
* patchvars/
  + PatchVarsS1 – example file with parameters/variables for each patch with different options for up to 3 species.
* PatchVarsS1\_climate.csv – example file for how to implement CDClimate module at the patch level.
* Otherfiles/Patch\_r.csv – example correlation matrix for 8 parameters.

## **Example run**

### Command line run

The example runs are for 7 patches representing up to 3 species and individuals with an effective distance matrix calculated using a least-cost path algorithm through riverine distance (Fig. 6).



**6**

**4**

**5**

**1**

**2**

**3**

**7**

Figure 6: Example riverine system with 7 patches centroids as red. Variables and parameters correspond to each patch location (PatchVars.csv).

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 ‘PopVars.csv’ and ‘RunVars.csv’ files in the data directory specify the parameters that can be changed and used in a sample CDMetaPOP run. Open these files in your editor of choice. A spreadsheet program like Microsoft Excel, allows for easy editing of the tabular values.
3. The file ‘RunVars.csv’ defines which ‘PopVars.csv’ file(s) to use in the first column. For multiple species, each species’ PopVars filepath should be separated by a semicolon. The first column of the PopVars files then points to the patch files. The included files ‘PatchVars.csv’ will be in the same folder (../data/patchvars). ‘ClassVars.csv’ files are in turn specified in the ‘PatchVars.csv’ file and example ‘ClassVars.csv’ will be in the ../data/classvars/ folder.
4. Each row below the header line in the RunVars and PopVars file represents a ‘batch-run’ with information corresponding to a unique CDMetaPOP run. Batches in RunVars can supply all new parameters (including PopVars files), whereas batches in PopVars will follow parameters from the relevant RunVars batch. See Figure 7 for a flowchart containing the CDMetaPOP file structure, and Table 1 which contains a breakdown for each column header and the parameters that can be changed. The ‘Input’ (bolded) in the table listed is for the first row in the file. Make sure you save input files in the ‘.csv’ format – a comma delimited file – when you make changes to input parameters. Do not change the ‘Input’ (first row) labeling. Select ‘Yes’ or ‘OK’ for any Excel questions about saving in this format.
5. 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 CDMetaPOP src home directory (i.e., > cd C:\”homedirectorylocation”\src).
6. 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 CDMetaPOP.py C:/”homedirectorylocation”/data RunVars.csv output\_test”. Or a short-cut if your data is located at the same folder level as the src folder: “python CDMetaPOP.py ../data/ PopVars.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”.
* “CDMetaPOP.py” runs the CDMetaPOP program.
* “C:/’homedirectorylocation’/data” is the directory location of the input test files. You can point this directory to other project files, for example, So as projects accumulate you can rename input folders that contain the project specific files (e.g., dataWestslope or dataBullTrout). We suggest not having any spaces in your directory names.
* “RunVars.csv” is the parameter file (comma delimited) which can be renamed (e.g., “RunVars\_WCT.csv”). Caution should be taken when going between operating systems and saving this file as a .csv.
* “output\_test” is the name of the directory that will be created to house CDMetaPOP output in the specified home directory location.

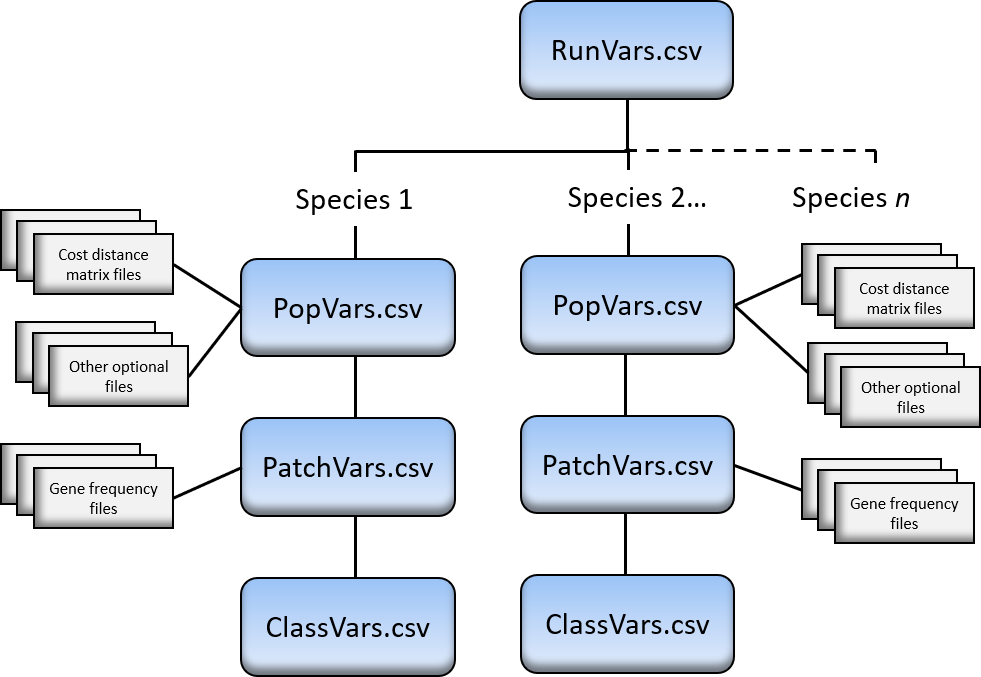


Figure 7. Input file structure for CDMetaPOP. All files should be housed in the ./data directory or subdirectories that can then be pointed to in the input files.

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 and folders (e.g., run0batchrun0mcrun0, run0batchrun0mcrun1, run0batchrun0mcrun2, run0batchrun1mcrun0, and run0batchrun2mcrun0) will be created in your CDMetaPOP home directory to store output from the separate batch and/or Monte-Carlo runs (each row in the RunVars and PopVars file corresponds to a separate batchrun and the number of ‘mcruns’ is specified in the RunVars file for each batch). Output folders will be located in the home directory location specified in the above step. The name of the output folder will include a unique date/time stamp in case you want to execute multiple CDMetaPOP runs in this same directory. CDMetaPOP will also produce a log file with program steps in your specified output directory. If parameters are such that all species become extinct before the specified generation time, then program will end. CDMetaPOP will provide error and feedback for parameters that are outside of an appropriate range or incorrectly entered.

# **Input**

This section lists the specific information for each input file and corresponding parameters used in CDMetaPOP. See examples provided for formatting (RunVars.csv, PopVars.csv, PatchVars.csv, and ClassVars.csv). The file headers listed are for the first row in each input file describing each file or parameter (do not modify this line in the input files).

Note on delimiters: Different delimiter options can be specified for different fields. In general, the delimiter ‘|’ or ‘bar’ is used for changing parameters over time and corresponds to the cdclimgentime field. The delimiter ‘~’ (‘tilda’) is used to assign parameters based on sex. The delimiter ‘;’ is the default and used in various contexts to split fields and parameters for patches primarily. The delimiter ‘:’ is also used in some instances to split parameters used for functions. Note that Loo uses one instance of ‘;’ in an Hindex option. See each specific field and header for the options.

**\*\***Parameters available for delineation by sex (~ delimiter) include default maturation age/size (mature\_default), movement parameters in the Popvars.csv file, maturation equation parameters in the Popvars.csv file, size/age mortality in the Classvars.csv file, migration, straying, and dispersal, and capture probability in the Classvars.csv file.

**†**Indicates a parameter that can vary over time using pipe delimiter (‘|’)

**Table 1**. List of all input parameters in each of the 4 primary input files along with example inputs and explanations for each option.

|  |  |
| --- | --- |
| **RunVars.csv file – run parameters apply to all species** | |
| *See RunVars.csv as the example input file for run level controls that apply to all species.* | |
| **File header in csv file (bold) and example input** | **Parameter description** |
| **Popvars**  ‘popvars/PopVarsS1.csv’ – Standard run parameters for a 1-species run  ‘popvars/PopVarsS1.csv;popvars/PopVarsS2.csv’ – Standard population parameters for a 2-species run  ‘PopVarsS1\_Climate.csv’ – Vary parameters over time  ‘PopVarsS1\_testMloci.csv’ – Multilocus (polygenic) application  ‘PopVarsS1\_MultiClassVars.csv’ – Hybrid application  ‘PopVarsS1\_IntroducePopulation.csv’ – Introduce individuals to the simulation over time | File that contains all population level parameters that apply to individuals in all patches. See section 3.2 for a description of each parameter in this file. |
| **sizecontrol**  ‘Y’ or ‘N’ | If ‘Y’ is specified, then body size will drive processes that are linked to that size class, e.g., in ClassVars.csv file – migration, straying, mortality – in PopVars.csv file – fecundity, maturation and growth parameters. E.g., for size 83 mm, all individuals will be sorted and those closest in size to 83 mm will be binned together. Then the mortality, migration, and straying values for row in input file with size 83 mm will be used for those individuals. Tracking numbers (see Section 4) are then summarized around these size class bins.  If ‘N’ is specified, the age will drive processes linked in ClassVars.csv file. Tracking numbers (see Section 4) are then summarized around the age classes. |
| **constMortans**  ‘1’ or ‘2’ | This option applies how the patch, size and/or age constant mortality value events are considered and compounded.   * ‘1’ – refers to mutually exclusive events or mortality percentages are added (i.e., P(A) + P(B)). * ‘2’ – refers to independent events or mortality percentages are multiplied. |
| **mcruns**  Any positive integer e.g., ‘5’ | The repeated number of simulations to be conducted for the Monte Carlo method (i.e., the number of replicates for 1 batch of parameters). Rows in the ‘PopVars.csv’ file indicate a new simulation run or batch run. Labeling begins counting with 0. In the output folder, and example with mcruns = 2 with 3 rows of parameters would give the following folder output: batchrun0mcrun0, batchrun0mcrun1, batchrun1mcrun0, batchrun1mcrun1, batchrun2mcrun0, batchrun2mcrun1 |
| **runtime**  Any positive integer e.g., ‘5’ | Simulation run time [generation or year]. File output indexed from 0 – (runtime-1). For example ind0.csv, ind1.csv, ind2.csv, ind3.csv, and ind4.csv would be outputted for a runtime of 5. |
| **output\_years**  ‘0’  ‘1’  ‘0|3|4’ | 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). |
| **Gridformat**  ‘genepop’  ‘genalex’  ‘structure’  ‘genepop’ or  ‘cdpop’ | This is the genes output format option:   * ‘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 ind{year}.csv. Note that this format can now be read into many population genetics programs (e.g., PopGraph in R (Dyer 2005)). * ‘general’ – for a general genotype output. The general format will follow Locus1a, Locus1b, Locus2a, Locus2b, …, LocusNa, LocusNb. Output will be labeled general\_ind{year}.csv * ‘genalex’ – for the program GENALEX (Peakall and Smouse 2006, 2012). Output will be labeled genalex\_ind{year}.gen * ‘structure’ – for the program STRUCTURE (Pritchard et al. 2000). Output will be labeled structure\_ind{year}.stru * ‘genepop’ – for the program GENEPOP (REF) or related programs. Output will be labeled genepop\_ind{year}.gen |
| **Gridsampling**  ‘N’ or  ‘Sample’ | An option to output genotypes before the Immigration step or when they are away from their natal grounds.   * Enter ‘N’ for output when individuals are at their natal grounds. E.g., ind0.csv, ind1.csv, … * Enter ‘Sample’ for output when individuals are away from their natal grounds. Two files for each year specified will be seen in the output file: e.g., ind0.csv, indSample0.csv, ind1.csv, indSample1.csv, … |
| **summaryOutput**  ‘Y’ or ‘N’ | Option to output summary metrics for each patch at the given time intervals. See Section 4 on output metrics. Depending on the size of the system and simulation time, these summary output files can be very big, so consider time and space limitations on your local machines when selecting this answer.   * Enter ‘Y’ to produce output summary metrics for each time interval and patch. * Enter ‘N’ to not produce output metrics. |
| **cdclimgentime**  ‘0|5|10’ or  ‘0’ | To initiate the CDClimate module, this is the generation or year that the next variable or effective distance matrix will be switched.   * Place only a ‘0’ here to start simulations with one surface (and initial parameters) and continue to use throughout runtime. This is the default value. * You can specify multiple years by separating each generation by ‘|’. Then in the following surface columns (described immediately below), a separate file can be given for each generation. |
| **startcomp**  Any positive integer .g., ‘5’ | Year of the simulation at which Lotka-Volterra competition begins. Prior to this year, no interaction will occur among species. |
| **implementcomp**  **‘**Back’ | Option for when to implement Lotka-Volterra competition. Currently, the only available option is when individuals are ‘back’ on natal grounds. |
| **Run parameters and output – PopVars.csv file** | |
| *See PopVarsS1.csv for an example input file. This file controls species-specific population level parameters.* | |
| **xyfilename**  ‘PatchVarsS1.csv’, ‘PatchVarsS2.csv’, ‘PatchVarsS3.csv’ – see example supplied for 7 patches for up to 3 species.  ‘PatchVars\_climate.csv’ – an example on how to use cdclimate module changing parameters.    ‘PatchVars\_IntroducePopulation.csv’ – an example for how to introduce additional individuals at given time units.  ‘PatchVars\_MultiClassVars.csv’ – an example for how to use multiple ClassVars files to simulate more than one species or groups of individuals.  ‘PatchVars\_MultiLocus.csv’ – Example for use of multilocus (polygenic) selection | The *n-(x,y)* patch locations with 48 columns/fields of information for each patch. This is a comma delimited file. Some fields are optional and will not be considered – leave default values.  \*parameters can vary through time either systematically or through temporal variation. Parameters with ‘\_Std’ can be used for temporal variation and to create a covariance matrix with selected parameters (e.g., correlation\_matrix: Patch\_r.csv). See CDClimate section for details on how to implement systematic temporal variation. |
| **Popmodel**  ‘N’  ‘logistic’  ‘packing’ or  ‘packing\_1’ | The choice of population growth models.   * Enter ‘N’ – this population model is essentially exponential growth where Nj(t+1) = birth-rate \* Nj(t) – death-rate \* Nj(t). Population numbers can reach the set carrying capacity for each patch, but not exceed this number. This model is considered for all patches after migration out and after migration back. If Nj > Kj, then a random sample from the patch is removed. * Enter ‘packing’ to consider the density-dependent class specific population model, where competition is simulated among all classes simultaneously (section 1.2.1.5.4). * Enter ‘packing\_1’ to consider the density-dependent class specific population model, where competition is simulated only between adjacent classes. * Enter ‘logistic’ to use the density-dependent age-structured Leslie matrix model modified for logistic growth. Note that this model is only applied when individuals are ‘BACK’ and no density-dependent model is applied when individuals are ‘OUT’. Leslie matrix survival(in the form of mortality percentages) and fecundity values must be entered in the ClassVars file fields ‘Age Mortality Back %’ and Fecundity Leslie, respectively. If density-independent mortality is wished, then it will only be applied during the DoMortality() ‘OUT’ phase to avoid conflicting mortality parameters in both fields. |
| **Popmodel\_par1**  ‘-0.6821’ | The parameter used for the ‘packing’ and ‘packing\_1’ options for Popmodel that shapes the ideal class distribution. See section 1.2.1.5.4 for more details. |
| **Correlation\_matrix**  ‘Patch\_r.csv’ or  ‘N’ | File name for the correlation matrix used to generate correlated variables at the patch level. Currently, there are 8 parameters in which the user can specify correlation structure: K, popmort\_out, temperature\_out, growdays\_out, popmort\_back, temperature\_back, growdays\_back, eggmort. See ‘otherfiles/Patch\_r.csv’ for the order and example of each potential correlated variable. Enter ‘N’ and the covariance matrix will not be used. |
| **Subpopmort\_file**  ‘3popmort\_PhaseI.csv’ or  ‘N’ | Percent mortality for a dispersing individual into another subpopulation designated by the SubPatchID column. This is a matrix of mortalities that can be asymmetrical. Columns in the matrix correspond to TO, for example, in the matrix for 3 SubPatchIDs:  100 50 25  0 100 50  100 75 5  The added mortality for an individual located in SubPatchID 2 that attempts to move to SubPatchID 3 would be 75%. This function is also applied during the Mate module. For example if a male (or pollen) from SubPatchID 3 attempts to mate with a female from SubPatchID 1, then there would be a 25% chance of this mate pair not occurring.  Note that the matrix order corresponds to the ordered SubpatchIDs counting from 1, 2, 3, …, so this option would not work correctly if the SubpatchID is not consecutive.  Example matrices are given in ‘otherfiles/3popmort\_PhaseI.csv. This field is also linked to the cdclimate module and files can be swapped out at the given cdclimategentime. |
| **egg\_delay**  Any integer ≥ 0 e.g., ‘1’ | Number of years between mating and gestation/emergence. |
| CDClimate and movement surfaces with functions | |
| *See PopVarsS1.csv rows that contain multiple values in the cdclimgentime field, e.g., 0|1 and PatchVars\_climate.csv as an example for how to implement systematic changing variables and surfaces. Examples of commonly changed variables include*   * *Movement surfaces, functions, and thresholds (in PopVars.csv input and described below)* * *K (in PatchVars.csv input)* * *N0 for translocation/stocking scenarios* * *Mortality at the patch control (in PatchVars.csv input)* * *Migration and straying probabilities at the patch control (in PatchVars.csv input)* * *Patch specific temperature and grow days for two times of the year (e.g., summer and winter) (in PatchVars.csv).* * *Spatial selection surfaces for genotype-environment, -maturation, and –growth processes (in PatchVars.csv and described in Section 3.8)* * *SubPatchID mortality matrix.*   **\*\*** Indicates that a parameter can be different for each sex class (~ delimiter).  **†** Indicates that a parameter can vary with time (| delimiter). | |
| **NOTE: If turning off migration, straying, or local dispersal movement is desired, the best way to accomplish this is to set probabilities for migration, straying, or dispersal to 0 in the ClassVars file.** | |
| **Mate\_cdmat†**  ‘Patch7\_CdmatrixS1.csv’ – an example effective distance matrix used for mating movement.  ‘Patch7\_Probmatrix.csv’ – Contains probabilities rather than cost distances – for use with movement option 9.  ‘Patch7\_Probmatrix\_onewayBarriers.csv’ – An example of how to set up a landscape with one-way barriers such as dams  ‘Patch7\_Probmatrix\_onewayBarriersXRiverineS1.csv’ – An example of how to set up a landscape with one-way barriers combined with cost distances. | An *[n x n]* effective distance matrix for mating movement, where *n* is the number of patches on the landscape. This is a comma delimited file. See the example given for formatting this file in the data/cdmat/ folder. This file can be calculated from any program you choose (e.g., PATHMATRIX, CIRCUITSCAPE, UNICOR, COSTDISTANCE, and with R gdistance functions).  This file can be asymmetrical (see gdistance package for calculating asymmetrical cost) and the order is assumed to read columns, e.g., costVal[1,2] > costVal[2,1] if for example, patch 1 is higher in elevation than patch 2. When using asymmetrical cost, then double check that this is in the correct column order.  For mating movement, this matrix is from the perspective of the female. That is, given a female in patch 1, then the matrix is used to find all possible male mates or how far does the male move to find the female. Furthermore, males return to their original patch location after mating. Therefore, caution is advised when using asymmetrical matrices with the mate movement process. If asymmetrical movement is not understood for a system, then it is strongly encouraged to use matemovement option ‘6’ (described below).  If the CDClimate module was initiated with multiple years in the RunVars field ‘cdclimgentime’, then the same number of surfaces may be given here separated by a ‘|’, e.g., ‘Patch7\_Cdmatrix.csv|Patch7\_Cdmatrix.csv|Patch7\_Cdmatrix.csv’ could correspond to the example given for reading in a new surface at generations ‘0|5|10’. This same method can be used for all cost matrix fields.  Note that you can place these files in a separate folder, just make sure that the field still points to the correct files. |
| **migrateout\_cdmat\*\*†**  ‘Patch7\_CdmatrixS1.csv’ or ‘N’ | Cost matrix for migration out of each subpopulation. Same description as ‘mate\_cdmat’ except this matrix is from the perspective of each individual’s potential migratory ability. It also can be the same file as the mate, migration back, straying, or dispersal effective distance matrix.  Note, this module, DoEmigration() can be skipped by entering ‘N’ here. Careful use of this option is advised, as the entire module will be skipped, as well as population model check and ‘Out’ selection options in this module. Tracking numbers during this module will be displayed as ‘NA’ (see output section on more information). |
| **migrateback\_cdmat\*\*†**  ‘Patch7\_CdmatrixS1.csv’ | Used for migration movement back to natal subpopulation. |
| **stray\_cdmat\*\*†**  ‘Patch7\_CdmatrixS1.csv’ | Used for movement of straying to other patches.  Here, in order to stray individuals randomly, use the movement function answer ‘4’ below. You can use a matrix of 0s and 1s to control for random dispersal while neglecting some patches. |
| **dispLocal\_cdmat\*\*†**  ‘Patch7\_CdmatrixS1.csv’ | Used for movement of dispersal to other patches. If turning off local dispersal is desired, set dispersal probabilities to 0 in ClassVars. |
| **Matemoveno†, migrateOutno\*\*†, migrateBackno\*\*†, StrayBackno\*\*†,**  **dispLocalno\*\*†**  ‘6’ – Movement function answer for probability. | You can control the probability of movement and effective distance distribution by transforming the cost matrix through functions. 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 a cost value exceeds the threshold provided by the user, then probability = 0.   * ‘1’ = Linear: probability = (1 – (1/Threshold) \* Effective Distance) * ‘2’ = Inverse Square: probability = (1 / (Effective Distance^2)). This function gets rescaled between 0-1 using the min and threshold of the inverse square effective distance. * ‘3’ = Nearest Neighbor: Moore neighborhood random neighbor chosen. Often the case however, is if patches are not on a uniform grid, then the closest neighbor will be chosen. * ‘4’ = Random Mixing: Will consider the cost distance threshold. Use the maximum effective distance in the threshold field if you want to consider the entire population as random movement. * ‘5’ = Negative Exponential: probability = (A \* 10^(-B \* Effective Distance)). This function gets rescaled between 0-1 using the min and threshold of the negative exponential effective distance. * ‘6’ = Random Patch: No movement out of current patch. The function could be used to test defined patches with no movement between, or to force mating within a patch. * ‘7’ = Gaussian function: probability = A \* exp ( - (Cost Distance – B)^2 / (2 \* C^2)). This function gets rescaled between 0-1 using the min and threshold of the Gaussian effective distance. * ‘8’ = Use the cost distance matrix: probability = 1 – the cost distance matrix rescaled to the min and threshold. * ‘9’ = Provide a probability matrix rather than a cost distance matrix in the cdmat column: no function is applied to values and the straight probability matrix is used. * ‘10’ = Pareto function: For divide by zero issue, Cost distance = cost distance + b and probability =(a\*(b^a))/(cost distance^(a+1)). * ‘11’ = FIDIMO distribution function, used to predict a leptokurtic dispersal distribution based on stream order, fish size, and aspect ratio of the caudal fin (Radinger and Wolter 2014). a = proportion of the population belonging to stationary component, b = mean dispersal distance of stationary component, c = mean dispersal distance of mobile component. See R package ‘fishmove’ to obtain values for b and c. mu indicates upstream/downstream movement, currently set to 0. Probability = a \* (1/sqrt(2pi\*b^2))\*exp(-((x-mu)^2)/(2b)) + (1-a) \* (1/sqrt(2pi\*c^2))\*exp(-((x-mu)^2)/(2c)) |
| **matemoveparA†, migrateOutparA\*\*†, migrateBackparA\*\*†, StrayBackparA\*\*†, disperseLocalparA\*\*†**  ‘0.0005’ | This is the A parameter used for the function answer ‘5’, ‘7’, ‘10’, and ‘11’. Specify same number of values here if CDClimate is initiated, e.g., 0.05|0|0. |
| **matemoveparB†, migrateOutparB\*\*†, migrateBackparB\*\*†, StrayBackparB\*\*†, disperseLocalparB\*\*†**  ‘0.01’ | This is the B parameter used for the function answer ‘5’, ‘7’, ‘10’, and ‘11’. Specify same number of values here if CDClimate is initiated, e.g., 0.05|0|0. |
| **matemoveparC†, migrateOutparC\*\*†, migrateBackparC\*\*†, StrayBackparC\*\*†, disperseLocalparC\*\*†**  ‘0.01’ | This is the C parameter used for the function answer ‘7’ and ‘11’. Specify same number of values here if CDClimate is initiated, e.g., 0.05|0|0. |
| **Matemovethresh†, migrateOutthresh\*\*†, migrateBackthresh\*\*†, StrayBackthresh\*\*†, disperseLocalthresh\*\*†**  ‘max’, ‘10max’, or cost units | A threshold option (in effective distance units) for how far an individual can search for a mate (i.e., mate selection), migrate, stray, or disperse to a new patch location.   * Enter ‘max’ to consider all patches for movement or all individuals for mate selection. * Enter ‘%max’ to consider a percent cost distance movement for mating. The % is an integer value between 1 and 100. 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. * Enter a value in cost distance units to consider movement within a set threshold. |
| **HomeAttempt**  ‘mortality’, ‘stray\_emiPop’, or ‘stray\_natalPop’ | There is a possibility that a migrant that did not become a strayer attempts to immigrate back to its original natal patch but cannot. This can occur either because K = 0 at that original natal patch location, e.g., a catastrophic event, or the cost to move back to the original natal patch exceeds the specified movement threshold (e.g., an individual emigrated too far perhaps down over a barrier). The following options are available for modeling these special case individuals:   * ‘mortality’ – the individual is assumed to die. * ‘stray\_emiPop’ – the individual will attempt to stray to accessible patch locations using the stray matrix movement rules and from its emigrated patch. Note that there still is a possibility that the stray matrix rules will produce inaccessible locations in which case the individual is assumed to die. * ‘stray\_natalPop’ – the individual will attempt to stray to accessible patch locations using the stray matrix movement rules and from its natal patch. The difference here and rationale is that the individual wants to stray to a patch closest to its natal location. A recursive operation is used to check to make sure there are indeed patch locations in which probability is greater than 1 to ensure that an individual does not travel up over a barrier. Note that there still is a possibility that the stray matrix rules will produce inaccessible locations in which case the individual is assumed to die. |
| Reproduction options | |
| **sex\_chromo**  ’2’,’3’,’4’ | Enter the number of sex chromosome combinations.   * For the simplest option, enter ‘2’ for XX females and XY males * Enter ‘3’ to add YY males * Enter ‘4’ to add YY females   Initial sex ratios are specified in ClassVars. If different values by sex are desired for any given parameter, the number of values split by the ‘~’ delimimter must match the value in **sex\_chromo**. If only a single value is provided for a given parameter, that value will be applied to all sexes. |
| **sexans**  ‘Y’, ‘N’, or ‘H’ | This option determines heterosexual or asexual reproduction   * Enter ‘Y’ for sexual reproduction. In sexual reproduction, mated pairs consider males and females with or without replacement. Only mature individuals are considered in the pairing. * Enter ‘N’ for asexual reproduction. All mature individuals are considered for mating 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. * Enter ‘H’ for hermaphroditic reproduction. Individuals are initialized either hermaphrodites or as males (using sex ratio in ClassVars see below). All mature individuals are considered for mate selection. Hermaphrodites will self fertilize with probability of mating with males. All offspring for this mating structure will either be hermaphroditic or males using the egg sex percent option (described below). |
| **selfans**  ‘Y’  ‘N’  [0-1] probability | Consider selfing, i.e., individuals exchange genes with self.   * Enter ‘Y’ to consider selfing * Enter ‘N’ to turn off selfing. * Enter a probability value between 0 and 1 corresponding to the sexans 'H' or hermaphroditic reproduction system. This probability is used to self-fertilize individuals that are hermaphrodites in the 'H' initialized systems, else mate selection with males occurs. |
| **Freplace**, **Mreplace**  ‘Y’ or ‘N’ | These options determine the mating structure, i.e., monogamy, polygyny, or polyandry.   * Enter ‘Y’ to consider females or males to mate with replacement (i.e., multiple mates). * Enter ‘N’ to consider females or males to mate without replacement (i.e., multiple mates). |
| **AssortativeMate\_Model**  ‘1’  ‘2’  ‘3a’ ‘3b’  ‘4’  ‘5’ | Assortative mating potential based off of the M’Gonigle and Fitzjohn 2010 models. The difference is that a Hybrid Index is calculated for each individual and used to sort similarity of individuals and preferential mating. Five options include:   * ‘1’ – Random mating. * ‘2’ – Strict self-preference mating. That is, if there is not a similar ‘hindex’ in the neighborhood search, then that female will not have a mate pairing. * ‘3’ – Self-preference model. A ‘c’ assortative mating factor is used, along with the male hindex frequencies. The difference from ‘3’ and ‘2’ is that mating can still occur if there is not a similar hindex pairing. The equation is:   Phi\_i,j = AssortativeMate\_Factor ^ Kronecher’s Delta \* fj, where fj denotes the frequency of males with hindex j in the females patch, and Kronecker’s Delta is 1 when i = j and zero otherwise. Phi\_i,j are standardized across all males within the females patch.   * + Note there is an option here for ‘3a’ and ‘3b’. ‘3a’ represents the standard model described above. ‘3b’ can be used in the case of hybridization between species, where Kronecker’s Delta becomes 1 for every pairing except for the extreme hindex values of 1.0 and 0.0. See the Assortative Mating description for more information. * ‘4’ – Dominance-preference model from M’Gonigle and Fitzjohn 2010. Any hybrid individuals are phenotypically indistinguishable from individuals with hindex =1.0. Therefore, all individuals with hindex > 0.0 mate with equal probability, and individuals with hindex = 0.0 have reduced probability of mating with all individuals where hindex > 0.0. * ‘5’ – Linear-preference model similar to M’Gonigle and Fitzjohn 2010, but modified to pair according to hindex. The modified equation is:   Phi\_i,j = (1 + (1 – abs(Female\_hindex – male\_hindex)) \* (AssortativeMate\_Factor – 1)) \* fj. |
| **AssortativeMate\_Factor**  ‘1’-‘1000000’ | Assortative mating factor *c* in equation [21] in text and used for options 3-5 above. Self-preferential mating can be considered based on the genotypes *AA, Aa,* and *aa* using the hybrid index of an individual (e.g., a hybrid index of 1.0 corresponds to the original parent genotype *AA*, a hybrid index of 0.0 corresponds to the original genotype *aa,* and a hybrid index in between 0 and 1 corresponds to a hybrid individual).   * A value of ‘1’ will be random mating and similar to model option ‘1’. * Values between ‘1’ – ‘1000000’ will increase the probability of self-preference mating.   \* Note that if *c*is not random or strict self-preference mating, then a patch is chosen from which a male will be drawn and if there are males there, then a pairing will occur regardless if there is the same male genotype to match with the female. |
| **mature\_default\*\***  ‘age6’, ‘size200’, ‘N’  \*\* This field can be different for each sex class (~ delimiter). | Allows the user to set a default age or size for reproductive maturation. Value must include the word ‘age’ or ‘size’ to indicate whether age or size will dictate default maturation.  If an individual reaches this age/size, the program assumes that the individual will become mature if not already. If you wish to ignore this option, enter ‘N’. |
| **mature\_eqn\_slope\*\*,**  **mature\_eqn\_int\*\***  ‘0.5’, ‘-0.1’  \*\* This field can be different for each sex class (~ delimiter). | If size control is specified, then these are the slope and intercept parameters that describe probability of maturation as a function of size. These values can be modeled separately for males and females. The function used is from Downs et al. 1997:  P(mature) = exp(mature\_eqn\_int + mature\_eqn\_slope \* Length) / (1 + exp(mature\_eqn\_int + mature\_eqn\_slope \* Length))  If age control is operating, then these values are ignored and the probability values entered in ‘ClassVars.csv’ are used.  Once an individual becomes mature, then it stays mature. |
| Offspring/litter/egg/sex ratio options | |
| **offno**  ‘1’, ‘2’, ‘3’, or ‘4’ | Draw for number of offspring (e.g., eggs or litter or clutch size) for each mate pair.  Only used if age control is specified for **sizecontrol** response, then this parameter combines with the **Fecundity Ind** mean and standard deviation values in the ‘ClassVars.csv’ file for the corresponding draw choice below.  If size control is specified, then the **Egg\_Mean** parameter options in PopVars.csv will determine the mean fecundity value instead of **offno** (described in more detail next).   * ‘1’ – random draw between 0 and mean fecundity value. * ‘2’ - Poisson draw of mean fecundity value. * ‘3’ - constant number of offspring of mean fecundity value. * ‘4’ - normal draw with mean fecundity and standard deviation. |
| **offans\_InheritClassVars**  ‘random’, ‘Hindex’, or ‘mother’ | If multiple genotype and ClassVars files are specified, then this is the option for how offspring will inherit their parents ClassVars files. If multiple classvars files are specified, mutation options are unavailable.   * ‘random’ - equal probability of receiving its mother’s or father’s ClassVars file and associated parameters. * ‘Hindex’ - weight the probability of receiving ClassVars files associated with Hindex values. That is, in the first patch, the first allele frequency file could have an Hindex of 1.0 and the second file an Hindex of 0.0. Offspring with given Hindex values will then be assigned either the first or second set of ClassVars parameters depending on a random draw weighted by Hindex. * All offspring inherit the mother’s classvars, useful for Myy applications. |
| **equalClutchSize**  ‘Y’ or ‘N’ | Determines whether each mate pair will have equal clutch size.   * Enter ‘Y’ for an equal clutch size for each female of mean fecundity value resulting from ‘**offno**’ choice above. * Enter ‘N’ for an unequal clutch size for each female.   For example, if a female mates with 4 males and ‘**offno**’ = 3 with fecundity value of 2, then with an equal clutch size (‘Y’) she will have 2 offspring total and the fathers and resulting genes are randomly selected from the 4 mating events. She will have 8 offspring (2 from each mate event) with ‘N’ entered. |
| **eggFrequency**  ‘0’ - '1’ | Probability with which the female will lay eggs. This value ranges between 0 – 1. E.g., 0.5 would mean that a mature female will lay eggs every other year. |
| **Egg\_Mean\_ans**  ‘exp’, ‘linear’, or ‘pow’ | If size control is specified, then this function choice is used to produce the number of offspring a female will have. If age control is specified, then **offno** and **Fecundity Ind** values in ClassVars are used.   * ‘exp’ – *E\_mu = Egg\_Mean\_par1 \* exp (Egg\_Mean\_par2 \* Size)* * ‘linear’ - *E\_mu = Egg\_Mean\_par1 + Egg\_Mean\_par2 \* Size* * *‘pow’ - E\_mu = Egg\_Mean\_par1 \* Size ^ Egg\_Mean\_par2* |
| **Egg\_Mean\_par1**, **Egg\_Mean\_par2**,  ‘-790’, ‘6.2’ | If size control is specified in **sizecontrol** column in RunVars, then these are the parameters values used to fit size-based fecundity function above. |
| **Egg\_Mortality**,  **Egg\_Mortality\_StDev**  [0-1], e.g., ‘0.62’, ‘0.18’ | Mortality of eggs will occur around **Egg\_Mortality** value with given standard deviation. If a nonzero value is placed in **Egg\_Mortality\_StDev**, then stochastic variability will occur every time step (a new value will be drawn around **Egg\_Mortality** with given standard deviation). Note this can be compounded with the use of the patch level egg mortality and events will be considered mutually exclusive. |
| **Egg\_FemaleProb**  [0-1] or ‘WrightFisher’ | * [0-1] Defines sex ratio at birth (for eggs/litter) and is the probability that an individual born in a litter will be female (1 – ‘Egg\_FemaleProb’ gives the probability of males born in the litter). * ‘WrightFisher’ - A special case for Wright-Fisher assumption can be specified here by entering and only should be used when considering a panmictic population (see section on *Special Cases* for more details). |
| Genetic options | |
| **startGenes**  ‘0’ - runtime | The year/generation at which genetic exchange will begin. For example, use a later year to begin swapping genes while population dynamics stabilizes.  Note: A special case occurs during this period when ‘Hindex’ parameters are specified. Offspring will inherit their parents hindex when strict assortative mating (**AssortativeMate\_Model** =‘2’) and **growth** = ‘temperature\_hindex’. Similarly, **mtDNA** = ‘Y’, the mother’s last locus is passed on to offspring (see below). |
| **loci**  ‘2’ – max number dependent on computer resources. | The number of loci (microsatellites/snps). Recommended maximum number of loci dependent on computer resources and recommend test runs for tradeoffs in performance.  If a file is specified in the ClassVars.csv input file, then the number of loci entered here must match this file. |
| **alleles**  ‘2’ – max number dependent on computer resources. | The number of starting alleles per locus.  If a file is specified in the PatchVars.csv input file (**Genes Initialize** column), then the number of alleles entered here must match this file. Polymorphism or varying number of alleles can be used by specifying the maximum number of alleles here and filling in allele frequency values of 0 for ‘filler’ alleles in other loci locations.  Alternatively, variable number of alleles per locus can be specified here and separated with a ‘:’, e.g., 2:5:3 for 3 loci. Then random\_var must be used in the field **Genes Initialize** in the PatchVars file or a matching allele frequency file. |
| **muterate**  0 - 1 | Allele mutation rate. |
| **mutationtype**  ‘random’, ‘forward’, ‘backward’, ‘forwardbackward’ | Mutation model:   * ‘random’ – kth-allele mutation model. * ‘forward’ – step-wise mutation in which an allele can mutate forwards only (i.e., to the right). * ‘backward’ – step-wise mutation in which an allele can mutate backwards only (i.e., to the left). * ‘forwardbackward’ – step-wise mutation in which an allele can mutate forward or backwards only (i.e., to the left or right with equal probability). * ‘forwardAbackwardBrandomN’ – special case for the 2-locus 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. |
| **mtdna**  ‘Y’ or ‘N’ | Tracking maternal genes:   * ‘Y’ - the last locus becomes the maternal marker (mtDNA) and every offspring inherits this locus from its mother only. * ‘N’ - regular Mendel inheritance occurs for the last locus. |
| Spatial selection options | |
| **cdevolveans**  ‘N’, ‘1’, ‘2’, ‘M’, ‘G’, ‘MG\_ind’, ‘MG\_link’, ‘1\_mat’, ‘2\_mat’,  ‘stray’,  ‘1\_G\_ind’, ‘1\_G\_link’  ‘Hindex’, ‘P\_X{n}\_L{l}\_A{a}\_Model{XY}’  ‘M\_X{n}\_L{l}\_A{a}\_Model{XY}’, | Selection model and mechanism. Only the first 2 alleles of each locus are used.   * ‘N’ - turn off selection (CDEVOLVE). * ‘1’ - selection with 1 locus A. * ‘2’ - selection with 2 loci (A and B). * ‘M’ - Locus A (first locus) tied to maturation probability curves. * ‘G’ - Locus B (second locus) tied to growth curves. * ‘MG\_ind’ - both maturation and growth independently linked to Locus A and Locus B, respectively. * ‘MG\_link’ - both maturation and growth associated to Locus A. E.g., If AA genotype, then the respective maturation parameter values are used for this genotype and linked to growth values listed for BB genotype. Aa is linked to Bb, and aa is linked to bb. * ‘1\_mat’ - consider selection with 1 locus (A) with only mature individuals. * ‘2\_mat’ - consider selection with 2 loci (A and B) with only mature individuals. * ‘stray’ - consider stray rate associated to Locus A (e.g., If AA genotype, then the respective stray rate value will be used for this genotype). * ‘1\_G\_ind’ - uses the ‘1’ option described above (spatial selection tied to Locus A) and the ‘G’ option described above for Locus B. Since 2 loci are used, the ‘\_ind’ tells the program to run these processes independently. * ‘1\_G\_link’ - uses the ‘1’ option described above (spatial selection tied to Locus A) and the ‘G’ option described above for Locus B. Since 2 loci are used, the ‘\_link’ tells the program to run these processes tied to Locus A (e.g., if AA genotype, then the respective spatial fitness values are used for this genotype and linked to growth values listed for BB genotype. Aa is linked to Bb, and aa is linked to bb). * Multiple loci/allele selection model can be considered by entering ‘M\_X{n}\_L{l}\_A{a}\_Model{XY}’ for n environmental variables, l loci, a alleles, either Model X or Model Y. Environmental variable information will be entered at the end of the XY file with a column corresponding to each variable considered in the selection model at each spatial location. Fitness is incorporated through a linear additive model. Model X codes the alleles 2, 1, or 0 and Model Y codes the alleles 1 or 0. The equation is rescaled based on the hypothetical maximum/minimum GXE space (static value calculated before time loop begins). * ‘Hindex\_Gauss\_mintemp:maxtemp:C:p:minparent:maxparent’\* - apply spatial selection as a function of each individual’s ‘HIndex’ using a Gaussian function. For example, enter ‘Hindex\_Gauss\_8:10:0.5:0.5:0.1:0.9’ for spatially explicit temperature values. ‘Minparent’ and ‘maxparent’ are the cutoff Hindex values that define parental species. Then, if Hindex > minparent or Hindex < maxparent, then p = 1, else the value of p entered is used for hybrid Hindex values.   Fitness = p \* exp(-(Xj – (mintemp + (maxtemp - mintemp) \* Hindex)^2 / (2 \* C^2) )  Then, similar to all other modules a differential mortality is applied via 1 – Fitness.   * ‘Hindex\_Para\_p:h:k’\* - apply selection as a function of each individual’s ‘HIndex’ using a parabolic function. Note that here, no spatial or environmental influence is needed. For example, enter ‘Hindex\_Para\_1:0.5:0’.   Fitness = k + (Hindex - h)^2 / (4 \* p)  Then, similar to all other modules a differential probability of mortality is applied via 1 – Fitness.   * ‘Hindex\_Step\_a:b:c’\* - apply a selection as a function of each individual’s ‘HIndex’ using a step function. Note that here, no spatial or environmental influence is given. If Hindex <= a, then fitness = b, else fitness = c. * ‘Hindex\_Linear\_p:h’\* - apply selection as a function of each individual’s ‘HIndex’ using a linear function. Note that here, no spatial or environmental influence is needed. For example, enter ‘Hindex\_Linear\_0.5:0.2’.   Fitness = h + (p \* Hindex)  Then, similar to all other modules, a differential probability of mortality is applied via 1 – Fitness.   * ‘F\_Linear\_p:h’ - apply selection as a function of each individual’s observed homozygosity using a linear function. \*Note the result of this function is differential mortality, not fitness. Differentialmortality is confined between 0 and 1. For example, enter ‘F\_Linear\_1:2’.   Differential mortality = p \* Ohomo + h   * ‘F\_Logistic\_p:h’ - apply selection as a function of each individual’s observed homozygosity using a logistic function. \*Note the result of this function is differential mortality, not fitness. Differentialmortality is confined between 0 and 1. For example, enter ‘F\_Logistic\_1:2’.   exp(h + p\*Ohomo)/ (1 + exp(h + p\*Ohomo))= differentialmortality  (1.-(1.-differentialmortality\_F)\*(1.-differentialmortality\_H))  Differentialmortality is confined between 0 and 1. For example, enter FHindex\_Linear\_1:2\_Linear\_1:2   * ‘P\_X{n}\_L{l}\_A{a}\_Model{XY}’ - Multiple loci/allele selection model can be considered by entering for n environmental variables, l locus, a alleles, either Model X or Model Y. Environmental variable information will be entered at the end of the XY file with a column corresponding to each variable considered in the selection model at each spatial location. Fitness is incorporated through a linear additive model. Model X codes the alleles 2, 1, or 0 and Model Y codes the alleles 1 or 0. The equation is rescaled based on the hypothetical maximum/minimum GXE space (static value calculated before time loop begins).   Spatial selection values (differential mortality for options ‘1’ and ‘2’ and parameter values for options ‘M’ and ‘G’) for each selection model and corresponding genotype are then entered for each patch (see Patch control and PatchVars.csv file).  \*Note that for all ‘HIndex’ settings, users will need to specify the initial ‘HIndex’ values for each allele frequency file used using the first loci. |
| **startSelection**  ‘0’ – runtime | This is the time unit that the selection surface will begin operating on loci under selection, specified in previous field (**cdevolveans**). If **cdevoloveans** = ‘N’, then this field is ignored.  Note that ‘startSelection’ must be >= ‘startGenes’. |
| **implementSelection**  ‘Out’,  ‘Back’,  ‘Eggs’,  ‘packing’  ‘Out\_{age}’  ‘Back\_{age}’ | The option to apply selection at specific timing events:   * ‘Out’ – individuals are migrated out and away from natal grounds. * ‘Back’ – individuals are back at natal grounds. * ‘Eggs’ – applies only to the egg class. * ‘packing’ – Applies selection during packing mortality to simulate the effects of competition * ‘Out\_{age}’ – individuals of age = {age} are only used in selection options * ‘Back\_{age}’ – individuals of age = {age} are only used in selection options   Note all or any of the timing options can be specified with ‘:’, e.g., Out:Back:Eggs would apply selection at every instance. |
| **betaFile\_selection**  ‘N’  ‘betaFile\_1L2A.csv’ | For use with polygenic selection module, this file specifies beta values for each locus and allele. |
| Phenotypic Plasticity options | |
| **plasticgeneans**  ‘{signal type}\_{allele response}\_{response reduction}’  ‘N’ | ‘Temp\_dom\_0’ to apply a behavioral plasticity process as a function of the individuals first locus alleles and temperature values.  First term defines whether plastic behavior is related to temperature or habitat values in the patchvars file (‘Temp’ or ‘Hab’).  Second terms is allele response(one of “dom”/”rec”/”codom”). dom: individual only needs one turned on allele (“2”) to avoid patches above the PlasticBehaviorResponse. rec: individual needs two “2”s to avoid patches. codom: individual with two “2”s will avoid based on the full reduction value entered, whereas those with one “2” will have the reduction value entered in the next term reduced by 50%.  Third term is the response reduction is the value multiplied to the probability array for patches that are higher than the PlasticBehaviorResponse. This will be the amount to reduce the probability matrix.  Examples:  Temp\_dom\_0: Temperature at each patch will trigger the response and it will be the patch variable individuals will avoid if they have at least one “allele”. The probability array values for patches above the PlasticBehaviorResponse will be zero (original value \* 0).  Temp\_dom\_0.5: Same as above, but probability array values for those above the PlasticBehaviorResponse reduced by 50% (original value \* 0.5). |
| **plasticSignalResponse** | Value in a patch that will trigger the plastic locus region alleles to ‘turn-on’ during either the second or third DoUpdate() modules of ‘Back’ or ‘Out’ options dependent on implementPlasticgene input. Both alleles will be turned on. Any patch above this value will turn the “allele” from “1” to “2” (the “on” state). |
| **PlasticBehaviorResponse** | If an individual has the first allele (A; a single or double copy), then patches with temperature values >= Behavioral Response Threshold value will be considered as a reduced value considered in the probability distribution from which to move to. Occurs at “Out” or “Back” depending on user input at implementPlasticgene. |
| **startPlasticgene**  ‘0’ – runtime | This is the time unit that the plastic process will begin operating on the plastic locus region, specified in previous field (plasticans). If ‘N’ is specified for ‘plasticans’, then this field is ignored.  Note that ‘startPlastic’ must be >= ‘startGenes’. |
| **implementPlasticgene**  ‘Back’, ‘Out’, | The option to apply plasticity at specific timing events:   * ‘Out’ – individuals are migrated out and away from natal grounds. * ‘Back’ – individuals are back at natal grounds.   Note, all or any of the timing options can be specified with ‘:’, e.g., Out:Back would apply the plastic response at every instance. |
| Infection options | |
| **cdinfect**  ‘Y’ or ‘N’ | Implements vertical disease transmission in the population. A column in ind{}.csv denotes the infection status (0 – no infection, 1 - infected) at each year for every individual.   * ‘Y’ - a random status infection (0 or 1) is created and initialized for each individual. Vertical transmission at given rate below will then occur. * ‘N’ - all individuals will have 0 infection and this module is turned off. |
| **transmissionprob**  [0-1] | Vertical transmission probability from an infected to the offspring. |
| Growth options (see Section 1.2.1.6) \*\*Warning. If value for cdevolveans points to fitness-based growth, growth parameter values present in the fitness columns in patchvars will be used and growth parameter values present here in popvars will be ignored. | |
| **growth\_option**  ‘N’, ‘known’ ‘vonB’, ‘temperature’, ‘temperature\_hindex’, or ‘bioenergetics’ | Growth function option. Parameters are specified in the fields described below. These functions can work for either size or age control. If size control, then the size classes are binned following a nearest neighbor routine to the closest size given in the ClassVars.csv file. Then those individuals are assigned a new ‘adjusted’ age (i.e., class).   * ‘N’ - turn off growth and the rest of the growth parameters are ignored. * ‘known’ - assign each individual’s size by a known amount each year. Given the age of the individual, the size class entered in the ClassVars field is used. This will only be applied at the second DoUpdate. * ‘vonB’ - von Bertalanffy equation for growth. Newsize = size\_Loo \* (1 – exp( -size\_R0 \* (‘adjusted’ age + 1))). * ‘temperature’ - the von Bertalanffy function is modified by parameters that are fit to temperature as well. * ‘temperature\_hindex’ - the above temperature growth model is used with the individual’s HIndex which adjusts the Loo parameter. |
| **growth\_Loo\*\*,**  **growth\_R0\*\***  ‘266’ or ‘144;266’  ‘0.19’  \*\* This field can be different for each sex class (~ delimiter). | The von Bertalanffy L infinity and growth rate parameters used to grow individuals via:  L = Loo \* (1 – exp(– R0 \* (i + 1-t0)))  where age can be the new binned ‘adjusted age’ if size control is operating.  If 2 values are entered separated by a ‘;’ (note this is the only parameter field that uses a semicolon), then these values correspond to Loo\_1 and Loo\_2 (Loo\_1;Loo\_2) and used with growth\_option ‘temperature\_hindex’ adjusting the Loo parameter based on the individuals hIndex:  Loo = hIndex \* (Loo\_2 – Loo\_1) + Loo\_1 |
| **growth\_temp\_max\*\*,**  **growth\_temp\_CV\*\*, growth\_temp\_t0\*\***  \*\* This field can be different for each sex class (~ delimiter). | Fit parameters for incremental growth based on temperature. Temperature growth values are specified for two different migration stages (Temperature Out and Temperature Back) in the PatchVars.csv file. See section 1.3.1.6 for equations. |
| **Patch level controls – PatchVars.csv file** | |
| *See PatchVarsS1.csv as the example input file for patch level controls. An easy way to create this file is to establish first X,Y locations for each patch and develop raster surfaces for each variable (temperature, grow days, fitness values, and mortality values linked to climate or harvesting spatially). Then use a software program (‘raster’ in R or ArcGIS) to extract variables under each X,Y patch location, such as temperature, grow days, and fitness values. Through these software programs, additional fields can be added to help populate K, N0, or Gene Initialization.* | |
| **File Header in patchvars.csv**  **Example** | **Description** |
| **PatchID** | A unique numerical identifier for each patch. Begin label 1 through *n* in consecutive order. |
| **X, Y** | Coordinate locations for each patch |
| **SubpatchID** | A unique text/string identifier for each patch used to identify individuals labeled to a particular region. This field is used as a ‘tag’ ID for each individual and reported also in output files. |
| **K†, K StDev†** | Carrying capacity ± standard deviation that will vary annually. A value of 0 can be included here and individuals will not move into (migration out, stray, disperse) to these patches. |
| **N0†** | Number of individuals to initialize the patch at year 0.   * cdclimate module can be initiated and multiple N0 values can be specified, e.g., for cdclimategen = 0|5|10, N0 = 100|10|10. This has a unique meaning, in that, it represents introduced individuals into that patch. A ClassVars or GenesInit file can be used to initialize these individuals. See PatchVars\_IntroducePopulation.csv as an example for how to implement this option. |
| **Natal Grounds** | Identifies natal ground locations.  ‘1’ – individuals can occupy this patch for e.g., spawning.  ‘0’ – individuals cannot occupy this patch when they are ‘back’.  **N0** cannot be > 0 (initialization value starting individuals at natal grounds), if Natal grounds = 0. |
| **Migration** | Identifies suitable migration patches.  ‘1’ – individuals can occupy this patch during DoEmigration() process (e.g., overwintering).  ‘0’ - individuals cannot occupy this patch when they are ‘out’. |
| **Genes Initialize** | Determines genotype initialization for each individual. Different initialization can be used for each patch.   * ‘random’ or ‘random\_var’ - random assignment of alleles (number of alleles and loci specified in PopVars.csv). Patch will be at maximum genetic diversity with this option. ‘random\_var’ will give a variable allele / locus option specified in the PopVars file field **alleles**. ‘random’ will assume the number of alleles per locus are all the same. * allele frequency file name - alleles are drawn from the allele frequency distribution in file. Examples of format in comma delimited form are given in the data/genes/ folder location. This file is a column of allele frequencies where the length of the column matches starting loci \* starting alleles given in the PopVars.csv file. Different allele frequency files can be given for different patches, making sure they are the same size (loci \* alleles). * multiple allele frequency files - This is achieved by using a ‘;’ to separate each file (e.g., allelefrequecyA.csv;allelefrequencyB.csv;allelefrequencyC.csv would randomly choose one of these three files to use for gene initialization for an individual). Note that this is a different delimiter than the ‘|’ which is linked to temporal changing files. |
| **Class Vars** | ClassVars input file that governs this patch location. See more on ClassVars specific parameters in next section 3.4. Note that different ClassVars files can be given at each patch, as well as through time with introduced populations.  You can choose to initialize each patch with multiple ClassVars files that are linked to the multiple allele frequency files given in previous field. This is achieved by using a ‘;’ to separate each file. E.g., allelefrequecyA.csv;allelefrequencyB.csv;allelefrequencyC.csv could be associated with ClassVars files in the same patch, e.g.,  ClassVarsA.csv;ClassVarsB.csv;ClassVarsC.csv. An individual in a patch would get a random initial allele frequency assignment and then corresponding ClassVars assignment. Note that this species/space delimiter (‘;’) is a different delimiter than the ‘|’ which is linked to temporal changing files.  **Warning:** ClassVars split by time with ‘|’ will only apply when introducing new individuals. In this case, the only values used will be the first 5 columns of ClassVars used for initializing individuals. All other ClassVars values will remain static. |
| **Mortality Out†, StDev†**  **Mortality Back†, StDev†**  **Mortality Eggs†, StDev†**  [0-1]  ‘N’  ‘E’ | Density-independent mortality   * [0-1] - mortality percentage [0-1] applied to each patch in the rearing/overwintering/foraging (out), spawning (back) stage, or egg/litter stage (eggs). * ‘N’ - this field is not considered. Note this is compounded with age and size level mortalities with given **constMortans**. * ‘E’ - the special case in which e.g., eradication as well as suppression operating at the class level. This will override any class level mortality values (described below).   Standard deviation around these parameters ‘Mortality Out/Back/Eggs %’ [0-1] is applied to each patch each year. Enter 0 here and this field will be deterministic, using the ‘Mortality Out/Back/Eggs %’ value at each time step.  Note that there is also a population wide level consideration on egg mortality (see below) in PopVars file. If used together, events will be considered mutually exclusive (additive). |
| **Migration†** | Emigration probability [0-1] applied before moving to rearing/overwinter grounds. If just age/size class migration is wanted, then set these values to 1. |
| **Set Migration** | If an individual becomes a migrant, it can stay a migrant with probability 1.0.   * ‘Y’ - turn on set migration. * ‘N’ - turn off set migration. |
| **Straying†** | Straying probability   * [0-1] - applied before moving back from emigrated patch to original/natal grounds. * ‘1’ - Age/size class governs straying probability through Classvars. |
| **GrowthTemperatureOut†, StDev†**  **GrowthTemperatureBack†, StDev†** | Temperature values that influence body size growth of individuals at patch location after DoEmigration() or migration out of natal grounds (out, e.g., during fall/winter for a spring breeder) or after DoImmigration() or movement back to natal grounds (back).   * Float value - spatial temperature values * ‘N’ - turn off temperature-dependent patch-specific growths at this time of year.   Standard deviations around each temperature value are applied annually to each patch. Enter 0 here and this field will be deterministic, using the ‘Growth Temperature Out’ value at each time step. |
| **GrowDaysOut†, StDev†**  **GrowDaysBack†, StDev†** | * [0-365] - Patch site growing days corresponding to the growth time ‘out’ or ‘back’ for growth option ‘temperature’. * ‘N’ -This parameter is ignored.   Standard deviations around each grow days value are applied annually to each patch. Enter 0 here and this field will be deterministic, using the ‘Growth Days Out/Back’ value at each time step. |
| **Capture Probability Out†, Back†** | Probability of detection, capture, or sampling for each patch when individuals are back at natal grounds (e.g., for spring spawning fish during the summer). This occurs right before the individuals are preparing to migrate out of their natal grounds.   * [0-1] - probability * ‘N’ - Ignore capture probability. * ‘1’ - capture probability will be controlled by ClassVars.   \*Warning: If values for capture occur in both patchvars and classvars, these will be multiplied to achieve a final capture probability. If ‘N’ is entered in either spot, all capture is ignored for the relevant patch. |
| **HabitatOut, HabitatBack** | Habitat quality values for use with the plasticity module during the Out and Back steps of the simulation. Only impacts probability matrix if the first term of plasticgeneans is “Hab”. |
| **Fitness\_AA†**  **Fitness\_Aa†**  **Fitness\_BB†, etc.**  [0-1] for survival options  Alternatively, values for growth, maturation, and straying (see description). | *Fitness\_AA, Fitness Aa, Fitness\_aa* – When **cdevolveans** = 1 (in PopVars.csv file), then this is the offspring viability selection values for AA, Aa, and aa, respectively (i.e., one-locus selection model). These are differential mortality values tied to each genotype. You can link the selection via mortality to spatial environmental-genotype processes.  When **cdevolveans** = ‘M’ or ‘MG’ (in PopVars.csv file), then these 3 genotypes are used to enter 2 to 6 maturation response parameters in the order and s slope:intercept. The 2 to 6 values must be entered and separated by a ‘:’. If 2 values are entered, then all sex classes will follow the same logistic maturation curve. If 4 are entered, then female\_slope:female\_intercept~male\_slope:male\_intercept. If 6 values are entered, then YY males will be considered for a different maturation curve, female\_slope:female\_intercept~male\_slope:male\_intercept~ YYmale\_slope:YYmale\_intercept. CDCLIMATE can still be considered, e.g., 0.06:23~0.08:21|0.08:24~0.10:22 using ‘|’ to separate each parameter group to be considered.  When **cdevolveans** = ‘stray’ (in PopVars.csv file), then these genotypes are used to enter stray probabilities (e.g., *AA* could be 0.05, *Aa* could be 0.01 and *aa* could be 0.00 indicating individuals with the genotype *AA* having a higher stray tendency than individuals with *Aa* or *aa*).  *Fitness\_BB, Fitness Bb, Fitness\_bb* – When **cdevolveans** = ‘G’ or ‘MG’ (in PopVars.csv file), then these 3 genotypes are used to enter 5 growth rate response parameters in the order  temperature:353:0.57:13:0.33:-0.196. The 6 values must be entered and separated by a ‘;’. CDCLIMATE can still be considered, e.g., temperature:353:0.57:13:0.33:-0.196| temperature:353:0.57:13:0.33:-0.196 using the ‘|’ to separate each parameter group to be considered.  \*(*Fitness\_AABB, Fitness AaBB, Fitness\_aaBB, Fitness AABb, Fitness AaBb, Fitness aaBb, Fitness Aabb, Fitness Aabb, Fitness aabb*) – When **cdevolveans** = 2, then this is the offspring viability selection values for each of the 9 genotypes (i.e., two-locus selection model). These are differential mortality values tied to each genotype. You can link the selection via mortality to spatial environmental-genotype processes. For example, to consider one genotype under directional selection, e.g., AABB, enter spatial mortality values here [0-100] and all other genotypes having 0% mortality. |
| **comp\_coef** | Lotka-Volterra competition coefficient (α) for multispecies applications, representing the effect of other species on the target species. If more than 2 species, multiple α values are separated by a ‘;’, in the order in which the Popvars files are listed in the **Popvars** column in the RunVars file. For example, in a 3 species application, species 1 may have values of ‘0.5;0.9’, where the effect of species 2 on species 1 = 0.5 and the effect of species 3 on species 1 = 0.9. |
| ***Class level controls – ClassVars.csv file*** | |
| *See ../classvars/ClassVarsA.csv as an example input file for the age/size parameters and formatting (e.g., comma delimited with the below headings). These files determine age/size class information for the program. Supply a file in the /data/ folder. However, input file locations are specified in the PatchVars.csv input file (see above). Hence, a separate ClassVars input file can be specified for each patch if wished, and used for initialization purposes for each individual. In addition, multiple ClassVars files can be specified (separated by ;) and linked to gene frequency files (separated by ;), thus, distinguishing ‘species’ using HIndex. Most all of the parameters can have different values for each sex as well and specified with ~ in the order FXX~MXY~MYY~FYY. If only one value is given, then that value will be used for all sexes.*  *If it is desired to change parameter values in ClassVars over time using the cdclimate module (e.g., age-based mortality), this has to be done by listing multiple ClassVars files in the PatchVars file separated by ‘|’. Parameter values in the ClassVars file itself do not accept temporal changes via ‘|’.* | |
| **Age class** | Number of age classes by sequential class numbers – 0, 1, 2, … |
| **Body Size and StDev** | Mean body size and standard deviation at initialization (e.g., length in mm). If size control is specified, then individuals are grouped around these size classes specified here and associated parameters in the rest of this file used. In addition, if ‘known’ **growth** option is specified, then these values will be used as the size at given age. |
| **Distribution** | These values specify the class distribution at initialization. |
| **Sex Ratio** | Initializes the sex of the population which can give a ratio of females to males per age class. Up to 4 values may be specified here: e.g., 0.5~0.5~0, for females~males~trojan YY males. The values must equal 1, be separated with a tilda (~), and the number of values provided must be equal to the value given for the **sex\_chromo** variable in the PopVars file.  \*A special case for Wright Fisher assumption can be specified here by entering ‘WrightFisher’ and only should be used when considering a panmictic population (see section on *Special Cases* for more details here). |
| **Age Mortality Out/Back and StDev** | Age-specific mortality probability [0-1] and standard deviation applied in overwintering/ rearing/foraging patches (out) after DoEmigration() module and in natal patches (back) after DoImmigration() module.   * [0-1] – Probability of mortality * ‘N’ - ignore this mortality. * Note this is compounded with patch and size level mortalities with given **constMortans**. * 3 sex class values can be used here, as well, separated with ‘~’. |
| **Size Mortality Out/Back and StDev** | Size class-specific mortality percentages [0-1] and standard deviation applied in overwintering/ rearing/foraging patches (out) after DoEmigration() module and in natal patches (back) after DoImmigration() module.   * [0-1] – Probability of mortality * ‘N’ - ignore this mortality. * Note this is compounded with patch- and size-level mortalities with given **constMortans**. 3 sex class values can be used here, as well, separated with ‘~’. * 3 sex class values can be used here, as well, separated with ‘~’. |
| **Migration Prob** | Probability [0-1] of an individual from a given class (size/age) migrating out of original/natal population during the DoEmigration() module. Set these values to 1 for patch level control on migration (see PatchVarsS1.csv). 3 sex class values can be used here, as well, separated by ‘~’. |
| **Straying Prob** | Probability [0-1] of a migrant from a given class (age/size) straying to a patch other than their natal patche during the DoImmigration() module. Set these values to 1 for patch level control on straying (see PatchVarsS1.csv). 3 sex class values can be used here, as well, separated by ‘~’. |
| **Dispersal Prob** | Probability [0-1] of an individual from a given class (age/size) undergoing annual dispersal from their natal/spawning patch following the DoImmigation() processes. Set these values to 1 for all individuals to disperse. 3 sex class values can be used here, as well, separated by ‘~’. |
| **Maturation** | Probability of being a reproductively mature individual and then stay a reproductively mature individual. If size option is specified (**sizecontrol** in RunVars), then these values are not used and population fit parameters based on size/length relationships are used instead (PopVarsS1.csv). 3 sex class values can be used here, as well, separated by ‘~’. |
| **Fecundity Ind and StDev** | Litter size or egg number with standard deviation option. Poisson, random, constant, or normal draw options available (in PopVars.csv file). If size option is specified, then these values are not used and population fit parameters based on size are used (PopVars.csv). |
| **Fecundity Leslie and StDev** | Fecundity values (± annual standard deviation) specific to the Leslie matrix model which also requires values in the Age Mortality Back column to specify Leslie matrix survival. Note that values are also required in the Fecundity Ind column to produce individual-based reproduction and genetic inheritance. Leslie matrix projections will only be realized if the Fecundity Ind processes first produce enough offspring, followed by removal of individuals to match Leslie projections. |
| **Capture Out/Back Probability** | Probability of detection, capture, or sampling for each class when individuals are at natal grounds (Back) (e.g., for spring spawning fish during the summer) or at migration/overwintering grounds (out). This occurs right before the individuals are preparing to emigrate out of their natal grounds or immigrate back to natal grounds. 3 sex class values can be used here, as well, separated with ‘~’.   * [0-1] - probability * ‘N’ - ignore capture. * ‘1’ - Patch level capture probability will operate (PatchVars.csv).   \*Warning: If values for capture occur in both patchvars and classvars, these will be multiplied to achieve a final capture probability. If ‘N’ is entered in either spot, all capture is ignored for the relevant patch. |

# **Output**

Folders will be created in your project directory labeled with a unique time stamp (dos convention), e.g., ‘..\data\output\_test\_1332964297\run0batchrun0mcrun0’. Monte Carlo runs will be designated mcrun0, mcrun1,… with the number of runs specified in the RunVars.csv file. Each batch run in PopVars will be labeled batchrun0, batchrun1, … with the number of batch runs corresponding to the number of parameter input rows in the PopVars.csv file. Each batch run in RunVars will be labeled run0, run1, … with the number of batch runs corresponding to the number of rows in the RunVars.csv file. The following is a description of the files that can appear in each run{}batchrun{}mcrun{} folder.

## **Individual files**

### ind.csv or indSample.csv

For the specified ‘output\_years’ (or years to output given in the RunVars.csv file), files with information on each individual are produced. For example, if the user specifies a list for the years to report, e.g., ‘0|4|10’, then ind0.csv, ind4.csv, and ind10.csv will be written to each Monte Carlo folder. The first *DoUpdate()* module will automatically write out starting information for each individual to the file ind-1.csv. This is the information for each individual when they are back at their natal grounds.

If the user specifies ‘Sample’ for the ‘gridSampling’ option in RunVars, then indSample0.csv, indSample4.csv, and indSample10.csv will be written to each Monte Carlo folder. This is the information for each individual after they have migrated away from or resided at their natal grounds. This file will include the fingerlings (Age0 class) that enters into the population at this point. See Figure 5 flow diagram for ‘DoUpdate’ processes for order of processes and respective sample of the individuals.

The following is a description of what is included in each of these files:

**PatchID** – This designates the patch in which each individual is sampled(patch numbers given in the PatchVars.csv file).

**XCOORD** and **YCOORD** – The reference for each patch location given in the PatchVars.csv file.

**ID** – a unique identifier for each individual that can be tracked through time. For the ind.csv files: {movement alpha character (see below) and patch currently in}\_{emigrated from patch}\_(‘P’ and natal patch born in}}\_{‘Y’ and year born}\_{unique individual number}. E.g., R2\_F2\_m3f4\_P1\_Y10\_150 would mean that this is the 150th unique individual that was a residor in patch 2, that moved previously from from patch 2 (residor should have the same patch number for these first 2 fields), born to a mother in patch 3 and father in patch 4, and born in year 10 in natal patch 1. Initialization of individuals will have Y-1, m-1f-1, and will be ‘R’. Therefore, a query for year, patch, and number will track individual locations. The first alpha character tracks the individuals movement. Here is the list of alpha characters for identification:

**R** – Residor; stayed in current patch,

**E** – Emigrator; moved out from natal patch (note that an individual could attempt to move out, but still settle in the same natal patch due to effective distance matrix),

**S** – Strayer; strayed to another natal ground location (note that an individual could attempt to stray, but settle still in natal patch due to effective distance matrix),

**I** – Immigrator; migrated out and returned back to original natal patch,

**Z** – Attempted to immigrate back to natal grounds, but was forced to stray elsewhere due to either K = 0 at original patch (e.g., catastrophic event) or the movement cost exceeded the specified movement threshold (e.g., the individual traveled too far away from natal grounds).

**RD** – Residor with dispersal; Resident individual but dispersal movement applied.

**ID** – Immigrator with dispersal; migrated out and returned successfully back to original natal patch area, but an addition dispersal movement applied.

**Age0** – emerged offspring within natal patch. This identification will only occur in the ind().csv files.

**N** – New individuals introduced at the specified year using the cdclimate module. This alpha will be tagged to the {unique individual number}, e.g., N0, N1, N2.

**F** – Indicates the patch that the individual migrated from.

**P** – Natal patch

**Y** – Year born

**U** – unique ID

**UO** – unique offspring ID

**UN** – unique introduced individual ID

**m{}f{}** – Mother’s patch and Father’s patch.

**MID/FID** – Unique identifier associated with the mother/father of the individual.

**sex** – XX for female, XY for male, and YY for Trojan males or females.

**age** – Age of each individual.

**size** – Size of each individual.

**mature** – Whether or not that individual is capable of reproducing.

**newmature** – Whether or not the individual became mature this year.

**capture** – 0 for not sampled and 1 for sampled.

**recapture** – 0, 1, 2, … value for how many times this individual was captured.

**infection** – Whether or not that individual was infected through vertical transmission (see transmission probabilities in Table of input variables for PopVars.csv).

**CDist** – If this individual was a migrator (Emigrator or Immigrator) or disperser, then the total cost distance that this individual traveled (derived from the cost distance matrix).

**HIndex** – This is the hybrid index (Allendorf et al. 2001) or allele frequency of A (p) for each individual. It is the average of the parent’s hybrid index where (at initialization) AA = 1.0, Aa = 0.5, and aa = 0.0.

**Species** – This is an ID [1-N] for species. Species are delineated by patch in the N0 PatchVars variable using ‘;’. Currently, the only process operating is strict mating by species ID.

**ClassFile** – This individual inherited class variables from this given file.

**SubPatchID**- The unique label for each patch given in PatchVars file – used to group patches together.

**L0A0**, **L0A1**, …, **L0AN**, **L1A0**, **L1A1**, …, **LMAN** – The diploid genotypes of this individual for N alleles and M loci given. For each locus values of 0, 1, or 2 can be given. A value of 2 represents a homozygous locus at the given allele location, else two 1’s will specify the allele locations for a heterozygous locus. 0’s will be given for every other allele not present.

### Genetics format option

The user also has the option to output genotypes in a number of different formats. See ‘gridformat’ option in the Table of input parameters for these options. For example, if ‘structure’ is specified then the files structure\_ind0.csv, structure\_ind4.csv, and structure\_ind10.csv will be produced.

## **Summary measures**

The following is a description of various output files to summarize each Monte Carlo run either by class, patch and through time.

### Time versus class

Each folder will contain a summary file for each year and for various measures across the classes (age or size) labeled ‘summary\_classAllTime.csv’. Each row represents a year, and every measure will be separated by ‘|’ distinguishing the class breaks in ascending order given in the ClassVars.csv file. The following is a description of each measure and in order of processes (see Figure 5):

**Year** – This is the time unit (year or generation).

**Ages** – A column of age classes.

**N\_Initial\_Age** – N counts for age classes at start of each time unit.

**AgeSize\_Mean** – Size mean of age classes.

**AgeSize\_Std** – Standard deviation around mean size of age classes.

**Size\_Classes** – A column of size classes.

**N\_Initial\_Class** - N counts for size classes at start of each time unit.

**ClassSize\_Mean** – Size mean of classes.

**ClassSize\_Std** – Standard deviation around mean size of size classes.

\*\*\*\* Note: If size class switch is specified, then the following tracking numbers are then binned by Size\_Class above, else binned by Age. An exception here is when logistic density dependent population model is applied, and tracking reverts to actual age bins.\*\*\*

**Births** – The number of births.

**N\_GrowthBack** – N counts after the first growth stage – second DoUpdate().

**Capture\_Back** – Number of individuals captured at spawning grounds by class.

**N\_BeforePacking\_AddAge0s** – N counts following addition of age 0 indivdiuals but before density-dependent mortality.

**PackingDeaths\_Emigration** – The number of individuals that died during the density dependent packing event after migration out of natal grounds (i.e., after DoEmigration()).

**N\_AfterEmigration** – N counts after DoEmigration().

**N\_GrowthOut** – N counts after the second growth stage – third DoUpdate().

**Capture\_Out** – Number of individuals captured following migration.

**PackingDeaths\_Immigration** – The number of individuals that died during density-dependent mortality after migration back to natal grounds (i.e., after DoImmigration()).

**N\_Immmigration** – N counts after DoImmigration() (age 0s counted here).

**Deaths\_AfterImmiMort\_Age** – The number of deaths by age from the

DoMortality() module (density-independent patch or class deaths) after the DoImmigration() process.

**Deaths\_AfterImmiMort\_Size** – The number of deaths by size due to density-independent factors.

### Class by given time

Using the ‘summaryOutput’ option, each given year specified will produce a file summarizing measures for each class. They will be labeled ‘summary\_class{year}\_{foldertime}.csv’. All the above section 4.2.1 measures will be listed within these files, separated by class.

### Time versus patch

Each folder will contain a summary file for each year and for various measures across the patches (i.e., subpopulations) labeled summary\_popAllTime.csv. Each row represents a year, and every measure will be separated by ‘|’ distinguishing the patch breaks in ascending order given in the PatchVars.csv file. The first number, unless specified otherwise, is the total population value. The following is a description of each:

**Year** – This is the time unit (year or generation).

**K** - Carrying capacity.

**GrowthRate** – N\_Initial at the end of the year divided by N\_Initial at the beginning of the year.

**N\_Initial** – N at the beginning of each time unit.

**PopSizes\_Mean** – Mean size of individuals per patch (no total reported)

**PopSizes\_Std** – Standard deviation around mean size of individuals per patch (no total reported)

**N\_Females** – The total number of females (‘XX’). First number is total.

**N\_Males** – The total number of males (‘XY’). First number is total.

**N\_YYMales** – The total number of YY males (‘YY’). First number is total.

**N\_YYFemales** – The total number of YY females (‘YY’). First number is total.

**N\_MatureFemales** – The total number of breeding class (size or age) females (‘XX’).

**N\_MatureMales** – The total number of breeding class (size or age) males (‘XY’).

**N\_MatureYYMales** – The total number of breeding class (size or age) males (‘YY’).

**N\_MatureYYFemales** – The total number of breeding class (size or age) males (‘YY’).

**MatureCount** – The number of reproductive individuals.

**ImmatureCount** – The number of immature individuals.

**EggLayEvents** – The number of breeding events for females (can be less than the number of mature females when egg frequency is less than 1 or some females cannot find mates).

**Births** – The number of eggs or number of individuals in litters.

**EggDeaths** – The number of deaths from eggs/litter to Age0 recruitment. Births – EggDeaths is the survival to age 0 (e.g., fingerling, young of year).

**MyyProgeny** – The number of offspring from YY males, following egg mortality.

**FyyProgeny** – The number of offspring from YY females, following egg mortality.

**Capture\_Back** – The number of individuals captured while at natal/spawning grounds.

**SelectionDeaths**\_**Emigration** – The number of deaths due to differential mortality (spatial selection) during the migration out period.

**MoveDeaths**\_**Emigration** – The number of individuals that could not migrate due to cost of moving to open patches exceeding the user-provided distance threshold.

**Kadj\_Emi** – Adjustment to P/K ratio due to presence of competitors for Lotka-Volterra competition

**N\_beforePacking\_AddAge0s** – The number of individuals present following addition of age 0 offspring to the population but prior to density-dependent mortality during the migration out period.

**PackingDeaths**\_**Emigration** – The number of individuals that died during density-dependent mortality after the migration out period.

**YYSelectionPackingDeaths**\_**Emi** – The number of YY individuals that died during the packing selection algorithm after the migration out process.

**WildSelectionPackingDeaths**\_**Emi** – The number of wild type individuals that died during the packing selection algorithm after the migration out process.

**N**\_**Emigration** – N counts after DoEmigration() process.

**Deaths**\_**EmiMort** – The number of deaths from the DoMortality() module (constant patch or class deaths) after the DoEmigration() process.

**N**\_**EmiMortality** – N counts after the DoMortality() module after the DoEmigration() process.

**Capture\_Out** - The number of individuals captured during the migration out period.

**SelectionDeaths**\_**Immigration** – The number of deaths due to differential mortality (spatial selection) during the migration back process.

**MoveDeaths**\_**Immigration** – The number of individuals that could not migrate back due to high cost to moving to open patches

**Kadj\_Immi** -

**PackingDeaths**\_**Immigration** – The number of individuals that died during the density dependent packing event after the migration back process.

**YYSelectionPackingDeaths**\_**Immi** – The number of YY individuals that died during the packing selection algorithm after the migration back process.

**WildSelectionPackingDeaths**\_**Immi** - The number of wild type individuals that died during the packing selection algorithm after the migration back process.

**SelectionDeaths\_Age0s\_Immigration** – The number of eggs that died as a result of selection.

**N**\_**Immigration** – N counts after the DoImmigration() process or when individuals have migrated back to their natal grounds. Age 0s counted here.

**Deaths\_ImmiMort** - The number of deaths from density-independent morality (DoMortality() module) after the DoImmigration() process.

**N**\_**ImmiMortality** – N counts after the DoMortaltiy() module following the DoImmigration() process.

**Alleles** – The number of unique alleles.

**He** - The expected heterozygosity value.

**Ho** – The observed heterozygosity value.

**p1** – The allele frequency of A or the first allele in the first locus (used for CDEVOLVE).

**p2** – The allele frequency of a or the second allele in the first locus (used for CDEVOLVE).

**q1** – The allele frequency of B or the first allele in the second locus (used for CDEVOLVE).

**q2** – The allele frequency of b or the second allele in the second locus (used for CDEVOLVE).

**\*\*\*Note**: The following movement distances are all split between females and males (YY individuals are ignored). Cost distances are reported except for options ‘4’, ‘6’, and ‘9’ which return probabilities.**\*\*\***

**MateDist** and **MateDiststd** – The mean and std {cost} distance individuals travel to mate.

**EmigrationDist** and **EmigrationDist**\_**SD** – The mean and std {cost} distance individuals migrate from their natal location during the DoEmigration() process. (This is the distance from ‘NatalPop’ to ‘ImmiPop’.)

**FromHomeDist** and **FromHomeDist**\_**SD** – The mean and std {cost} distance individuals moved during the DoImmigration() process. This is for all local dispersing individuals, except ‘Age0s’ and ‘I’ individuals (the distance these individuals moved was captured in EmigrationDist). Note also when dispersal is included, this will be the settled final patch location. (This is the distance from ‘ImmiPop’ to ‘NatalPop’ the dispering individual settled). **StrayerDist** and **StrayerDist**\_**SD** – The mean and std {cost} distance individuals stray during the DoImmigration() process instead of immigrating back to their natal patch. (This is the distance from ‘EmiPop’ to ‘ImmiPop’.)

**HomeAttemptStrayerDist** and **HomeAttemptStrayerDist**\_**SD** – The mean and std {cost} distance individuals stray after they attempt (and fail) to return to their natal grounds during the DoImmigration() process. (This is the distance from ‘EmiPop’ to ‘ImmiPop’.)

**Infected** – The number of individuals that are infected (used for CDINFECT) with given transmission probability (vertical infection).

**Residors** – Number of individuals after immigration process that are resident to each natal population (i.e., did not emigrate). Note if dispersal is operating, then RD individuals are included in this count.

**Strayers**\_**1** – Number of individuals that strayed into each population instead of immigrating back to their natal patch.

**Strayers**\_**2** – Number of individuals that attempted to immigrate home, but could not, then strayed into each population during immigration back to natal patches.

**Immigrators** – Number of individuals that emigrated then immigrated back to a natal patch. Note if dispersal is operating, the ID individuals are included in this count.

**ResidentDispersers** – Number of individuals after immigration process that were resident and locally dispersed to a new natal location.

**ImmigrantDispersers** – Number of individuals that immigrated back to original natal patch and local dispersed to a new natal patch.

**Allele-specific\_Mates** – Number of mating pairs between individuals of each combination of alleles at the first locus (e.g., AAxaa, AaxAa, etc.)

### Patch by given time

Using the ‘summaryOutput’ option, each given year specified will produce a file summarizing measures for each patch. They will be labeled ‘summary\_pop{year}\_{foldertime}.csv. All of the above section 4.2.3 measures will be listed here, separated by patch.

# **Special Cases**

This section provides examples of specific and commonly used simulation scenarios. Parameter values and input setup are given for each.

## **Non-overlapping generations**

Control non-overlapping generations through the ClassVars.csv file. The ClassVars.csv file must contain two rows for Age 0 (i.e., offspring population) and Age 1 (i.e., mature adult population) classes. Initialize the distribution of Age 0 = 0 and then the distribution of Age 1 = 1. For mortality, use 100% for the ‘Mortality Out’ option for Age 1. Set the maturation rate at 1 for Age 0 for both female and male. To control a constant population, you must balance your Leslie matrix or to control for a constant population at K, see Wright-Fisher assumptions below.

## **Wright-Fisher assumptions**

The following is a bullet list of parameters that must be set for Wright-Fisher assumptions.

In the ClassVars.csv file:

Use two age classes: Age 0 and Age 1.

Set the distribution for Age 0 to 1 and Age 1 to 0.

Set the ‘mortality out’ for Age 0 to 0% and Age 1 to 100%.

Set all other mortalities to 0%.

Set the maturation rate for Age 0 to 1 and Age 1 to 0 for both female and male.

Set the fecundity for Age 0 to 0 and Age 1 to 1.

Set the ‘Sex Ratio’ option to WrightFisher – this will ensure that the initial starting population will have equal females and males in each population, as well as ensuring that the replacement population is also equal females and males in each population.

In the PatchVars.csv file:

You must consider a panmictic population: all patches can mate with all other patches and locally disperse or stray freely between patches. Alternately, you could consider one big population or just one patch.

In the PopVars.csv file:

Set the reproduction to male and female mating with replacement (Y/Y).

Set the ‘offno’ choice to 3 for constant litter size

Set the ‘equalClutchSize’ to ‘N’.

No mutation.

Set the egg mortality to 0 and specify ‘WrightFisher’ in Egg\_FemalePercent.

Comparisons of simulation output with theoretical equations illustrate the validation, usefulness, and reliability of such programs. Genetic exchange is simulated in 1 population with 500 individuals. Wright-Fisher assumptions were used for 300 generations, 50 Monte Carlo replicates, and genotypes initialized with 30 loci and 30 alleles per locus. We provide basic graphics, including comparison of results to theoretical predictions for loss of heterozygosity over time (Fig. 7).

He and Ho are calculated at each generation as the estimated proportion of individuals that are expected heterozygous based on a Hardy-Weinberg population and the proportion of individuals that are observed heterozygous, respectively. He and Ho results are shown in Figure 7, compared with heterozygosity produced according to the following equation (Crow and Kimura 1970) adjusted for a small sample size,

Ht = (1 – (1 / (2\*Ne + 1)))t \* Ho(0), [25]

where the effective population size for separate sexes (Hedrick 2011) is defined as

Ne = (4\*NM\*NF) / (NM + NF), [26]

and Ht is the theoretical rate of decay after t generations, Ho(0) is initial heterozygosity, NM is the male total, and NF is the female total.



Figure 7: CDMetaPOP heterozygosity. Equation [25] Ht (solid line) show the theoretical rate of decay in the panmictic scenario, as expected for one subpopulation (N = Ne = 500) that matches the simulations observed (dotted-line) and expected (dashed-line) heterozygosity values.

## **Deterministic and stochastic variability**

Controlling for deterministic versus stochastic variability can be accomplished for most parameters. Here, we show how each parameter can be used for deterministic events with case examples (i.e., sensitivity) of chosen parameters. The following is a table with parameters for certain types of variability.

Table 2: Initial Parameter Space for the WCT model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Description** | **Demographic variability** | **Spatial variation** | **Temporal variation** | **Correlation structures** |
| **A1 ~N(0,1)** | Autocorrelation deviate - Random normal deviate used to create autocorrelation structures in other variables | N/A | N/A | Primarily auto-  correlation | Specify ϕ as necessary to simulate environmental data (e.g. temperature; ϕ~0.8) |
| **E1 ~N(0,1)** | Environment deviate 1. Used to capture epistemic uncertainty in the model | N/A | N/A | Could be used to simulate trend where µ~f(t) | Specify correlation coefficient rho (ρ) with A1 as required. |
| **K (µ,σ)** | Carrying capacity specified for each patch | N/A | Heterogeneously distributed as a function of patch size and empirical habitat data[[1]](#footnote-1). | Environmental variation simulated as a normal deviate specified with σ. | Since K is primarily a habitat variable, stochasticity is expected to be spatially highly correlated, as well as temporally autocorrelated. Link to E1 |
| **Nt** | Initial population size | N/A | Initially distributed as K/2 | N/A | N/A |
| **Meggs**  **~N(µ,σ)** | Egg mortality | Random Normal draw based on empirical data | N/A in current model[[2]](#footnote-2) | Implicit in demographic variation | Variation in egg mortality might be expected to be negatively correlated to environmental variation. i.e. when conditions are good mortality should decline.  ~f(E1); ρ~ -0.5 |
| **Mageclass**  **~N(µ,σ)** | Age/stage specific mortalities | Random Normal draw as specified | Uniform. [[3]](#footnote-3) | Implicit | N/A |
| **Mout**  **Mback**  **~N(µ,σ)** | Additional mortality associated with movement | Uniform random draw | Explicit patch level control | Implicit as function of σ | ~f(E1); ρ~ -1 |
| **Migration**  **and Stray rates** | Population level controls for migration rate and probability of straying | Uniform random draw | Uniform | Constant | Not implemented but could be explored as a negative correlate of E1 |
| **Growth:**  **L∞**  **k**  **t0**  **T\_optimal** | Population variables that are modified as a function of Temperature and duration of growing season | Determin-istic | Growth is individual-ly calculated as a function of Temperature and location | Implicit as a function of environ-mental Temperature | N/A |
| **Maturity** | Probability of maturity as a logistic function of length | Uniform random draw | Uniform | Constant[[4]](#footnote-4) | Not implemented but could be explored as environmentally correlated |
| **Fecundity** | Mean number of eggs per female as function of length | Poisson draw | N/A | N/A | N/A |

For each input file the parameters can be specified as deterministic as follows.

In the PopVars.csv file:

Deterministic runs were tested for size control on (‘Y’).

Movement can be controlled to occur within each patch, or demes/metapopulations with no movement between. Set all movement numbers to ‘6’.

Deterministic runs were tested for female and male with replacement for mate selection.

Maturation – in the fields mature\_length\_female and mature\_length\_male, choose a length at which individuals will become mature. Then the probability of maturation curves are ignored.

Fecundity – the egg mean to length relationship can still be used, but set the draw to be a constant mean given size (offno = ‘3’) with equal clutch size (equalClutch == ‘Y’) and egg frequency of 1.

Egg mortality – set standard deviation to 0.

In the PatchVars.csv file:

K can be set to any value, but set K StDev to 0.

N can be set to any value.

Mortality values – Set all StDev values to 0.

Movement (migration and straying) is controlled in the PopVars.csv file, so these field will be irrelevant.

Growth Temperature and grow days – Set all StDev values to 0.

In the ClassVars.csv file:

Body Size – Enter 0 for the Body Size StDev. This will ensure that individuals are initialized with the given Body Size number.

Mortalities – Set all mortality StDev values to 0.

Movement (migration and straying) is controlled in the PopVars.csv file and these fields will be irrelevant.

Maturation and fecundity – These fields are ignored when size control is operating.

### Deterministic example runs

#### Temperature and grow days

Here, we present an example sensitivity run for how stochasticity in temperature and grow days effects results using mean patch growth rate and initial population abundance as responses. 406 patches were used and no movement was considered between patches to simulate 406 independent metapopulations. K was set to 200 for each patch and initial N set at 100. Initial body sizes to set 8 size classes were 31, 83, 130, 162, 181, 192, 199, and 206 mm. Males and females matured at the set given length of 134 mm and 156 mm, respectively. Females had a constant mean egg number (option ‘3’) given their length and following the Downs et al. 1997 exponential equation (Egg\_Mean\_par1 = 9.55 and Egg\_Mean\_par2 = 0.018). Egg mortality was set at 62%. Because we were interested in the effect temperature and grow days had on growth, we used the fit age-length-temperature data equation (growth\_Loo = 353, growth\_R0 = -0.57, growth\_temp\_max = 13, growth\_temp\_CV = 0.33, growth\_temp\_T0 = -0.196). We allowed packing to operate.

We considered 4 scenarios using the PatchVars file information with the temperature values Out = 2.95 and Back = 11.18, and grow day values of Out = 242 and Back = 123 for each patch: (1) Constant temperature (TConst) and constant grow days (GDConst), (2)Variable temperatures (TVar) around means given and constant grow days (GDConst), (3) Constant temperature (TConst) and variable grow days (GDVar) around means given, and (4) Variable temperatures (TVar) and variable grow days (GDVar) around means given above. Each scenario was run for 20 years with 20 Monte Carlo replicates. Figure 8 and 9 show mean patch growth rate and initial population N results for the 20 replicates (confidence intervals too small to be seen at this scale).



Figure 8: Mean patch growth rate for sensitivity around temperature and grow days at the patch level. TConst – constant temperature, GDConst – constant grow days, TVar – variable temperature, GDVar – variable grow days.



Figure 9: Initial population N for sensitivity around temperature and grow days at the patch level. TConst – constant temperature, GDConst – constant grow days, TVar – variable temperature, GDVar – variable grow days.

#### K / patch

Using all of the same parameters as mentioned in 5.3.1.2, while holding temperature and grow days constant, we considered 5 scenarios for starting K values: K = 100, 300, 500, 700, 900. Each scenario was run for 20 years with 20 Monte Carlo replicates. Figures 10 and 11 show mean patch growth rates and initial population N results for the 20 replicates (confidence intervals too small to be seen at this scale).



Figure 10: Mean patch growth rate for sensitivity around starting K values.



**Population**

Figure 11: Initial population N for sensitivity around starting K values.

#### Initial N

Using all the same deterministic parameters mentioned above and K = 200, we looked at 5 scenarios for sensitivity around starting N: N0 = 25, 50, 100, 150, and 175. Each scenario was run for 20 years with 20 Monte Carlo replicates. Figure 12 and 13 show mean patch growth rate and initial population N results for the 20 replicates (confidence intervals too small to be seen at this scale).



Figure 12: Mean patch growth rate for sensitivity around starting N0 values.



Figure 13: Initial population N for sensitivity around starting N0

## **Terrestrial Example**

CDMetaPOP was originally developed for species living in dynamic riverine landscapes (‘riverscapes’; Ward 1998; Fausch et al. 2002). However, the model is suitable for a spectrum of spatial processes that operate in any spatially heterogeneous environment (e.g., terrestrial; Landguth et al. 2015 or aquatic; Schlosser 1995; Rieman and Dunham 2000) where spatial arrangement of habitats, habitat-specific demographic variables, and behavioral variables describing the dispersal characteristics of the species can be defined (Pulliam et al. 1992). Each of the modules described in this user manual can be essentially turned on or off depending on the objective of the simulation study. Here, we show a terrestrial example using Whitebark Pine (WBP; *Pinus Albicaulus*) and the added influence of White Pine Blister Rust (WPBR; *Cronartium ribicola*) mortality with resistant genetics in the US Northern Rocky Mountains.

We created a WBP system model that required the development and parameterization of a number of species-specific processes (see Fig. 14). After initialization of the model (e.g., stands, stage structure, and genetics; where forest stands are considered ‘patches’), pollen dispersal was considered during the summer in which the age 0 fertilization event occurs. Then, age 1 cones from the previous year’s fertilization event began to emerge on each tree and dispersed in the fall. Overwinter, stage-structured density-independent mortality is considered while checking each stand’s carrying capacity. Growth and new mature individuals occurred by spring with the additional WPBR mortality on mature individuals implemented. More detailed methods with data sources for developing the model are outlined below.



Figure 14. CDMetaPOP flow diagram with respect to Whitebark Pine major processes.

Stands and carrying capacity: The WBP simulation extent constrained to the Northern Rockies seed zones extent (Mahalonvich and Hopkins 2011; Fig. 15) contained 1,059 initial spatially-delineated stand locations separated by 5 km. Presence or absence of WBP was determined by a 0.5 probability cutoff using habitat suitability data (Landguth et al. 2015). We assumed a carrying capacity (K) of 100 at each stand location.



Figure 15: Study area defined by the northern Rockies seed zones (Malahovich and Hopkins 2011) with initial 1,059 stand locations. WPBR relative spatial selection mortality shown for each stand.

Stage classes: We initialized the model at time = 0 with a random distribution of 500 age classes. After 25 years, a stable age distribution was reached at which time we began genetic exchange. We assumed that age 0 individuals were fertilized individuals and 12 months later emerged as age 1 cones on the trees that could then be dispersed. Size classes increased monotonically at a diameter at breast height (DBH) of 0.2 cm each year (Keane et al. 2007). As trees progressed through each size class, size-linked parameters (e.g., probability of mortality, probability of maturation, and fecundity) varied.

Genetics: Genetic exchange was initiated 25 years into the simulations in order to stabilize the population dynamics first before introducing spurious genetic bottlenecks due to transient population dynamics. We initialized the model with allele frequency files generated for each seed zone (Mahalonvich and Hopkins 2011) that comprised 16 loci with at most 9 polymorphic alleles per locus. In addition, we added a selection-driven locus. This locus was putatively adaptive with 0.01 and 0.99 frequency for the first and second allele, respectively (see below description on White Pine Blister Rust selection for more details on how adaptive genetic processes were implemented). We did not consider mutation, which is reasonable considering the short simulation time period.

Maturation and fecundity: Mature individuals were assumed to be > age 20. Although WBP may typically take longer to reach maturity when growing at poor sites, we used a lower bound of 20 years drawn from literature in order to allow for more generations in the short time frame simulated (Fire Effects Information System, accessed September 2015). A size-based fecundity model was implemented to determine the number of seeds produced at a given basal area per stand following the individual tree DBH conversion to basal area: Basal Area = 0.00007854 \* DBH2 (Keane et al. 2007). Then, we used the value of 500 cones per 1 basal area (Barringer et al. 2012) multiplied by 20 seeds per cone, which yielded a size-based seed production per individual tree.

Mortality: Mortality in the model is regulated by both density-dependent and density-independent processes. In order to simplify processes and isolate the effects of White Pine Blister Rust mortality, we only considered density-independent processes using class-based mortalities. We applied a 99% probability of mortality to the age 0 class to mimic 1% probability of seed survival (Izlar 2002). We implemented a cumulated 35% probability of survival for age classes 1 – 15 (Izlar 2002). Trees age 500 and older underwent a 25% probability of survival, which allowed for occasional long-lived trees given the length of the simulation time. If a stand happened to reach K, then a random removal of excess individuals was conducted (e.g., Balloux 2001).

Reproduction and pollen dispersal: Reproduction within and across stands was monecious with selfing allowed. Once a pollen spore settled at a respective female counterpart, a fertilization event was assumed, meaning gene exchange was considered and stored as age 0 individuals (not contributing to K) while incubation for 12 months would occur. Pollen movement during summer months was assumed to follow an isolation-by-distance model: probability of pollen dispersal to a respective female cone locations was a function of the inverse-square Euclidean distance with a 50 % maximum study area distance threshold (450 km).

Cone/seed dispersal: Emerged age 1 cones from the previous year were dispersed from individual trees following an isolation-by-distance movement pattern similar to pollen dispersal: probability of cone dispersal to a new stand location was a function of the inverse-square Euclidean distance with a 30 km maximum distance threshold (Lorenz et al. 2011). This produced the majority of cones staying in the same stand or nearest neighbor stands (i.e., dropping near parent tree) with occasional longer distance cone dispersal (e.g., Clark’s Nutcracker). We assumed a cone produced 20 seeds. Seed survival was assumed to be 1 % and the ability for a seed to establish in a new stand location was determined based on resource availability (i.e., carrying capacity not exceeded in stand location cone dispersed to).

White Pine Blister Rust mortality and resistance: We used WPBR probability of occurrence map values (Landguth et al. 2015) at each stand as a proxy for differential mortality applied to mature trees only (Fig. 1). Depending on the simulation scenario (see following section), values at each stand were implemented depending on the genotype of each individual. This allowed us to isolate the process for a simple model of WPBR resistance, by defining a priori a locus under selection with a genotype being selected for.

Simulation scenarios and analysis: We conducted simulation scenarios to check the overall influence of WPBR presence via added mortality at each stand and then, introduce a gene that was resistant to this mortality (Table 3). We ran simulations for 130 years, with the first 25 years considered ‘burn-in’ for the population dynamics and to stabilize age distributions. We plotted overall mean population abundance, allelic diversity, and heterozygosity for each scenario and used 10 replicates to assess variation in each metric.

Table 3: Simulation scenarios

|  |  |  |
| --- | --- | --- |
|  | **Scenario Name** | **Description** |
| **WPBR mortality** | No Mortality | The null model in which no WPBR mortality considered. |
| All Stands Gene Resistance | All stand locations applied WPBR mortality (Fig. 1). One genotype assumed to be resistant to WPBR. |
| All Mortality | All stand locations applied WPBR mortality (Fig. 1) regardless of genetic makeup. |

Results: Overall population mean abundance for each scenario is shown in Figure 16. ‘No Mortality’ (Fig. 16; dashed black line) shows stable population dynamics with a declining population for the extreme scenario ‘All Mortality’ (Fig. 16 red line). When we introduced a gene for WPBR selection applied at every stand (Fig. 16 blue line), population abundance still declined, but higher than the ‘All Mortality’ scenario. Overall population mean allelic diversity is shown in Figure 17 and shows similar patterns to that of the population abundance graphs.



Figure 16. Population abundance through time for each scenario.



Figure 17. Allelic diversity through time for each scenario.

# **General Issues**

## **How to obtain CDMetaPOP**

The program is freeware and can be downloaded at https://github.com/ComputationalEcologyLab/CDMetaPOP.

## **Debugging and troubleshooting**

Please report problems including any bugs, to erin.landguth@mso.umt.edu.

## **How to cite CDMetaPOP**

The reference to cite is as follows, substituting the version number:

Landguth EL, Bearlin A, Day C, Dunham J (2017) CDMetaPOP: an individual-based, eco-evolutionary model for spatially explicit simulation of landscape demogenetics. Methods in Ecology and Evolution 8(1):4-11. v2.54.

## **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 x.x). CDMetaPOP 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. Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

We strongly urge you to read the entire documentation before ever running CDMetaPOP. 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 CDMetaPOP 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 CDMetaPOP. 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 CDMetaPOP 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 CDMetaPOP in ecological investigation and management application. Therefore, we encourage you to contact us and describe your application after using CDMetaPOP.

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

# **References**

Allendorf FW, Leary RF, Spruell P, Wenburg JK (2001) The problems with hybrids: setting conservation guidelines. Trends in ecology & evolution, 16, 613-622.

Balloux F. (2001) EASYPOP (version 1.7): a computer program for population genetic simulations. Journal of Heredity, 92, 301–302.

Barringer, LE, Tomback, DF, Wunder, MB, McKinney, ST (2012) Whitebark Pine Stand Condition, Tree Abundance, and Cone Production as Predictors of Visitation by Clark's Nutcracker. PLOS ONE. doi: 10.1371/journal.pone.0037663.

Bear EA (2005) Effects of temperature on survival and growth of westslope cutthroat trout and rainbow trout: Implications for conservation and restoration. Montana State University Master of Science Thesis.

Bowcock AM, Ruiz-Linares A, Tomfohrde J et al. (1994) High resolution of human evolutionary trees with polymorphic microsatellites. Nature, 368, 455–457.

Bowerman T, "A Multi-Scale Investigation of Factors Limiting Bull Trout Viability" (2013). All Graduate Theses and Dissertations. Paper 1524. http://digitalcommons.usu.edu/etd/1524

Bertalanffy, L. von (1938) A quantitative theory of organic growth(Inquiries on growth laws. II). Human Biol. 10, 181-213.

Castillo JA, Epps CW, Davis AR, Cushman SA (2014) Landscape effects on gene flow for climate-sensitive montane species, the American pika. Molecular Ecology, 23, 843-856.

Day CC, Landguth EL, Bearlin A, Holden ZA, Whiteley AR (2018) Using simulation modeling to inform management of invasive species: A case study of eastern brook trout suppression and eradication. Biological Conservation 221:10-22.

Day CC, Landguth EL, Simmons RK, Baker WP, Whiteley AR, Lukacs PM, Bearlin A (2020) Simulating effects of fitness and dispersal on the use of Trojan sex chromosomes for the management of invasive species. Journal of Applied Ecology 57(7):1413-25.

Downs CC, White RG, Shepard BB (1997) Age at sexual maturity, sex raio, fecundity, and longevity of isolated headwater populations of westlope cutthroat trout. North American Journal of Fisheries Management, 17, 85-92.

Dyer R (2014) popgraph: This is an R package that constructs and manipulates population graphs. http://cran.r-project.org/web/packages/popgraph/index.html

Ettan J van (2015) gdistance: Distances and routes on geographical grids. <http://cran.r-project.org/web/packages/gdistance/index.html>.

Elith J., Leathwick J.R. (2009). "Species distribution models: ecological explanation and prediction across space and time". Annual Review of Ecology, Evolution, and Systematics40: 677–697. doi:10.1146/annurev.ecolsys.110308.120159

Frank, B.M., Piccolo, J.J., & Baret, P.V. (2011) A review of ecological models for brown trout: towards a new demogenetic model. Ecology of Freshwater Fish, 20, 167-198.

Izlar, D.K. (2002) Assessment of Whitebark Pine seedling survival for Rocky Mountain plantings. M.S. Thesis Humboldt State Univeristy.

Jones MR, Forester BR, Teufel AI, Adams RV, Anstett DN, Goodrich BA, Joost S, Manel S, Landguth EL (2014) Integrating spatially-explicit approaches to detect adaptive loci in a landscape genomics context. Evolution (Special Issue: Evolutionary Landscape Genetics), 67, 3455-3468.

Keane, R., K. Gray and L. Dickinson (2007). Whitebark Pine Diameter Growth Response to Removal of Competition. RMRS General Technical Report Research note RN-32.

Labonne, J., Ravigné, V. Parisi, B., & Gaucherel, C. (2008) Linking dendritic network structures to population demogenetics: the downside of connectivity. Oikos, 117, 1479-1490.

Landguth EL, Cushman SA (2010) CDPOP: a spatially explicit cost distance population genetics program. Molecular Ecology Resources, 10, 156-161.

Landguth EL, Cushman SA, Schwartz MK, Murphy M, McKelvey KS, Luikart G (2010) Quantifying the lag time to detect barriers in landscape genetics. Molecular Ecology, 19, 4179-4191.

Landguth EL, Balkenhol N (2012) Relative sensitivity of neutral versus adaptive genetic data for assessing population differentiation. Conservation Genetics, 13, 1421-1426.

Landguth EL, Muhlfeld CC, Luikart G (2012a) CDFISH: an individual-based, spatially-explicit, landscape genetics simulator for aquatic species in complex riverscapes. Conservation Genetics Resources, 4, 133-136.

Landguth EL, Cushman SA, Johnson NA (2012b) Simulating natural selection in landscape genetics. Molecular Ecology Resources, 12, 363-368.

Landguth EL, Hand BK, Glassy JM, Cushman SA, Sawaya M (2012c) UNICOR: a species corridor and connectivity network simulator. Ecography, 12, 9-14.

Landguth EL, Fedy B, Garey A, Mumma M, Emel S, Oyler-McCance S, Cushman SA, Wagner HH, Fortin MJ (2012) Effects of sample size, number of markers, and allelic richness on the detection of spatial genetic pattern. Molecular Ecology Resources, 12, 276-284.

Landguth EL, Holden ZA, Mahalovich MF, Cushman SA (2017) Using landscape genetics simulations for planting blister rust resistant whitebark pine in the US Northern Rocky Mountains. Frontiers in genetics 2017 8:9.

Leslie, P.H. (1945) On the use of matrices in certain population mathematics. *Biometrika,* **33,** 183-212.

Lorenz, T.J.; Sullivan, K.A.; Bakian, A.V.; Aubry, C.A. 2011. Cache-site selection in Clark’s nutcracker (*Nucifraga columbiana*). Auk. 128, 237−247.

Mahalovich, M.F., Hipkins, V.D. (2011) Molecular genetic variation in whitebark pine (*Pinus albicaulis Engelm.)* in the Inland West. In: Keane, R.E. (ed.) "High-Five Symposium: The Future of High-Elevation Five-Needle White Pines in Western North America. 2010 June 28-30, Missoula, MT, USA. Proc: RMRS-P-63. For Collins, CO: USDA Forest Service, Rocky Mountain Research Station, p. 124-139.

McRae BH (2006) Isolation by resistance. Evolution 60:1551-1561.

M’Gonigle LK, FitzJohn RG (2009) Assortative mating and spatial structure in hybrid zones. Evolution, 64, 444-455.

Miller DH, Ankley GT (2004) Modeling impacts on populations: fathead minnow (Pimephales promelas) exposure to the endocrine disruptor 17β-trenbolone as a case study. Ecotoxicology and Environmental Safety, 59, 1-9.Oliphant TE (2006). Guide to NumPy, Provo, UT., Brigham Young University, <http://www.tramy.us/>.

Mims MC, Day CC, Burkhart JJ, Fuller MR, Hinkle J, Bearlin A, Dunham JB, DeHaan PW, Holden ZA, Landguth EE. (2019) Simulating demography, genetics, and spatially explicit processes to inform reintroduction of a threatened char. Ecosphere 10(2):e02589.

Oyler-McCance SJ, Fedy BC, Landguth EL (2013) Sample design effects in landscape genetics. Conservation Genetics, 14, 275–285.

Peacock MM, Dochtermann NA (2012) Evolutionary potential but not extinction risk of Lahontan cutthroat trout 9Oncorhynchus clarkia henshawi) is associated with stream characteristics. Can. J. Fish. Aquat. Sci. 69, 615-626.

Peakall R and Smouse PE (2006) GENALEX 6: genetic analysis in Excel. Population genetic software for teaching and research. Molecular Ecology Notes. 6, 288-295.

Peakall R and Smouse PE (2012) GenAlEx 6.5: genetic analysis in Excel. Population genetic software for teaching and research-an update. Bioinformatics 28, 2537-2539.

Piper R.G., McElwain I.B., Orme L.E., McCraren J.P., Fowler L.G., and Leonard J. R. (1982) Fish hatchery management. USFWS, 517 pp.

Pritchard JK, Stephens M, Donnelly PJ (2000) Inference of population structure using multilocus genotype data. Genetics 155: 945-959.

Prunier J.G., Kaufmann B., Fenet S., Picard D., Pompanon F., Joly P., Lena J.P., 2013. Optimizing the trade-off between spatial and genetic sampling efforts in patchy populations: towards a better assessment of functional connectivity using an individual-based sampling scheme. Molecular Ecology 22, 5516–5530

R Development Core Team (2012) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Radinger J. and Wolter C, 2014. Patterns and predictors of fish dispersal in rivers. Fish and Fisheries, 15:456-473.

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

Ray C, Peacock MM, Dunham JB (2007) (2007). Demography and population dynamics of Lahontan cutthroat trout (Oncorhynchus clarkii henshawi) stream populations in eastern Nevada. Reno, United States Fish and Wildlife Service. Cooperative Agreement FSW 14-48-0001-95646.

Raymond M. & Rousset F, 1995. GENEPOP (version 1.2): population genetics software for exact tests and ecumenicism. J. Heredity, 86:248-249

Rousset, F., 2008. Genepop'007: a complete reimplementation of the Genepop software for Windows and Linux. Mol. Ecol. Resources 8: 103-106.

Rideout RM, Tomkiewicz J (2011) Skipped spawning in fishes: More common than you might think. Marine and coastal fisheries: Dynamics, management, and ecosystem science, 3:1, 176-189.

Rossum, G van. and Drake, F. L. (eds), 2006. Python Reference Manual, Python Software Foundation,. http://docs.python.org/ref/ref.html.

Schill DJ, LaBar GW, Mamer ERJM, Meyer KA (2010) North American Journal of Fisheries Management 30:1352–1363.

Scipy 1.0: fundamental algorithms for scientific computing in Python. (2020) Nature Methods 17, 261-272.

Todd CR, Ng MP (2001) Generating unbiased correlated random survival rates for stochastic population models. Ecological Modelling 144, 1-11.

West Fork Environmental. 2013. Salmonid tissue sampling in the Boundary Hydroelectric Project Area, NE Washington. Prepared for: City of Seattle, Seattle City Light Department

Wright S (1932) The roles of mutation, inbreeding, crossbreeding and selection in evolution. Proceedings of the Sixth International Congress of Genetics, 1, 356–366.

# **Acknowledgements**

We thank Helen Neville, Dan Dauwalter, Seth Wegner, Harry Rich, Al Solonsky, and Robin Bjork for fruitful discussions during model development. We thank the class members of FNR 691 from Purdue University, and three anonymous reviewers for helpful manuscript comments. We thank Zack Holden for data contributions for example applications. We thank Lucas Nathan, Marco Alejandro Escalante, and Patrick James for beta testing. Funding for this research comes from Seattle City Light and NASA grant NNX14AC91G.

1. In later models, spatial distribution of K could be used to simulate other factors such as habitat degradation or competition with other species. [↑](#footnote-ref-1)
2. Could be implemented as a function of surrogate variables such temperature or flow [↑](#footnote-ref-2)
3. Spatial structure in age/stage specific mortalities could be used to simulate fishing pressure or other specific source of additional mortality. [↑](#footnote-ref-3)
4. Potential target for selective/adaptive control [↑](#footnote-ref-4)