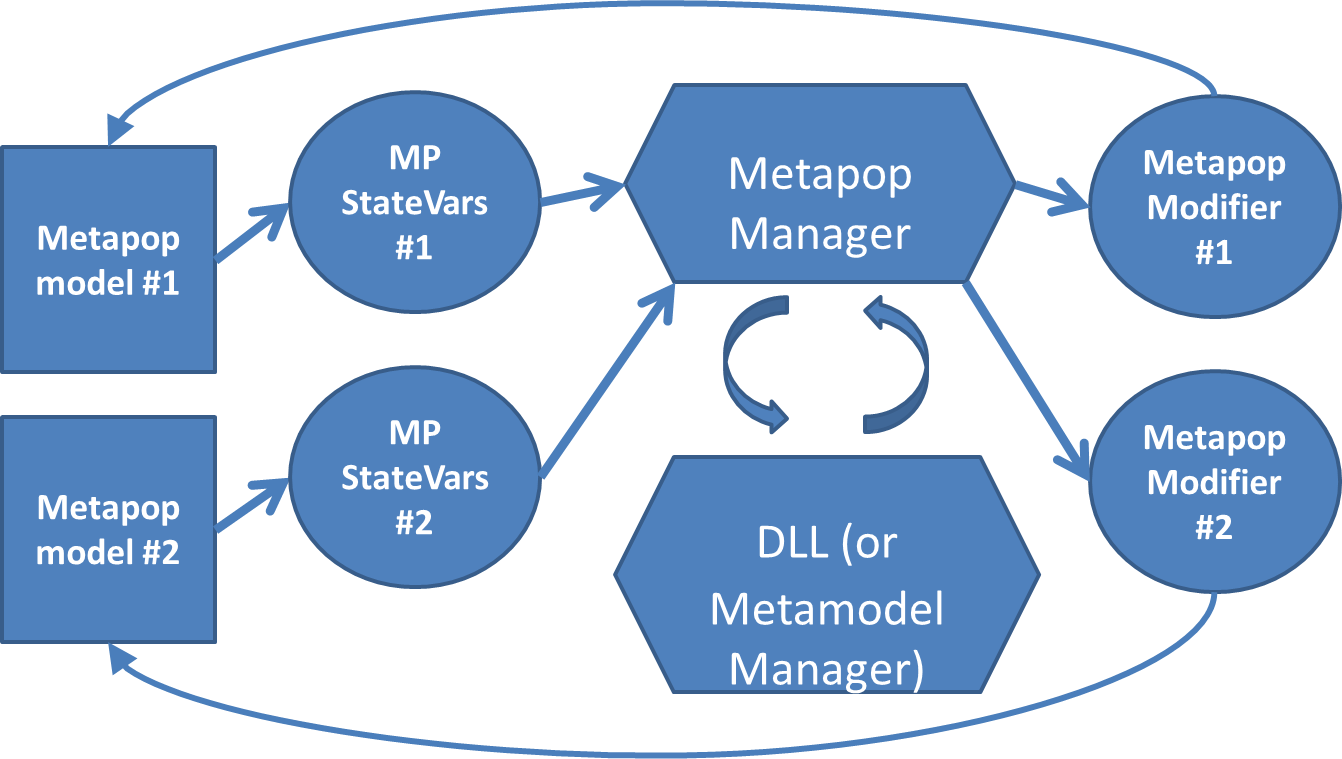
Enabling metamodel linkages using Ramas Metapop software

**Make changes so that not all changes are multipliers**

**Strategy:**  To enable metamodel linkages in Ramas Metapop, we (1) implemented a new "metamodel mode" (batch mode command: RPC=T; no changes should be made to the Metapop GUI) that instigates a suite of changes within Metapop that accommodates information transfer among multiple instances of Metapop (see below), (2) built a "Metapop Manager" software to manage information transfer between multiple (for now, one or two) instances of Metapop, pausing and initiating simulations as necessary, (3) specified an "Metapop StateVars" storage structure to communicate information from Metapop to Metapop Manager (and subsequently used to inform metamodel linkages), and (4) specified a new "Metapop Modifier" storage structure to communicate information from Metapop Manager back to Metapop. The "Metapop StateVars" stores information from each time step that may be used as independent variables for metamodel linkages. The "Metapop Modifier" simply provides a set of modified parameters that (once read into Metapop) will update existing Metapop parameters. The Metapop Modifier alters parameters in Metapop using a mode of action similar to the various forms of "catastrophes" already implemented in Metapop. Like catastrophes, the Metapop Modifier can modify abundance, vital rates, carrying capacities or dispersal rates via conditional expressions analogous to those that govern catastrophes. This provides substantial flexibility for modifying Metapop parameters (as currently implemented, catastrophes can alter abundance, vital rates, carrying capacity, and dispersal rates), and should be compatible with most of the existing functionality of Ramas Metpop, including many existing linkages with Ramas GIS (e.g., survival and fecundity modifiers) and Ramas Landscape (implemented as ".XCH" files).



*Fig 1. Schematic of a proposed method for implementing metamodel linkages among instances of Ramas Metapop.*

BOX 1

**Specification for a "Metapop Modifier" object (one modifier object specified per Metapop instance):**

Fields:

1. Type: Categorical indicator of what is to be modified: abundance, vital rates, carrying capacity or dispersal (ultimately, it should be possible to modify more than one; therefore, an array of Booleans may be most appropriate)
2. Timestep: Last year for which complete simulation data are available for all component models (scalar integer).
3. Complete: Boolean indicator of whether this modifier object is complete and ready to be used in a Metapop simulation (single Boolean).
4. mmActive: Boolean indicator of whether this modifier object is currently in effect (single Boolean). [[for now, always TRUE]]
5. VarList: List of optional variables that can be monitored, and thereby used to modify any of the metamodel linkage options specified in Type (vector of Booleans of length "number of possible variables"- see Metapop Manager description below for a more detailed list of variables).
6. (if applicable) mmAbund: modified stage-specific abundances (matrix of dimension "nPopulations", "nStages"). *Replaces stage-specific abundances for each population. Not compatible with catastrophes affecting abundance.*
7. (if applicable) mmVital: modified vital rates (matrix of dimension "nPopulations", "nStages", "nStages"). *Replaces mean population-specific transition matrix values. Not compatible with temporal changes in vital rates or catastrophes that affect vital rates.*
8. (if applicable) mmK: modified mean population-specific carrying capacities (vector of length "nPopulations"). *Replaces mean population-specific K values. Not compatible with "temporal change in K" or catastrophes that affect K.*
9. (if applicable) mmDisp: dispersal rate modifier (vector of length "nPopulations"). *Modifies dispersal rates relative to the rate expected for a given stage and population pairing. Not compatible with catastrophes that affect dispersal. [[this should probably be tabled for now, until we can clarify the details]]*

The application of the Metapop Modifier is fairly straightforward; in this case stage-specific abundances from the previous year are simply replaced by the number specified by the modifier object. For metamodel linkages that modify carrying capacity, the new carrying capacity stored in the Metapop modifier object simply replaces the mean K value specified for that population (specified in the Populations dialog; this option is not compatible with the "temporal trend in K" option or carrying capacity change specified with a .KCH file or catastrophes that affect K). For metamodel linkages that modify vital rates, the new vital rate matrix replaces the population-specific mean transition matrix (a compound parameter computed within Metapop as the product of the specified stage matrix values, any vital rate multipliers from catastrophes within Metapop, and any population-specific survival or fecundity modifiers [relative fecundity, relative survival]). Vital rate change (.SCH and .FCH) files are not compatible with metamodel linkages that modify vital rates. Like catastrophes that affect vital rates, scramble and contest density dependence are not compatible with metamodel linkages that affect vital rates. Finally, dispersal rate modifiers are applied to the mean dispersal rate for a given population pairing (from distance function), source population, and stage-specific modifier.

**BOX 2:**

**Specification for "MP StateVars" object (one storage structure specified per Metapop instance):**

Note: not all of these variables will be stored for each metamodel. Instead, the user (ultimately, through Metapop Manager) must specify which variables are to be stored. Information about which variables to store is then stored in the "Metapop modifier" object (see Box 1). [[To facilitate temporal mismatches, future versions may store data for the current and previous year(s) (or a moving average of multiple years?). However, the current version will only store the current year's data]]

*Global state variables (read from Metapop, cannot be modified from outside but may be used in forging metamodel linkages). These data are always stored by Metapop Manager but only need to be read in once.*

* nPopulations: Number of populations (scalar) (must be read in each year)
* nStages: Number of stages (scalar)
* nMP: Number of Metapop instances being run simultaneously (scalar).
* constraintsMat: Constraints matrix (matrix of dimension "nStages", "nStages")
* stageMass: Stage-specific average mass (vector of length "nStages")
* stageBreed: Stage-specific proportion breeding (vector of length "nStages")
* stageRelDisp: Stage-specific relative dispersal (vector of length "nStages")
* stageDD: Basis for density dependence (vector of length "nStages")
* nCatastrophes: Number of defined catastrophes (scalar)
* catType: Indicates whether catastrophe affects survival, abundance, or K (Boolean array of dimension "nCatastrophes", 4 [4 types of catastrophe])
* catStages: Stage-specific catastrophe multiplier (array of dimension "nCatastrophes","nStages")

*Population state variables (read from Metapop, cannot be directly modified from outside but are optionally available for use in forging metamodel linkages). These variables are stored only if a user has specified to do so.*

* popAbundTot: Total abundance for each population (vector of dimension "nPopulations")
* popAbundFem: Total number of adult females (vector of dimension "nPopulations")
* popAbundMale: Total number of adult males (vector of dimension "nPopulations")
* PBM: percent breeders that have mates (vector of dimension "nPopulations")
* disperserMat: Disperser matrix (array of dimension "nPopulations", "nPopulations", "nStages")
* dispersalMat: Dispersal matrix (matrix of dimension "nPopulations", "nPopulations")
* NDF: correlated standard normal deviate for fecundity (vector of length "nPopulations")
* NDS: correlated standard normal deviate for survival (vector of length "nPopulations")
* NDK: correlated standard normal deviate for carrying capacity (vector of length "nPopulations")
* popEmigrants: Number of emigrants (array of dimension "nPopulations", "nStages")
* popImmigrants: Number of immigrants (array of dimension "nPopulations", "nStages")
* popCatStatus: Catastrophe status (array of dimension "nPopulations", 2)
* popTimeSince: Time since last catastrophe (array of dimension "nPopulations", 2)
* popCatProb: Catastrophe probability (vector of length "nPopulations")
* popRmax: Maximum growth rate (vector of length "nPopulations")
* popHarvest: number of individuals harvested (array of dimension "nPopulations", "nStages")
* popSupplement: number of individuals supplemented (array of dimension "nPopulations", "nStages")
* popK2: actual or realized K (with stochasticity).
* popStMat2: Realized transition matrices, including stochasticity …

*Population state variables (read from Metapop: modifications to these variables are translated into a set of multipliers stored in the "Metapop modifier" object). These variables*

* popAbundSt: Stage-specific abundances for each population (matrix of dimension "nPopulations","nStages") [[modifications result in specification of "abundance modifier" ]]
* popStMat: Mean transition matrices for each population (array of dimension "nPopulations", "nStages", "nStages") [[modifications result in specification of "vital rate modifier"]]
* popK: Mean population-specific carrying capacity (vector of length "nPopulations"). [[modifications result in specification of "K modifier"]]
* relDispersal: Relative dispersal rate (matrix of dimension "nPopulations", "nStages") [[modifications result in specification of "dispersal modifier"]]

**Detailed Algorithm for RAMAS Metapop, with modifications for implementing metamodel linkages.**

**Set up simulation (only performed once, at the initialization step)**

1. Check stage structure and make necessary corrections:
   1. All elements of stage matrices and standard deviations matrix must be nonnegative. Positive standard deviations for vital rates with zero means are ignored.
   2. If constraints are not to be ignored, survival rates (stage matrix elements multiplied by the corresponding elements of the Constraints matrix) must be between 0 and 1, and the sum of survival rates in each *column* must be between 0 and 1.
   3. For each population, the total of all stage abundances must be equal to the total (initial) abundance.
   4. If model includes sex structure (separate stages for females and males), the stage matrix elements must be consistent with the mating system, constraints matrix and the proportion breeding in each stage (see help file).
2. Check dispersals
   1. Each dispersal rate must be between 0 and 1, and total dispersal rate from each population must be less than 1.
   2. If dispersal is density dependent, checks in step 2a must hold with close to 0 population size as well as with population sizes above the carrying capacity.
3. Check correlations
   1. Correlation matrix must be symmetric and positive-definite.
   2. Use the correlation matrix to calculate the decomposition matrix that will be used in producing correlated random deviates in step vii. (see Burgman et al. 1993).
4. Initialize variables for storing results
5. IF in Metamodel Mode: write time- and replicate-independent state variables to Metapop Manager
   1. Share global state variables with Metapop Manager: nPopulations, nStages, constraintsMat, stageMass, stageBreed, stageRelDisp, stageDD, nCatastrophes, catType, catStages (see detailed Metapop Manager description, below, for variable descriptions)
   2. Write population state variables to Metapop Manager: popRMax.

**Replications:** for each replicate simulation run, repeat the following steps

1. Initialize population sizes and structures from initial abundances; initialize replication variables; initialize counters for timesteps since catastrophe (from Catastrophes tab of the Populations dialog).
2. If in metamodel mode, initialize the arrays for storing data to be transferred to the external "manager" software (Metapop manager, Metamodel Manager). See detailed algorithm for Metapop manager, below.
3. ***Time steps:*** for each time step, repeat the following steps
   1. ***IF in metamodel mode and IF specified to do so by Metapop Manager (mmActive==TRUE):***
      1. ***Pause the simulation***
      2. ***When ready (Complete=TRUE), read the "Metapop modifier" object (see Box 1) from MP manager and resume the simulation. Check to make sure the current year of the simulation matches with (Year – 1).***
   2. Increment counters for timesteps since catastrophe.
   3. If a catastrophe is spreading, then compound local probability with "additional probability" due to spread form other populations (which was calculated in the previous time step and stored).
   4. Sample uniform random numbers *rand1* and *rand2* for regional catastrophes (based on the correlation between the two catastrophes, if both are regional).
   5. Regional catastrophe abundances
      1. If the simulation is stochastic, and catastrophe 1 extent is regional, and catastrophe affects abundances, and *rand1* < *CR1*, reduce number of individuals in each stage:

*Npj*’ = *Npj* · *CE1p* · *CS1j*

and reset counter for time since catastrophe.

* + 1. If the simulation is stochastic, and catastrophe 2 extent is regional, and catastrophe 2 affects abundances, and *rand2* < *CR2*, reduce number of individuals in each stage:

*Npj*’ = *Npj* · *CE2p* · *CS2j* and reset counter for time since catastrophe.

* + 1. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record catastrophe status for catastrophes 1 and 2 for use by Metapop Manager (popCatStatus).
    2. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record time since last catastrophes 1 and 2 for use by Metapop Manager (popTimeSince)
  1. If the simulation is stochastic, produce 3 sets of *n* correlated normal deviates:

*NDFp*, *NDSp*, and *NDKp*, for *p*=1...*n*, where *n* is the total number of populations (see Burgman et al. 1993). The number of independent sets depends on within-population correlation: If "All correlated", then *NDFp* = *NDSp* = *NDKp* (only 1 independent set). If "F and S correlated", then *NDFp* = *NDSp*, but *NDKp* are independent (two independent sets). If "F, S, K uncorrelated", then *NDFp*, *NDSp*, *NDKp* are independent (three independent sets).

* + 1. If in metamodel mode, and if required (i.e., these variables are listed in VarList), record correlated deviates *NDFp*, *NDSp*, *NDKp*, for use by Metapop Manager (NDF, NDS, NDK).
  1. ***Population growth***: for each population *p*, repeat the following steps
     1. Sample uniform random numbers *rand1* and *rand2* for local catastrophes (based on the correlation between the 2 catastrophes, if both are local), and store them in arrays
     2. Local catastrophe - abundances
        1. If the simulation is stochastic, and catastrophe 1 extent is local, and catastrophe 1affects abundances, and *rand1* < *CL1p*, reduce number of individuals in each stage:

*Npj*’ = *Npj* · *CE1p* · *CS1j*

and reset counter for time since catastrophe 1.

* + - 1. If the simulation is stochastic, and catastrophe 2 extent is local, and catastrophe 2 affects abundances, and *rand2* < *CL2p*, reduce number of individuals in each stage:

*Npj*’ = *Npj* · *CE2p* · *CS2j \*

and reset counter for time since catastrophe 2.

* + - 1. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record catastrophe status for catastrophes 1 and 2 for use by Metapop Manager (popCatStatus).
      2. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record time since last catastrophes 1 and 2 for use by Metapop Manager (popTimeSince)
    1. If in metamodel mode: apply Metapop modifier – abundances
       1. If an abundance modifier is in effect, then replace the number of individuals in each stage:

*Npj*’ = *mmAbundpj*

* + 1. Catastrophe - carrying capacities
       1. If "Temporal trend in K" is a number, then:
          1. If the simulation is stochastic, and there is a catastrophe 1 (regional or local), and catastrophe 1 affects carrying capacities, and *CE1p* < 1, reduce the carrying capacity:

*Kp*’ = *Kp* · *CE1p*

and reset counter for time since catastrophe.

* + - * 1. If the simulation is stochastic, and there is a catastrophe 2 (regional or local), and catastrophe 2 affects carrying capacities, and *CE2p* < 1, reduce the carrying capacity:

*Kp*’ = *Kp* · *CE2p*

and reset counter for time since catastrophe.

* + - 1. If "Temporal trend in K" is a file, then:
         1. If the simulation is stochastic, and there is a catastrophe 1 (regional or local), and catastrophe 1 affects carrying capacities, and *CE1p* < 1, then reset counter for time since catastrophe.
         2. If the simulation is stochastic, and there is a catastrophe 2 (regional or local), and catastrophe 2 affects carrying capacities, and *CE2p* < 1, then reset counter for time since catastrophe.
      2. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record catastrophe status for catastrophes 1 and 2 for use by Metapop Manager (popCatStatus).
      3. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record time since last catastrophes 1 and 2 for use by Metapop Manager (popTimeSince)
    1. If in metamodel mode, and if a carrying capacity modifier is in effect:
       1. Store K information for export (in MPStateVars):

*popK* = *Kp*

* + - 1. apply Metapop modifier – carrying capacities: replace the carrying capacity based on information in Metapop modifier:

*Kp*’ = *mmKp*

* + - * 1. Note: ignore any temporal trend in K. Issue a warning if "Temporal trend in K" is not equal to zero or if a .KCH file is in effect.
    1. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record mean population-specific K (popK) for use by Metapop Manager:
       1. If carrying capacity modifier is NOT in effect:
          1. If "Temporal trend in K" is a number, then:

*popK = Kp* + (*TTp* · *timestep*)

* + - * 1. If "Temporal trend in K" is a file, then:

*popK = KT*[*p*,*timestep*]

* + 1. Calculate current carrying capacity of this population, incorporating its temporal trend and (if the simulation is stochastic) correlated random variation. If temporal trend in K is linear or zero, then

*Kp*(*t*) = *Kp* + (*TTp* · *timestep*) + (*NDKp* · *KDp*),

If temporal trend in K is read from a file, then

*Kp*(*t*) = *KT*[*p*,*timestep*] + (*NDKp* · *KDp*)

where *timestep* is the number of time steps since the beginning of the simulation, or since the last catastrophe that affected carrying capacity of this population, whichever

is less, and *KT*[*p*,*t*] is the, and *CEx p*<1 for that catastophe (x), then the sequence of *K*s carrying capacity for population *p* for timestep *t* read from a file. Note that if temporal trend in K is read from a file, and there was a catastrophe that affected carrying capacity of this population for *p* read from the file is reset to its beginning.

* + 1. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record actual (stochastic) population-specific K (PopK2) for use by Metapop Manager:

*PopK2 =* *Kp*(*t*)

* + 1. Density dependence (see also *Details* in Chapter 8).
       1. Set transition matrix ***T*** equal to stage matrix for population *p*, modified by relative fecundity and relative survival parameters for stage *j* of population *p* for timestep *t*, where *t* is the number of time steps since the last catastrophe that affected the vital rates of this population (if the "Reset" option is selected in "Advanced Catastrophe Settings"), or since the beginning of the simulation (otherwise).
       2. For user-defined density dependence, call the user DLL (which may modify the transition matrix ***T***), and skip to step (vi.)
       3. For scramble and contest type density dependences (with or without Allee effects), calculate deterministic growth rate based on population size (total, based on selected stages, or fecundity-weighted, as specified in the Density dependence dialog) at this time step:

[[see functions in manual]]

* + - 1. For scramble and contest type density dependences (with or without Allee effects), calculate constant *m*, which, when multiplied with the transition matrix as described below (in (v)), will give a matrix with a dominant eigenvalue equal to *R*(*t*) calculated above (i.e., will give a growth rate that incorporates effects of density dependence). For ceiling or exponential types of density dependence (with or without Allee effects), *m*=1.
      2. Modify mean transition matrix ***T*** by multiplying with constant *m* the elements (fecundities, survivals, or both) that are affected by density dependence
      3. Check the consistency of the mean transition matrix
      4. If Allee effects are specified, modify elements *T ij* of the transition matrix (fecundities, survivals, or both) that are affected by density dependence:

*Tij*’ = *Tij* · (*Bp*/(*Ap*+*Bp*)).

* + 1. **If in metamodel mode, and if vital rate modifier is in effect: apply Metamodel modifier- vital rates**
       1. **If a vital rate modifier is in effect, replace elements of the mean transition matrix:**

***Tij*’ = *mmVitalpij***

* + 1. **If in metamodel mode, and if vital rate modifier is NOT in effect~~,~~ record mean population-specific stage matrix for use by Metapop Manager (PopStMat):**

***PopStMatpij* = *T***

**Note: this ignores any potential catastrophes and env var in vital rates, but does include DD and pop-specific multipliers, which may be time-varying (i.e., it may also include deterministic temporal trends (rel. surv., rel. fec.)).**

* + 1. Catastrophes affecting vital rates
       1. If the simulation is stochastic, and there is a catastrophe 1 (i.e., either *rand1* < *CR1* or *rand1* < *CL1p*, depending on catastrophe 1 extent), and catastrophe 1 affects vital rates, modify mean values of the transition matrix elements:

*Tij*’ = *Tij* · *CE1p* · *CM1ij*

and reset counter for time since catastrophe 1.

* + - 1. If the simulation is stochastic, and there is a catastrophe 2 (i.e., either *rand2* < *CR2* or *rand2* < *CL2p*, depending on catastrophe 2 extent), and catastrophe 2 affects vital rates, modify mean values of the transition matrix elements:

*Tij*’ = *Tij* · *CE2p* · *CM2ij*

and reset counter for time since catastrophe 2.

* + - 1. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record catastrophe status for catastrophes 1 and 2 for use by Metapop Manager (popCatStatus).
      2. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record time since last catastrophes 1 and 2 for use by Metapop Manager (popTimeSince)
    1. **~~If in metamodel mode, and if vital rate modifier is NOT in effect, and if required (i.e., this variable is listed in VarList), record mean population-specific stage matrix for use by Metapop Manager (PopStMat):~~**

***~~PopStMatpij~~* ~~=~~ *~~Tij~~*~~’~~**

**~~Note: In addition to PopStMat above, this also includes DD and catastrophes that affect vital rates..~~**

* + 1. Environmental stochasticity
       1. Generate the current transition matrix based on the mean values as modified in steps (v.) (for density dependence) and (vi.) (for catastrophes affecting vital rates).
       2. If a stochastic simulation, for all columns *j*, and for rows *i*:

*Tij*’ = normal( *Tij* , *SDij* ),

where normal(*m,s*) is a correlated normal deviate with mean *m* and standard deviation *s*:

*Tij*’ = *Tij* + *SDij* · *NDFp* , for fecundities,

*Tij*’ = *Tij* + *SDij* · *NDSp* , for survivals, if "Advanced stochasticity settings"=None (see help file for other settings). Lognormal deviates are similar but more complicated (see Ferson and Akçakaya 1990; Burgman et al. 1993).

* + 1. Check the consistency of the realized transition matrix (see step 1.1)
    2. **If in metamodel mode, and if required (i.e., this variable is listed in VarList), record realized (stochastic) vital rates (PopStMat2) for use by Metapop Manager:**

***PopStMat2pij = Tij’***

* + 1. Project stage abundances
       1. If sex structure is modeled, calculate the number of breeding males and breeding females (based on "Breeding" in Stages dialog, and on the current number of individuals in each stage). Use these numbers to calculate *PBM*, the proportion of breeders that have mates (based on the mating system and the number of mates per male or female). If there is no sex structure, *PBM*=1.
       2. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record number of breeding males and females (and/or PBM) for use by Metapop Manager (popAbundFemale, popAbundMale, PBM).
       3. Stage transitions:
          1. No demographic stochasticity: *Npi*’ = *PBM* · *Tij* · *Npj*, for fecundities, *Npi*’ = *Tij* · *Npj*, for survivals
          2. With demographic stochasticity: *Np*i’ =poi(*PBM*·*T*i*j*·*Npj*), for fecundities, *Npi*’ = bin(*Tij* , *Npj*), for survivals, where poi and bin are functions that return random deviates from Poisson and binomial distributions, respectively (see Price et al. 1986). Note that if the simulation is deterministic (i.e., number of replications specified as 0 in the General information dialog box), then demographic stochasticity is not used, even if specified. Before *Npj* are updated, check consistency of projected abundances, if constraints are not to be ignored (see section 7.7).
          3. If density dependence is Ceiling or Ceiling-Allee, and *N p*>*Kp*(*t*), then reduce abundances of (all or selected, as specified) stages by multiplying each with *Kp*(*t*)/*Np* .
    2. ***Population management (before dispersal)***
       1. Apply population management actions (in the same order as in the Population management dialog box) for which When=Before dispersal, if they are not ignored. Each management action may modify the number of individuals; the modified values are then used in the rest of the management actions.
       2. If in metamodel mode, and if required (i.e., this variable is listed in VarList), record total or stage-specific numbers of animals harvested or supplemented for use by Metapop Manager (popHarvest).
    3. ***Dispersal***
       1. Initialize dispersal variables
       2. For each *source* population *p*:
          1. If there is a catastrophe that affects dispersal rates, modify dispersal rates accordingly
          2. If in metamodel mode, and if a dispersal rate modifier is in effect, modify dispersal rates accordingly [[note: we need to clarify the details of how this works]]
          3. If simulation is stochastic, and CV for dispersal, CVdisp>0 then sample dispersal rates from a normal distribution

*Mip*’ = normal( *Mip* , *Mip* \* CVdisp ), where normal(*m,s*) is a normal deviate with mean *m* and standard deviation *s*

* + - * 1. Calculate the total dispersal rate from this population to all target populations *i*:

*totmig* = *i Mip*

* + - * 1. If dispersal depends on source population size, modify *totmig* as a function of density, *N*:

*totmigD* = *totmig* + (*N\**

*p Kp*(*t*)) · *DMp*.

* + - * 1. Calculate current dispersal rate to each target population *i:*

*Mip*’ = *totmigD*/*totmig* · *Mip*.

* + - * 1. If dispersal depends on target population K, then modify *Mip*’ as a function of target population’s carrying capacity, *Ki*(*t*) (see help file)
        2. Calculate the number of dispersers (emigrants), *NMji*, from each stage *j* of this population to each other population *i*:

No demographic stochasticity: *NMji* = *Mip* · *PMjp*(*k*) · *SMj* · *Npj*

With demographic stochasticity: *NMji* =binomial((*Mip*·*PMjp*(*k*)·*SMj*),*Npj*)

* + - * 1. If in metamodel mode, and if required, record total or stage-specific numbers of migrants for use by Metapop Manager.
        2. Check consistency of the number of dispersers (emigrants) *NMji* from this population
        3. If a catastrophe is spread by dispersers, and it occurred in this population (*p*) in *this* time step (*t*), then calculate the "additional" probability of the catastrophe for all other populations *i* to which there are dispersers from this population (i.e., *NMji*>0 for any stage *j*). This "additional" probability will be used in the next timestep, *t*+1 (see step 2.2.2)
        4. Record total number of emigrants from each stage of this population, and update the number of immigrants to each stage of each target population.
      1. Migrate all: For each population *p*, update the number in each stage *j* by subtracting the number of emigrants (*Epj*) and adding the number of immigrants (*I pj*):

*Npj*’ = *Npj Epj* + *Ipj*

* + - 1. For each population *p*: If a catastrophe is spread by a vector, and it occurred in this population (*p*) in *this* time step (*t*), then calculate the "additional" probability of thecatastrophe for all other populations *i*, based on the distance between *p* and *i*, and on the probability-distance function. This "additional" probability will be used in the next timestep, *t*+1 (see step 2.2.2)
    1. ***Population management (after dispersal)***
       1. Apply population management actions (in the same order as in the Population management dialog box) for which When=After dispersal, if they are not ignored. Each management action may modify the number of individuals; the modified values are then used in the rest of the management actions.
       2. If in metamodel mode, and if required, record total or stage-specific numbers of animals harvested or supplemented for use by Metapop Manager.
    2. Calculate summary statistics for this time step. If a population abundance is below its local threshold, and the parameter **When abundance is below local threshold** is specified as "assume dead" (in the **Stochasticity** dialog box), then set all stage abundances in that population to zero.
    3. **If in metamodel mode, and if required, record stage-specific abundances and other state variables that have not yet been recorded for use by Metapop Manager. See detailed algorithm below.**
    4. IF in metamodel mode, export all information from this time step to Metapop manager/Metamodel manager.
  1. Calculate summary statistics for this replication

**Calculate summary statistics for the simulation; store results**

**Technical overview of "Metapop Manager" operations:**

1. User opens (minimalistic) GUI and specifies:
   1. One or two component MP files (e.g., predator.mp, prey.mp)
   2. (For now) DLL for implementing the specific linkage (e.g., predPrey\_test.dll)
   3. (for each MP file) Variables to store in order to implement metamodel linkage (select from among a large number of possibilities: see description of MP StateVars)
   4. (for each MP file) Variable to be modified (one of four choices: abundance, carrying capacity, vital rates and dispersal; see Metapop modifier description)
   5. "Access privileges" defining linkages among subpopulations in the two Metapop models.
2. Load and initialize one or two instances of Metapop, as specified by user.
3. For each Metapop instance, initialize "MP StateVars" storage structure (storing "population" and "global" state variables; see Box 2): Only store those state variables that are used in functions or that are needed for text output.
4. Initialize and populate "access privileges", which is a single variable (unlike the above, which are defined for each Metapop instance) that defines which populations in the Species 1 metapopulation are accessible by the Species 2 metapopulation and vice versa.
   1. accessPrivileges: Boolean array defining which populations are accessible by Species 2 (Boolean array of dimension "nPopulations" [MP File 2], "nPopulations" [MP File 1] )
   2. [[note: ultimately we may be able to use RAMAS GIS spatial module or other spatial software to define access privileges and/or relative strength of interaction among populations of different species]]
5. If Metapop instances have not yet been run: Load "Year 0" data for all used state variables from the component Metapop instances (e.g., initial abundances and transition matrices etc). Disperser matrix and management actions should remain at 0.
6. If Metapop instances have been run: Load data for all required state variables from the component Metapop instances (e.g., initial abundances and transition matrices etc).
7. If possible, run some checks:
   1. Throw an error if carrying capacity is modified by both MP manager and by Metapop (temporal change in abundance or catastrophe that affects abundance).
   2. Throw an error if vital rates are modified by both MP manager and by Metapop (vital rate change files or vital rate catastrophe)
   3. Throw an error if dispersal rates are modified by both MP manager and by Metapop
   4. Throw an error if ceiling, scramble, or user-defined DD are specified and vital rate modifier is in effect.
8. Send all MP\_StateVars objects to DLL (along with "access privileges"). The DLL will then implement changes in vital rates, carrying capacity, abundance, or dispersal rates as a function of these input state variables. [[ultimately all functional linkages will hopefully be performed in Metamodel Manager, but for the time being, we might just create a DLL that performs the metamodel linkages for a couple test cases (implements user-defined functions)]]
9. Read from DLL (ultimately from Metamodel Manager?):
   1. the new target values for the modified parameter:
      1. for an abundance modifier, read in the new stage-specific abundances for each population (popAbundSt)
      2. for a K modifier, read in the new population-specific K values (popK)
      3. for a vital rate modifier, read in the new baseline stage matrix for each population (popStMat)
      4. for a dispersal modifier, read in the new relative dispersal rates (relDispersal)
10. For each Metapop instance, convert modified parameters to "Metapop modifier" object that can be read and understood by Metapop (see Box 1):
11. Initiate next time step for user-specified MP files.
12. Update "MP StateVars" storage variables
13. Pause Metapop instances at the beginning of the next year, awaiting completion of all component models.
14. Run steps 8-13 until all Metapop files have completed.

Note: the DLL (ultimately Metamodel Manager) will allow users to specify which results to store (results will be stored in a text file (actually I think MMM will automatically store all relevant state variables in a master text file).