Song Evolution Model Walk-Through*

R Version: 1.0

C. Robinson

May 2019

*Published in conjunction with Robinson and Creanza 2019 (Pending)

Contents

1 Overview of the Model

1.1 Description

The Song Evolution Model initializes a population of male birds with several song-learning traits and one song, which exist in a matrix with discrete boundaries. Each time step, birds can (A) die, or survive to potentially (B) learn and/or (C) father chicks.

A) Birds are first chosen to die randomly. However, birds with greater learning age thresholds and can learn for longer periods or time have a greater chance of being chosen to die than those with smaller learning age thresholds. The amount this chance is increased is dependent on both the size of the learning age threshold and the value of the learning penalty parameter. The number of birds chosen to die is either 1) a set percentage or 2) a proportion of the birds in each age group depending on the maximum age of the population.

B) If oblique learning is enabled, any remaining birds that are younger than their learning age threshold have the capacity to learn. Learners chose a tutor male from the population and learn all or part of the tutor's song using one of four strategies: 1) Add, 2) AddForget, 3) Forget, or 4) Consensus.

C) After the learning step, vacancies left by deceased birds are filled by generating chicks fathered by the remaining birds. Birds are chosen as fathers based on sexual selection pressures which include: 1) preference for larger repertoire sizes, 2) preference for repertoires that match a song template, and/or 3) uniform preference for factors not coded into the model, such as ornate plumage or dance (called "noise preference" in the model). Fathers may also be preferentially chosen based on their location in the matrix. Birds chosen to be fathers produce one or more chicks, which inherit their father's song-learning traits with a parameterized amount of random noise. Chicks start with an empty song template and, if enabled, vertically learn their father's song with an accuracy based on their song-learning ability.

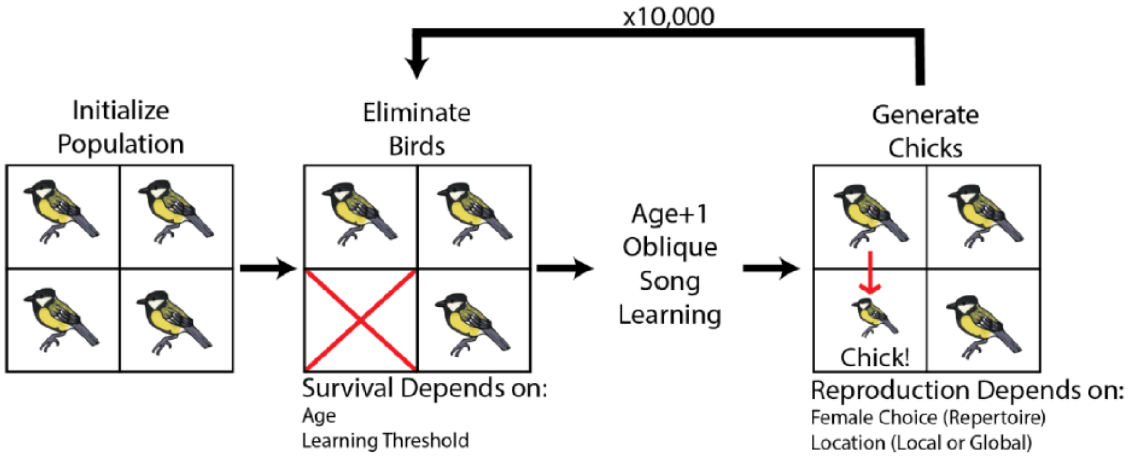


Figure 1: Schematic of the model.

1.2 Conceptualizing the Bird Matrix

Although a matrix is never literally created during the simulation, the code operates as though birds are in a matrix with discrete edges. **Figure 2** shows the assumed positions of birds [0,99] if the matrix dimensions are 10 rows by 10 columns. Thus, if birds are considered to be local when they are one "step" or territory away from a target bird, corners will have three local territories (illustrated by cell 0 in panel A), edges will have 5 local territories (illustrated by cell 6), and center pieces will have 8 local territories(illustrated by cell 67) (blue in **Figure 2A**). The combination of the blue and red territories in **Figure 2A** shows the local birds if local territories can be two steps away. When there is more than one dialect in the simulation, dialect regions are constructed such that each region contains an equal number of birds, and the region itself is as square as possible (**Figure 2B** shows 4 dialects). This means that the number of dialects in a matrix must be a factor of the total number of territories. For the 10x10 matrix example, that means that 1, 2, 4, 5, 10, 20, 25, 50, and 100 are acceptable numbers of dialects. However, adding more than 5 dialects to such a small matrix is likely to negate the purpose of adding dialects.

A

0	10	20	30	40	50	60	70	80	90
1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
5	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99

B

0	10	20	30	40	50	60	70	80	90
1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
5	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99

Figure 2: Matrix (A) shows which territories are considered local when Par.Steps=1 (blue) or 2 (blue and red) for a corner (0), edge (6) and center piece (67). Matrix(B) shows how 4 dialects would be positioned in this matrix. Dialect 1=white, 2=red, 3=blue, 4=purple.

1.3 Implementing a Type II Survival Curve

1.3.1 Overview of Type II Survival Curves

Chicks often have a low chance of surviving their first year (e.g. [1], [2], [3], and [4]). However, after the first year, mortality in birds is best modeled by a type II survival curve [5], wherein the mortality rate is not dependant on an organism's age. Instead, a constant proportion of individuals are lost every year. On a semi-log plot, where the y axis has been log-transformed, a Type II survival curve appears to be a straight line (Fig 3A).

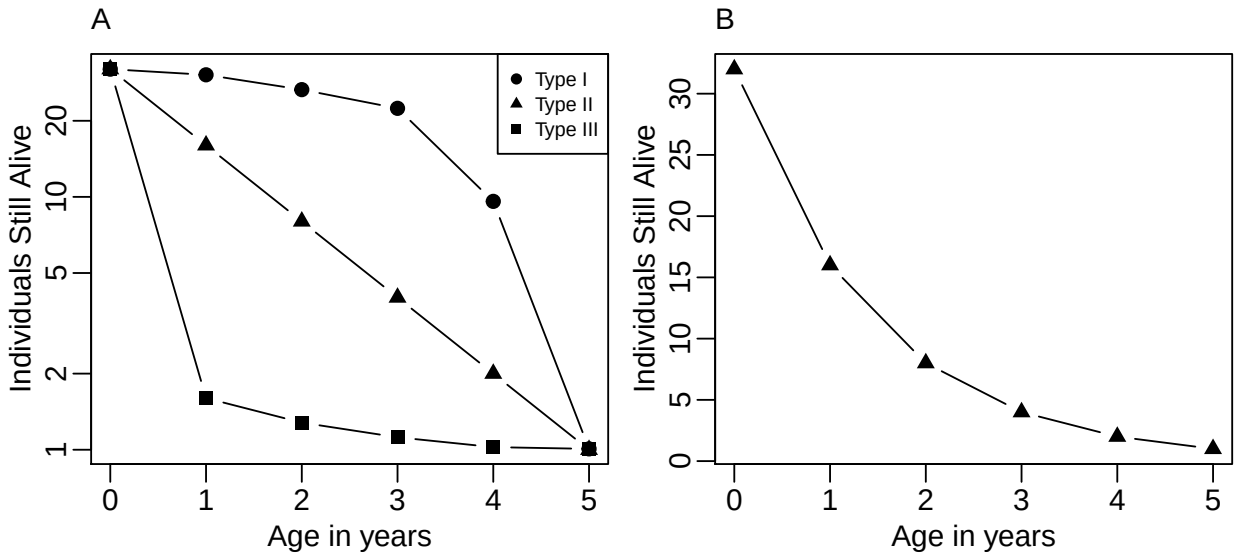


Figure 3: Survival Curves. A) Type I survival curves model organisms with a low mortality rate when young with a high mortality when older. Type III survival curves fit organisms with a high mortality rate when young, and a low mortality rate when older. Type II survival curves show no age dependence; a constant proportion of individuals die every year. In semi-log space, Type II curves appear linear. In linear space, Type II curves are not linear and approach, but never reach, 0.

Despite the appearance of linearity on a semi-log plot **Figure 3A**, on normal axes, a type II survival curve is indeed a curve, as can been seen in linear space **Figure 3B**. This curve fits an exponential equation of the form:

$$y = n_1 P^x \tag{1}$$

Where:

$$n_1 = \text{the number of age 1 adult males in a generation} \tag{1a}$$

$$P = \text{the proportion of adult males lost at each time step} \tag{1b}$$

and

$$0 < P < 1 \tag{1c}$$

1.3.2 Implementation and Model Assumptions

To mimic the mortality rates seen in real birds, our model allows a user-defined proportion of male chicks to survive to year 1 and then calculates mortality at subsequent ages using a Type II survival curve. This affects two components of our model. It is used to calculate what proportion of males in a given age step will survive to the next age step. It is also used to determine the initial distribution of males present in each age group at the start of the simulation. It is used at this initialization step, because if the initial population of birds is inappropriately distributed throughout the age classes, this leads to different numbers of birds dying in each time step. This, in turn, creates chaotic generation sizes. This unwanted chaos can be avoided if the ages of the initial generation of birds are distributed as defined by (1), which would be the case if the population had reached a mortality equilibrium.
The design of our model imposes the following constraints:

1. There is a set number of territories as paramaterized by the user. Each territory is claimed by a single male bird. There are no males without territories. Thus, we know total population of male birds (N).

2. There is maximum lifespan for all birds, which is parameterized by the user (m). No birds are permitted to survive past age m.

3. Because Equation (1) never reaches zero, a death threshold must be defined (t) at which point the probability of survival is considered to be 0.

4. In adult birds, the proportion of birds lost at a given time step (P) is equal to the proportion lost at every other time step in a generation except from age group 0 to 1.

5. The proportion of chicks lost in the 0 to 1 step is parameterized by the user (P_c)

If n_0 is known, which is the case for all generations after the initial time step, we can also calculate n_1 ($C * n_0$). Subsequent survival (n_2 to $n_m + 1$) is calculated using (1) ($|_0^{m+1}$). This requires knowing P, which can be calculated in this case, because at age $n_{m+1} = t$. However, we do not know the constants n_0 or P for generating the initial age distribution of adults. Thus, additional logic is required to solve the equation in that case.

1.3.3 Finding P and n0 for the Initial Age Distribution

Because of the way we have defined our model, we know that the total number of males in the population (N) must be equal to the number of birds in each age step (n_i) summed:

$$N = \sum_{i=0}^{m+1} n_i \tag{2}$$

For an example case, when the maximum age for the population is 2 years ($m = 2$):

$$N = n_0 + n_1 + n_2 + n_3 \tag{3}$$

Where n_0 is the number of newly generated chicks that survive, and n_1 and n_2 are the number of adults from previous generations remaining at ages 1 and 2 respectively. n_3 is the number of birds of age 3, which logically would be 0, because no birds live to age 3 when $m = 2$. However, type-II survival curves can only approach 0, so n_3 is a non-zero value that will be equivalent to the death threshold t. We know that n_1 is some proportion of n_0, n_2 is some proportion of n_1, and n_3 is some proportion of n_2. We can use the property of multiplicative identity to rewrite this equation as:

$$N = n_0 + \frac{n_1}{n_0}n_0 + \frac{n_2}{n_1}n_1 + \frac{n_3}{n_2}n_2 \tag{4}$$

To simplify, these proportions can be rewritten as p_{i-1}:

$$N = n_0 + p_1 n_0 + p_2 n_1 + p_3 n_2 \tag{5}$$

$n_i = p_i n_0$, so the equation cane be rewritten such that only n_0 remains:

$$N = n_0 + p_1 n_0 + p_1 p_2 n_0 + p_1 p_2 p_3 n_0 \tag{6}$$

Assume $p_1 = P_c$, the proportion of chicks that survive, which is a known value. All other proportions are equal, because one of the defining traits of a Type II survival curve is that a constant proportion of individuals die at each time step. To take advantage of this, we will assume that the population has stabilized its generation size, such that n_0 was equivalent for each generation. Therefore, p_2 and p_3 are equivalent (P). The equation can now be rewritten as:

$$N = n_0 + P_c n_0 + P_c P n_0 + P_c P^2 n_0 \tag{7}$$

We can take this example and generalize it to:

$$N = n_0 + n_0 P_c \sum_{i=0}^{m} P^i \tag{8}$$

At this point, n_0 and P are still unknowns. However, as explained in Section 1.3.2:

$$t = n_{m+1} \tag{9}$$

We showed in equations (3)-(7) that we can know the number of individuals in any age class using the following:

$$n_i = n_0 P_c P^{i-1} \tag{10}$$

This logic combined with (9) allows us to create the equation:

$$t = P_c P^m n_0 \tag{11}$$

Solving (11) for n_0 allows us to rewrite (8) as:

$$N = \frac{t}{P_c P^m} + \frac{t}{P_c P^m} P_c \sum_{i=0}^{m} P^i \tag{12}$$

This can be simplified and rewritten as:

$$0 = (t - N) + t\frac{1}{P} + t\frac{1}{P}^2 + ... + (t + \frac{t}{P_c})\frac{1}{P}^m \tag{13}$$

Thus, the equation can be solved for $1/P$, to obtain a real, positive root that can be converted to P. Thus, only unknown variable in (11) is now n_0, so this equation can be solved when given user defined t, m, P_c and N.

2 Using the Code

Once the library in called, the code is ready to run. Although all simulation types can be called directly, there are also stored in the SEMSimulation wrapper.

2.1 Simulation Functions

2.1.1 SEMSimulation

A wrapper that runs the other simulation types. It times the simulation and saves and/or returns data to a timestamped or user-named folder. It has one required argument, 5 named, optional arguments, and takes ellipses arguments for specific simulations (see specific simulations for more information):

1. P (Required): a parameter data structure
2. type (Default='Basic'): what type of simulation to run. Can be Basic, Light, Insult, or Invasion.
3. folderName (Default=NA): if save=TRUE, the name of the folder to save data to. If left as NA, the folder name will be a timestamp
4. save (Default=TRUE): whether to create .csv files of the results
5. return (Default=FALSE): whether to return the simulation results
6. verbose (Default=TRUE): whether to print timing information

2.1.2 BasicSimulation

Runs a simulation where the values for all males are saved every [freq] time steps.

1. P (Required): a parameter data structure
2. freq (Default=1): how often to save data.

2.1.3 LightSimulation

Runs a simulation where the population average is saved every [freq] time steps.

1. P (Required): a parameter data structure
2. freq (Default=200): how often to save data.

2.1.4 InvasionSimulation

Runs an invasion simulation where the time to conversion and final average value of the trait is returned. [numInvader] invaders have their [trait] changed to [invaderStat] and their age reset to 1 time step [when]. Simulation ends when invaders are expelled, take over, or P$nSim time steps have passed.

1. P (Required): a parameter data structure to use at the start of the simulation
2. numInvader: the number of invaders to add
3. trait: the stat to change (LrnThsh, Acc, ChanceInv, or ChanceFor)
4. invaderStat: the stat to change (LrnThsh, Acc, ChanceInv, or ChanceFor)
5. when: the time step at which to introduce the invaders

2.1.5 InsultSimulation

Runs a simulation where the only average values are saved every [freq] time step. Parameters change from P to insultP at time step [when].

1. P (Required): a parameter data structure to use at the start of the simulation
2. insultP (Required): a parameter data structure to switch to part way through the simulation
3. when (Required): the timestep at which to switch from P to insultP
4. freq (Default=200): how often to save data.

2.2 .SEMP Files

Song **E**volution **M**odel **P**aramterset (.SEMP) files are text files in the format of [parameter]=[value]. Both the R and C# versions of the model can generate and read .SEMPS that are cross-language compatible. The parameters in .SEMPs are:

R=Rows in the bird matrix
int, [3,∞]; Default=20

C=Columns in the bird matrix
int, [3, ∞]; Default=20

numBirds=NumBirds, number of birds in the matrix
int, R*C

Steps=Steps, number of territories away that are considered local
int [1,max(R,C)-1]; Default=1

RSize0=InitialSyllableRepertoireSize, the number of syllables a bird has a 90% chance to know at the start of the simulation
int, [1,MaxRSize]; Default=5

PerROh=PercentSyllableOverhang, this fraction*RSize0 is the number of syllables that a bird has a 10% and 1% chance to know at the start of the simulation
float, [0,(MaxRsize-RSize0)/(2*RSize0)]; Default=0.2

MaxRSize=MaxSyllableRepertoireSize, the limit to the number of syllables an individual bird in the population can learn; Default=500

Acc0=InitialAccuracy, the mean starting value for the population accuracy
float, [MinAcc,MaxAcc]; Default=0.7

IAccN=InheritedAccuracyNoise, the range around the mode (initial accuracy or the father's accuracy) that can be sampled to pull an initial male's accuracy or a chick's inherited accuracy, respectively
float, [0,(MaxAcc-MinAcc)/2]; Default=0.15

MinAcc=MinimumAccuracy, the absolute minimum value for accuracy
float, [0,MaxAcc); Default=0

MaxAcc=MaximumAccuracy, the absolute maximum value for accuracy
float, (MinAcc,1]; Default=1

MAge=MaxAge, the maximum age
int,[1,∞]; Default=20

LrnThrsh0=InitialLearningThreshold, the mean starting value for the population learning age
float, [MinLrn,MaxLrn]; Default=2

ILrnN=InheritedLearningThresholdNoise, the range around the mode (initial learning threshold or the father's learning threshold) that can be sampled to pull an initial male's learning threshold or a chick's inherited learning threshold, respectively
float, [0,(MaxLrn-MinLrn)/2]; Default=0.25

MinLrn=MinimumLearningThreshold, the absolute minimum value for the learning age threshold
float, [0,MaxLrn); Default=0

MaxLrn=MaximumLearningThreshold, the absolute maximum value for the learning age threshold
float, (MinLrn,MaxAge]; Default=MaxAge

CtI0=InitialChancetoInvent, the mean starting value for the population chance to invent; It is not the final percentage
float, [MinCtI,MaxCtI]; Default=0.1

ICtIN=InheritedChancetoInventNoise, the range around the mode (initial chance to invent or the father's chance to invent) that can be sampled to pull an initial male's chance to invent or a chick's inherited chance to invent, respectively
float, [0,(MaxCtI-MinCtI)/2]; Default=0

MinCtI=MinumumChancetoInvent, the absolute minimum value for chance to invent
float, [0,MaxCtI); Default=0

MaxCtI=MaximumChancetoInvent, the absolute maximum value for chance to invent
float, (MinCtI,1]; Default=1

CtF0=InitialChancetoForget, the mean starting value for the population chance to forget
float, [0,MaxCtF/2]; Default=0.2

ICtFN=InheritedChancetoForgetNoise, the range around the mode (initial chance to forget or the father's chance to forget) that can be sampled to pull an initial male's chance to forget or a chick's inherited chance to forget, respectively
float, [0,(MaxCtF-MinCtF)/2]; Default=0

MinFtI=MinumumChancetoForget, the absolute minimum value for chance to forget
float, [0,MaxCtF); Default=0

MaxFtI=MaximumChancetoForget, the absolute maximum value for chance to forget
float, (MinCtF,1]; Default=1

LisThrsh=ListeningThreshold, if less than 1, the percentage of a song that a learner hears. .999 is treated as 100%. If an integer greater than 1, the number of syllables a learner hears
float $[0,.999] \cup$ ints $[1,\infty]$; Default=7

FLisThrsh=FatherListeningThreshold, if less than 1, the percentage of a song that a son hears. 0.999 is treated as 100%. If an integer greater than 1, the number of syllables a son hears
float [0,0.999]∪ints [1,∞]; Default=0.999

MinLrnSyl=MinLearnedSyls, the minimum number of syllables a male learns when a percentage strategy is employed for one or both of the listening thresholds
ints [0,∞]; Default=7

EnSuc=EncounterSucess, the chance that a learner meets a tutor
float [0,1]; Default=0.95

Lpen=LearningPenalty, the magnitude of the survival disadvantage placed on birds with learning thresholds greater than 1
float, [0,∞]; Default=0.75

DStrat=DeathStrategy, if true, death is based on the dynamics of a population with a type II survival curve, if false, every timestep PDead*numBirds birds die
bool ; Default=true

PDead=PrcntRandomDeath, if DStrat=false, the percentage of birds that die each time step.
float[0.01,0.9]; Default=0.1

DeadThrsh=DeathThreshold, when the number of birds in a generation is less than or equal to the Death Threshold, they are all considered dead; a key component for calculating the type II survival curve. This calculation may prevent birds from reaching the MaxAge if the threshold is set below 1.
float[1,∞]; Default=1

Pc=ChickSurvival, the percentage of chicks that survive to age 1
float [.1,1]; Default=0.3

InitProp=InitialSurvival, the proportion of adult birds that will survive to the next age. It is calculated internally based on the MaxAge, ChickSurvival, and DeathThreshold. Do not modify this in .SEMPs unless you want the birds to start in a non-equilibrium state, with nonuniform generations sizes.
float, **See Calculating a Type II Survival Curve**

ScopeB=LocalBreeding, whether mates are chosen from a local pool, or among all males
bool; Default=false

ScopeT=LocalTutor, whether tutors are chosen from a local pool, or among all males
bool; Default=false

Consen=Consensus, whether birds use the consensus strategy in learning. Automatically sets add=true and forget=true, but changing either one to false manually

will be accepted by the code.
bool

ConsenS=ConsensusStrategy, which variant of the consensus strategy to use. Percentage means there is a linear association between the commonality of a syllable and whether a learner attempts to learn it. AllNone means a learner only learns syllables that all tutors sang. Conform means that a learner attempts to learn a syllable based on a sigmoidal relationship.
Default=Conform

Add=Add, whether birds use the add strategy in learning
bool

Forget=Forget, whether birds use the forget strategy in learning
bool

ConNoTut=NumTutorConsensusStrategy, if using the consensus strategy, the number of tutors to find
int [1,numBirds-1]; Default=3

OvrLrn=OverLearn, whether chicks use the overlearn strategy
bool; Default=false

OLNoTut= numTutorOverLearn if using the overlearn strategy, the number of tutors to find
int [1,numBirds-1]; Default=3

Obliq=ObliqueLearning, whether birds engage in oblique learning
bool; Default=true

Vert=VerticalLearning, whether birds engage in vertical learning
bool; Default=true

VertLrnCut=VerticalLearningCutOff, birds with a learning threshold below this value cannot learn vertically
float [0,MaxAge]; Default=0.25

RepPref=RepertoireSizePreference, the percentage of female choice devoted to finding males with the largest repertoire sizes
float [0,1-(MatPref+NoisePref)]; Default=1

LogScl=TRUE

MatPref= MatchPreference, the percentage of female choice devoted to finding males with repertoires that best match her template
float [0,1-(RepPref+NoisePref)]; Default=0

NoisePref=NoisePreference, the percentage of female choice devoted to finding males who are superior in other traits (e.g. better dancing or territory quality)
float [0,1-(RepPref+MatPref)]; Default=0

UniMat=MatchUniform, whether all females in a dialect region have the same (true), or if there is noise between templates (false)
bool; Default=true

MScl=MatchScale, not currently implemented. This will hopefully allow for scaling the matches in a non-linear way in the future.

Dial=NumDialects, the number of different dialects in the matrix. Different dialects have no overlap in the syllable space
int [0,NumBirds], must be a factor of NumBirds, Default=1

MDial=MaleDialects, whether males in the starting generation have dialects matching females. If "None," male songs are generated in the first slots of the syllable space, putting them in the same syllable space as dialect 1. If "Same," a male's song is a duplicate his female's song template. If "Similar," a male's song is generated within the same syllable space as his female's song template, but a new song is generated, so they are not duplicates. Note that is there is only one dialect, then "None" and "Similar" produce the same effect.
string ["None","Same","Similar]; Default="None"

FEvo=FemaleEvolution, whether females can change their templates over time. If true, when a male dies, his female also dies. When a male chick is created, a new female with a template matching different father is created.
bool; Default=false

ChoMate=ChooseMate, allows females to pick a new mate based on their preferences each time step.
bool; Default=FALSE

SMat=SaveMatch, whether to save matching data. If null, the code decides whether to save based on other parameters.
nullable bool; Default=null

SAcc=SaveAccuracy, whether to save accuracy data. If null, the code decides whether to save based on other parameters.
nullable bool; Default=null

SLrn=SaveLearningThreshold, whether to save learning age threshold data. If null, the code decides whether to save based on other parameters.
nullable bool; Default=null

SCtI=SaveChancetoInvent, whether to save chance to invent data. If null, the code decides whether to save based on other parameters.
nullable bool; Default=null

SCtF=SaveChancetoForget, whether to save chance to forget data. If null, the code decides whether to save based on other parameters.
nullable bool; Default=null

SNam=SaveNames, whether to save name and father name GUID data.
bool; Default=false

SAge=SaveAge, whether to save age data.
bool; Default=false

SMSng=SaveMSong, whether to save male song data.
bool; Default=false

SFSng=SaveFSong, whether to save female song data.
bool; Default=false

SimStep=SimStep, the current step in the simulation. Used for error handling and reset to 1 when the parameters are saved.
int [1,∞]; Default=1

nSim=NumSim, the number of timestpes to run in a simulation.
int[1,∞]; Default=1000

Seed=Seed, for reproducibility. 0 is not an option, because NA is converted to 0 in this code due to strong typing, and thus 0 is considered to mean that no seed was passed.
int [1,∞] or NA; Default=NA

2.3 Vignette

1. If your computer is not already set up to use R, do so:
   https://www.r-project.org/

2. We also recommend that you install an IDE like RStudio or Visual Studio Code:
   https://www.rstudio.com/products/rstudio/download/
   or
   https://code.visualstudio.com/download

3. Install the SEM package from GitHub (you only need to do this once):

```
#install.packages('devtools') #if you don't already have devtools
library('devtools')
install_github('CreanzaLab/SongEvolutionModel/R-Package')
```

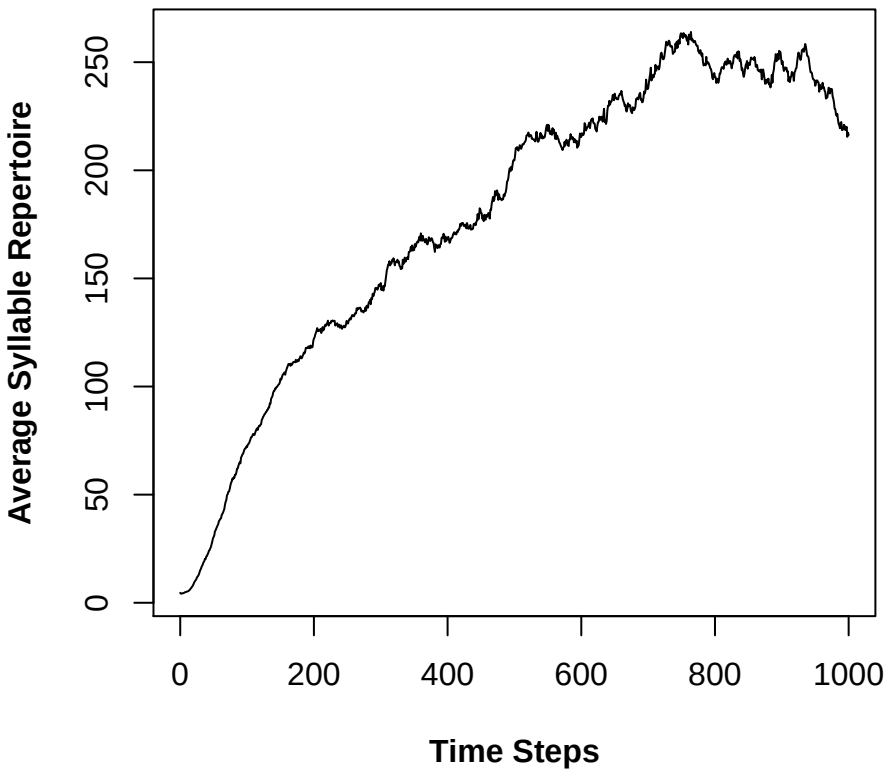
4. Load the library (you need to run this every time you start a new R session):

```
library('SEM')
```

5. Create parameters, run the simulation, and plot the data (will take approximate one minute):

```
P <- DefineParameters(Seed = 49)
SimResults <- SEMSimulation(P, 'Light', save = FALSE, return = TRUE, fr
plot(0:1000, SimResults$SylRep, type="l", xlab="Time Steps",
     ylab="Average Syllable Repertoire", font.lab=2)
```

   It should produce a plot that looks like this:


Average Syllable Repertoire
0
50
100
150
200
250
0
200
400
600
800
1000
Time Steps

3 Code Documentation

Please see the R manual generated via royxgen2!

4 Acknowledgements

Ray Yu wrote the seed code for the first version of the model in MATLAB. Jordan Singer helped develop the strategy and logic for implementing the type II survival curve. We used the `UUIDgenerate` function from package uuid (https://CRAN.R-project.org/package=uuid) and the `rpert()` function from package mc2d [7] (https://CRAN.R-project.org/package=mc2d).

5 References

[1] Naef-Daenze, B., Widmer, F., and Nube, M. (2001) .Differential post-fledging survival of great and coal tits in relation to their condition and fledging date. *Journal of Animal Ecology 70:*(5), 730-738. DOI: 10.1046/j.0021-8790.2001.00533.x

[2] Loison, A., Saether, B., Jerstad, K., and Rostad, O.W.. (2010). Disentangling the sources of variation in the survival of the European dipper. *Journal of Applied Statistics 1:*(4), 289-304. DOI: 10.1080/02664760120108665

[3] Stenzel, L.E., Page, G.W., Warriner, J.C., Warriner, J.S., George, D.E., Eyster, C.R., Ramer, B.A., and Neuman, K.K. (2007). PRBO Conservation Science, 3820 Cypress Drive Suite 11, Petaluma, California 94954, USA Kristina K. NeumanSurvival and natal dispersal of juvenile snowy plovers (*charadrius alexandrines*) in central coastal California. *The Auk 124:*(3), 1023-1036. DOI: 10.1642/0004-8038(2007)124[1023:SANDOJ]2.0.CO;2

[4] Tyler, G.A., and Green, R.E. (2003). Effects of weather on the survival and growth of corncrake *Crex crex* chicks. *Ibis 146:*(1), 69-76. DOI: 10.1111/j.1474-919X.2004.00225.x

[5] Begon, M., Harper, J.L., and Townsend, C.R. (1996). *Ecology: Individuals, populations and communities*, 3rd ed. Oxford:Blackwell Science Ltd.

[6] Wong, C.K. and Easton, M.C. (1980). An efficient method for weighted sampling without replacement. *SIAM Journal on Computing, 9:*(1), 111-113. DOI: 10.1137/0209009

[7] Pouillot, R. and Delignette-Muller, M.-L. (2010). Evaluating variability and uncertainty in microbial quantitative risk assessment using two R packages. *International Journal of Food Microbiology 142:*(3), 330-40. DOI: 10.1016/j.ijfoodmicro.2010.07.011