SEM

March 22, 2019

Title Song Evolution Model

Version 0.0.0.9000

Description Using an agent-based model, this package simulates birdsong evolution in response to various selection pressures. It allows the implementation of different song-learning styles, and fitness benefits depending on repertoire size or match to a female template. Users can also add a fitness cost on longer learning.

Depends R (>= 3.5.1)

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports mc2d, numbers, uuid

R topics documented:

AgeDeath 2
AssignFemale 3
BasicSimulation 3
BirthDeathCycle 4
CalcFractional 4
CalculateAllGen 4
CalculateProportion 5
CheckBool 5
CheckEncouter 5
CheckInsultPs 6
CheckMinMaxInt 6
CheckP 7
CheckTrait 7
ChooseFathers 8
ChooseTutors 8
ClusterCalc 9
ClusterPlot 9
ConsensusLearning 10
CreateFemaleSongs 10
DefineParameters 11
DropSyllables 14
EstablishDialects 14

FamilyTreePlot 14
FemaleEvolve 15
FinalDirections 15
GenerateAdultBirds 15
GenerateChicks 16
GenerateFounderMales 16
GenerateNovelSong 17
GetAgeGroup 17
GetAgeRates 17
GetLearners 18
GetMaxMat 18
GetProbability 19
InitAgeDistribution 19
InsultSimulation 19
LearningProcess 20
LearningThrshPenalty 20
LightSimulation 21
ListeningTest 21
LocalSearch 22
NextStepDirections 22
ObliqueLearning 23
OneStepDirections 23
OneTutorLearning 23
OverLearn 24
QuickClusterPlot 24
QuickSEMPlot 25
RandomDeath 26
ReloadParam 26
SaveParam 26
SEMSimulation 27
SongPlot 27
TerritoryHeatMap 28
TestLearningThreshold 28
TestMatch 29
TestRequirement 29
TraitPlot 30
UpdateProbabilities 30
UpdateSongTraits 31
VerticalSongLearning 31

Index **32**

`AgeDeath` *Age Death*

Description

Males are chosen to die based on their age using a type II survival curve.

Usage

```
AgeDeath(P, population)
```

Arguments

`P`	a list of parameters
`population`	the population of birds

`AssignFemale` *Assign Female*

Description

Assigns females to males based on template matching. Females are randomly chosen to pick a male from the population. Males that better match her template are more likely to be chosen. Assumes that there are as many females as there are males, and all birds are paired in the end.

Usage

```
AssignFemale(P, maleSong, femaleSong)
```

Arguments

`P`	a list of parameters
`maleSong`	a matrix of syllable vectors
`femaleSong`	a matrix of syllable vectors

`BasicSimulation` *Basic Simulation*

Description

Runs a simulation where individual values are saved for every timestep. No parameters change during the simulation.

Usage

```
BasicSimulation(P, freq = 1, saveInfo)
```

Arguments

`P`	a list of parameters
`freq`	how often to sample data from the simulation
`saveInfo`	a matrix of saving data made in the simulation wrapper

See Also

Other Sim Functions: `CheckInsultPs`, `InsultSimulation`, `LightSimulation`, `SEMSimulation`

`BirthDeathCycle` *Birth Death Cycle*

Description

A wrapper that allows birds to die, undergo oblique learning, be born, and undergo vertical learning (in that order).

Usage

```
BirthDeathCycle(P, population)
```

Arguments

`P`	a list of parameters
`population`	the population of birds

`CalcFractional` *Calculate Consensus Fraction*

Description

Calculates the probability that a bird will learn a syllable depending on how many tutors it was heard from.

Usage

```
CalcFractional(P, consensusSong)
```

Arguments

`P`	a list of parameters
`consensusSong`	vector of the number of tutors that sang each syllable.

`CalculateAllGen` *Calculate All Generations*

Description

Calculates the proportion of the population that is in each generation when the simulation starts.

Usage

```
CalculateAllGen(pa, pc, t, mAge)
```

Arguments

`pa`	the proportion of adults that survive to the next age
`pc`	the proportion of chicks that survive
`t`	the death threshold
`mAge`	the max age of the population

`CalculateProportion` *Calculate Adult Survival Proportion*

Description

Calculates the proportion of adults that survive from one timestep to the next.

Usage

```
CalculateProportion(n = 400, t = 1, pc = 0.3, mAge = 20)
```

Arguments

`n`	the population size
`t`	the death threshold
`pc`	the proportion of chicks that survive
`mAge`	the max age of the population

`CheckBool` *Check Boolean*

Description

Checks whether a value that should be Boolean is, allowing NA if required.

Usage

```
CheckBool(value, valueName, NAer = FALSE)
```

Arguments

`value`	user parameter to check
`valueName`	name of teh checked parameter
`NAer`	whther the value can be NA

`CheckEncouter` *Check Encouter*

Description

Tests whether a learner met tutors

Usage

```
CheckEncouter(P, learners)
```

Arguments

`P`	a list of parameters
`learners`	indicies of males that are alive and young enough to learn

CheckInsultPs *Insult Simulation*

Description

Tests whether the initial and insult parameters are compatable.

Usage

```
CheckInsultPs(P, insultP)
```

Arguments

`P`	a list of parameters
`insultP`	a list of parameters to switch to at timestep [when]

See Also

Other Sim Functions: `BasicSimulation`, `InsultSimulation`, `LightSimulation`, `SEMSimulation`

CheckMinMaxInt *Check Min/Max/Int*

Description

Checks whether values that should be integers are and ensures that are within the correct range.

Usage

```
CheckMinMaxInt(value, valueName, min = 0, max = 1, maxed = FALSE,
  int = TRUE)
```

Arguments

`value`	the user defined parameter
`valueName`	the name of the value being checked
`min`	minumum value for a feature
`max`	maximum value value for a feature
`maxed`	whether a value has a maximum
`int`	whether a value is an integer

`CheckP` *Check Parameters*

Description

Checks whether user defined parameters fit all requirements

Usage

```
CheckP(P)
```

Arguments

`P`	a list of parameters

`CheckTrait` *Check Trait*

Description

Checks whether user defined parameters make sense for each song-learning trait

Usage

```
CheckTrait(initial, noise, min, max, name, absMax = 1)
```

Arguments

`initial`	user defined initial value
`noise`	user defined noise value
`min`	user defined min
`max`	user defined max
`name`	name fo the trait in question
`absMax`	absolute max possible for parameter, usually 1

ChooseFathers *Choose Fathers*

Description

Chooses the males who will breed. Males must be alive and know at least one syllable. They can be chosen locally or globally. Males who best fit selection preferences are the most likely to father offspring. Males can father more than one chick.

Usage

```
ChooseFathers(P, population, vacancy)
```

Arguments

P	a list of parameters
population	the population of birds
vacancy	territories that need to be filled

ChooseTutors *Choose Tutors*

Description

Randomly chooses a tutor for each learner. Tutors must be alive, not be chicks, and must know at least one syllable. Tutors can be chosen locally or globally.

Usage

```
ChooseTutors(P, population, learners, vacancy, misc = rep(0,
  length(learners)))
```

Arguments

P	a list of parameters
population	the population of birds
learners	the indicies of birds that will attempt to learn
vacancy	the indicies of birds that are dead
misc	a matrix of positions of other birds that are excluded for some reason (e.g. already for consensus tutors)

`ClusterCalc` *Cluster Calculation*

Description

Calculates the cluster score of a matrix.

Usage

```
ClusterCalc(P, matrix)
```

Arguments

`P`	a list of parameters
`matrix`	a saved trait from the Basic sims (requires individual data)

See Also

Other Cluster Plots: `ClusterPlot`, `GetMaxMat`, `QuickClusterPlot`

`ClusterPlot` *Cluster Plot*

Description

Cluster Plots are normalized such that the minimal score (a smooth gradient) is zero. The black line shows the maximal score (a checkerboard pattern) while the grey line shows the average score (no pattern). Green line plots the score of the real data over time. The function also prints the mean probability of getting the real values given the Min, Max, and Mean values for the matrix at each timestep.

Usage

```
ClusterPlot(P, trait)
```

Arguments

`P`	a list of parameters
`trait`	a saved trait from the Basic sims (requires individual data)

See Also

Other Cluster Plots: `ClusterCalc`, `GetMaxMat`, `QuickClusterPlot`

`ConsensusLearning` *Consensus Learning*

Description

Allows birds to sample multiple tutors to create a consensus song as a template to decide what to learn. Birds then learn.

Usage

```
ConsensusLearning(P, population, learners, vacancy)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`learners`	the indicies of birds that will attempt to learn
`vacancy`	the indicies of dead birds

`CreateFemaleSongs` *Create Female Songs*

Description

A wrapper that creates identical (uniform) or noisy female songs, and offsets their position in the syllable vector to create dialects if necessary.

Usage

```
CreateFemaleSongs(P)
```

Arguments

`P`	a list of parameters

`DefineParameters` *Define Parameters*

Description

Creates a parameter list and error checks chosen parameters. This is the only place where extensive error checking is done on parameters!

Usage

```
DefineParameters(Rows = 20, Cols = 20, Steps = 1,
  InitialSylRepSize = 5, PrcntSylOverhang = 0.2, MaxSylRepSize = 500,
  InitialAccuracy = 0.7, InherAccuracyNoise = 0.15,
  AccuracyLimits = c(0, 1), MaxAge = 20,
  InitialLearningThreshold = 2, InherLearningNoise = 0.25,
  LearningLimits = c(0, MaxAge), InitialChancetoInvent = 0.1,
  InherChancetoInventNoise = 0, ChancetoInventLimits = c(0, 1),
  InitialChancetoForget = 0.2, InherChancetoForgetNoise = 0,
  ChancetoForgetLimits = c(0, 1), ListeningThreshold = 7,
  EncounterSuccess = 0.95, LearningPenalty = 0.75, AgeDeath = TRUE,
  PrcntRandomDeath = 0.1, DeathThreshold = 1, ChickSurvival = 0.3,
  LocalBreed = FALSE, LocalTutor = FALSE, LearnerStrategy = "Add",
  ConsensusNoTut = 8, ConsensusStrategy = "Conform",
  OverLearn = FALSE, OverLearnNoTut = 3, VerticalLearnCutOff = 0.25,
  ObliqueLearning = TRUE, RepSizePrefer = 1, LogScale = TRUE,
  MatchPrefer = 0, UniformMatch = TRUE, MatchScale = 1,
  Dialects = 1, MaleDialects = "None", FemaleEvolve = FALSE,
  ChooseMate = FALSE, SaveMatch = NA, SaveAccuracy = NA,
  SaveLearningThreshold = NA, SaveChancetoInvent = NA,
  SaveChancetoForget = NA, SaveNames = FALSE, SaveAge = FALSE,
  SaveMaleSong = FALSE, SaveFemaleSong = FALSE, numSim = 1000,
  Seed = NA)
```

Arguments

`Rows`	the number of rows in the bird matrix
`Cols`	the number of columns in the bird matrix
`Steps`	the number of spaces away from a focal territory "local" is considered to be
`InitialSylRepSize`	the number of syllables birds have a 90% chance to know when the bird matrix is intialized
`PrcntSylOverhang`	the fraction of InitialSylRepsize that birds have a 10% and 1% chance to know when the bird matrix is intialized
`MaxSylRepSize`	the length of the syllable vector
`InitialAccuracy`	the mode value for accuracy when the bird matrix is intialized
`InherAccuracyNoise`	the area around the mode that can be sampled from for accuracy inheritance and establishing the initial distribution

`AccuracyLimits`	the absolute min and max values that accuracy can be
`MaxAge`	the maximum age of birds in the population (only in use with the type II survival curve)
`InitialLearningThreshold`	the mode value for the learning threshold when the bird matrix is intialized
`InherLearningNoise`	the area around the mode that can be sampled from for learning threshold inheritance and establishing the initial distribution
`LearningLimits`	the absolute min and max values that the learning threshold can be
`InitialChancetoInvent`	the mode value for the chance to invent when the bird matrix is intialized
`InherChancetoInventNoise`	the area around the mode that can be sampled from for chance to invent inheritance and establishing the initial distribution
`ChancetoInventLimits`	the absolute min and max values that the chance to invent can be
`InitialChancetoForget`	the mode value for the chance to forget when the bird matrix is intialized
`InherChancetoForgetNoise`	the area around the mode that can be sampled from for chance to forget inheritance and establishing the initial distribution
`ChancetoForgetLimits`	the absolute min and max values that the chance to forget can be
`ListeningThreshold`	the max absolute number or fraction of syllables a bird hears from one oblique tutor
`EncounterSuccess`	the chance that a male finds suitable tutors
`LearningPenalty`	an artitrary scale for how severly longer learning is punished
`AgeDeath`	whether to model death on a type II survival curve (TRUE) or random death (FALSE)
`PrcntRandomDeath`	the percentage of birds that die each time step when death is random
`DeathThreshold`	the numbers of birds at which a group is considered to be extinquished (you probably should not change this)
`ChickSurvival`	the proportion of chicks that survive to age 1
`LocalBreed`	whether empty territory are filled by chicks from local males (TRUE) or any male (FALSE)
`LocalTutor`	whether obliqgue learners pick tutors from from local males (TRUE) or any male (FALSE)
`LearnerStrategy`	the mode by which birds learn; can be "Add", "Forget", "AddForget", or "Consensus"
`ConsensusNoTut`	the number of tutors sampled in the consensus strategy
`ConsensusStrategy`	the method by which consensus decisions are made; can be "Conform" (chance = based on conformity bias), "AllNone" (all tutors must sing the syllable), "Percentage" (chance = percent of tutor that sang a syllable)

`OverLearn`	whether males overlearn from many tutors as chicks
`OverLearnNoTut`	the number of tutors sampled in the overlearning strategy
`VerticalLearnCutOff`	this minimum value the learning window can be while still allowing males to learn vertically.
`ObliqueLearning`	whther the population undergoes oblique learning (TRUE) or not (FALSE)
`RepSizePrefer`	the fraction of female preference dedicated to larger repertoires
`LogScale`	whether females percieve repertoire size on a natural log scale (TRUE) or not (FALSE)
`MatchPrefer`	the fraction of female preference dedicated to template matching
`UniformMatch`	whether all females have the same song template (TRUE) or variations on a template (FALSE)
`MatchScale`	an equation for how matching is perceived; not yet implemented!
`Dialects`	the number of dialects; must be a factor of the matrix size
`MaleDialects`	whether males start the simulation with dialects; can be "None" (all males are similar to dialect 1), "Similar" (male songs are in teh correct syllable space, but are not identical to female songs), "Same" (male song temapltes are identicle to their female's template)
`FemaleEvolve`	whether the female templates can evolve (TRUE) or stay static throughout teh simmulation (FALSE)
`ChooseMate`	whether females can pick their mate (TRUE) or not (FALSE)
`SaveMatch`	whether to save matches; can be NA (the program decides based on other parameters) or TRUE/FALSE
`SaveAccuracy`	whether to save the accuracy values; can be NA (the program decides based on other parameters) or TRUE/FALSE
`SaveLearningThreshold`	whether to save the learning thresholds; can be NA (the program decides based on other parameters) or TRUE/FALSE
`SaveChancetoInvent`	whether to save the chance to invent; can be NA (the program decides based on other parameters) or TRUE/FALSE
`SaveChancetoForget`	whether to save the chance to forget; can be NA (the program decides based on other parameters) or TRUE/FALSE
`SaveNames`	whether to save the UID and father's UID of the birds; can be TRUE or FALSE
`SaveAge`	whether to save the age of the birds; can be TRUE or FALSE
`SaveMaleSong`	whether to save male song templates; can be TRUE or FALSE
`SaveFemaleSong`	whether to save female song templates; can be TRUE or FALSE
`numSim`	the number of sim steps to complete
`Seed`	seed to run simulation on for reproducibility

`DropSyllables` *Drop Syllables*

Description

Tests whether a learner forgets a syllable that he knows, but that his tutor did not sing.

Usage

```
DropSyllables(chanceFor, tutorSongs, learnerSongs)
```

Arguments

`chanceFor`	the learners' chance to forget
`tutorSongs`	a matrix of tutor syllable vectors
`learnerSongs`	a matrix of learner syllable vectors

`EstablishDialects` *Establish Dialects*

Description

Modifies a matrix of syllable vectors to create dialects (regions of syllables that are separated from one another in the syllable space). Regions are defined so that each dialect space is as square as possible.

Usage

```
EstablishDialects(P, fSongs)
```

Arguments

`P`	a list of parameters
`fSongs`	a matrix of syllable vectors

`FamilyTreePlot` *Family Tree Plot*

Description

Experimental plot that shows which birds sired which offspring over time. It starts from the tips, so all but one of the original lineage will be lost after enough time steps have passed.

Usage

```
FamilyTreePlot(path, byGens = TRUE)
```

Arguments

`path`	path to a fodl with 2 matricies of bird UIDs.
`byGens`	whether to plots the y axis by generation (TRUE) or timestep (FALSE)

`FemaleEvolve` *Female Evolve*

Description

Replaces females that lived on the same territory as a dead male. New female song templates are created based on fathers that are different form the father that sired the male on her territory. Fathers must be alive and know at least one syllable. One created, the match between these new females and their males are recalculated.

Usage

```
FemaleEvolve(P, population, vacancy, fatherInd)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`vacancy`	indicies of territories where male chicks were born
`fatherInd`	the indicies of the male that fathered the resident male chicks

`FinalDirections` *Final Directions*

Description

A wrapper that calls StepOne() and EachStep() to create lists of locality data.

Usage

```
FinalDirections(P)
```

Arguments

`P`	a list of parameters

`GenerateAdultBirds` *Generate Adult Birds*

Description

Generates the features for each bird in the population at timestep 0.

Usage

```
GenerateAdultBirds(P, songs)
```

Arguments

`P`	a list of parameters
`songs`	a matrix of syllable vectors

`GenerateChicks` *Generate Chicks*

Description

Creates chicks that are similar to their fathers.

Usage

```
GenerateChicks(P, fatherInd, territorialMales, vacancy)
```

Arguments

`P`	a list of parameters
`fatherInd`	the index of the fathers
`territorialMales`	the population
`vacancy`	the index of future chicks (aligned with fathers)

`GenerateFounderMales` *Generate Founder Males*

Description

Creates the population at time step 0. Generates 1) a matrix of male syllable vectors, 2) an optional matrix of female syllable vectors, 3) a dataframe of bird features, 4) an optional locality list, and 5) structures for keeping track of bird survival if a type II survival curve is implemented.

Usage

```
GenerateFounderMales(P)
```

Arguments

`P`	a list of parameters

`GenerateNovelSong` *Generate Novel Song*

Description

Generates one or more song templates based on parameters in P; tests of inheriting [RSize0] syllables (90%), [RSize0]*[PerROh] syllables (10%), and [RSize0]*[PerROh] syllables (1%), by random sampling. Creates a numeric vector where learned syllables are 1, and unlearned syllables are 0. It then appends 0s to the end, so the vector is of length [MaxRSize].

Usage

```
GenerateNovelSong(P, numTemplates)
```

Arguments

`P`	a list of parameters
`numTemplates`	the number of song templates to create

`GetAgeGroup` *Get Age Group*

Description

Given the number of birds that should be in each generation, creates a vector of ages and scrambles them for random assignment.

Usage

```
GetAgeGroup(P, ageRates)
```

Arguments

`P`	a list of parameters
`ageRates`	the output from GetAgeRates()

`GetAgeRates` *Get Age Rates*

Description

Calculates the number of birds that should be in each generation.

Usage

```
GetAgeRates(P)
```

Arguments

`P`	a list of parameters

`GetLearners` *Get Learners*

Description

Returns the indices of males that are alive, young enough to learn, and met tutor males.

Usage

```
GetLearners(P, population, vacancy)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`vacancy`	the indicies of dead birds

`GetMaxMat` *Get Max Matrix*

Description

Creates a matrix that has the maximum score given the trait data.

Usage

```
GetMaxMat(trait, R, C)
```

Arguments

`trait`	a saved trait from the Basic sims (requires individual data)
`R`	the rows in the bird matrix
`C`	the columns in the bird matrix

See Also

Other Cluster Plots: `ClusterCalc`, `ClusterPlot`, `QuickClusterPlot`

`GetProbability` *Get Probability of Reproducing*

Description

Calculates how well a male matches female preferences for repertoires and/or matching (and/or noise, which is added uniformly to all males) to determine their probability of fathering offspring.

Usage

```
GetProbability(P, population, usableInd)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`usableInd`	males that are alive and know at least one syllable

`InitAgeDistribution` *Initial Age Distribution*

Description

Creates the age distribution of the population. Either follows a type II survival curve, or uniformly samples from 1 to the max age.

Usage

```
InitAgeDistribution(P)
```

Arguments

`P`	a list of parameters

`InsultSimulation` *Insult Simulation*

Description

Runs a simulation where the only average values are saved every [freq] timestep. Parameters change at timestep [when].

Usage

```
InsultSimulation(P, insultP, when, freq = 200, saveInfo)
```

Arguments

`P`	a list of parameters
`insultP`	a list of parameters to switch to at timestep [when]
`when`	the timestep at which to introduce the insult
`freq`	how often to sample data from the simulation
`saveInfo`	a matrix of saving data made in the simulation wrapper

See Also

Other Sim Functions: `BasicSimulation`, `CheckInsultPs`, `LightSimulation`, `SEMSimulation`

`LearningProcess` *Core Learning Process*

Description

Tests whether learners successfully acquire new syllables from their tutor(s) and modifies their song if this occurs.

Usage

```
LearningProcess(P, newSongs, tutorSyllables, accuracy, chanceInv)
```

Arguments

`P`	a list of parameters
`newSongs`	the learner's plastic song
`tutorSyllables`	the syllables the learner wants to learn from the tutor
`accuracy`	the learner's accuracy
`chanceInv`	the learners' chance to invent

`LearningThrshPenalty` *Learning Threshold Fitness Penalty*

Description

Calculates the fitness penalty for longer learning which is used as the probability that a male will be chosen to die in that timestep.

Usage

```
LearningThrshPenalty(P, lrnThsh)
```

Arguments

`P`	a list of parameters
`lrnThsh`	a vector of learning thresholds in the population

`LightSimulation` *Light Simulation*

Description

Runs a simulation where the only average values are saved every [freq] timestep. No parameters change during the simulation.

Usage

```
LightSimulation(P, freq = 200, saveInfo)
```

Arguments

`P`	a list of parameters
`freq`	how often to sample data from the simulation
`saveInfo`	a matrix of saving data made in the simulation wrapper

See Also

Other Sim Functions: `BasicSimulation`, `CheckInsultPs`, `InsultSimulation`, `SEMSimulation`

`ListeningTest` *Listening Test*

Description

Tests which syllables a learner heard from his tutor.

Usage

```
ListeningTest(P, songs)
```

Arguments

`P`	a list of parameters
`songs`	a matrix of tutor syllable vectors

`LocalSearch` *Local Search*

Description

Given a target territory, find which local males are alive. If none are alive, extend the search by one step.

Usage

```
LocalSearch(P, population, targetMale, notAvailable)
```

Arguments

`P`	a list of parameters
`population`	the population
`targetMale`	index of the territory around which local birds should be found
`notAvailable`	vectors of males that cannot be chosen

`NextStepDirections` *Next Step Directions*

Description

Extends the locality data by one step.

Usage

```
NextStepDirections(currentStep, firstStep)
```

Arguments

`currentStep`	the current list of location data
`firstStep`	the output form OneStepDirections()

`ObliqueLearning` *Oblique Learning*

Description

A wrapper that checks which birds learn, allows them to do so, then updates syllable repertoire size and match (if needed).

Usage

```
ObliqueLearning(P, population, vacancy)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`vacancy`	the indicies of dead birds

`OneStepDirections` *One Step Directions*

Description

Creates the location data for what is one "step" away form each territory.

Usage

```
OneStepDirections(R, C)
```

Arguments

`R`	rows
`C`	columns

`OneTutorLearning` *One Tutor Learning*

Description

Allows for birds to learn from one tutor (Add, Add/Forget, or Forget strategies). It is also called multiple times in the OverLearn strategy, where chicks add syllables from oblique tutors.

Usage

```
OneTutorLearning(P, population, tutors, learners)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`tutors`	the indicies of tutor paired with each learner
`learners`	the indicies of birds that will attempt to learn

`OverLearn` *Over-Learn*

Description

Allows chicks to sample from tutors other than the father to add new syllables to their repertoire.

Usage

```
OverLearn(P, population, learners)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`learners`	the indicies of birds that will attempt to learn

`QuickClusterPlot` *Quick Cluster Plots*

Description

Creates Cluster Plotss for all saved data except MaleSongs and FemaleSongs. See ClusterPlot for info on plot interpretation.

Usage

```
QuickClusterPlot(P, path, rep = TRUE, acc = P$SAcc, lrnThsh = P$SLrn,
  match = P$SMat, chanceInv = P$SCtI, chanceFor = P$SCtF,
  age = P$SAge, AutoLayout = TRUE)
```

Arguments

`P`	a list of parameters
`path`	location of a folder with simulation data
`rep`	whether to plot repertoire size data
`acc`	whether to plot accuracy data
`lrnThsh`	whether to plot learning threshold data
`match`	whether to plot matching data
`chanceInv`	whether to plot chance to invent data
`chanceFor`	whether to plot chance to forget data
`age`	whether to plot age data
`autoLayout`	whether to allow the function to figure out the layout (TRUE) or not (FALSE)

See Also

Other Cluster Plots: `ClusterCalc`, `ClusterPlot`, `GetMaxMat`

`QuickSEMPlot` *Quick SEM Plot*

Description

A method that plots whatever data was saved in the path location. It takes the column averages, so it works for Basic, Light, and Insult Sims, but not for Invasion Sims. For trait plots, black lines are the average, dark grey is the inner 50

Usage

```
QuickSEMPlot(P, path, rep = TRUE, acc = P$SAcc, lrnThsh = P$SLrn,
  match = P$SMat, chanceInv = P$SCtI, chanceFor = P$SCtF,
  age = P$SAge, mSong = P$SMSng, fSong = P$SFSng,
  autoLayout = TRUE, xlab = "Time Steps", thin = 10)
```

Arguments

`P`	a list of parameters
`path`	location of a folder with simulation data
`rep`	whether to plot repertoire size data
`acc`	whether to plot accuracy data
`lrnThsh`	whether to plot learning threshold data
`match`	whether to plot matching data
`chanceInv`	whether to plot chance to invent data
`chanceFor`	whether to plot chance to forget data
`age`	whether to plot age data
`mSong`	whether to plot male song data
`fSong`	whether to plot female song data
`autoLayout`	whether to allow the function to figure out the layout (TRUE) or not (FALSE)
`xlab`	x-axis label for plot()
`thin`	how often to sample a step of song data for the SongEvolve() plots; This is graphically intensive when there are a lot of syllables (default is 500), so ideally do not plot more than 100-200 time steps.

RandomDeath *Random Death*

Description

Randomly picks a percentage of males in the population to die. Current age is not a relevant factor in being chosen.

Usage

```
RandomDeath(P, population)
```

Arguments

`P`	a list of parameters
`population`	the population of birds

ReloadParam *Reload Parameters*

Description

Loads a .SEMP file and converts it into a list of parameters.

Usage

```
ReloadParam(filePath)
```

Arguments

`filePath`	the pather where a .SEMP is located

SaveParam *Save Parameterss*

Description

Saves a list of parameters as a .SEMP file.

Usage

```
SaveParam(P, folderName, fileName = "Parameters", type = "Basic")
```

Arguments

`P`	a list of parameters
`folderName`	where to save the .SEMP
`type`	the simulation type run (accepts any string)
`fName`	file name for the .SEMP

`SEMSimulation` *SEM Simulation*

Description

A wrapper that conveniently handles data saving and times the simulation

Usage

```
SEMSimulation(P, type = "Basic", folderName = NA, save = TRUE,
  return = FALSE, verbose = TRUE, ...)
```

Arguments

`P`	a list of parameters
`type`	what type of simulation to run ('Basic', 'Light', 'Insult)
`folderName`	where to save the simualtion data, defaults to a timestemp in the current directory
`save`	whether to write the data to .csvs
`return`	whether to return the data in R
`verbose`	whether to print the running time and folder name
`...`	arguments to the simulation types. See documentation for individual sim type arguments.

See Also

Other Sim Functions: `BasicSimulation`, `CheckInsultPs`, `InsultSimulation`, `LightSimulation`

Examples

```
P <- DefineParameters(RepSizePrefer = 0, MatchPrefer = 1, numSim=100)
SEMSimulation(P, 'Basic', 'Example', return=TRUE)
SEMSimulation(P, 'Interval', 'Example', return=TRUE, freq=2)

P2 <- DefineParameters(numSim=600, MatchPrefer = 1, RepSizePrefer = 0)
P3 <- DefineParameters(numSim=600, SaveMatch = TRUE)
SEMSimulation(P2, insultP=P3, 'Insult', when=100, freq=2, save=FALSE, return = TRUE)
```

`SongPlot` *Song Plot*

Description

Shows the prevalence of each syllabel across time. Darker color means that a syllable is more common.

Usage

```
SongPlot(P, songs, thin = 10, male = TRUE)
```

Arguments

`P`	a list of parameters
`songs`	male or female song data from simulation
`thin`	how often to sample a step of song data for the SongEvolve() plots; This is graphically intensive when there are a lot of syllables (default is 500), so ideally do not plot more than 100-200 time steps.
`male`	whether male songs are being plotted; affects the y-axis label

---

`TerritoryHeatMap` *Territory Heat Map*

Description

Creates a heat map showing the magnitude of a trait in each territory for a given timestep. Requires individual data.

Usage

```
TerritoryHeatMap(P, index = 1, trait, max = NA)
```

Arguments

`P`	a list of parameters
`index`	which column to plot
`trait`	a matrix of SEM data from a Basic sim (individual data)

`TestLearningThreshold` *Test Learning Threshold*

Description

Tests whether males are young enough to learn.

Usage

```
TestLearningThreshold(P, males)
```

Arguments

`P`	a list of parameters
`males`	the bird trait data.frame from the population of birds ($Males)

TestMatch *Test Match*

Description

Calculates how well the female template matches the male template. Mismatch is based how many syllables the female knows that the male does not (Missing) + how many more syllables does the male know than the female (Extra, min 0). Match = 1 - Mismatch/Number of Female Syllables.

Usage

```
TestMatch(P, maleSong, femaleSong)
```

Arguments

`P`	a list of parameters
`maleSong`	a syllable vector
`femaleSong`	a syllable vector

TestRequirement *Test Requirement*

Description

If a Save parameter is set to NA, checks whether they should be set to TRUE or FALSE.

Usage

```
TestRequirement(test, dependancy1 = 0, dependancy2 = FALSE)
```

Arguments

`test`	a SaveTrait
`dependancy1`	a value for trait noise
`dependancy2`	a secon dependancy that requires teh trait

`TraitPlot` *Trait Plot*

Description

Plots averages for a bird trait. Black lines are the average, dark grey is the inner 50

Usage

```
TraitPlot(trait, xlab = "Time Steps", ylab)
```

Arguments

`trait`	a trait matrix to plot
`xlab`	x-axis label for plot()
`ylab`	y-axis label for plot()

`UpdateProbabilities` *Update Survival Proportions*

Description

At the end of a timestep, it removes the adult survival proportion from the generation that has just been extinguished, and calculates the adult survival proportion for the chicks that have just been generated.

Usage

```
UpdateProbabilities(P, chicks, prob)
```

Arguments

`P`	a list of parameters
`chicks`	vector of chick indicies
`prob`	the data structure from the population that keeps track of survival probabilities

`UpdateSongTraits` *Update Song Traits*

Description

Updates the SylRep and Match traits for learners post learning.

Usage

```
UpdateSongTraits(P, population, learners)
```

Arguments

`P`	a list of parameters
`population`	the population of birds
`learners`	the indicies of birds that will attempt to learn

`VerticalSongLearning` *Vertical Song Learning*

Description

A wrapper that prepares chick and tutor template data during vertical learning.

Usage

```
VerticalSongLearning(P, templates, chicks)
```

Arguments

`P`	a list of parameters
`templates`	matrix of the fathers' syllable vectors
`chicks`	song-learning traits of chicks form the population ($Males)

Index

∗Topic **birth**
BirthDeathCycle, 4
ChooseFathers, 8
GenerateChicks, 16
GetProbability, 19
∗Topic **death**
AgeDeath, 2
BirthDeathCycle, 4
LearningThrshPenalty, 20
RandomDeath, 26
UpdateProbabilities, 30
∗Topic **error-check**
CheckBool, 5
CheckInsultPs, 6
CheckMinMaxInt, 6
CheckP, 7
CheckTrait, 7
TestRequirement, 29
∗Topic **female-choice**
AssignFemale, 3
GetProbability, 19
TestMatch, 29
∗Topic **initialize**
DefineParameters, 11
GenerateAdultBirds, 15
GenerateFounderMales, 16
∗Topic **locality**
FinalDirections, 15
LocalSearch, 22
NextStepDirections, 22
OneStepDirections, 23
∗Topic **read-write-run**
BasicSimulation, 3
InsultSimulation, 19
LightSimulation, 21
ReloadParam, 26
SaveParam, 26
SEMSimulation, 27
∗Topic **song-leanring**
UpdateSongTraits, 31
∗Topic **song-learning**
CalcFractional, 4
CheckEncouter, 5
ChooseTutors, 8
ConsensusLearning, 10
DropSyllables, 14
FemaleEvolve, 15
GetLearners, 18
LearningProcess, 20
ListeningTest, 21
ObliqueLearning, 23
OneTutorLearning, 23
OverLearn, 24
TestLearningThreshold, 28
VerticalSongLearning, 31
∗Topic **song-template**
CreateFemaleSongs, 10
EstablishDialects, 14
GenerateNovelSong, 17
∗Topic **stats-plotting**
ClusterCalc, 9
ClusterPlot, 9
FamilyTreePlot, 14
GetMaxMat, 18
QuickClusterPlot, 24
QuickSEMPlot, 25
SongPlot, 27
TerritoryHeatMap, 28
TraitPlot, 30
∗Topic **survival-curve**
CalculateAllGen, 4
CalculateProportion, 5
GetAgeGroup, 17
GetAgeRates, 17
InitAgeDistribution, 19
UpdateProbabilities, 30

AgeDeath, 2
AssignFemale, 3

BasicSimulation, 3, *6*, *20*, *21*, *27*
BirthDeathCycle, 4

CalcFractional, 4
CalculateAllGen, 4
CalculateProportion, 5
CheckBool, 5

CheckEncouter, 5
CheckInsultPs, *3*, 6, *20*, *21*, *27*
CheckMinMaxInt, 6
CheckP, 7
CheckTrait, 7
ChooseFathers, 8
ChooseTutors, 8
ClusterCalc, *9*, 9, *18*, *25*
ClusterPlot, *9*, 9, *18*, *25*
ConsensusLearning, 10
CreateFemaleSongs, 10

DefineParameters, 11
DropSyllables, 14

EstablishDialects, 14

FamilyTreePlot, 14
FemaleEvolve, 15
FinalDirections, 15

GenerateAdultBirds, 15
GenerateChicks, 16
GenerateFounderMales, 16
GenerateNovelSong, 17
GetAgeGroup, 17
GetAgeRates, 17
GetLearners, 18
GetMaxMat, *9*, 18, *25*
GetProbability, 19

InitAgeDistribution, 19
InsultSimulation, *3*, *6*, 19, *21*, *27*

LearningProcess, 20
LearningThrshPenalty, 20
LightSimulation, *3*, *6*, *20*, 21, *27*
ListeningTest, 21
LocalSearch, 22

NextStepDirections, 22

ObliqueLearning, 23
OneStepDirections, 23
OneTutorLearning, 23
OverLearn, 24

QuickClusterPlot, *9*, *18*, 24
QuickSEMPlot, 25

RandomDeath, 26
ReloadParam, 26

SaveParam, 26
SEMSimulation, *3*, *6*, *20*, *21*, 27
SongPlot, 27

TerritoryHeatMap, 28
TestLearningThreshold, 28
TestMatch, 29
TestRequirement, 29
TraitPlot, 30

UpdateProbabilities, 30
UpdateSongTraits, 31

VerticalSongLearning, 31