HCB

June 14, 2019

Title Human Cultural Boundaries

Version 0.0.0

Description Creates seed populations with phoneme inventories that can grow, migrate, and create offshoot populations. Phoneme inventories mutate when populations establish a new territory.

License What license it uses

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1.9000

Imports mc2d, randomcoloR, uuid, numbers, philentropy, ade4

R topics documented:

`AddBeringStrait` *Add Bering Strait*

Description

Removes connections at the FirstStep stage of the Local structure to create "barriers" between cells. Bering Strait Barriers are designed to create structures similar to the Bering strait entering North America, Traveling to Central America, then opening up into South America.

Usage

```
AddBeringStrait(P, firstStep)
```

Arguments

`P`	A list of parameters.
`firstStep`	The local directions created by OneStepDirections().

`AddShift` *Add Shift Phoneme*

Description

Allows a language to either gain a new phoneme and shift and existing phoneme to match another population.

Usage

```
AddShift(P, language, languages, local, phonemeRelatedness, index)
```

Arguments

`P`	A list of parameters.
`language`	The target language to be modified if possible.
`languages`	All languages
`local`	The local territories data structure.
`index`	The target territory whose language may change.
`phonemeProbab`	The probability of gaining each phoneme in the population.

AddSnakeBarriers *Add Snake Barriers*

Description

Removes connections at the FirstStep stage of the Local structure to create "barriers" between cells. Snake Barriers are lines with length and spacing defined by the parameters. The barriers jut out from the east and west walls, alternating east, west, east, west. This creates a snaking zig-zag pattern, hence the name.

Usage

```
AddSnakeBarriers(P, firstStep)
```

Arguments

P	A list of parameters.
firstStep	The local directions created by OneStepDirections().

CardinalDirections *Cardinal Directions*

Description

Calculates the terrritory numbers of locations around a target territory (also used for phoneme relatedness in the same way).

Usage

```
CardinalDirections(target, R, start, round, South, North, East, West, SE,
  NE, SW, NW)
```

Arguments

target	The territory around which to get local territoies.
R	The number of rows.
start	How much to offset numbers (for phoneme structures).
round	Whether to get a "round" set of territories (N, S, E, W only) for phonemes or a square set of territories (includes diagonals) for distance.
SE	Whether to get the southeasrern territory.
NE	Whether to get the northestern territory.
SW	Whether to get the southwestern territory.
NW	Whether to get the northwestern territory.
south	Whether to get the southern territory.
north	Whether to get the northern territory.
east	Whether to get the eastern territory.
west	Whether to get the western territory.

DefineParameters *Define Parameters*

Description

Creates a parameter data structure for running simulations.

Usage

```
DefineParameters(Rows = 40, Cols = 50, ChanceExpand = 0.8,
  PopulationStartIndex = c(1, 2), NumPopulationPhonemes = rep(NA,
  length(PopulationStartIndex)), UsePopSize = TRUE,
  IndividualsStEmSuEM = c(1000, 10, 20, NA), MutationRate = 15,
  PhonemeDitribution = c(12, 24, 133), Consonants = 750,
  Vowels = 100, MinConsonant = 6, MinVowel = 6,
  PhonemeProbabilityType = "RealMimic", GrowthRate = 5,
  Barriers = FALSE, BarrierLength = 30, BarrierBreaks = 4,
  MutationTypeChance = rep(1/5, 5), HorizontalRate = 0.1,
  Bias = TRUE, Steps = 1, HorizontalLocal = TRUE,
  NumberRandomHorizontal = 8, UpRoot = TRUE, Death = TRUE,
  Bering = FALSE, MigrationSimSteps = 300, HorizontalSimSteps = 400,
  Waves = FALSE, Seed = NA)
```

Arguments

Rows	The number of rows in the world matrix.
Cols	The number of columns in the world matrix.
ChanceExpand	The chance that a population will either move or send off a group of individuals to found a new population.
PopulationStartIndex	The position in the matrix where each seed population starts. The number of seed populations is defined by the number of starting indicies.
NumPopulationPhonemes	The number of phonemes in each starting population. If set to NA, this is decided by sampling from a distribution with min, mode, and made on the values from the PhonemeDistribution arguement.
UsePopSize	Whether to take into account the the population size (number of people) when making decisions about moving, immegrating, and phoneme loss/addition biases.
IndividualsStEmSuEM	Four related parameters: 1) The number of individuals a seed population stats with, 2) the minumum number of individuals required to make a founder party to settle a new territory, 3) the minumum number of individuals that must stay behind when a founder party is sent off, 4) the maximum number of individuals allowed to be in one founder party.
MutationRate	The rate at which phonemes mutate. E.g., if MutationRate==0.1, each phoneme in a populatiosn phoneme inventory has a 10% chance to mutate.
Consonants	The number of possible consonants in existence. Default based on real phoneme data.

`Vowels` The number of possible vowels in existence. Default based on real phoneme data.

`MinConsonant` The minumum number a Consonants that can be in a population's phoneme inventory. Default based on real phoneme data.

`MinVowel` The minumum number a vowels that can be in a population's phoneme inventory. Default based on real phoneme data.

`PhonemeProbabilityType`
The method by which phoneme probabilities are established.

`GrowthRate` When an integer, the nunmber of individuals added to each population every time step. When a fraction, the percent that a population increases each timestep.

`Barriers` Whether to create "snake barriers" that limit the direction of migration in the matrix.

`BarrierLength` The width of snake barriers.

`BarrierBreaks` The height of the space between snake barriers.

`MutationTypeChance`
The chance that each mutation type occurs. 1) Add, 2) Lose, 3) Split, 4) Join, 5) Shift.

`HorizontalRate` The fraction of the population that attempts to modify its phoneme inventory every horizontal timestep.

`Bias` Whether to randomly bias mutations towards either gains or losses when populations are small. Set to true based on previously published data.

`Steps` The number of distance steps away from a target location that are considered "local." Includes all 8 cardinal and ordinal directions around a target, so the local area is always a rectangle around the target location.

`HorizontalLocal`
Whether horizonta transfer occurs between local populations or globally. Set to FALSE as a control, as global horizontal transfer should abolish local patterns.

`NumberRandomHorizontal`
The number of locatiosn to conpare when HorizontalLocal==FALSE. Should be 8 when Steps==1, 24 when steps==2, 48 when Steps=3, ect.

`UpRoot` Whether establish populations can move (TRUE) or they remain in place for the entire simulation (FALSE).

`Death` Whether a population can die out.

`Bering` Whether to employ barriers that mimick the Bering Strait and Americas.

`MigrationSimSteps`
The number of time steps to run each wave of migration.

`HorizontalSimSteps`
The number of time steps to spend on horizontal transfer.

`Waves` Whether migration occurs in waves or all seed populations are added at the same time. If TRUE, there is one wave for each seed population.

`Seed` Sets a seed for reproducibility if an integer instead of NA.

`PhonemeDistribution`
The 1) min, 2) mode, and 3) max number of phonemes a population can have when sampling for seed population sizes and when preventing languages from gaining or losing too many phonemes. Default based on real phoneme data.

Emigrate *Emigrate*

Description

Picks which populations migrate, whether the entire population migrates or a founder party is sent off, and where the population mirgates to. Allows only one population to enter a territory. When multiple popultaions attempt to enter the same territory, one is randomly chosen to do so while the rest stay put.

Usage

```
Emigrate(P, occupied, local, populations)
```

Arguments

P	A list of parameters.
occupied	The territories with a population on them.
local	The local territories data structure.
populations	The data for all exising populations.

Extinction *Extinction*

Description

Tests which populations will die based on population size and random chance.

Usage

```
Extinction(populations, occupied)
```

Arguments

populations	The data for all exising populations.
occupied	The territories with a population on them.

GeneratePhonemeProbabilities *Generate Phoneme Probabilities*

Description

Genetrate a vector of the probability to know each phoneme.

Usage

```
GeneratePhonemeProbabilities(P)
```

Arguments

P	A list of parameters.

GenerateSeedLanguage *Generate Seed Language*

Description

Generate Seed Language

Usage

```
GenerateSeedLanguage(P, phonemeProbab, seedNum)
```

Arguments

P	A list of parameters.
phonemeProbab	The probability of gaining each phoneme in the population.
seedNum	Which population seed is having it's language generated.

GetAMutation *Get A Mutation*

Description

Returns a new Add, Loss, Split, Join, or Shift mutation. Biases the mutations towards gaining or losing syllables when the parameter Bias==TRUE.

Usage

```
GetAMutation(P, phonemes, phonemeProbab, phonemeRelatedness, gain)
```

Arguments

P	A list of parameters.
phonemes	The phonemes currently in the language.
phonemeProbab	The probability of gaining each phoneme in the population.
phonemeRelatedness	The phoneme relatedness list.
gain	Whether to bias the mutations towards gaining syllables (TRUE) or losing them (FALSE).

GetASplitShiftJoinMut *Get A Split Shift Join Mutation*

Description

Recursively calls itself until a phoneme is found that can be used the generate the mutation type of interest. Returns Null if no phoneme can mutate appropriately.

Usage

```
GetASplitShiftJoinMut(phonemes, phonemeRelatedness, unusable = NULL,
  type)
```

Arguments

phonemes	Phonemes that can be mutated.
phonemeRelatedness	The phoneme relatedness list.
unusable	Phonemes that cannot be used to obtain the correct type of mutation.
type	Which kind of mutation to find, Split, Join, or Shift.

GetFactorDim *Get Factor Dimentions*

Description

Given a number of consomants or vowel, create a datastructure that is as square as possible.

Usage

```
GetFactorDim(nPhonemes)
```

Arguments

nPhonemes	The number of Phonemes (vowels or consonants).

GetImmigrants *Get Immigrants*

Description

Tests which populations immigrate, removes that did not from the migraton data and splits them into populations that sent off founder parties and those that moved as a single population.

Usage

```
GetImmigrants(P, occupied, local, populations)
```

Arguments

occupied	The territories with a population on them.
local	The local territories data structure.
populations	The data for all exising populations.

GetRealPhonemeData *Get Real Phoneme Data*

Description

Uses the real Phoneme data from Creanza..... UPDATE THIS!!!! to determine the phoneme probabilities.

Usage

```
GetRealPhonemeData(nPhoneme, actual, vowel = FALSE)
```

Arguments

nPhoneme	The number of phonemes (vowels or consonants).
actual	Whether the data is Real (TRUE) or RealMimic (FALSE).
vowel	If true load the vowel data, otherwise load the cosonant data.

GetTerritory *Get Territory*

Description

Returns a territory a population can migrate to or NA if none are available.

Usage

```
GetTerritory(local, open)
```

Arguments

local	Territories that are within reach of the target territory.
open	Which territoies can be migrated to (i.e. no other population currently resides there).

HCBSimmulation *Human CUltural Boundaries Simulation*

Description

Runs a simulation

Usage

```
HCBSimmulation(P)
```

Arguments

P	A list of parameters.

HoritontalTransferRepeater

After migration, allow populations to exchnage phoneme information, losing or gaining syllables based on other populations in the simulation.

Description

After migration, allow populations to exchnage phoneme information, losing or gaining syllables based on other populations in the simulation.

Usage

```
HoritontalTransferRepeater(P, S)
```

Arguments

P	A list of parameters.
S	A list of the data structures.

HorizontalTransfer *Horizontal Transfer*

Description

A function wrapper that get the language to modify and allows the phoneme change to either add/shift or remove a phoneme if this can be done.

Usage

```
HorizontalTransfer(P, languages, local, phonemeRelatedness, phonemeProbab,
  index)
```

Arguments

P	A list of parameters.
languages	All languages.
local	The local territories data structure.
phonemeRelatedness	The phoneme relatedness list.
phonemeProbab	The probability of gaining each phoneme in the population.
index	The target territory whose language may change.

Initialize *Initialize*

Description

The function wrapper that makes calls to create the population and phoneme data structures and then populatas them with initial data.

Usage

```
Initialize(P)
```

Arguments

P	A list of parameters.

Lose *Lose Phoneme*

Description

Allows a language to either lose a phoneme to better match other populations.

Usage

```
Lose(P, language, phonemeProbab)
```

Arguments

P	A list of parameters.
language	The target language to be modified if possible.
phonemeProbab	The probability of gaining each phoneme in the population.

MakeLanguage *Make Language*

Description

Mutates the parent's langauge to create a new language for a founder party.

Usage

```
MakeLanguage(P, phonemeProbab, phonemeRelatedness, language, popSize)
```

Arguments

P	A list of parameters.
phonemeProbab	The probability of gaining each phoneme in the population.
phonemeRelatedness	The phoneme relatedness list.
language	The parent langauge to mutate into a new language.
popSize	The number of individuals in the parent poultaion.

`MakePopulation` *Make Population*

Description

Generates new population based on the parent population.

Usage

```
MakePopulation(P, population)
```

Arguments

`P`	A list of parameters.
`population`	The population data used to make a new population.

`Migration` *Migration*

Description

Main simulation function. Allows populations to migrate, split, and die.

Usage

```
Migration(P, S)
```

Arguments

`P`	A list of parameters.
`S`	A list of data structures.

`NextStepDirections` *Next Step Directions Expands the Steps list one more step out.*

Description

Next Step Directions Expands the Steps list one more step out.

Usage

```
NextStepDirections(firstStep, currentStep, start = 0)
```

Arguments

`firstStep`	The original StepOne.
`currentStep`	StepOne in its current state.
`start`	How much to offset numbers (for phoneme structures).

NextWave *Next Wave*

Description

Adds the seed data for the next wave to the population and language dataframes.

Usage

```
NextWave(P, S, i)
```

Arguments

P	A list of parameters.
S	A list of data structures.
i	The number of the next wave.

OneStepDirections *One Step Directions*

Description

One Step Directions

Usage

```
OneStepDirections(R, C, start = 0, round = FALSE)
```

Arguments

R	The number of rows.
C	The number of columns.
start	How much to offset numbers (for phoneme structures).
round	whether to make the spacing Round (Phonemes) or Sqaure (Territories).

PopulationGrowth *Population Growth*

Description

Adds new individuals to existing populations when population size is used in the simulation.

Usage

```
PopulationGrowth(growthRate, populationSizes, occupied)
```

Arguments

growthRate	The population growth rate parameter.
populationSizes	The number of people live on each territory.
occupied	The indicies of territories with people living on them.

RemoveHorizontalConnections *Remove Horizontal Connections*

Description

Affects local territories below/South (and perhaps to the Southeast and Southwest) the target territory (index) and above/North (perhaps Northwest and Northeast) of index +1.

Usage

```
RemoveHorizontalConnections(R, index, firstStep, right = TRUE,
  left = TRUE)
```

Arguments

R	The number of rows in the population matrix.
index	The target territory.
firstStep	The local directions created by OneStepDirections().
right	Whether to remove the right diagonal.
left	Whether to remove the left diagonal.

RemoveVerticalConnections *Remove Vertical Connections*

Description

Affects local territories right/East (and perhaps to the Northeast and Southeast) the target territory (index) and left/West (perhaps Northwest and Southwest) of index + R.

Usage

```
RemoveVerticalConnections(R, index, firstStep, above = TRUE,
  below = TRUE)
```

Arguments

R	The number of rows in the population matrix.
index	The target territory.
firstStep	The local directions created by OneStepDirections().
above	Whether to remove the upper diagonal.
below	Whether to remove the lower diagonal.

ResetPopulation *Reset Population*

Description

Blanks phoeneme and population data in the original territory whe an entire population moes to a new territory.

Usage

```
ResetPopulation()
```

ShiftDirections *Shift Directions*

Description

Returns the relationships between phonemes with an offset of start.

Usage

```
ShiftDirections(nPhonemes, start = 0)
```

Arguments

nPhonemes	The number of phonemes.
start	Where to start number the phonemes (0 for consonants, number of consonants +1 for vowels).

`StepDirections` *Step Directions*

Description

A wrapper that calls StepOne(), add barriers if required, then expands StepOne as many steps as the Steps parameter calls for.

Usage

```
StepDirections(P)
```

Arguments

`P`	A list of parameters.

`UpdateExistingPhonemes` *Update Existing Phonemes*

Description

Change the language to incorporation new mutations.

Usage

```
UpdateExistingPhonemes(existingPhonemes, newMut, index)
```

Arguments

`newMut`	The new mutation generated by GetAMutation().
`index`	Whether to enact changes to the langauge based on the first or section member of the mutation structure.
`ExistingPhonemes`	The Phonemes currently in the language.

UpdateStructuresMove *Update Structures Move*

Description

Copies population data from one territory to another when the entire population migrates and then erases the original data.

Usage

```
UpdateStructuresMove(S, move, former)
```

Arguments

S	A list of data structures.
move	The indicies of territories
former	The indicies of territories

UpdateStructuresRemove *Update Structures Remove*

Description

Deletes population and language information for specified territories.

Usage

```
UpdateStructuresRemove(S, remove)
```

Arguments

S	A list of data structures.
Remove	The indicies of territories whose data should be erased.

Index