**Classes.jl**

**Tipos utilizados e suas Funções associadas**

**Host**

Definidos por:

uma sequência de arrays booleanos L x 2

um contador do número de vezes que o hospedeiro resistiu à infecção por um Path

Funções:

*resisted!*

Atualize o contador do Host em 1

*reproduce*

Escolha um alelo aleatório de cada um de dois Hosts

Considerando a taxa de mutação da população de Hosts, mute cada um dos alelos selecionados (ver *mutate\_haplotype*)

Crie um novo Host utilizando os alelos potencialmente mutados e inicializando seu contador como 0

**PopHost**

Definidos por:

Uma array de Hosts

Um taxa de mutação

Funções:

*number\_of\_alleles*

Verifique o tamanho dos loci (L)

Inicialize um vetor (alleles) booleano do tamanho do número de alelos possíveis na população, incialmente, com todas as entradas como '*false*'

Para cada hospedeiro na população

Pensando em cada alelo como um uma representação binária de um número inteiro (note que não há degenerescência), verifique qual é esse inteiro (ver *to\_Integer*) e marque como '*true*' na posição do vetor booleano (isto é, tal alelo está presente)

Conte quantas entradas '*true*' existe no vetor booleano. Esse será o número de alelos distintos na população

*fitness*

Verifique qual o tamanho da população

Inicialize um vetor (fit) com todas as entradas nulas, que armazenará a fitness dos indivíduos da população

Para cada hospedeiro i na população

fit(i) armazenará o número de vezes que ele resistiu ao contato com patógenos ao quadrado

Caso nenhum hospedeiro tenha resistido (sum\_of\_squares = 0)

Todos devem possuir a mesma fitness: 1 / tamanho da população

Do contrário

Calcule a fitness de fato (ver comentários gerais sobre isso no outro arquivo), como fit(i) / sum\_of\_squares para cada hospedeiro i

*random\_hosts*

Calcule a fitness da população

Retorne uma amostra (sem reposição) de tamanho size\_of\_sample da população, retirada com probabilidades dadas pela fitness

*reproduce*

Verifique o tamanho da população

Inicialize uma nova população com todos os hospedeiros iguais (para alocar o espaço na memória)

Para cada “espaço” na nova população nova

Escolha dois indivíduos aleatóriamente (ver *random\_hosts*) na população antiga

Crie um novo hospedeito da população nova a partir desses dois indivíduos escolhidos (ver *reproduce* em Host)

**Path**

Definidos por:

Uma sequência de arrays booleanos L x N\_pep

Um contador do número de vezes que o patógeno infectou um Host

Funções:

*infected!*

Atualize o contador do número de vezes que o patógeno em 1

*reproduce*

Retorne um Path idêntico ao original, a menos da possibilidade de mutação em cada alelo (a partir da taxa de mutação na população, ver *mutate\_haplotype*) e do contador, que deve ser inicializado como 0.

*pathogen\_infects\_host*

Calcule o número de peptídeos que cada Path possui (N\_pep)

Para cada peptídeo do Path em questão

Verifique se o Host em questão é capaz de apresentá-lo (ver *host\_presents\_peptide*)

Se o Host apresenta algum dos peptídeos, retorne '*false*'. Se não apresenta nenhum, retorne '*true*'

**PopPath**

Definidos por:

Uma array de Paths

Um taxa de mutação

Funções:

*fitness*

Totalmente análoga à função *fitness* do tipo PopHost, mas considerando o número de vezes que cada Path infectou, ao invés do número de vezes que o Host resistiu

*random\_pathogen*

Retorne uma amostra (sem reposição) de tamanho size\_of\_sample da população de patógenos, onde cada Path possui probabilidade de ser escolhido igual à sua fitness

*reproduce*

Verifique o tamanho de PopPath e inicialize uma nova população

Para cada “espaço” na nova população

Escolha um patógeno da população antiga

Gere um novo usando a função *reproduce*

Retorne a nova população

**Demais funções**

*mutate\_alele*

Determine o tamanho do alelo

Sorteie um número aleatório entre 0 e 1. Se ele for menor que a taxa de mutação

Gere uma sequência de booleanos completamente nova para aquele alelo

*mutate\_haplotype*

Para cada alelo de um haplótipo (sequência de alelos adjacentes)

Aplique a função *mutate\_alele* para esse alelo

*match*

Inicialize maximum\_match = 0

Determine o tamanho da sequencia

Inicialize máximo atual = 0

Para cada entrada da sequência

Se entrada de cada uma das duas sequências for diferente (ou seja, complementares)

adicione 1 ao máximo atual

Se o máximo atual for maior que o maximum\_match

o maximum\_match recebe o valor do máximo atual

Do contrário

máximo atual = 0

Retorne '*true*' se maximum\_match for maior ou igual que o comprimento mínimo de complementariedade, e '*false*', caso contrário

*host\_presents\_peptide*

Para os dois alelos do Host

Se houve match entre o alelo e o peptídeo

Retorne '*true*'

Se nenhum alelo for (suficientemente, ver *match*) complementar ao peptídeo

Retorne '*false*'

*to\_Intege*

Calcule o tamanho da sequência

Inicialize um inteiro = 0

Para cada entrada da sequência

Se for '*true*' (1, na base binária)

inteiro = inteiro + 2^(i-1)

Retorne (inteiro)