

```

1 // 後で他のソースコードから関数をモジュールとして参照するので、パッケージをexportしている。
2 // Go言語では、初めの文字が大文字の関数が公開され、小文字の関数は公開されない。
3 package locust
4
5 import (
6     "../matmul" // MatVec関数, MatMlt関数, Transpose関数, 定数N
7     "encoding/csv" // 計算結果をCSVに出力するため
8     "fmt" // Sprint関数を使うため
9     "os" // 計算結果をCSVに出力するため
10 )
11
12 // #define N 3 の代わり。
13 const N = matmul.N
14
15 func main() {
16     // c = 0.5, s = 0.05の場合の、各時刻における各地点の、バッタGの存在確率を計算している
17     res1 := Calculation(0.5, 0.05)
18     // c = 0.5, s = 0.15の場合の、各時刻における各地点の、バッタGの存在確率を計算している
19     res2 := Calculation(0.5, 0.15)
20     // c = 0.5, s = 0.5の場合の、各時刻における各地点の、バッタGの存在確率を計算している
21     res3 := Calculation(0.5, 0.5)
22
23     // 計算結果をCSVに出力している
24     WriteCSV("0.05", res1)
25     WriteCSV("0.15", res2)
26     WriteCSV("0.5", res3)
27 }
28
29 // 書き出すファイル名と、書き出す内容を含んだArrayを受け取り、
30 // CSVファイルに書き出す関数
31 func WriteCSV(filename string, resVec [61][N]float64) {
32     // CSVを書き出すファイルを指定
33     file, _ := os.Create("res/" + filename + ".csv")
34     writer := csv.NewWriter(file)
35     // CSVのヘッダーを指定
36     writer.Write([]string{
37         "t",
38         "point 0",
39         "point 1",
40         "point 2",
41         "point 3",
42         "point 4",
43         "point 5",
44     })
45     for t := 0; t <= 60; t++ {
46         // writer.Write 関数はstringのArrayしか受け取らないので、
47         // resVec[t]の各要素をfmt.Sprintf関数でstringに変換して、
48         // そのArrayを生成し、それをwriter.Writeに渡している。
49         writer.Write(
50             []string{
51                 fmt.Sprintf(t),
52                 fmt.Sprintf(resVec[t][0]),
53                 fmt.Sprintf(resVec[t][1]),
54                 fmt.Sprintf(resVec[t][2]),
55                 fmt.Sprintf(resVec[t][3]),
56                 fmt.Sprintf(resVec[t][4]),
57                 fmt.Sprintf(resVec[t][5]),
58             }
59         )
60     }
61     // バッファをファイルへ出力する
62     writer.Flush()
63 }
64
65 // パラメーターcとsを受け取り、0 <= t <= 60の範囲のバッタGの
66 // 各地点での存在確率を計算する
67 func Calculation(c float64, s float64) [61][N]float64 {
68     // 移動確率を表す行列を計算し、取得する
69     var probMat [N][N]float64 = getProbMat(c, s)
70
71     var resultVectors [61][N]float64 // resultVectors[t]: 時刻tに於ける、Gの地点0~5での存在確率を表すベク
72     resultVectors[0] = [N]float64{1, 0, 0, 0, 0, 0} // t = 0のとき、バッタGの地点0~5での存在確率は

```

```

1,0,0,0,0,0である
73 // t = 1, 2, ..., 60について、
74 // Gの存在確率ベクトルを求める
75 for t := 1; t <= 60; t++ {
76     // ある時刻tのGの各地点での存在確率は、
77     // 移動確率を表す行列と、t-1での存在確率ベクトルの、行列ベクトル積である。
78     resultVectors[t] = matmul.MatVec(probMat, resultVectors[t-1])
79 }
80 return resultVectors
81 }
82
83 // パラメーターcとsを受け取り、バッタの移動確率を表す行列の値を計算する
84 func getProbMat(c float64, s float64) [N][N]float64 {
85     // 問に示された「バッタGの振る舞い」行列を転置したものと、
86     // 時刻tでのバッタの存在確率ベクトルとの行列ベクトル積を取ると、
87     // その計算結果が時刻t+1でのバッタの存在確率ベクトルとなっている。
88     return matmul.Transpose([N][N]float64{
89         {s + (1-s)*(1-c), (1 - s) * c, 0, 0, 0},
90         {(1 - s) * (1 - c), s, (1 - s) * c, 0, 0},
91         {0, (1 - s) * (1 - c), s, (1 - s) * c, 0},
92         {0, 0, (1 - s) * c, s, (1 - s) * (1 - c)},
93         {0, 0, 0, (1 - s) * c, s, (1 - s) * (1 - c)},
94         {0, 0, 0, 0, (1 - s) * c, s + (1-s)*(1-c)},
95     })
96 }

```