

Bertels et al., Estimating duplication rates of bacterial mobile genetic elements

Supplementary Information

Defining functions and importing data

Functions and definitions

Function that calculates equilibrium frequencies for a given set of fitness values and a given mutation matrix

```
CalculateEquilibriumFrequencies[fitness_, mutmatrix_] :=
Module[{},
  W := Transpose[mutmatrix fitness];
  ESys := Eigensystem[W];
  sorted = Sort[ESys];
  Return[sorted[[2, 1]] / Total[sorted[[2, 1]]];
];
```

Function that calculates the fitness values for a given set of equilibrium frequencies and a given mutation matrix

```

CalculateFitness[frequencies_, mutmatrix_] :=
Module[{f0, f1, f2, f3, f4, phi},
x0 = frequencies[[1]];
x1 = frequencies[[2]];
x2 = frequencies[[3]];
x3 = frequencies[[4]];
x4 = frequencies[[5]];
f0 = .;
f1 = .;
f2 = .;
f3 = .;
f4 = 1; (* set lowest fitness to one*)
q = mutmatrix;
phi = f0 x0 + f1 x1 + f2 x2 + f3 x3 + x4 f4;
(*Solve quasispecies model for fitness*)
s = Solve[{0 == x0 (f0) (q[[1, 1]]) + (x1) f1 q[[2, 1]] - x0 phi,
0 == x0 (f0) (q[[1, 2]]) + (x1) f1 q[[2, 2]] + (x2) f2 q[[3, 2]] - x1 phi,
0 == x1 (f1) (q[[2, 3]]) + (x2) f2 q[[3, 3]] + (x3) f3 q[[4, 3]] - x2 phi,
0 == x2 (f2) (q[[3, 4]]) + (x3) f3 q[[4, 4]] + (x4) f4 q[[5, 4]] - x3 phi,
0 == x3 (f3) (q[[4, 5]]) + (x4) f4 q[[5, 5]] - x4 phi, x0 + x1 + x2 + x3 + x4 == 1,
x0 > 0, x1 > 0, x2 > 0, x3 > 0, x4 > 0}, {f0, f1, f2, f3}];
(*if there are fitness values below one; scale these to one*)
min = Min[Table[s[[1]][[j]][[2]], {j, 1, mutclasses - 1}], 1];

fitnessTable = Table[s[[1]][[j]][[2]]/min, {j, 1, mutclasses - 1}];
fitnessTable = Append[fitnessTable, 1/min];
(*add mutation rate to output*)
fitnessTable = Append[fitnessTable, u];
(*add observed frequencies to output*)
For[j = 1, j ≤ mutclasses - 1, j++,
fitnessTable = Append[fitnessTable, freqs[[i]][[j]] // N];
];
fitnessTable = Append[fitnessTable, x4 // N];
fitnessTable = Append[fitnessTable, names[[i]]];
Return[fitnessTable];
]

```

Number of sequence classes is given by *mutclasses* while the sequence length is denoted by *L*

```

mutclasses = 5;
L = 25;

```

Following *Wielgoss et al. 2011* the mutation rate in *E. coli* is set to *u*

```

u = 8.9 × 10-11;

```

Using all the above definitions we now define the mutation kernel as given by *mutmatrix*

```
mutmatrix[u_, L_] := {{1 - u L, u L, 0, 0, 0}, {u/3, 1 - u*(L - 1/3), u*(L - 1/3), 0, 0}, {0, 2u/3, 1 - 2u*(L - 2/3), u*(L - 2/3), 0}, {0, 0, 3u/3, 1 - 3u*(L - 3/3), u*(L - 3/3)}, {0, 0, 0, 0, 1}};
```

Note that we have defined the kernel for five fitness classes as that is the number of classes that we will deal with in this example

Importing data

Next we import the sequence frequencies from the real data. The format of the imported data is in the form of a table with three columns, each referring to

1. Organism name
2. Sequence Class
3. Sequence Frequency

```
realFreqs = Import[NotebookDirectory[] <> "File_S8.txt", "Table"];
realTab = Table[Table[realFreqs[[i + j]][[3]], {j, 0, mutclasses - 1, 1}],
{i, 1, Length[realFreqs], mutclasses}];
(*organism names*)
names = Table[realFreqs[[i]][[1]], {i, 1, Length[realFreqs], mutclasses}];
seqLength = Table[realFreqs[[i]][[4]], {i, 1, Length[realFreqs], mutclasses}];
names = StringReplace[names, "_largestCluster" → ""];
(*calculate relative frequencies from absolute frequencies*)
freqs = Table[realTab[[i]]/Total[realTab[[i]]], {i, 1, Length[realTab]}];
```

Model

Solving for the equilibrium frequencies of the classes given by x_0, x_1, x_2, x_3 and x_4 .

The fitness of the five classes are given by f_0, f_1, f_2, f_3 and f_4 where we set $f_4=1$.

```

outputUnscaled = {};
For[i = 1, i ≤ Length[freqs], i++,

(*equilibrium frequencies*)
x0 = .;
x1 = .;
x2 = .;
x3 = .;

x4 = 1 - x0 - x1 - x2 - x3;

q = mutmatrix[u, seqLength[[i]]];

x0 = freqs[[i]][[1]];
x1 = freqs[[i]][[2]];
x2 = freqs[[i]][[3]];
x3 = freqs[[i]][[4]];

fitnessTable = CalculateFitness[{x0, x1, x2, x3, x4}, q];
fitnessTable = Append[fitnessTable, seqLength[[i]]];
outputUnscaled = Append[outputUnscaled, fitnessTable];

]

```

Exporting raw output and scaled with mutation rate

```

Export[NotebookDirectory[] <> "quasispecies_fitness_unscaled.txt",
      outputUnscaled, "Table"];

```

What does scaling do?

If we want to determine the fitness values that give us comparable results but for a higher mutation rate then we need to do the following:

1. Calculate the number of generations one time step at the new mutation rate corresponds to under the old mutation rate.

```

(*scale results for a mutation rate of 10^-4*)
newU = 10^-4;
gens = newU / u;
(*calculate new fitness values*)

```

2. Scale up each fitness value by the number of generations we calculated above and calculate the new equilibrium frequencies for the new mutation rate.

```

output = outputUnscaled;
For[i = 1, i <= Length[freqs], i++,
  fitness = {};
  For[j = 1, j <= mutclasses, j++,
    newFit = output[[i]][[j]]^gens;
    output[[i]][[j]] = newFit;
    fitness = Append[fitness, newFit];
  ];
  output[[i]][[mutclasses + 1]] = newU // N;
(*calculate new equilibrium frequencies for the new fitness values*)

eqfreqs =
  CalculateEquilibriumFrequencies[fitness, mutmatrix[newU, seqLength[[i]]]];

For[j = 1, j <= mutclasses, j++,
  k = mutclasses + 1;

  output[[i]][[k + j]] = eqfreqs[[j]];
];
]
(*output the scaled fitness values and equilibrium frequencies*)
Export[NotebookDirectory[] <> "quasispecies_fitness.txt", output, "Table"];

```

Calculate error thresholds

For each species we will calculate the fitness value, at which the frequency of the master sequence drops below 1% by depressing the whole fitness landscape in increments of 10^{-10} .

```

error = outputUnscaled;
For[i = 1, i ≤ Length[freqs], i++,

(*equilibrium frequencies*)

q = mutmatrix[u, seqLength[[i]]];
f0 = outputUnscaled[[i]][[1]];
f1 = outputUnscaled[[i]][[2]];
f2 = outputUnscaled[[i]][[3]];
f3 = outputUnscaled[[i]][[4]];
f4 = outputUnscaled[[i]][[5]];
start = IntegerPart[(f0 - 1) 1012];
threshold = 0.01;
For[j = 1, j < start, j++,
  f1int = IntegerPart[(f1 - 1) 1012];
  f2int = IntegerPart[(f2 - 1) 1012];
  f3int = IntegerPart[(f3 - 1) 1012];
  f4int = IntegerPart[(f4 - 1) 1012];
  If[f1int > 0, f1 = 1 + (f1int - 1) 10-12];

```

```
If[f2int > 0, f2 = 1 + (f2int - 1) 10-12];
If[f3int > 0, f3 = 1 + (f3int - 1) 10-12];
If[f4int > 0, f4 = 1 + (f4int - 1) 10-12];
f0 = 1 + (start - j) 10-12;
fitness = {f0, f1, f2, f3, f4};

effreqs = CalculateEquilibriumFrequencies[fitness, q];

If[effreqs[[1]] < threshold,
  j = start;
]

];

error[[i]][[1]] = f0 // N;
error[[i]][[2]] = f1 // N;
error[[i]][[3]] = f2 // N;
error[[i]][[4]] = f3 // N;
error[[i]][[5]] = f4 // N;
For[j = 1, j <= mutclasses, j++,
  k = mutclasses + 1;

  error[[i]][[k + j]] = effreqs[[j]];
];
]

Export[NotebookDirectory[] <> "error_threshold.txt", error, "Table"];
```