README

MCMC DFE: ReadMe

Sebastian Matuszewski, Jeffrey D. Jensen and Claudia Bank

Lausanne, June 16, 2015

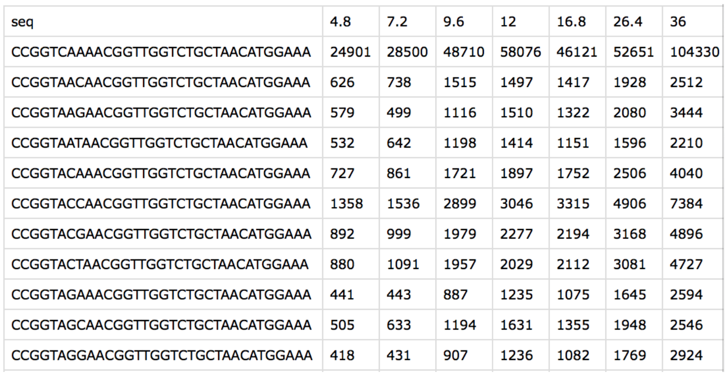
Contents

1 Introduction

This program serves as an extension of the Bayesian Monte Carlo Markov chain (MCMC) method described in Bank et al. (2014) for estimating selection coefficients (growth rates) from engineered-mutation-driven experimental evolution data. These data are based on methods – such as EMPIRIC – in which specific mutations are introduced and compared against each other and the wild-type. All mutants (and the wild-type) are assumed to have evolved together in bulk culture for a number of generations with samples taken throughout the course of the experiment. Growth rates – and thus the selection coefficient of each mutation – can then be estimated from the number of reads obtained from deep sequencing. The motivation for this MCMC–DFE software package is to provide an integrative framework for the analysis of experimental evolution data, and includes separate programs for processing raw sequence data and correcting for sequencing errors, obtaining statistically meaningful estimates of selection coefficients in a fast and efficient manner, and for providing ready-to-use summary statistics of the MCMC analysis and its associated parameter estimates. It is highly advisable to read the two accompanying papers by Bank et al. (2014) and Bank et al. (2014) (ADJUST) that describe the methods and underlying assumptions in more detail before using the software.

2 Input data

The MCMC program only takes files in csv-format (comma-separated values) as input, which, furthermore, need to match a specified format (e.g., line endings need to be UNIX specific, input data needs to be ordered in a specified way). However, as part of the MCMC software package, we provide a python script – MCMC–DFE–Input.py – that adjusts the raw data to match the specific input format needed for the MCMC simulation program (including other options discussed in more detail below). A minimal example of the raw data that is required to generate the MCMC input file is depicted in Figure 1. Note that the raw data itself also needs to be csv-formatted with column entries separated by a comma (’,’), a semicolon (’;’) or a tab (’
t’). Furthermore, the raw data file needs to have (exactly) one column either called ’sequence’, ’Sequence’, ’seq’ or ’Seq’ with at least two rows (the wild-type reference and a mutant) and at least additional 3 columns (i.e., time points) corresponding to the number of sequencing reads, with respectively named header cells.



seq	4.8	7.2	9.6	12	16.8	26.4	36
CCGGTCAAAACGGTTGGTCTGCTAACATGGAAA	24901	28500	48710	58076	46121	52651	104330
CCGGTAACAACGGTTGGTCTGCTAACATGGAAA	626	738	1515	1497	1417	1928	2512
CCGGTAAGAACGGTTGGTCTGCTAACATGGAAA	579	499	1116	1510	1322	2080	3444
CCGGTAATAACGGTTGGTCTGCTAACATGGAAA	532	642	1198	1414	1151	1596	2210
CCGGTACAAACGGTTGGTCTGCTAACATGGAAA	727	861	1721	1897	1752	2506	4040
CCGGTACCAACGGTTGGTCTGCTAACATGGAAA	1358	1536	2899	3046	3315	4906	7384
CCGGTACGAACGGTTGGTCTGCTAACATGGAAA	892	999	1979	2277	2194	3168	4896
CCGGTACTAACGGTTGGTCTGCTAACATGGAAA	880	1091	1957	2029	2112	3081	4727
CCGGTAGAAACGGTTGGTCTGCTAACATGGAAA	441	443	887	1235	1075	1645	2594
CCGGTAGCAACGGTTGGTCTGCTAACATGGAAA	505	633	1194	1631	1355	1948	2546
CCGGTAGGAACGGTTGGTCTGCTAACATGGAAA	418	431	907	1236	1082	1769	2924

Figure 1 – Schematic illustration of the minimal data needed to run the MCMC program.

2.1 MCMC–DFE–Input.py

The program MCMC–DFE–Input.py is written in Python (2.7) and serves as a link between the (raw) time-sampled sequence data (e.g., obtained from EMPIRIC) and the MCMC–DFE simulation program for the estimation of mutant growth rates 'r'. While it primarily ensures that the input data matches the input format required by the MCMC simulation program, it comes with additional options that will be detailed here.

The general usage is as follows: After opening a command-line interface (e.g., Shell, Terminal) and navigating to the location of the MCMC–DFE–Input.py file, the program can be executed by typing

```
python MCMC–DFE–Input.py [options] .
```

Note that this requires that the 'PATHVARIABLE' for Python has been set correctly on your system. Please consult the online Python documentation for further details (https://docs.python.org/2/). Without specifying any options the program will exit with an error and provide a short documentation on its usage, as it requires the name of a (raw) data input file and the start of the sequencing read data table to be specified (by invoking the '-f' and '-s' option, respectively). All options and their usage are given in Table 1.

Table 1 – A summary of the options of the MCMC–DFE–Input program.

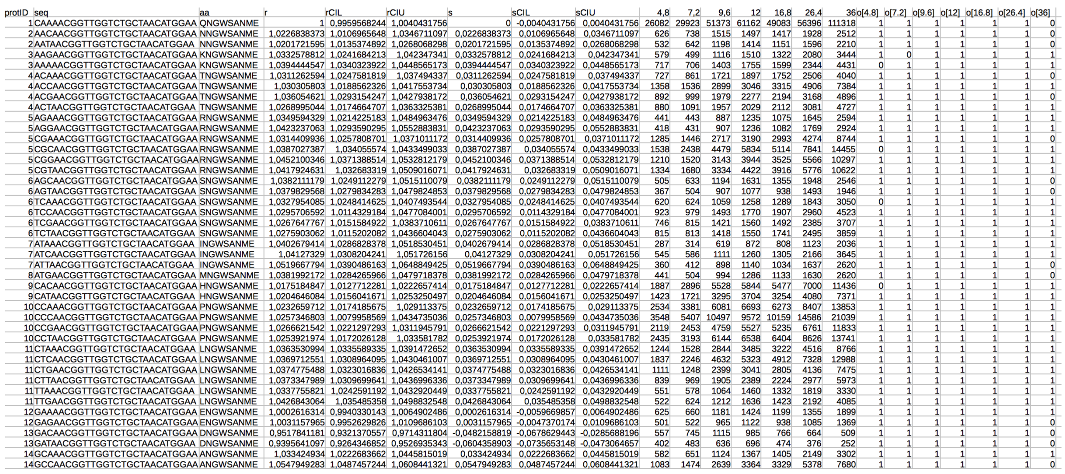
Short/Long option	Accepted values	Description
-h, –help	none	When the '-h'-option is invoked, a short documentation on the usage of the program is shown. Note that, if this option is invoked, the python program is not executed.

-f, –file=	string	The '-f'-option is a mandatory option, which passes the name of the (raw) data input file (csv formatted) to the python program. Files created by the python program will take the name of the input file and add option-dependent specific file identifiers.
-s, –skipcol=	integer	The '-s'-option is a mandatory option, which takes an integer value corresponding to the number of descriptive columns that precede the actual 'data matrix' of sequencing read counts. For example, for the raw data depicted in Figure 1, the user would have to pass '-s 1' (or equivalently '–skipcol=1'). Please note that the data matrix must always span all remaining columns.
-o, –outlier=	'detect' or 'impute'	When the '-o'-option is invoked, the python program will perform an outlier analysis. If '-o detect' (or equivalently '–outlier=detect'), the python program performs a log-linear regression analysis for all mutants. Data points are then classified as outliers on the basis of the DFBeta statistic with a cut-off value of 2 (with data points surpassing this cut-off regarded as outliers). For more details please consult Bank et al. (2014). If '-o impute' (or equivalently '–outlier=impute'), the python program performs a log-linear regression analysis for all mutants. Data points are then classified as outliers on the basis of the DFBeta statistic with a cut-off value of 2 and their studentized residuals with a cut-off value of 3 (with data points surpassing *both* cut-offs are regarded as outliers) and imputed as described in Bank et al. (2014) (ADJUST). Furthermore, an additional output file – whose name consists of 'ImputedData' along with the input file name – is produced that lists all imputed data points. In particular, the output is a simple matrix where rows denote different mutants and columns correspond to the different time points. An entry of '1' indicates that the data has *not* been imputed; an entry of '0' indicates that the data point has been imputed.
-l, –leadseq=	integer	When the '-l'-option is invoked, the first 'integer' characters, that precede the original mutant sequence (e.g., sites that function as DNA barcode or sequence tag), are removed.
-t, –trailseq=	integer	When the '-l'-option is invoked, the first 'integer' characters, that trail the original mutant sequence (e.g., sites that function as DNA barcode or sequence tag), are removed.
-p, –pool	none	When the '-p'-option is invoked, the DNA-sequences (characterizing the different mutants) are translated into amino acids. The data is then pooled based on their amino acid sequence, assuming that identical amino acid sequences, though differing in their DNA sequence (synonymous mutants), have the same growth rate (but different initial population sizes). Note that even if the '-p'-option is not invoked, data is pooled based on the sequence name (which can be any string and not only letters from the DNA alphabet).

-g, –group=	integer	When the '-g'-option is invoked, the data is grouped into subsets of mutants each of minimal size 'integer'. This results in more data sets with less mutants, such that the per-data set computation time is reduced, without affecting parameter estimates or the shape of the log-likelihood surface (compared to analysis of the full data set). The program also ensures, that mutants with identical mutant or protein ID (i.e., mutants that have an identical DNA- or amino acid sequence) remain in the same data sub-set as they are assumed to evolve under an identical growth rate (r). In general, the last line of the resulting input file serves as a summary line (indicated by a '-1' as mutant/protein ID and by 'XXX' in the sequence column) that gives the number of all sequencing reads not in the current sub-set (but which are still needed to be accounted for). Estimates for the growth rates and the selection coefficients for the summary line are not calculated (but simply equated to those of the reference strain), since MCMC simulation estimates for this 'summary mutant' will later be discarded. The name of the output file (i.e., the MCMC input file) is composed of the standard output file identifier 'MCMCInput', the grouping identifier (a consecutive number of the sub data sets created) and the name of the input file.
-i, –initialize	none	When the '-i'-option is invoked, and additional input file is created that specifies the initial growth rates (r) and initial population sizes (c) for all mutants based on he log-linear regression. Note that this could potentially bias the MCMC algorithm, since the initial starting point of the Markov chain could be trapped in a local log-likelihood optimum. Often, however, the median of the growth rates and the initial population sizes from the MCMC-DFE simulations are close enough to the corresponding estimates from the log-linear regression such that starting at these values could shorten the burn-in period and, thus, reduce the run time. For mutants with identical DNA or amino acid sequence, the mean initial population size is calculated (from these mutants) and taken as the starting value for the MCMC simulation program. Given the estimated mean initial population size, the growth rate is estimated from the log-linear regression. The name of the output file is composed of the standard output file identifier 'MCMCInput', an optional grouping identifier (see '-g'-option), the name of the input file, and an initialization file identifier '_inputRC.txt'.

An illustration of the output file produced by the python program (i.e., the input file for the MCMC simulation program) is depicted in Figure 2.

Depending on the invoked options, the name of the MCMC simulation input file that is produced by the python program is given by the standard output file identifier 'MCMCInput', an optional grouping identifier (see '-g'-option) and the name of the raw data file.



protID	seq	aa	r	rCIL	rCIU	s	sCIL	sCIU	4,8	7,2	9,6	12	16,8	26,4	36	o[4.8]	o[7.2]	o[9.6]	o[12]	o[16.8]	o[26.4]	o[36]
1	CAAAACGGTTGGTCTGCTAACATGGAA	QNGWSANME	1	0,9959568244	1,0040431756	0	-0,0040431756	0,0040431756	26082	29923	51373	61162	49083	56396	111318	1	1	1	1	1	1	0
2	AACAACGGTTGGTCTGCTAACATGGAA	NNGWSANME	1,0226838373	1,0106965648	1,0346711097	0,0226838373	0,0106965648	0,0346711097	626	738	1515	1497	1417	1928	2512	1	1	1	1	1	1	0
2	AATAACGGTTGGTCTGCTAACATGGAA	NNGWSANME	1,0201721595	1,0135374892	1,0268068298	0,0201721595	0,0135374892	0,0268068298	532	642	1198	1414	1151	1596	2210	1	1	1	1	1	1	0
3	AAGAACGGTTGGTCTGCTAACATGGAA	KNGWSANME	1,0332578812	1,0241684213	1,042347341	0,0332578812	0,0241684213	0,042347341	579	499	1116	1510	1322	2080	3444	1	0	1	1	1	1	1
3	AAAAACGGTTGGTCTGCTAACATGGAA	KNGWSANME	1,0394444547	1,0340323922	1,0448565173	0,0394444547	0,0340323922	0,0448565173	717	706	1403	1755	1599	2344	4431	0	1	1	1	1	1	1
4	ACAAACGGTTGGTCTGCTAACATGGAA	TNGWSANME	1,0311262594	1,0247581819	1,037494337	0,0311262594	0,0247581819	0,037494337	727	861	1721	1897	1752	2506	4040	1	1	1	1	1	1	0
4	ACCAACGGTTGGTCTGCTAACATGGAA	TNGWSANME	1,030305803	1,0188562326	1,0417553734	0,030305803	0,0188562326	0,0417553734	1358	1536	2899	3046	3315	4906	7384	1	1	1	1	1	1	1
4	ACGAACGGTTGGTCTGCTAACATGGAA	TNGWSANME	1,036054621	1,0293154247	1,0427938172	0,036054621	0,0293154247	0,0427938172	892	999	1979	2277	2194	3168	4896	1	1	1	1	1	1	0
4	ACTAACGGTTGGTCTGCTAACATGGAA	TNGWSANME	1,0268995044	1,0174664707	1,0363325381	0,0268995044	0,0174664707	0,0363325381	880	1091	1957	2029	2112	3081	4727	1	1	1	1	1	1	1
5	AGAAACGGTTGGTCTGCTAACATGGAA	RNGWSANME	1,0349594329	1,0214225183	1,0484963476	0,0349594329	0,0214225183	0,0484963476	441	443	887	1235	1075	1645	2594	1	1	1	1	1	1	1
5	AGGAACGGTTGGTCTGCTAACATGGAA	RNGWSANME	1,0423237063	1,0293590295	1,0552883831	0,0423237063	0,0293590295	0,0552883831	418	431	907	1236	1082	1769	2924	1	1	1	1	1	1	1
5	CGAAACGGTTGGTCTGCTAACATGGAA	RNGWSANME	1,0314409936	1,0257808701	1,0371011172	0,0314409936	0,0257808701	0,0371011172	1285	1446	2717	3190	2993	4274	8744	1	1	1	1	1	1	0
5	CGCAACGGTTGGTCTGCTAACATGGAA	RNGWSANME	1,0387027387	1,034055574	1,0433499033	0,0387027387	0,034055574	0,0433499033	1538	2438	4479	5834	5114	7841	14455	0	1	1	1	1	1	1
5	CGGAACGGTTGGTCTGCTAACATGGAA	RNGWSANME	1,0452100346	1,0371388514	1,0532812179	0,0452100346	0,0371388514	0,0532812179	1210	1520	3143	3944	3525	5566	10297	1	1	1	1	1	1	1
5	CGTAACGGTTGGTCTGCTAACATGGAA	RNGWSANME	1,0417924631	1,032683319	1,0509016071	0,0417924631	0,032683319	0,0509016071	1334	1680	3334	4422	3916	5776	10622	1	1	1	1	1	1	1
6	AGCAACGGTTGGTCTGCTAACATGGAA	SNGWSANME	1,0382111179	1,0249112279	1,0515110079	0,0382111179	0,0249112279	0,0515110079	505	633	1194	1631	1355	1948	2546	1	1	1	1	1	1	0
6	AGTAACGGTTGGTCTGCTAACATGGAA	SNGWSANME	1,0379829568	1,0279834283	1,0479824853	0,0379829568	0,0279834283	0,0479824853	367	504	907	1077	938	1493	1946	1	1	1	1	1	1	0
6	TCAAACGGTTGGTCTGCTAACATGGAA	SNGWSANME	1,0327954085	1,0248414625	1,0407493544	0,0327954085	0,0248414625	0,0407493544	620	624	1059	1258	1289	1843	3050	0	1	1	1	1	1	1
6	TCCAACGGTTGGTCTGCTAACATGGAA	SNGWSANME	1,0295706592	1,0114329184	1,0477084001	0,0295706592	0,0114329184	0,0477084001	923	979	1493	1770	1907	2960	4523	1	1	1	1	1	1	1
6	TCGAACGGTTGGTCTGCTAACATGGAA	SNGWSANME	1,0267647767	1,0151584922	1,0383710611	0,0267647767	0,0151584922	0,0383710611	746	815	1421	1560	1492	2385	3707	1	1	1	1	1	1	1
6	TCTAACGGTTGGTCTGCTAACATGGAA	SNGWSANME	1,0275903062	1,0115202082	1,0436604043	0,0275903062	0,0115202082	0,0436604043	815	813	1418	1550	1741	2495	3859	1	1	1	1	1	1	1
7	ATAAACGGTTGGTCTGCTAACATGGAA	INGWSANME	1,0402679414	1,0286828378	1,0518530451	0,0402679414	0,0286828378	0,0518530451	287	314	619	872	808	1123	2036	1	1	1	1	1	1	1
7	ATCAACGGTTGGTCTGCTAACATGGAA	INGWSANME	1,04127329	1,0308204241	1,051726156	0,04127329	0,0308204241	0,051726156	545	586	1111	1260	1305	2166	3645	1	1	1	1	1	1	1
7	ATTAACGGTTGGTCTGCTAACATGGAA	INGWSANME	1,0519667794	1,0390486163	1,0648849425	0,0519667794	0,0390486163	0,0648849425	360	412	898	1140	1034	1637	2620	1	1	1	1	1	1	0
8	ATGAACGGTTGGTCTGCTAACATGGAA	MNGWSANME	1,0381992172	1,0284265966	1,0479718378	0,0381992172	0,0284265966	0,0479718378	441	504	994	1286	1133	1630	2620	1	1	1	1	1	1	0
9	CACAACGGTTGGTCTGCTAACATGGAA	HNGWSANME	1,0175184847	1,0127712281	1,0222657414	0,0175184847	0,0127712281	0,0222657414	1887	2896	5528	5844	5477	7000	11436	0	1	1	1	1	1	1
9	CATAACGGTTGGTCTGCTAACATGGAA	HNGWSANME	1,0204646084	1,0156041671	1,0253250497	0,0204646084	0,0156041671	0,0253250497	1423	1721	3295	3704	3254	4080	7371	1	1	1	1	1	1	1
10	CCAAACGGTTGGTCTGCTAACATGGAA	PNGWSANME	1,0232659712	1,0174185675	1,029113375	0,0232659712	0,0174185675	0,029113375	2534	3381	6081	6693	6273	8407	13853	1	1	1	1	1	1	1
10	CCCAACGGTTGGTCTGCTAACATGGAA	PNGWSANME	1,0257346803	1,0079958569	1,0434735036	0,0257346803	0,0079958569	0,0434735036	3548	5407	10497	9572	10159	14586	21039	1	1	1	1	1	1	1
10	CCGAACGGTTGGTCTGCTAACATGGAA	PNGWSANME	1,0266621542	1,0221297293	1,0311945791	0,0266621542	0,0221297293	0,0311945791	2119	2453	4759	5527	5235	6761	11833	1	1	1	1	1	1	1
10	CCTAACGGTTGGTCTGCTAACATGGAA	PNGWSANME	1,0253921974	1,0172026128	1,033581782	0,0253921974	0,0172026128	0,033581782	2435	3193	6144	6538	6404	8626	13741	1	1	1	1	1	1	1
11	CTAAACGGTTGGTCTGCTAACATGGAA	LNGWSANME	1,0363530994	1,0335589335	1,0391472652	0,0363530994	0,0335589335	0,0391472652	1244	1528	2844	3485	3222	4516	8766	1	1	1	1	1	1	1
11	CTCAACGGTTGGTCTGCTAACATGGAA	LNGWSANME	1,0369712551	1,0308964095	1,0430461007	0,0369712551	0,0308964095	0,0430461007	1837	2246	4632	5323	4912	7328	12988	1	1	1	1	1	1	1
11	CTGAACGGTTGGTCTGCTAACATGGAA	LNGWSANME	1,0374775488	1,0323016836	1,0426534141	0,0374775488	0,0323016836	0,0426534141	1111	1248	2399	3041	2805	4136	7475	1	1	1	1	1	1	1
11	CTTAACGGTTGGTCTGCTAACATGGAA	LNGWSANME	1,0373347989	1,0309699641	1,0436996336	0,0373347989	0,0309699641	0,0436996336	839	969	1905	2389	2224	2977	5973	1	1	1	1	1	1	1
11	TTAAACGGTTGGTCTGCTAACATGGAA	LNGWSANME	1,0337755821	1,0242591192	1,0432920449	0,0337755821	0,0242591192	0,0432920449	551	578	1064	1460	1332	1819	3330	1	1	1	1	1	1	1
11	TTGAACGGTTGGTCTGCTAACATGGAA	LNGWSANME	1,0426843064	1,035485358	1,0498832548	0,0426843064	0,035485358	0,0498832548	522	624	1212	1636	1423	2192	4085	1	1	1	1	1	1	1
12	GAAAACGGTTGGTCTGCTAACATGGAA	ENGWSANME	1,0002616314	0,9940330143	1,0064902486	0,0002616314	-0,0059669857	0,0064902486	625	660	1181	1424	1199	1355	1899	1	1	1	1	1	1	1
12	GAGAACGGTTGGTCTGCTAACATGGAA	ENGWSANME	1,0031157965	0,9952629826	1,0109686103	0,0031157965	-0,0047370174	0,0109686103	501	522	965	1122	938	1085	1369	1	1	1	1	1	1	0
13	GACAACGGTTGGTCTGCTAACATGGAA	DNGWSANME	0,9517841181	0,9321370557	0,9714311804	-0,0482158819	-0,0678629443	-0,0285688196	557	745	1115	985	766	664	509	1	1	1	1	1	1	0
13	GATAACGGTTGGTCTGCTAACATGGAA	DNGWSANME	0,9395641097	0,9264346852	0,9526935343	-0,0604358903	-0,0735653148	-0,0473064657	402	483	636	696	474	376	252	1	1	1	1	1	1	0
14	GCAAACGGTTGGTCTGCTAACATGGAA	ANGWSANME	1,033424934	1,0222683662	1,0445815019	0,033424934	0,0222683662	0,0445815019	582	651	1124	1367	1405	2149	3302	1	1	1	1	1	1	1
14	GCCAACGGTTGGTCTGCTAACATGGAA	ANGWSANME	1,0547949283	1,0487457244	1,0608441321	0,0547949283	0,0487457244	0,0608441321	1083	1474	2639	3364	3329	5378	7680	1	1	1	1	1	1	0

Figure 2 – Schematic illustration of the output produced by the python program and which serves as input for the MCMC simulation program. This input data has been created using the minimal raw data shown in Fig. 1, where the first five bases and the last base have been discarded (barcodes; '-l' and '-t' option). DNA sequences ('seq' column) have been translated to amino acids ('aa' column) and pooled ('-p' option), such that identical amino acid sequences have the same protein ID ('protID' column). Estimates of the growth rates 'r' ('r' column) and the selection coefficient 's' ('s' column) along with the 95%-confidence intervals are based on the log-linear regression (where 'rCIL' and 'rCIU' give the lower and upper boundary of the confidence interval for the growth rate r, respectively. Notation is analogous for the selection coefficient 's'.). Please note, that while the MCMC simulation program assumes that mutants with identical sequence information (i.e., for sequences with identical 'protID') evolve at the same growth rate, log-linear estimates for 'r' and 's' are based on individual mutants. The columns '4.8', '7.2', '9.6', '12', '16.8', and '26.4' give the number of sequencing reads obtained from sampling at these time points for each mutant (row). If the '-o detect'-option is invoked, the matrix of sequencing reads is followed by an outlier matrix for the corresponding time points and mutants, where '0' indicate data points that where classified as outliers.

3 Usage

We provide executables for (Mac OS X), Windows and Linux. The C++ source code along with a system specific makefile are provided under a GNU General Public License as published by the Free Software Foundation. If you do not need to compile the program yourself you can skip the next subsection.

3.1 Compilation

Note that compilation requires that the (Gnu Scientific Library) (gsl-library) is installed on your system. Information on how to install the gsl-library can be found under `http://www.gnu.org/software/gsl/`. On Windows the easiest way to obtain the gsl-library is to install Cygwin (`http://www.cygwin.com/`) including the developers (all) packages. Alternatively, MinGW (`http://www.mingw.org/`) provides a "minimalist GNU for Windows" development environment. Under MinGW though, the gsl-library needs to be installed independently. A short instruction is given in the following paragraph.

Compilation of the program has successfully been tested on MacOSX (10.9.5) using 'clang' (version 6.0) and 'gcc' (version 4.9), on Ubuntu (14.04 LTS) using 'g++' (version 4.8.2), and on Windows (8.1) using 'g++' (version 4.8.1 under MinGW; version 4.9.2 under Cygwin).

Windows

Cygwin Please note that Cygwin is a large collection of GNU and open source tools, which provide functionality similar to a Linux distribution on Windows. Thus, when using Cygwin, commands and folder navigation are different to those from a native Windows Shell (e.g., Powershell). For example, under Cygwin you first need to type

```
cd /cygdrive/c
```

to navigate to the drive 'C:\'.

The easiest way to compile the program is by using the provided makefile. After navigating to the folder where the source code is stored, simply type

```
make -f Makefile_MCMC_DFE_Cygwin .
```

Please note that you might want to adjust the makefile, in particular, to change the name of the executable which is by default set to 'MCMC_DFE_Run_Cygwin.exe'.

Alternatively, you can manually compile the program from source by navigating to the folder where the source code is stored and by typing

```
g++ MCMC-DFE.cpp -lgsl -lgslcblas -O3 -o 'NameOfProgram' ,
```

where 'NameOfProgram' should be replaced by the actual name of the compiled program (e.g., MCMC_DFE_Run).

Note that Cygwin does not allow to link libraries statically.

MinGW Please note that MinGW provides a complete open source programming tool set for the development of native Windows applications (including both different compilers and a "minimal system" bourne shell command line interpreter system MSYS).

Compilation and installation of the gsl-library from source Before compilation and installation of the gsl-library please read and follow the installation instructions provided with the most recent version of gsl.

1. Download the latest version of gsl from `http://ftpmirror.gnu.org/gsl/`
2. Navigate to the place where the downloaded tar archive is stored and unpack it by typing

   ```
   tar -zxvf gsl-x.xx.tar.gz ,
   ```

   where 'x.xx' should be replaced by the version number.
3. Navigate to 'gsl-x.xx/' and carefully read the 'INSTALL' document and follow the instructions to configure, make and install the gsl-library:

   ```
   ./configure
   make
   make install
   ```

4. GSL binaries, headers and library files are installed automatically in the 'bin/', 'include/gsl/', and 'lib/' subdirectories (if not specified otherwise; in that case you would also need to adjust the linker and compiler flags in the makefile).

Compilation of the MCMC Open a MSYS shell and type

```
cd /c
```

to navigate to the drive 'C:\'.

The easiest way to compile the program is by using the provided makefile. After navigating to the folder where the source code is stored simply type

```
make -f Makefile-MCMC-DFE-WINDOWS .
```

Please note that you might want to adjust the makefile, in particular, to change the name of the executable which is by default set to 'MCMC_DFE_Run.exe'.

Alternatively, you can manually compile the program from source by navigating to the folder where the source code is stored and by typing

```
g++ MCMC-DFE.cpp -lgsl -lgslcblas -O3 -o 'NameOfProgram' ,
```

where 'NameOfProgram' should be replaced by the actual name of the compiled program (e.g., MCMC_DFE_Run).

Linux

The easiest way to compile the program is by using the provided makefile. Use the Shell to navigate to the folder where the source code is stored and simply type

```
make -f Makefile-MCMC-DFE-Linux .
```

Please note that you might want to adjust the makefile, in particular, to change the name of the executable which is by default set to 'MCMC_DFE_Run'.

Alternatively, you can manually compile the program from source by navigating to the folder where the source code is stored and by typing

```
g++ MCMC-DFE.cpp -lgsl -lgslcblas -O3 -o 'NameOfProgram' ,
```

where 'NameOfProgram' should be replaced by the actual name of the compiled program (e.g., MCMC_DFE_Run).

Mac OS X

The easiest way to compile the program is by using the provided makefile. Use the Terminal to navigate to the folder where the source code is stored and simply type

```
make -f Makefile-MCMC-DFE-MacOSX .
```

Please note that you might want to adjust the makefile, in particular, to change the name of the executable which is by default set to 'MCMC_DFE_Run'.

Alternatively, you can manually compile the program from source by navigating to the folder where the source code is stored and by typing

```
g++ MCMC-DFE.cpp -lgsl -lgslcblas -O3 -o 'NameOfProgram' ,
```

where 'NameOfProgram' should be replaced by the actual name of the compiled program (e.g., MCMC_DFE_Run).

3.2 Execution

Note that the 'MCMC_DFE' program is a command line program which is run from a command-line interface (e.g., Shell, Terminal, Powershell), with arguments and parameters being passed over the command line. An overview and description of the parameters can be found in Table 2. Depending on your operating system there are different ways to call the program. These will be explained in the subsequent paragraphs.

Windows

Cygwin There are two ways to execute the program. First, the easiest way is to use the provided bash script 'RunMCMC_Script_Cygwin.sh', where the parameters (e.g., 'PathToDataFile', skipCol, burnin) can be specified by the user. To begin, navigate to the folder where the executable and the bash script are stored and type

```
./RunMCMC-Script-Cygwin.sh .
```

Second, you can manually execute the program by using Cygwin to navigate to the folder where the executable is stored and typing

```
./'NameOfProgram' 'PathToDatafile' outfileName skipCol
    outliersPresent burnin subsampling noSets set
    terminateESS proposalDistCScale proposalDistRSD
    initialRC printLogLTS printESS printOutput flushcontrol
    seed ,
```

where the parameters (e.g., 'PathToDataFile', skipCol, burnin) need to be specified by the user.

Note that under Cygwin the 'PathToDataFile' needs to be preceded by '/cygdrive/c/' (if the datafile is stored on the 'c' drive).

Powershell There are two ways to execute the program. First, the easiest way is to use the provided powershell script 'RunMCMC_Script_Windows.ps1', where the parameters (e.g., 'PathToDataFile', skipCol, burnin) can be specified by the user. To begin, open Powershell and navigate to the folder where the executable and the powershell script are stored and type

```
& RunMCMC-Script-Windows.ps1 .
```

Second, you can manually execute the program by using Powershell to navigate to the folder where the executable is stored and typing

```
& 'NameOfProgram' 'PathToDatafile' outfileName skipCol
    outliersPresent burnin subsampling noSets set
    terminateESS proposalDistCScale proposalDistRSD
    initialRC printLogLTS printESS printOutput flushcontrol
    seed ,
```

where the parameters (e.g., 'PathToDataFile', skipCol, burnin) need to be specified by the user.

Linux

There are two ways to execute the program. First, the easiest way is to use the provided bash script 'RunMCMC_Script_Linux.sh', where the parameters (e.g., 'PathToDataFile', skipCol, burnin) can be specified by the user. To begin, navigate to the folder where the executable and the bash script are store and type

```
./RunMCMC-Script-Linux.sh .
```

Second, you can manually execute the program by using the shell to navigate to the folder where the executable is stored and typing

```
./'NameOfProgram' 'PathToDatafile' outfileName skipCol
    outliersPresent burnin subsampling noSets set
    terminateESS proposalDistCScale proposalDistRSD
    initialRC printLogLTS printESS printOutput flushcontrol
    seed ,
```

where the parameters (e.g., 'PathToDataFile', skipCol, burnin) need to be specified by the user.

Mac OS X

There are two ways to execute the program. First, the easiest way is to use the provided bash script 'RunMCMC_Script_MacOSX.sh', where the parameters (e.g., 'PathToDataFile', skipCol, burnin) can be specified by the user. To begin, navigate to the folder where the executable and the bash script are store and type

```
./RunMCMC_Script_MacOSX.sh .
```

Second, you can manually execute the program by using the terminal to navigate to the folder where the executable is stored and typing

```
./'NameOfProgram' 'PathToDatafile' outfileName skipCol
    outliersPresent burnin subsampling noSets set
    terminateESS proposalDistCScale proposalDistRSD
    initialRC printLogLTS printESS printOutput flushcontrol
    seed ,
```

where the parameters (e.g., 'PathToDataFile', skipCol, burnin) need to be specified by the user.

Table 2 – A summary of the parameters of the MCMC program.

Category **Parameter**	**Accepted values**	**Description**
Data		
datafile	string	Give the full path to the datafile (e.g., /users/me/PathToData/reads.csv).
outfileName	string	Name of the output file. The program will take the name of the input file and add the 'outfileName', the time of execution and the identifier (e.g., '_C'). This produces for example /PathToData/'datafile'_'outfileName'_'date&time'_C.txt.
skipCol	integer, $ ≥0	Number of columns to skip in data file before read numbers start.
outliersPresent	bool	If '0' there is no outlier matrix in the data file. If '1' there is an outlier matrix in the data file.
MCMC		
burnin	integer, $ ≥ 0	Number of accepted values that are discarded (burn-in period). During the burn-in period the parameters of the proposal distribution are adjusted.
subSampling	integer, $ ≥ 0	After the burn-in period only every 'subSampling' accepted value is recorded (i.e., written to file).
noSets	integer, $ 0	Number of output data sets that are recorded each of size size 'setSize'. The total chain length is given by 'burnin'+'noSets'×'setSize'×'subSampling'.

setSize	integer, $ 0	Number of recorded samples per set. The total chain length is given by 'burnin'+'noSets'×'setSize'×'subSampling'. Output to file is written every 'setSize' accepted and recorded samples.
proposalDistCScale	double, $ > 0	Scale parameter of the proposal distribution of initial population sizes 'c' (Cauchy distribution) .
proposalDistRSD	double, $ > 0	Standard deviation of the proposal distribution of growth rates 'r' (Normal distribution) .
inititalRC	string	An alternative way to initialize the growth rates 'r', the initial population sizes 'c' and to (optionally) set the parameters of the proposal distributions. This option allows, for example, to continue an MCMC run that has not been run long enough from the previous accepted sample. Note that for continuing an MCMC run the burn-in has to be set to 0. If 'initialRC' = −1, the default initialization is used (i.e., r=1 and initial population size = first observed read number for all mutants). When 'initialRC' ≠ −1 the program will search for the initialization file 'datafile'_'initialRC'. In particular if the 'datafile' has been specified as '/users/me/PathToData/reads.csv' and the name of the initialization file is 'reads_initialRC.txt', the initialRC parameter passed to the program should be 'initialRC.txt'. Note that this file needs to be located in the same directory as the datafile and that the name initialization file should be of the form XYZData_'initialRC'.
Output		
printLogLTS	bool	If '0', no time series of log-likelihoods is written. If '1', a time series of log-likelihoods is written.
printESS	bool	If '0', no ESS statistics are written. If '1', ESS statistics are written. The effective sample size (ESS) is calculated every 1000 accepted samples.
printOutput	bool	If '0', no additional output will be written to screen. If '1', additional output will be written to screen. Mainly for inspection purposes.
Random numbers		
seed	integer	Sets the random number seed. If 'seed'≤ 0 a random number seed is created automatically based on computer run time.

4 MCMC output

The program creates different kinds of output files that contain time-series data for different parameters, summary and diagnostic statistics for those parameters and for the entire MCMC run. Note that, even though all output files are plain 'txt'-files, they are all formatted as '.tsv'-files (tab separated) and can be displayed nicely with any spreadsheet application (e.g., Excel). Furthermore, all output files are structured such that they start with a list of the input/parameters for the MCMC run which are followed by the simulation results.

Table 3 – A summary of the output of the MCMC program.

File / Parameter	Description
.*_R	
sample	Consecutive number of samples. Sample ’0’ gives the initial values.
r.*	Sampled value for the growth rate ’r’ for all mutants.
.*_C	
sample	Consecutive number of samples. Sample ’0’ gives the initial values.
c.*	Sampled value for the initial population size ’c’ for all mutants.
.*_logLTS	
sample	Consecutive number of samples. Sample ’0’ gives the initial values.
logL	Log-likelihood for the current sampled values for the initial population size ’c’ and the growth rate ’r’.
.*_R_quantiles	
protID	Protein ID as specified by the input file.
mutant	Consecutive number of mutant identifier ’r.*’.
i%	Values for the i%-quantile of all samples of the growth rate ’r’ for each mutant, where $i = 0, 1, 2.5, 5, 25, 50, 75, 95, 97.5, 99, 100$.
.*_C_quantiles	
protID	Protein ID as specified by the input file.
mutant	Consecutive number of mutant identifier ’c.*’.
i%	Values for the i%-quantile of all samples of the initial population size ’c’ for each mutant, where $i = 0, 1, 2.5, 5, 25, 50, 75, 95, 97.5, 99, 100$.
.*_logL_quantiles	
logL	Log-Likelihood identifier.
i%	Values for the i%-quantile of all samples of the log-likelihood, where $i = 0, 1, 2.5, 5, 25, 50, 75, 95, 97.5, 99, 100$.
.*_ess	
sample	Number of samples after which effective sample size (ESS) is calculated. Note that the ESS is calculated every ’setSize’ samples.
minESS	Minimum effective sample size computed for any parameter of interest (i.e., growth rate ’r’, initial population size ’c’ and log-likelihood). Note that ESS $\leq$ *sample*.
r.*	ESS for growth rate ’r.* for all mutant’.
c.*	ESS for initial population size ’c.*’ for all mutants.
logL	ESS for log-likelihood.
acceptRatio	Overall acceptance ratio. To ensure high efficiency of the MCMC, the width of the proposal distributions should be chosen such that the acceptance ratio is between $0.15 - 0.45$. Performance is maximal with an acceptance ratio around 0.25. During the burn-in period the width of the proposal distributions is automatically tuned – based on the acceptance ratio – such that the acceptance ratio, when recording samples, has close to maximal efficiency. Thus, a sufficiently long burn-in period not only increases the chance that recorded samples are actually taken from the posterior distribution, but also that the width of the proposal distribution is set appropriately.

.*_Diag_R	
protID	Protein ID as specified by the input file.
mutant.*	Consecutive number of mutant identifier 'r.*'.
HD(*)	Hellinger distance (HD) between sets of samples from two probability distributions. Note that HD is bounded by $0 \leq \text{HD} \leq 1$ and can be used to inspect the similarity between two distributions, where HD = 0 corresponds to no divergence and HD = 1 corresponds to no common support between the distributions. The HD can be used to diagnose the MCMC in terms of its burn-in and whether samples obtained at different points of time came (most likely) from the same (posterior) distribution. Note that one cannot determine if the MCMC chain has truly converged, but only if a chain is internally similar. Here, the HD is calculated for up to 10 equally sized sets of consecutive samples from the MCMC simulation. To obtain sufficient statistical power, the HD between two sets of samples is calculated only if each set consisted of at least 1000 samples. If the total number of samples is less than $10 \times 1000 = 10000$, the number of batches is chosen such that the total number of samples is divided into sets of samples of size 1000 each. If the total number of samples exceeds 10000, the number of samples per set is given by the total number of samples divided by 10 (i.e., the maximal number of batches). If the HD between sets of samples is less than 0.1 the distribution of posterior samples shows a high degree of similarity; if $0.1 \leq HD \leq 0.3$ the distribution of posterior samples are still quite similar, but may require closer inspection; if $0.3 \leq HD \leq 0.5$ sets of samples are vaguely similar and should be inspected more closely; a $HD > 0.5$ indicates strong dis-similarity between sets of samples and could be an indicator that all samples that were taken before might not be from the posterior distribution and should be discarded as burn-in. Note that the HD depends on the degree of autocorrelation between samples. Thus, a high HD might not necessarily indicate that samples were obtained from different sampling distributions, but poor mixing (i.e., a low ESS) for the parameter of interest. For details see Boone et al. (2014).
mean	The mean of the posterior distribution for the parameter of interest.
SD	The standard deviation (SD) of the posterior distribution for the parameter of interest calculated with respect to the total number of samples.
median	The median of the posterior distribution for the parameter of interest.
2.5%	The 2.5% quantile of the posterior distribution for the parameter of interest.
97.5%	The 97.5% quantile of the posterior distribution for the parameter of interest.
ESS	The effective sample size for the parameter of interest.
minHD	The minimum HD calculated between consecutive batches. If there are not enough samples (more than 2000) to calculate the HD this field will read -1.
maxHD	The minimum HD calculated between consecutive batches. If there are not enough samples (more than 2000) to calculate the HD this field will read -1.

.*_Diag_C	
protID	Protein ID as specified by the input file.
mutant	Consecutive number of mutant identifier 'c.*'.

HD(*)	Hellinger distance (HD) between sets of samples from two probability distributions. Note that HD is bounded by $0 \leq \text{HD} \leq 1$ and can be used to inspect the similarity between two distributions, where $\text{HD} = 0$ corresponds to no divergence and $\text{HD} = 1$ corresponds to no common support between the distributions. The HD can be used to diagnose the MCMC in terms of its burn-in and whether samples obtained at different points of time came (most likely) from the same (posterior) distribution. Note that one cannot determine if the MCMC chain has truly converged, but only if a chain is internally similar. Here, the HD is calculated for up to 10 equally sized sets of consecutive samples from the MCMC simulation. To obtain sufficient statistical power, the HD between two sets of samples is calculated only if each set consisted of at least 1000 samples. If the total number of samples is less than $10 \times 1000 = 10000$, the number of batches is chosen such that the total number of samples is divided into sets of samples of size 1000 each. If the total number of samples exceeds 10000, the number of samples per set is given by the total number of samples divided by 10 (i.e., the maximal number of batches). If the HD between sets of samples is less than 0.1 the distribution of posterior samples shows a high degree of similarity; if $0.1 \leq HD \leq 0.3$ the distribution of posterior samples are still quite similar, but may require closer inspection; if $0.3 \leq HD \leq 0.5$ sets of samples are vaguely similar and should be inspected more closely; a $HD > 0.5$ indicates strong dis-similarity between sets of samples and could be an indicator that all samples that were taken before might not be from the posterior distribution and should be discarded as burn-in. Note that the HD depends on the degree of autocorrelation between samples. Thus, a high HD might not necessarily indicate that samples were obtained from different sampling distributions, but poor mixing (i.e., a low ESS) for the parameter of interest. For details see Boone et al. (2014).
mean	The mean of the posterior distribution for the parameter of interest.
SD	The standard deviation (SD) of the posterior distribution for the parameter of interest calculated with respect to the total number of samples.
median	The median of the posterior distribution for the parameter of interest.
2.5%	The 2.5% quantile of the posterior distribution for the parameter of interest.
97.5%	The 97.5% quantile of the posterior distribution for the parameter of interest.
ESS	The effective sample size for the parameter of interest.
minHD	The minimum HD calculated between consecutive batches. If there are not enough samples (more than 2000) to calculate the HD this field will read -1.
maxHD	The minimum HD calculated between consecutive batches. If there are not enough samples (more than 2000) to calculate the HD this field will read -1.

.*_Diag_logL	
logL	Log-likelihood tag.
mutant	Consecutive number of mutant identifier 'c.*'.

HD(*)	Hellinger distance (HD) between sets of samples from two probability distributions. Note that HD is bounded by $0 \leq \mathrm{HD} \leq 1$ and can be used to inspect the similarity between two distributions, where $\mathrm{HD} = 0$ corresponds to no divergence and $\mathrm{HD} = 1$ corresponds to no common support between the distributions. The HD can be used to diagnose the MCMC in terms of its burn-in and whether samples obtained at different points of time came (most likely) from the same (posterior) distribution. Note that one cannot determine if the MCMC chain has truly converged, but only if a chain is internally similar. Here, the HD is calculated for up to 10 equally sized sets of consecutive samples from the MCMC simulation. To obtain sufficient statistical power, the HD between two sets of samples is calculated only if each set consisted of at least 1000 samples. If the total number of samples is less than $10 \times 1000 = 10000$, the number of batches is chosen such that the total number of samples is divided into sets of samples of size 1000 each. If the total number of samples exceeds 10000, the number of samples per set is given by the total number of samples divided by 10 (i.e., the maximal number of batches). If the HD between sets of samples is less than 0.1 the distribution of posterior samples shows a high degree of similarity; if $0.1 \leq HD \leq 0.3$ the distribution of posterior samples are still quite similar, but may require closer inspection; if $0.3 \leq HD \leq 0.5$ sets of samples are vaguely similar and should be inspected more closely; a $HD > 0.5$ indicates strong dis-similarity between sets of samples and could be an indicator that all samples that were taken before might not be from the posterior distribution and should be discarded as burn-in. Note that the HD depends on the degree of autocorrelation between samples. Thus, a high HD might not necessarily indicate that samples were obtained from different sampling distributions, but poor mixing (i.e., a low ESS) for the parameter of interest. For details see Boone et al. (2014).
mean	The mean of the posterior distribution for the parameter of interest.
SD	The standard deviation (SD) of the posterior distribution for the parameter of interest calculated with respect to the total number of samples.
median	The median of the posterior distribution for the parameter of interest.
2.5%	The 2.5% quantile of the posterior distribution for the parameter of interest.
97.5%	The 97.5% quantile of the posterior distribution for the parameter of interest.
ESS	The effective sample size for the parameter of interest.
minHD	The minimum HD calculated between consecutive batches. If there are not enough samples (more than 2000) to calculate the HD this field will read -1.
maxHD	The minimum HD calculated between consecutive batches. If there are not enough samples (more than 2000) to calculate the HD this field will read -1.

.*_Diag_summary	
samples	Absolute number of accepted samples taken during the MCMC run.
minESS(c)	The minimum effective sample size (ESS) that was observed for any initial population size 'c.*'.
maxACT(c)	The maximal auto-correlation time (ACT) that was observed for any initial population size 'c.*'.
minESS(r)	The minimum effective sample size (ESS) that was observed for any growth rate 'r.*'.
maxACT(r)	The maximal auto-correlation time (ACT) that was observed for any growth rate 'r.*'.
minESS(logL)	The minimum effective sample size (ESS) that was observed for the log-likelihood.
maxACT(r)	The maximal auto-correlation time (ACT) that was observed for the log-likelihood.
minESS(all)	The minimum effective sample size (ESS) that was observed for all parameters.
maxACT(all)	The maximal auto-correlation time (ACT) that was observed for all parameters.

acceptRatio	The overall acceptance ratio of accepted (and recorded) samples. To ensure high efficiency of the MCMC, the width of the proposal distributions should be chosen such that the acceptance ratio is between $0.15 - 0.45$. Performance is maximal with an acceptance ratio around 0.25. During the burn-in period the width of the proposal distributions is automatically tuned – based on the acceptance ratio – such that the acceptance ratio, when recording samples, has close to maximal efficiency. Thus, a sufficiently long burn-in period not only increases the chance that recorded samples are actually taken from the posterior distribution, but also that the width of the proposal distribution is set appropriately.
jumpSDR	Standard deviation of the proposal distribution of growth rates 'r' (Normal distribution) after auto-tuning.
jumpSDC	Scale parameter of the proposal distribution of initial population sizes 'c' (Cauchy distribution) after auto-tuning.

.*_initialRC

This file prints the last accepted values of the MCMC run so that these could be used as initial values, e.g., to continue an MCMC run that has not yielded enough independent samples. The first line gives the last sampled growth rates 'r.*', the second line gives the last sampled initial population sizes 'c.*', and the third line gives the standard deviation of the proposal distribution of growth rates 'r' (Normal distribution) and the scale parameter of the proposal distribution of initial population sizes 'c' (Cauchy distribution) after auto-tuning.

References

Bank, C., R. T. Hietpas, A. Wong, D. N. Bolon, and J. D. Jensen, 2014. A bayesian mcmc approach to assess the complete distribution of fitness effects of new mutations: Uncovering the potential for adaptive walks in challenging environments. Genetics 196:841–852.

Boone, E. L., J. R. Merrick, and M. J. Krachey, 2014. A hellinger distance approach to mcmc diagnostics. Journal of Statistical Computation and Simulation 84:833–849.