flan: An R Package for Inference on Mutation Models

Adrien Mazoyer, Rémy Drouilhet, Stéphane Despréaux, Bernard Ycart

flan: An R Package for Inference on Mutation Models.

by Adrien Mazoyer, Rémy Drouilhet, Stéphane Despréaux, and Bernard Ycart

Abstract

This paper describes **flan**, a package providing tools for fluctuation analysis of mutant cell counts. It includes functions dedicated to the distribution of final numbers of mutant cells. Parametric estimation and hypothesis testing are also implemented, enabling inference on different sorts of data with several possible methods. An overview of the subject is proposed. The general form of mutation models is described, including the classical models as particular cases. Estimating from a model, when the data have been generated by another, induces different possible biases, which are identified and discussed. The three estimation methods available in the package are described, and their mean squared errors are compared. Finally, implementation is discussed, and a few examples of usage on real data sets are given.

1 Introduction

Mutation models are probabilistic descriptions of the growth of a population of cells, where mutations occur randomly during the process. Data are samples of integers, interpreted as final numbers of mutant cells. These numbers may be coupled with final numbers of cells (mutant and non mutant). The frequent appearance in the data of very large mutant counts, usually called "jackpots", evidences heavy-tailed probability distributions. The parameter of interest is the mutation probability for a mutant cell to appear upon any given cell division, denoted by π. In practice, π is typically of order 10^{-9}–10^{-11}. Computing robust estimates for π is of crucial importance in medical applications, like cancer tumor relapse or multidrug resistance of *Mycobacterium Tuberculosis* for instance.

Any mutation model can be interpreted as the result of the three following ingredients:

- a random number of mutations occurring with small probability among a large number of cell divisions. Due to the law of small numbers, the number of mutations approximately follows a Poisson distribution. The expectation of that distribution, denoted by α, is the product of the mutation probability π with the total number of divisions;
- from each mutation, a clone of mutant cells growing for a random time. Due to exponential growth, most mutations occur close to the end of the experiment, and the developing time of a random clone has exponential distribution. The rate of that distribution, denoted by ρ, is the relative fitness, i.e. the ratio of the growth rate of normal cells to that of mutants;
- the number of mutant cells that any clone developing for a given time will produce. The distribution of this number depends on the distribution of division times of mutants.

Using the theory of continuous time branching processes [6, 4], and under specific modeling assumptions, it can be proved that the asymptotic distribution of the final number of mutants has an explicit form. A first mutation model with explicit distribution is the well known Luria-Delbrück model [21]. Other mathematical models were introduced by Lea and Coulson [20], followed by Armitage [3] and Bartlett [5]. In these models, division times of mutant cells were supposed to be exponentially distributed. Thus a clone develops according to a Yule process, and its size at a given time follows a geometric distribution. The distribution of final mutant counts is also explicit when division times are supposed to be constant. This latter model is called Haldane model by Sarkar [24]; an explicit form of the asymptotic distribution is given in Ycart [29]. General division times have been studied by

Ycart [29], but no explicit distribution is available apart from the exponential and constant division times.

The first estimation method was given by Luria and Delbrück [21]. It is based on the simple relation between the probability of null counts in the sample, and the mutation probability, and it is called P0 method. Of course, if the sample does not contain null counts, the method cannot be applied. Apart from the P0 method, all other methods couple the estimation of π or α, with the estimation of ρ. When the distribution of final numbers has an explicit form, the Maximum Likelihood (ML) is an obvious optimal choice [22, 33]. However, because of the jackpots, likelihood computation can be numerically unstable. There are several ways to reduce tail effects [28, Sec. 2.2], among which "Winsorization" consists in truncating the sample beyond some maximal value. Another estimation method uses the probability generating function (GF) [23, 17]. The estimators of α and ρ obtained with the GF method proved to be close to optimal efficiency, with a broad range of calculability, a good numerical stability, and a negligible computing time. For the three methods, P0, ML, and GF, the estimators of α and ρ are asymptotically normal. Thus confidence intervals and p-values for hypothesis testing can be computed, for one sample and two sample tests.

The problem with classical mutation models, is that they are based on quite unrealistic assumptions: constant final number of cells [2, 19, 31], no cell deaths (Angerer [2, Sec. 3.1]; Dewanji et al. [10], Komarova et al. [19], and Ycart [30]), fully efficient plating [26, 25, 1], or, as mentioned above, exponential distribution of division times. Using a model for estimation, when the data have been generated by another one, necessarily induces a bias on estimates. For instance, if cell deaths are neglected, mutation probability will be underestimated.

Several informatic tools have already been developed for fluctuation analysis [32, 16, 15]. These tools are quite user-friendly. However, they do not take into account all the possible model assumptions mentionned above. The package **flan** described here, is dedicated to mutation models, and parameter estimation with the three methods P0, ML, and GF. It includes a set of functions for the distribution of mutant cell counts (`dflan`, `pflan`, `qflan`, `rflan`) and a graphic function (`draw.clone`). They treat general models, with fluctuating final numbers, cell deaths, and other division time distributions than exponential and constant. The general estimation function is `mutestim`. It returns estimates for the parameters α, π and ρ, with the three estimation methods, constant or exponential division times, and cell deaths. As a wrapper, a hypothesis testing function (`flan.test`) is provided. In order to make the package user-friendly, the functions have been designed to resemble classical R functions, like `t.test` or `rnorm`.

The paper is organized as follows. Section 2 is devoted to the probabilistic setting: the hypotheses of the different models are described, and the asymptotic results are explained. In section 3, the three estimation methods are exposed, and the biases described above are discussed. A comparison of the three methods in terms of mean squared errors is provided. The user interface and the **Rcpp** [11] implementation is treated in section 4; examples of execution are shown in section 5.

2 Mutation models

In this section, probabilistic mutation models are described. The basic modeling hypotheses are the following:

- at time 0 a homogeneous culture of n_0 normal cells is given;
- the lifetime of any normal cell is a random variable with distribution function F;
- upon completion of the generation time of a normal cell:
  - with probability π one normal and one mutant cell are produced;
  - with probability $1-\pi$ two normal cells are produced;
- the lifetime of any mutant cell is a random variable with distribution function G;
- upon completion of the lifetime of a mutant cell:
  - with probability δ the cell dies out;
  - with probability $1-\delta$ two mutant cells are produced;

- all random variables and events (division times, mutations, and deaths) are mutually independent.

Consider that the initial number n_0 tends to infinity, the mutation probability $\pi = \pi_{n_0}$ tends to 0, and the time $t = t_{n_0}$ at which mutants are counted tends to infinity. The scale of time is supposed to be adjusted so that the exponential growth rate of mutants is 1; thus the exponential growth rate of normal cells is ρ. See Athreya and Ney [4, Chap. IV Sec. 4] or Hamon and Ycart [17] for the definition of the growth rate (also called "Malthusian parameter"). The expected number of mutations before t_{n_0} is proportional to $n_0\pi_{n_0}\mathrm{e}^{\rho t_{n_0}}$, and the asymptotics are assumed to be such that this number converges as n_0 tends to infinity to α, positive and finite.

Under the above hypotheses, as n_0 tends to $+\infty$, the final number of mutants converges in law to the distribution with PGF:

$$g(z) = \exp\left(-\alpha(1-h(z))\right)\,, \tag{1}$$

with

$$h(z) = \int_0^\infty \psi(z,t)\rho\mathrm{e}^{-\rho t}\mathrm{d}t\,, \tag{2}$$

where $\psi(z,t)$ is the PGF of the number of cells at time t in a mutant clone, starting from a single cell at time 0. Observe that it depends on the lifetime distribution of normal cells F only through ρ. The above result is deduced from the theory of continous time branching processes [17]). The expressions (1) and (2) translate the three ingredients described in the introduction:

1. the Poisson distribution with intensity α models the total number of mutations which occur during the process;
2. the exponential distribution with rate ρ is that of the time during which a random clone develops;
3. the distribution with PGF $\psi(\cdot,t)$ is that of the number of cells in a random clone developing during a time interval of length t. The PGF ψ is the solution of a Bellman-Harris equation [6] in terms of δ and G.

Hence the expressions of h as an exponential mixture, and of g as a Poisson compound. In practice, the plating process can be less than 100% efficient. In that case, a random number of mutants will not be counted: if only a proportion ζ of the final population is plated, then each cell will be observed with probability ζ. Denote by M_{tot} and M the total and the observed numbers of mutants. Given $M_{\mathrm{tot}} = m$, M follows the binomial distribution with parameters m and ζ. Thus, the PGF g of M is given by:

$$\begin{aligned} g(z) &= \mathbb{E}\left[\mathbb{E}\left[z^M \mid M_{\mathrm{tot}}\right]\right] \\ &= \exp\left(-\alpha\left(1-h(1-\zeta+\zeta z)\right)\right)\,. \end{aligned} \tag{3}$$

The PGF (3) defines a parametrized family of distributions, denoted hereafter by $MM(\alpha,\rho,\delta,\zeta,G)$ (Mutation Model). This is a family of heavy-tailed distributions, with tail exponent ρ: the higher the fitness, the heavier the tail. This directly influences the number and the amount of jackpots.

At this point, the PGF ψ can be given as an explicit expression only for two particular lifetime distributions of mutants: exponential, and Dirac (constant lifetimes). The corresponding mutation models will be denoted respectively by $LD(\alpha,\rho,\delta,\zeta)$ (Luria-Delbrück), and $H(\alpha,\rho,\delta,\zeta)$ (Haldane). The functions `dflan`, `pflan`, and `qflan` compute densities, probabilities, quantiles of LD and H distributions (with $\zeta = 1$).

Assuming that a consistent estimator of α has been defined, the problem in practice is to compute reliable estimates of the mutation probability π. The simplest approach assumes that the final number of cells, denoted by N, is constant. An estimate of π is then obtained by dividing the estimate of α by N. However, even under close experimental monitoring, assuming that the final number of cells is a constant is quite unrealistic. Thus, N must be viewed as a random variable with a certain probability distribution function K on $[0,+\infty)$. By analogy with (1), the conditional PGF of the number of mutants given $N = n$, can be given by the following expression:

$$g\left(z \mid N = n\right) = \exp\left(-\pi n(1-h(z))\right)\,.$$

Or else, the conditional distribution of the number of mutants given $N = n$ is the distribution $MM(\pi n, \rho, \delta, \zeta, G)$. The distribution function K is supposed to be known and its Laplace transform is denoted by $\mathcal{L}$:

$$\mathcal{L}(z) = \mathbb{E}\left[\mathrm{e}^{-zN}\right] = \int_0^\infty \mathrm{e}^{-zn}\mathrm{d}K(n)\,,$$

Thus the PGF of the final number of mutants is given by:

$$\begin{aligned} g(z) &= \int_0^\infty g\left(z \,|\, N = n\right) \mathrm{d}K(n) \\ &= \mathcal{L}\left(\pi(1 - h(z))\right)\,. \end{aligned} \tag{4}$$

Remark that if N is constant, (4) reduces to (1) with $\alpha = \pi N$. In general, the PGF (4) defines a new parametrized family of mutation distributions, denoted herafter by $MMFN(\pi, \rho, \delta, \zeta, G, K)$ (Mutation Models with Fluctuating Numbers of cells).

The two particular cases for the distribution G previously mentioned above (exponential and Dirac) will be denoted by $LDFN(\alpha, \rho, \delta, \zeta, K)$ (Luria-Delbrück with Fluctuating Number of cells) and $HFN(\alpha, \rho, \delta, \zeta, K)$ (Haldane with Fluctuating Number of cells). As will be shown in section 3, estimating π by the ratio of an estimate of α by the expectation of N induces a negative bias.

The function `rflan` outputs samples of pairs (mutant counts–final counts) following $MMFN$ distributions where G is an exponential, Dirac, log-normal or gamma distribution, and K is a log-normal or Dirac distribution.

3 Statistical inference

Here the three estimation methods P0, ML and GF are described. The main features and the limitations of each method are discussed. The three methods compute estimates of α and ρ, under the LD and H models. When couples (mutant counts–final numbers) are given, estimates of π and ρ are calculated under the $LDFN$ or HFN models.

Even if the probabilities and their derivatives with respect to δ for LD and H distributions can be computed, the variations of the whole distribution as a function of δ are too small to enable estimation in practice (see Ycart [30] for more details). Thus, the parameter δ is supposed to be known for the three methods.

In the rest of this section, the three estimators are described, their performances compared in terms of MSE, and the possible sources of biases discussed.

3.1 Estimators

P0 estimator: The first method was introduced by Luria and Delbrück [21] when $\delta = 0$. In that case, the probability of null counts in the sample is $\mathrm{e}^{-\alpha}$. Hence α can be estimated taking the negative logarithm of the relative frequency of zeros among mutant counts. Hence the method cannot be applied if the sample does not contain null counts.

If $\delta > 0$, the probability of null counts in the sample depends also on δ. Assuming $\delta < 1/2$, a fixed point of the PGF $\psi(\cdot, t)$ is the extinction probability of a mutant clone [4, Theorem 1, Chap.I]:

$$\delta_* = \frac{\delta}{1 - \delta}\,.$$

By definition, δ_* is also a fixed point of the PGF (2). Then the probability of null counts in the sample is $\mathrm{e}^{-\alpha(1-\delta_*)}$. A consistent and asymptotically normal estimator of α is given by:

$$\hat{\alpha}_0 = \frac{-\log\left(\hat{g}(\delta_*)\right)}{1 - \delta_*}\,, \tag{5}$$

where $\hat{g}$ denotes the empirical PGF of the final number of mutants.

Consider now that $\zeta < 1$. Ignoring the the unefficient plating will induce a negative bias. A correction has been proposed by Stewart et al. [26, eq. (41)]. However, it can be used only under model $LD(m, 1, 0)$. Indeed, the general expression of the probability of null counts is $\mathrm{e}^{-\alpha(1-h(1-\zeta))}$, which depends on the fitness ρ. It is still possible to extend the estimator (5) to the case where $\zeta < 1$:

$$\hat{\alpha}_0 = \frac{-\log\left(\hat{g}\left(\delta_*^{(\zeta)}\right)\right)}{1-\delta_*}\,, \tag{6}$$

with

$$\delta_*^{(\zeta)} = \frac{\delta_* - (1-\zeta)}{\zeta}\,.$$

Remark that (6) makes sense only if $|\delta_*^{(\zeta)}| \leqslant 1$. In particular, if $\delta = 0$, the plating efficiency ζ has to be greater than 0.5. Therefore, the

Notice that the P0 method does not directly yield an estimator of ρ. If an estimate is desired, the ML method can be used for ρ only, setting $\alpha = \hat{\alpha}_0$.

ML estimators: Since algorithms [13, 33, 17, 31] enable to compute the probabilities of the LD, and H models, the ML method seems to be an obvious choice. It can be used on two kinds of samples:

1. sample of mutant counts: In that case, the likelihood is computed with the probabilities of the model LD or H. The parameter of interest is α.
2. sample of pairs of (mutant counts–final numbers): In that case, the likelihood is computed with the probabilities of the model $LDFN$ or HFN. The parameter of interest is π.

In both cases, ρ can also be estimated.

However, when the sample maximum is large, sums of products of small terms must to be computed [17]. The procedure can be very long and numerically unstable. Thus, the ML estimators can fail for large α and small ρ. In practice, this instability problem is avoided using Winsorization [28, Sec. 2.2], which consists in replacing any value of the sample that pass a certain bound by the bound itself. The bound is 1024 by default, and it could be necessary to increase it. All information above the bound is lost, and in an extreme case where the sample minimum is greater than the bound, irrelevant results will be returned.

In theory, it is also possible to explicit the probabilities of the LD model when $\zeta < 1$. However, these computations have not been done for the H model. Thus the plating process is assumed to be fully efficient when the ML method is used.

GF estimators: The GF method uses the PGF to estimate the parameter of a compound Poisson distribution [23, 17]. Let $0 < z_1 < z_2 < 1$ and z_3 in $(0\,;1)$. The estimators of α and ρ are the following:

$$\hat{\alpha}_{GF}(z_3) = \frac{\log\left(\hat{g}(z_3)\right)}{h_{\hat{\rho}_{GF}(z_1,z_2)}(z_3) - 1} \quad \text{and} \quad \hat{\rho}_{GF}(z_1, z_2) = f^{-1}_{z_1,z_2}(\hat{y})\,,$$

where $\hat{g}$ denotes the empirical PGF of the final number of mutants, h_x is the PGF (2) with $\rho = x$, and:

$$f_{z_1,z_2}(x) = \frac{h_x(z_1) - 1}{h_x(z_2) - 1} \quad \text{and} \quad \hat{y} = \frac{\log\left(\hat{g}(z_1)\right)}{\log\left(\hat{g}(z_2)\right)}\,. \tag{7}$$

From Rémillard and Theodorescu [23], it can be proved that the couple of estimators $(\hat{\alpha}_{GF}, \hat{\rho}_{GF})$ is strongly consistent and asymptotically normal, with explicit asymptotic variance [17].

The GF estimators depend on the three arbitrary values of z_1, z_2, z_3. Those tuning parameters are set to $z_1 = 0.1$, $z_2 = 0.9$, and $z_3 = 0.8$. For more details about the choice of those values, see Hamon and Ycart [17].

In practice, the GF estimators are quite comparable in precision to ML estimators, with a much broader range of calculability, a better numerical stability, and a negligible computing time, even in the case where the ML method fails. For that reason, we have chosen to initialize the ML optimization by GF estimates, to improve both numerical stability and computing time.

The only practical limitation of this method is the following. A zero of the monotone function $f_{z_1,z_2}(\rho) - \hat{y}$ must be computed. An upper bound for the domain of research must be given, which can be a problem if the sample does not contain jackpots. However in that case, a mutation model is not adapted.

According to (3), the GF estimators or α and ρ can be extended to the case where $\zeta \leqslant 1$ as follows:

$$\hat{\alpha}_{GF}(z_3) = \frac{\log\left(\hat{g}(z_3)\right)}{h_{\hat{\rho}_{GF}(z_1,z_2)}(1-\zeta+\zeta z_3) - 1} \quad \text{and} \quad \hat{\rho}_{GF}(z_1, z_2) = f^{-1}_{1-\zeta+\zeta z_1, 1-\zeta+\zeta z_2}(\hat{y})\,,$$

where f_{z_1,z_2} and $\hat{y}$ are still given by (7). By the same reasoning as Hamon and Ycart [17], strong consistency and asymptotic normality of the couple $(\hat{\alpha}_{GF}, \hat{\rho}_{GF})$ can be proved.

The function `mutestim` computes estimates and their respective standard deviations for α, π and ρ according to the type of input. Moreover, the estimators mentioned here are asymptotically normal. Thus, one and two sample tests can be performed, using the function `flan.test`. The null hypothesis will be either fixed theoretical values of α, π, ρ in the one sample case, or a difference of the same in the two sample case.

3.2 Comparison of the three estimators

The Figures 1 and 2 (drawn using **ggplot2**) show "maps of usage" of the estimation methods under LD and H models. The methods are compared in terms of the relative MSE of $(\hat{\alpha}, \hat{\rho})$ defined as:

$$\sqrt{\left(1 - \frac{\hat{\alpha}}{\alpha}\right)^2 + \left(1 - \frac{\hat{\rho}}{\rho}\right)^2}\,. \tag{8}$$

The RGB code is used: red for GF, green for P0, blue for ML. Twenty values of α between 0.5 and 10, and as many values of ρ from 0.2 to 5, were chosen. Thus 400 couples were considered. For each of them, the following procedure was applied:

1. draw 10^4 samples of size 100 of the $LD(\alpha, \rho, 0, 1)$;
2. for each sample, compute ML, GF and P0 estimates of (α, ρ) under LD model;
3. from the 10^4 estimates, compute the relative MSEs of each method;
4. assign a RGB color according to the MSEs. For each method:
   - if the MSE is less than 0.05, assign 1 to the corresponding RGB component;
   - if the MSE is greater than 1, assign 0 to the corresponding RGB component;
   - else, assign 1 minus the MSE to the corresponding RGB component.

The above experience is also performed considering H model instead of LD. The maps have been drawn with a $\log_5$-scale for ρ (y-axis). The map can be roughly divided into four distinct parts:

- For $(\alpha, \rho) \in (0.5\,;3) \times (0.2\,;2.5)$, the color is essentially grey: the three methods are more or less equivalent.
- For $(\alpha, \rho) \in (3\,;10) \times (0.2\,;3.5)$, the color is magenta: the ML and GF methods are equivalent. The P0 method provides estimates with large MSEs or cannot be used because of the absence of null counts.
- For small values of ρ, the color is mainly red: The GF method is the only method with an acceptable MSE. Small values of ρ induce large jackpots. Moreover, the number of jackpots increases with α. Because of the winsorization, the ML and P0 method (which uses ML to estimate ρ) provide estimates with very large MSEs.
- For ρ large, the color is darker and tends to black: the three methods provide estimates with large MSEs, specially for $\rho \in (3.5\,;5)$, where jackpots are very small or absent. In those cases, estimating ρ with the GF method is not possible in practice (see previous sub-section). Consequently, the GF method will provide a biased estimate for α. The ML method, which

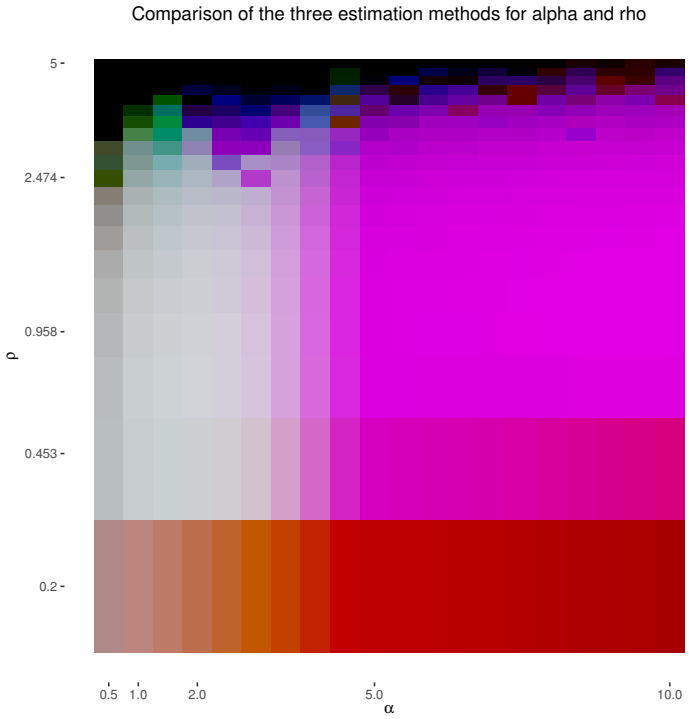

Figure 1: Map of usage of the estimation methods under *LD* model. The map compares the three methods according to their relative MSE (8). For each of 400 couples of parameters $\alpha = 0.5 \ldots 10$ (x-axis) and $\rho = 0.2 \ldots 5$ (y-axis, $\log_5$-scale), 10^4 samples of size 100 of the $LD(\alpha, \rho, 0, 1)$ distribution were simulated. The estimates of (α, ρ) were calculated with the three methods. Each method is represented by a color: red for GF, green for P0, blue for ML.

uses the GF estimates to initialize the optimization of the log-likelihood, also provides biased estimates. The P0 method can provide good estimates of α whatever the value of ρ, which explains the presence of green areas at the top of the map. In a case where no jackpots are present in the sample it should be considered that a (heavy tailed) mutation model is not adapted.

Figure 2 is quite similar to Figure 1. There are still two remarkable differences:

- For ρ small and $m \leqslant 2$, the three methods seem to be more equivalent under H model than under LD model.
- For ρ large, GF method seems to provide better estimates under H model than under LD model.

The three methods should also be compared in terms of computational time. An illustration on real data will be given in section 5. The slowest method is ML, for the reasons discussed in the previous section. It is even slower when the estimates are calculated under Haldane models H or HFN, when δ is positive, or if the initialization of ρ with the GF method fails. The GF method computes estimates of α and ρ (when possible) in negligible time. The P0 method outputs estimates of α in negligible time, but estimates of ρ are as slow as with ML.

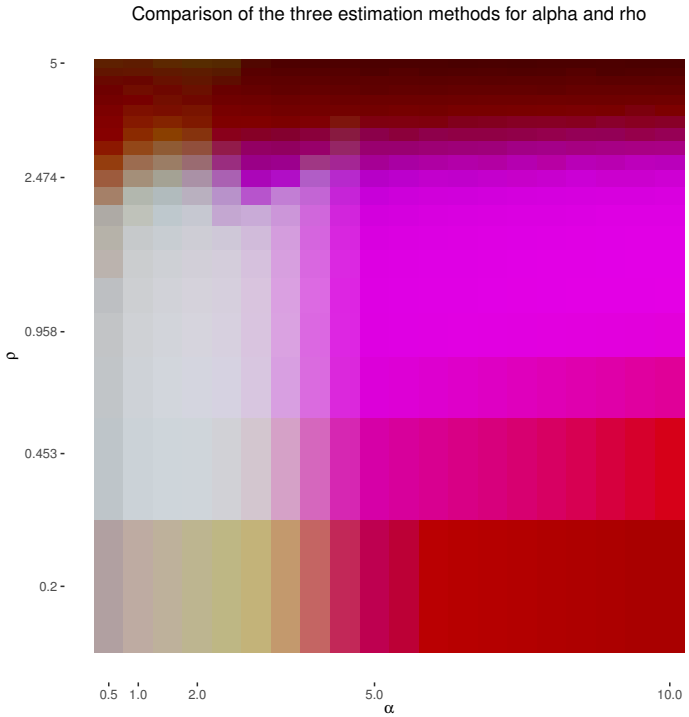

Figure 2: Map of usage of the estimation methods under H model. The map compares the three methods according to their relative MSE (8). For each of 400 couples of parameters $\alpha = 0.5 \ldots 10$ (x-axis) and $\rho = 0.2 \ldots 5$ (y-axis, $\log_5$-scale), 10^4 samples of size 100 of the $H(\alpha, \rho, 0, 1)$ distribution were simulated. The estimates of (α, ρ) were calculated with the three methods. Each method is represented by a color: red for GF, green for P0, blue for ML.

3.3 Bias evidence

If the model used for the estimation does not correspond to the theoretical model, the estimates can be biased. Four different sources of bias are considered:

1. the final counts are random in the data, constant for the estimation model;
2. cell deaths occur in the data, not in the estimation model;
3. the lifetime distribution is different in the data and the estimation model;
4. the plating process is less than 100% efficient.

In each case, simulation experiments have been made along the following lines:

1. draw 10^4 samples of size 100, under one model;
2. for each sample, compute estimates using another model and the true one if available;
3. compare the empirical distributions of $\hat{\theta}/\theta$, where $\hat{\theta}$ is estimator and θ the true value.

For each figure, red lines mark unit, blue lines mark relative biases of 0.9 and 1.1. According to Figures 1 and 2, the GF method is at least equivalent in terms of the relative MSE (8) to the ML and P0 methods. Moreover, it is also the best in terms of computational time. Therefore, the bias evidences will be mainly illustrated with GF method.

Fluctuation of final counts: When N is constant, the estimate of π is derived by dividing the estimate of α by N. As mentioned in previous section, if N is a random variable, the relation between α and π can be explicit if the distribution K is known. However, this is not the case in practice. Usually, estimates of the expectation and variance of N are available at best. Assume that only the first two moments μ and σ^2 of N are known. Then a first order approximation of the Laplace transform $\mathcal{L}$ can be used to reduce the bias. This method is explained in Ycart and Veziris [31] for the P0 method. It has been adapted to ML and GF estimates. Figure 3 shows the influence of the coefficient of variation $C = \sigma/\mu$ on the ML estimate of π. The estimates were calculated with three different approaches:

- divide ML estimates of α by the empirical mean of N and ignore fluctuations of N (left boxplots);
- directly compute ML with the sample of pairs (mutant counts–final counts) (center boxplots);
- derive from ML estimates of α, taking into account of the empirical fluctuations of N (right boxplots).

According to the visual observations, the bias reduction seems to be working well when either the product $\pi\mu$ or C are small: most of the estimates have a relative bias smaller than 10%. However, the efficiency of the correction decreases as product $\pi\mu$ and C increase. In particular, for larger values of $\pi\mu$, the bias seems to be smaller without correction. It could be improved with a better approximation of $\mathcal{L}$, that implies knowing or estimating higher moments of N. Another solution is to improve the estimation of C. Here C was estimated by the ratio of the empirical standard deviation of the empirical mean, which is known to be a bad method in terms of MSE [8].

Cell deaths: The PGFs (1), (4) and (2) depend on δ. Ignoring cell deaths involves a negative bias on the estimate of α. Assuming the exact value is known, this bias is removed. Figure 4 shows the influence of the death parameter δ on the GF estimate of α. The estimates are calculated with two different approaches:

1. computing GF estimates of α with $\delta = 0$ (left boxplots);
2. computing GF estimates of α with theoretical value of δ (right boxplots).

The visual results show that the negative bias induced by ignoring cells death increases with the value of δ: the relative bias can easily exceed 10% for large values of δ. From the theory of branching processes, the growth process of a mutant clone is supercritical and δ has to be smaller than 0.5. In practice δ is smaller than 0.3. According to the boxplots, the relative bias induced by ignoring cell deaths can reach 0.80. These experiments illustrate also the difficulty to estimate δ. For example, the boxplots at the top right of the figure seems to show that the value of the likelihood for $\alpha = 4$ and $\delta = 0$ is very close to its value for $\alpha = 4$ and $\delta = 0.05$.

Lifetime distribution: As mentioned earlier, the PGF h is explicit only for the LD and H distributions, i.e. when lifetimes are either exponential or constant. This is not the case in practice. If another lifetime distribution is used to simulate the data, and either LD or H are used to estimate the parameters, a bias will be induced on α and ρ. Figure 5 illustrates these observations. It shows the influence of the lifetime distribution on the GF estimates of α and ρ. The samples are drawn assuming the lifetimes are log-normally distributed. The estimates of α and ρ are calculated under LD (left boxplots) and H models (right boxplots).

From the visual observations, the LD and H models can be seen as extreme values for the lifetime distribution:

- both models correctly estimate α;
- the LD model overestimates ρ and has a rather large dispersion of estimated values. The bias seems to increase as α increases;
- the H model correctly estimates ρ.

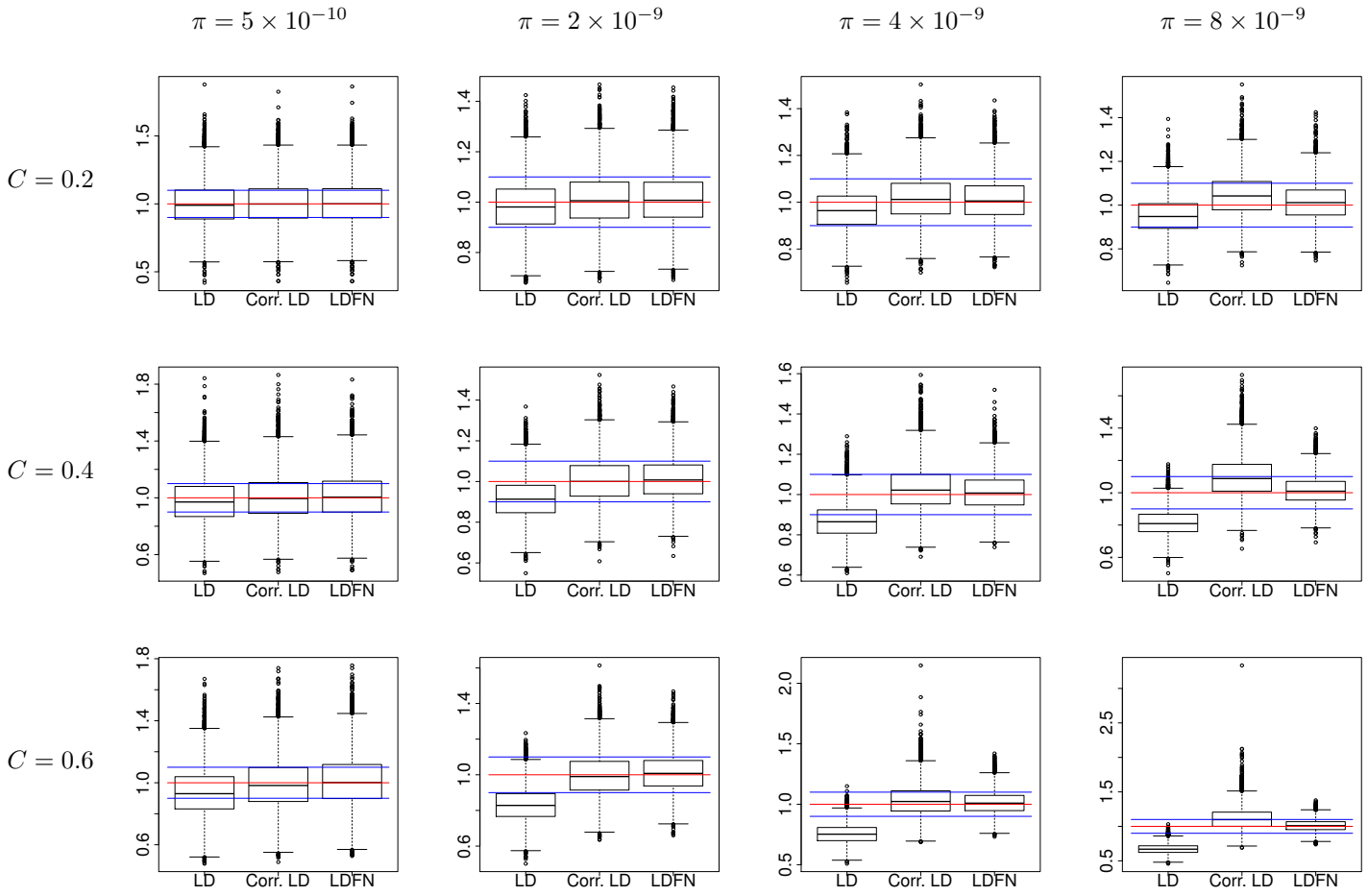

Figure 3: ML estimates of π ignoring fluctuations of final numbers or not. Red horizontal lines mark unit. Blue horizontal lines mark relative bias of 0.9 and 1.1. For each of the 12 sets of parameters $\pi = (0.5/\mu, 2/\mu, 4/\mu, 8/\mu)$ (columns), and $C = (0.2, 0.4, 0.6)$ (rows), 10^4 samples of size 100 of the $LDFN(\pi, \rho, 0, 1, K)$ distribution were simulated, with $\rho = 1$ and K being the Log-normal distribution adjusted to mean $\mu = 10^9$ and coefficient of variation C. The ML estimates of π and ρ were calculated under model LD or $LDFN$. Each boxplot represents the distribution of the 10^4 ratio $\hat{\pi}/\pi$ obtained with LD model with $C = 0$ (left), LD model with bias reduction (center), $LDFN$ model (right).

Plating efficiency: Ignoring the plating efficiency induces a negative bias on α and a positive bias on ρ. In practice, the exact value is known. Figure 6 shows the influence of ζ on the GF estimates of α. The estimates are calculated with two different approaches:

1. computing GF estimates with $\zeta = 1$ (left boxplots) ;
2. applying the correction of Stewart et al. [26, eq. (41)] to GF estimates (center boxplots);
3. computing GF estimates with known value of ζ (right boxplots).

The visual results illustrate first the fact that the correction of Stewart et al. [26] should not be used when $\rho \neq 1$: the induced bias on α is negative when $\rho > 1$, positive when $\rho < 1$. However, the variance of the estimates obtained with these rectification is smaller than that of the estimates calculated with GF method.

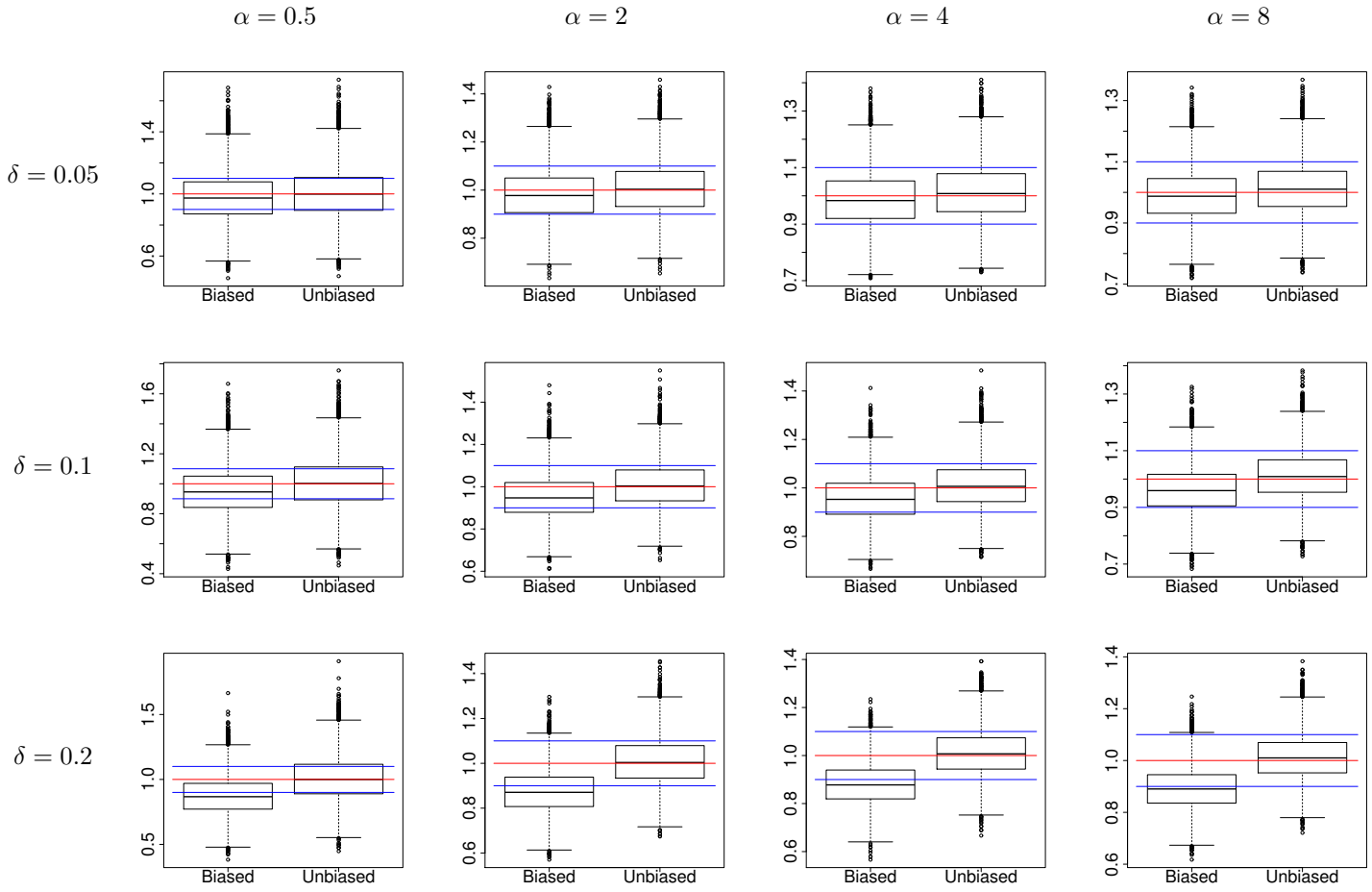

Figure 4: GF estimates of α ignoring cell deaths or not. Red horizontal lines mark unit. Blue horizontal lines mark relative bias of 0.9 and 1.1. For each of the 12 sets of parameters $\alpha = (0.5, 2, 4, 8)$ (columns), and $\delta = (0.05, 0.1, 0.2)$ (rows), 10^4 samples of size 100 of the $LD(\alpha, \rho, \delta, 1)$ distribution were simulated, with $\rho = 1$. The GF estimates of α and ρ were calculated under LD model. Each boxplot represents the distribution of the 10^4 ratio $\hat{\alpha}/\alpha$ of the estimates obtained without taking account of cells death (left) and with the theoretical value of δ (right).

4 Implementation details

The available functions are described here; more details are given in the manual. The behavior of inference functions for inputs which are out of practical limitations is described. Some details about the **Rcpp** implementation are also provided.

4.1 User interface

flan can be split into two distinct parts: the distribution of the final number of mutants, and statistical inference. The functions `dflan`, `pflan`, `qflan` compute densities, probabilities and quantiles of LD and H distributions. The function `rflan` outputs samples of pairs (mutant counts–final counts) following $LDFN$, HFN, or $MMFN$ where G is log-normal or gamma distribution and K is log-normal or Dirac distribution. K is adjusted to the mean and coefficient of variation provided by the

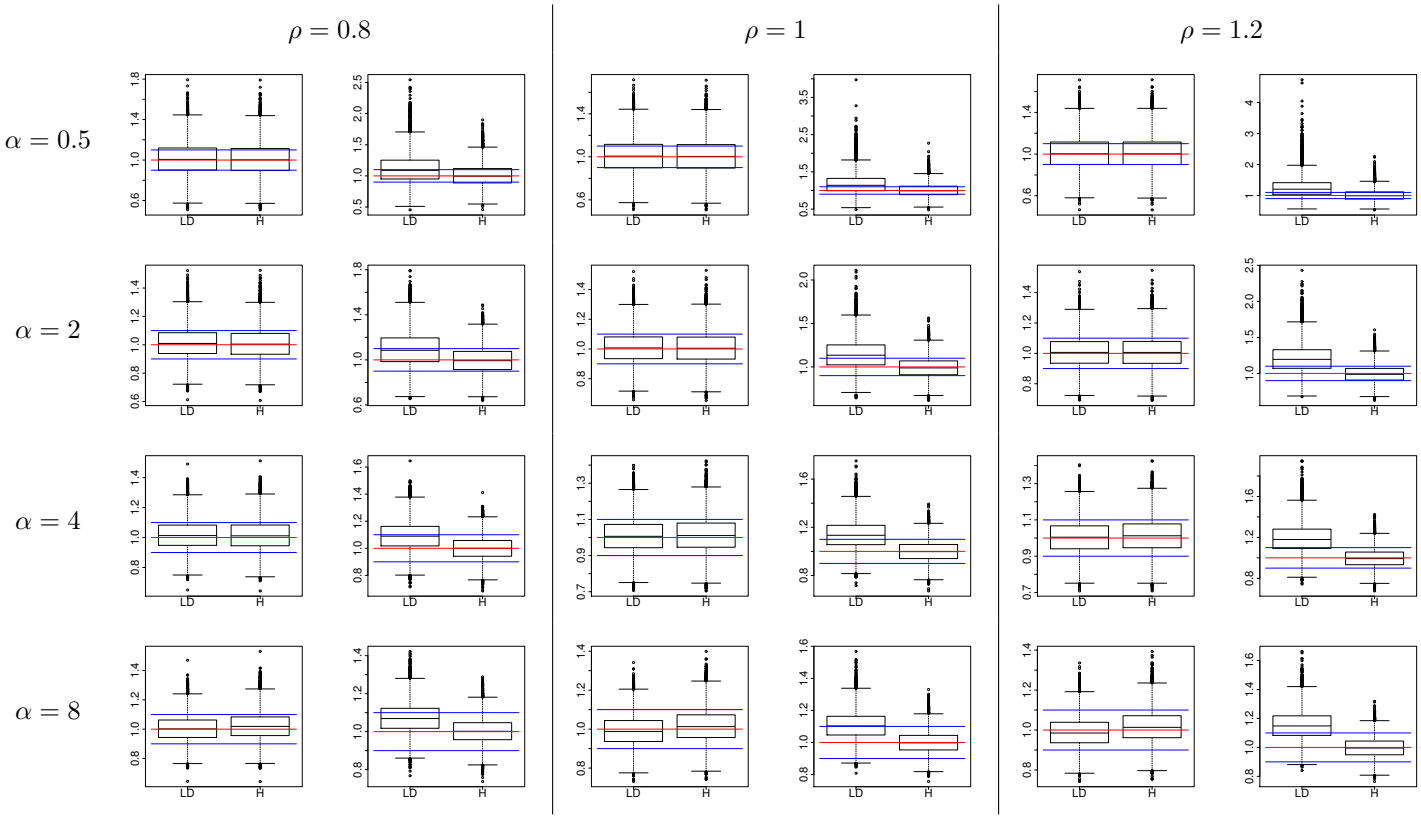

Figure 5: GF estimates of α and ρ under LD and H models on data drawn with MM model. Red horizontal lines mark unit. Blue horizontal lines mark relative bias of 0.9 and 1.1. For each of the 12 sets of parameters $\alpha = (0.5, 2, 4, 8)$ (rows), and $\rho = (0.8, 1, 1.2)$ (columns), 10^4 samples of size 100 of the $MM(\alpha, \rho, 0, 1, G)$ distribution were simulated, G being the Log-normal distribution adjusted on Kelly and Rahn's data [18]. The GF estimates of α and ρ were calculated with the two distributions $LD(\alpha, \rho, 0, 1)$ and $H(\alpha, \rho, 0, 1)$. For each couple (α, ρ), the two first boxplots represent the distribution of the 10^4 ratio $\hat{\alpha}/\alpha$ obtained by the LD model (left) and the H model (right); the two last boxplots represent the distribution of the 10^4 ratio $\hat{\rho}/\rho$ obtained by the LD model (left) and the H model (right)

user. Those functions have been designed on the principle of the classical distribution functions of R. A graphic function `draw.clone` is also provided. It represents with a binary tree the growth of a clone starting from a single normal cell with mutation occurrences until a finite time. The function `mutestim` computes estimates of α or π and ρ, using LD or H models. The three estimation methods are available. Fluctuations of final numbers and cells death are included. It returns estimate(s) of the parameter(s) of interest and the standard deviations. The function `flan.test` uses asymptotic normality to perform one or two sample hypothesis testing. It has been designed on the principle of the classical hypothesis testing functions of R, such as `t.test`.

As mentioned in section 3, there are practical limitations for each estimation method. If the inputs of the `mutestim` function do not respect those limitations, it will output errors or warning messages:

- If $\delta = 0$, the P0 method can not be used if the sample does not contain any null counts. In that case, the `mutestim` function will return an error message.
- For now, an unefficient plating (i.e. $\zeta < 1$) can be taken into account only with the GF method. If another method is specified, the `mutestim` function will return a warning message and set ζ at 1.

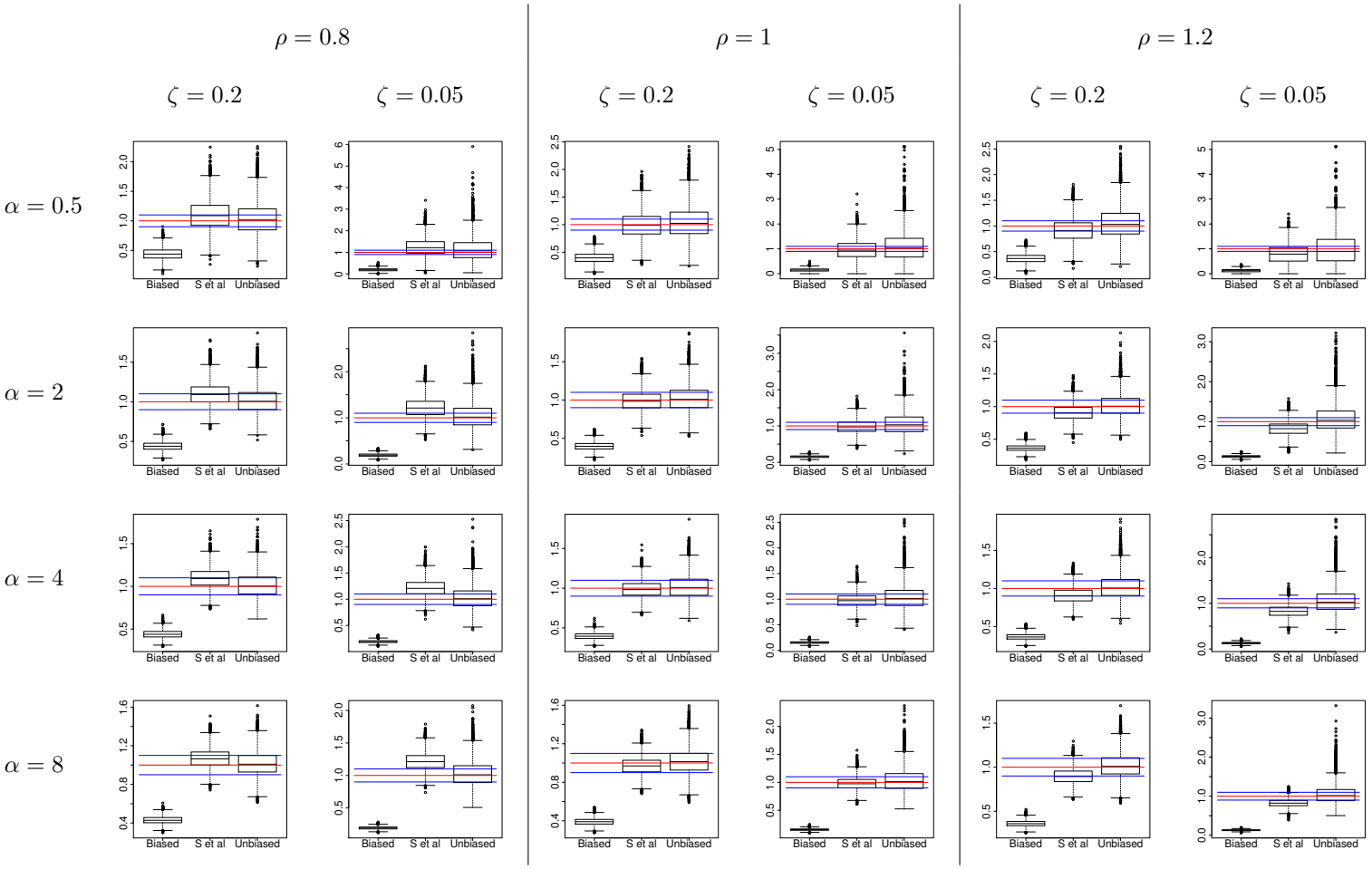

Figure 6: GF estimates of α ignoring plating efficiency or not. Red horizontal lines mark unit. Blue horizontal lines mark relative bias of 0.9 and 1.1. For each of the 12 sets of parameters $\alpha = (0.5, 2, 4, 8)$ (rows), and $\rho = (0.8, 1, 1.2)$ (columns), 10^4 samples of size 100 of the $LD(\alpha, \rho, 0, \zeta)$ distribution were simulated, with $\zeta = 0.2$ and $\zeta = 0.05$. The GF estimates of α and ρ were calculated under LD model. Each boxplot represents the distribution of the 10^4 ratio $\hat{\alpha}/\alpha$ obtained with $\zeta = 1$ (left), using the Stewart et al. [26] correction (center), and with the true value of ζ (right).

- Issues of the Winzorization parameter w of ML method:
  1. If the minimum of the sample is larger than w, then the sample of mutant counts will be constant.
  2. If w is too large, then the optimization process can be very long.
  3. In the `mutestim` function w is set at 1024 by default.
- The GF method does not have limitations of usage, even for extreme cases where the ML estimators fail, i.e. samples with theoretical large α and small ρ. However, estimating ρ requires to solve the zero equation discussed in section 3, which is theoretically solvable on $\mathbb{R}^+$. In practice the interval of research is bounded. Thus, if the sample does not contain any jackpot, which means that ρ is very large, the zero equation may not have any solution on the interval. In that case, the function will return a warning message, and set the estimate of ρ at 1, and the estimate of the standard deviation at 0. In the `mutestim` function, the domain of research is

$[0.01\,;100]$.

- Moreover, the initialization of the ML method is done with GF method. Then the domain of optimization is $[0.1 \times \hat{\theta}_{GF}\,; 10 \times \hat{\theta}_{GF}]$, where $\hat{\theta}_{GF}$ is the GF estimate(s) of the parameter(s) of interest. Then, if the GF method does not succeed to estimate ρ, there is no chance to estimate it with ML. A warning message is returned if the initialization of the estimate of ρ with GF fails.

The function `flan.test` is a wrapper function of `mutestim`. It will ouput the same errors or warning messages if its inputs do not respect the practical limitations.

4.2 Implementation

Since most functions involve loops that are more expensive in R than in C, **flan** has been implemented with **Rcpp** modules. This paradigm provides an easy way to expose C++ functions and classes to R. There are four main classes in the C++ implementation:

- `FLAN_Sim`: random generation for MM and $MMFN$ distributions. One of its members is a variable of the following type;
- `FLAN_SimClone`: random generation for clone size distribution according to the lifetimes distribution;
- `FLAN_MutationModel`: computation of the descriptive functions (probabilities, PGF,...) for LD and H distributions. One of its members is a variable of the following type;
- `FLAN_Clone`: computation of the descriptive functions for clone size distribution according to the lifetimes distribution.

The **Rcpp** interface enables also to import into the C++ code any R function. In particular, it is interesting to import the R functions which are already implemented in C. Thus no external C/C++ library is required. The installation remains basic, and the size of the installed package is reduced. For example, the computations of LD distributions involve numeric integrations. The C libraries **integration** and **alglib** compute integrals with an accuracy close to machine precision. We could use those libraries but the R function `integrate` is actually implemented in C. The R function can then be directly called into the C++ code. However, such imports increase the computational time and memory consumption. This is thus unsatisfactory. Another solution uses the package **RcppGSL** [11, chap. 11], which creates an interface between **Rcpp** and the C library **gsl**. Several integration methods are thus available. Since it requires only that **gsl** is correctly installed, this solution is a good compromise between easy setup and computational cost.

Computing the probabilities for the H distribution with $\delta > 0$ involves squaring high degree polynomials. Such polynomials are easily treated by the package **polynom**. However its implementation raises memory issues, because of the degree of the polynomials involved. A more efficient way is to use the Fast Fourier Transform. It is provided by the C library **fftw3**, which can raise some installation issues. The R function `fft` could be directly called into the C++ code. For the same reasons as for the `integrate` function, it is more adequate to use the package **RcppArmadillo** (Eddelbuettel [11, chap.10] ; Eddelbuettel and Sanderson [12]). This package links **Rcpp** to the C++ library **armadillo**, which is dedicated to linear algebra. In particular, it includes a performing Fast Fourier Transform.

Finally, likelihood optimizations in the ML and P0 methods are done with a bounded BFGS optimizer. The package **lbfgsb3** provides the eponymous function which is implemented in Fortran. It is much faster than the basic R function `optim`.

5 Examples of usage

Some examples on the real data included in **flan** are provided. Practical limitations, influence of bias sources and comparison of the estimation methods in terms of computational time are illustrated. Consider first the eleventh sample of mutant counts of the `werhoff` data [27]:

```
werhoff$samples$W11$mc
```

```
##  [1] 0 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 4 4 4 4 5 5
```

This sample does not contain any jackpot, then the theoretical fitness in a mutation model, should be very large. If the GF method is used, it will output a warning message and set the fitness at $\rho = 1$, as customarily done in the litterature [14]:

```
W <- werhoff$samples$W11$mc

# Compute GF estimates of mutations number and fitness
mutestim(mc = W, method = "GF")
```

```
## Warning in mutestim(mc = W, method = "GF"):  Impossible to estimate 'fitness' with 'GF'-method:
'fitness' is set to default value 1 and only mutations number is estimated.
```

```
## $mutations
## [1] 0.8792917
##
## $sd.mutations
## [1] 0.260549
##
## $fitness
## [1] 1
##
## $sd.fitness
## [1] 0
```

Notice that Table 1 of [27] shows mutants counts and mean final numbers of cells for 1 mL of solution. However, the total volume of the culture is 5 mL. In other words, a plating efficiency of 20% has been applied. The fact is the fitness parameter can not be estimated, even taking into account the plating efficiency:

```
# Volume of each sample: 1 mL
# Total volume of each culture: 5 mL
# i.e. plating efficiency of 0.2
pef <- 0.2

# Compute GF estimates of mutation probability and fitness
# taking into account the plating efficiency
mutestim(mc = W, plateff = pef, method = "GF")
```

```
## Warning in mutestim(mc = W, plateff = pef, method = "GF"):  Impossible to estimate 'fitness'
with 'GF'-method:  'fitness' is set to default value 1 and only mutations number is estimated.
```

```
## $mutations
## [1] 2.804244
##
## $sd.mutations
## [1] 0.74011
##
## $fitness
## [1] 1
##
## $sd.fitness
## [1] 0
```

Using the P0 method is a way to realize that setting $\rho = 1$ by default can be misleading. Since

this method does not depend on the lifetime distribution, the estimate of α will not depend on the value of ρ.

```
# Compute P0 estimate of mutations number
mutestim(mc = W, fitness = 1, method = "P0")
```

```
## $mutations
## [1] 2.525729
##
## $sd.mutations
## [1] 0.678233
```

The P0 estimate of α is very different from the GF estimate (when $\zeta = 1$). Consider now the sample of David [9, Tab. 2], which includes rifampin-resistant bacteria counts.

```
david$D11
```

```
## $mc
##  [1] 4 0 1 0 1 0 0 0 0 0
##
## $fn
##  [1] 1.3e+09 9.2e+08 1.3e+09 2.5e+09 1.3e+09 1.6e+09 1.3e+09 2.5e+09
##  [9] 2.5e+09 2.0e+09
```

Since the 4 value can be seen as a jackpot, the GF method can be used to estimate α and ρ. Now let us compute the ML estimates of π and ρ taking into account or not of the final counts, under the LD model.

```
D <- david$D11
Dmfn <- mean(D$fn)

# Compute ML estimates and confidence intervals of mutation probbility and fitness
# with empirical mean and nul coefficient of variation
ft <- flan.test(mc = D$mc, mfn = Dmfn)
ft$estimate
```

```
## mutation probability              fitness
##         2.067641e-10         2.214676e+00
```

```
ft$conf.int
```

```
##      mutation probability  fitness
## bInf         0.000000e+00 0.000000
## bSup         4.423916e-10 8.109577
## attr(,"conf.level")
## [1] 0.95
```

```
# Compute ML estimates and confidence intervals of mutation probbility and fitness
# with empirical mean and empirical coefficient of variation
ft <- flan.test(mc = D$mc, mfn = Dmfn, cvfn = sd(D$fn)/Dmfn)
ft$estimate
```

```
## mutation probability              fitness
##         2.092720e-10         2.214676e+00
```

```
ft$conf.int
```

```
##      mutation probability  fitness
## bInf         0.000000e+00 0.000000
## bSup         4.506155e-10 8.109577
## attr(,"conf.level")
## [1] 0.95
```

```r
# Compute ML estimates and confidence intervals of mutation probability and fitness
# with couples (mc,fn)
ft <- flan.test(mc = D$mc, fn = D$fn)
ft$estimate
```

```
## mutation probability              fitness
##         1.977135e-10         2.048984e+00
```

```r
ft$conf.int
```

```
##      mutation probability  fitness
## bInf         0.000000e+00 0.000000
## bSup         4.078591e-10 7.127426
## attr(,"conf.level")
## [1] 0.95
```

The sample of final counts is denoted by $D_{11}^{(FN)}$. The empirical mean of the final counts is denoted by $\overline{\mu}$, the empirical coefficient of variation by $\overline{C}$. Table 1 displays the ML estimates of π and ρ, in the same way as for Figure 3. Their 95% confidence intervals are provided. Comparing the second row to the fourth, one can see that neglecting final number fluctuations induces a bias of order 5% on π, 10% on ρ. From the third row, it turns out that the correction taking into account $\overline{C}$, has not improved the estimate of π. Notice also that due to the small size sample, the confidence intervals are quite large.

	Estimates of π	Confidence intervals (95%) of π	Estimates of ρ	Confidence intervals (95%) of ρ
ML using $\mu = \overline{\mu}$, $C = 0$	2.07×10^{-10}	$[0\,; 4.22 \times 10^{-10}]$	2.21	$[0\,; 8.11]$
ML using $\mu = \overline{\mu}$, $C = \overline{C}$	2.09×10^{-10}	$[0\,; 4.51 \times 10^{-10}]$	2.21	$[0\,; 8.11]$
ML using $D_{11}^{(FN)}$	1.98×10^{-10}	$[0\,; 4.08 \times 10^{-10}]$	2.05	$[0\,; 7.13]$

Table 1: ML estimates of π and ρ of data set from [9, Tab. 2] ignoring fluctuations of final numbers or not. Each row shows the ML estimates of π and ρ and their 95% confidence intervals, deducing from LD model with $C = 0$ (first row), $C = \overline{C}$ (second row), and directly with $LDFN$ model (third row).

Consider finally the data from Boe et al. [7, Tab. 4]. The author studied mutations of *Escherichia coli* from sensitivity to nalidixic acid resistance. 23 samples of resistant bacteria counts are provided. As in the original paper, the 23 samples are concatenated as one. The three estimation methods are compared in terms of computational time on the resulting sample of size 1104. The package **microbenchmark** is used, evaluating 10^4 times each method on the sample. The methods are compared when only α is estimated ($\rho = 1$), and when the couple (α,ρ) is estimated. In both cases the estimates are computed under the model LD with $\delta = 0$ and $\zeta = 1$.

```r
B <- unlist(boeal)       # Concatenation of the 23 samples
require(microbenchmark)

# Comparing the methods in terms of computational performance
# (mutations number only)
```

```
microbenchmark(P0 = mutestim(mc = B, fitness = 1, method = "P0"),
               GF = mutestim(mc = B, fitness = 1, method = "GF"),
               ML = mutestim(mc = B, fitness = 1, method = "ML"),
               unit = "ms", times = 1e4)
```

```
## Unit: milliseconds
##  expr       min        lq      mean    median        uq       max neval
##    P0  0.063696  0.0800720  0.09701435  0.0899955  0.0986275  2.149537 10000
##    GF 0.327989  0.3694555  0.41309310  0.3850380  0.4035050 26.774004 10000
##    ML 8.381844 10.5333670 12.23137063 10.9118690 11.3860045 43.592177 10000
```

```
# Comparing the methods in terms of computational performance
microbenchmark(P0 = mutestim(mc = B, method = "P0"),
               GF = mutestim(mc = B, method = "GF"),
               ML = mutestim(mc = B, method = "ML"),
               unit = "ms", times = 1e4)
```

```
## Unit: milliseconds
##  expr       min        lq      mean    median        uq       max neval
##    P0  8.451605 11.004419 12.832827 11.848562 13.082066  43.00397 10000
##    GF  2.274944  2.419663  2.688851  2.481529  2.549597  33.19333 10000
##    ML 73.704656 79.581238 83.841150 81.802464 84.323398 120.65768 10000
```

The results are shown on Figure 7, as boxplots of timing distributions. Times are in milliseconds and plotted on log-scale.

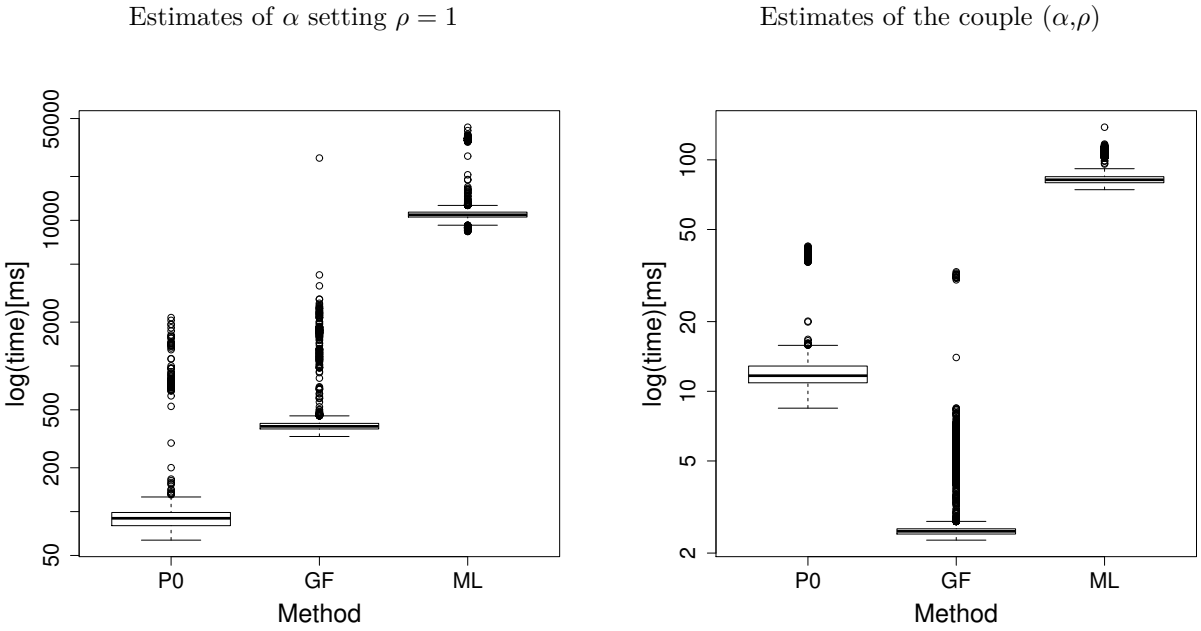

Figure 7: Computational time of the three methods on data from Boe et al. [7, Tab. 4]. Data consist in the 23 samples of `boeal` concatenated as one. For each method, the estimates of α setting $\rho = 1$ (left boxplots) or of α and ρ (right boxplots) have been computed under model LD. The timings have been returned with **microbenchmark**, evaluating 10^4 times each method. Times are in milliseconds and plotted on log-scale.

As mentioned earlier, the ML method is the slowest. The GF method seems to be quite faster

than the P0 method. However, the estimates of ρ of the P0 method is calculated maximizing the likelihood. This step is more costly in term of computational time. If only α has to be estimated, the P0 method is faster than the GF method.

References

[1] W.P. Angerer. "A note on the evaluation of fluctuation experiments". In: *Mutation Research* 479 (2001), pp. 207–224.

[2] W.P. Angerer. "An explicit representation of the Luria-Delbrück distribution". In: *J. Math. Biol.* 42.2 (2001), pp. 145–174.

[3] P. Armitage. "The statistical theory of bacterial populations subject to mutation". In: *J. R. Statist. Soc. B* 14 (1952), pp. 1–40.

[4] K.B. Athreya and P.E. Ney. *Branching Processes*. Berlin Heidelberg: Springer, 1972.

[5] M. S. Bartlett. *An introduction to stochastic processes, with special reference to methods and applications*. 3rd. Cambridge University Press, 1978.

[6] R. Bellman and T. Harris. "On age-dependent binary branching processes". In: *Ann. Math.* 55.2 (1952), pp. 280–295.

[7] L. Boe, T. Tolker-Nielsen, K. M. Eegholm, H. Spliid, and A. Vrang. "Fluctuation analysis of mutations to nalidixic acid resistance in *Escherichia Coli*". In: *J. Bacteriol.* 176.10 (1994), pp. 2781–2787.

[8] R. Breunig. "An almost unbiased estimator of the coefficient of variation". In: *Econ. Lett.* 70.1 (2001), pp. 15–19.

[9] H. L. David. "Probability distribution of drug-resistant mutants in unselected populations of *Mycobacterium tuberculosis*". In: *Appl. Microbiol.* 20.5 (1970), pp. 810–814.

[10] A. Dewanji, E.G. Luebeck, and S.H. Moolgavkar. "A generalized Luria-Delbrück model". In: *Math. Biosci.* 197.2 (2005), pp. 140–152.

[11] D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. New York: Springer, 2013.

[12] D. Eddelbuettel and C. Sanderson. "RcppArmadillo: Accelerating R with high-performance C++ linear algebra". In: *Comput. Stat. Data An.* 70 (2014), pp. 1054–1063.

[13] P. Embrechts and J. Hawkes. "A limit theorem for tails of discrete infinitely divisible laws with applications to fluctuation theory". In: *J. Austral. Math. Soc. Series A* 32 (1982), pp. 412–422.

[14] P.L. Foster. "Methods for Determining Spontaneous Mutation Rates". In: *Method. Enzymol.* 409 (2006), pp. 195–213.

[15] A. Gillet-Markowska, G. Louvel, and G. Fisher. "bz-rates: a web-tool to estimate mutation rates from fluctuation analysis". In: *G3* 5.11 (2015), pp. 2323–2327.

[16] B.M. Hall, C.X. Ma, P. Liang, and K.K. Singh. "Fluctuation AnaLysis CalculatOR: a web tool for the determination of mutation rate using Luria–Delbrück fluctuation analysis". In: *Bioinformatics* 25.12 (2009), pp. 1564–1565.

[17] A. Hamon and B. Ycart. "Statistics for the Luria-Delbrück distribution". In: *Elect. J. Statist.* 6 (2012), pp. 1251–1272.

[18] C. D. Kelly and O. Rahn. "The growth rate of individual bacterial cells". In: *J. Bacteriol.* 23.2 (1932), pp. 147–153.

[19] N. L. Komarova, L. Wu, and P. Baldi. "The fixed-size Luria-Delbrück model with a nonzero death rate". In: *Math. Biosci.* 210.1 (2007), pp. 253–290.

[20] D.E. Lea and C.A. Coulson. "The distribution of the number of mutants in bacterial populations". In: *Journal of Genetics* 49.3 (1949), pp. 264–285.

[21] S.E. Luria and M. Delbrück. "Mutations of bacteria from virus sensitivity to virus resistance". In: *Genetics* 28.6 (1943), pp. 491–511.

[22] W. T. Ma, G. v. H. Sandri, and S. Sarkar. "Analysis of the Luria-Delbrück distribution using discrete convolution powers". In: *J. Appl. Probab.* 29.2 (1992), pp. 255–267.

[23] B. Rémillard and R. Theodorescu. "Inference based on the empirical probability generating function for mixtures of Poisson distributions". In: *Statist. Decisions* 18 (2000), pp. 349–366.

[24] S. Sarkar. "Haldane's solution of the Luria-Delbrück distribution". In: *Genetics* 127 (1991), pp. 257–261.

[25] F.M. Stewart. "Fluctuation analysis: the effect of plating efficiency". In: *Genetica* 84.1 (1991), pp. 51–55.

[26] F.M. Stewart, D.M. Gordon, and B.R. Levin. "Fluctuation analysis: the probability distribution of the number of mutants under different conditions". In: *Genetics* 124.1 (1990), pp. 175–185.

[27] J. Werngren and S. E. Hoffner. "Drug susceptible *Mycobacterium tuberculosis Beijing* genotype does not develop motation-conferred resistance to Rifampin at an elevated rate". In: *J. Clin. Microbiol.* 41.4 (2003), pp. 1520–1524.

[28] R. Wilcox. *Introduction to Robust Estimation and Hypothesis Testing.* 3rd. Amsterdam: Elsevier, 2012.

[29] B. Ycart. "Fluctuation analysis: can estimates be trusted?" In: *PLoS One* 8.12 (2013), pp. 1–12.

[30] B. Ycart. "Fluctuation analysis with cell deaths". In: *J. Appl. Probab. Statist* 9.1 (2014), pp. 13–29.

[31] B. Ycart and N. Veziris. "Unbiased estimates of mutation rates under fluctuating final counts". In: *PLoS One* 9.7 (2014), pp. 1–10.

[32] Q. Zheng. "Statistical and algorithmic methods for fluctuation analysis with SALVADOR as an implementation". In: *Math. Biosci.* 176.2 (2002), pp. 237–252.

[33] Q. Zheng. "New algorithms for Luria-Delbrück fluctuation analysis". In: *Math. Biosci.* 196.2 (2005), pp. 198–214.