Algorithms for Molecular Biology BioMed Central

Research

Open Access

Effective p-value computations using Finite Markov Chain Imbedding (FMCI): application to local score and to pattern statistics

Grégory Nuel*

Address: Laboratoire Statistique et Génome, UEVE, CNRS (8071), INRA (1152), Evry, France

Email: Grégory Nuel* - nuel@genopole.cnrs.fr

* Corresponding author

Published: 07 April 2006

Algorithms for Molecular Biology 2006, **1**:5 doi:10.1186/1748-7188-1-5

Received: 15 February 2006
Accepted: 07 April 2006

This article is available from: http://www.almob.org/content/1/1/5

Abstract

The technique of Finite Markov Chain Imbedding (FMCI) is a classical approach to complex combinatorial problems related to sequences. In order to get efficient algorithms, it is known that such approaches need to be first rewritten using recursive relations. We propose here to give here a general recursive algorithms allowing to compute in a numerically stable manner exact Cumulative Distribution Function (CDF) or complementary CDF (CCDF). These algorithms are then applied in two particular cases: the local score of one sequence and pattern statistics. In both cases, asymptotic developments are derived. For the local score, our new approach allows for the very first time to compute exact p-values for a practical study (finding hydrophobic segments in a protein database) where only approximations were available before. In this study, the asymptotic approximations appear to be completely unreliable for 99.5% of the considered sequences. Concerning the pattern statistics, the new FMCI algorithms dramatically outperform the previous ones as they are more reliable, easier to implement, faster and with lower memory requirements.

1 Introduction

The use of Markov chains is a classical approach to deal with complex combinatorial computations related to sequences. In the particular case of pattern count on random sequences, [5] named this method Finite Markov Chain Imbedding (FMCI, see [11] or [7] for a review). Using this technique it is possible to compute exact distributions otherwise delicate to obtain with classical combinatorial methods. More recently, [12] proposed a similar approach to consider local score on i.i.d. or Markovian ([13]) random sequences. Although these methods are very elegant, they could require a lot of time and memory if they are implemented with a naive approach. The authors of [6] first stated that recursive relation could be established for any particular case in order to provide an efficient way to perform the computations. We propose here to explore in detail this idea with the aim to provide fast algorithms able to compute with high numerical accuracy both CDF (cumulative distribution function) and CCDF (complementary CDF) of any general problem which can be written as a FMCI. We apply then these results to the particular cases of local score and pattern statistics. In each case, asymptotic developments are derived and numerical results are presented.

2 Methods

In this part, we first introduce in section 2.1 the FMCI and see the limits of naive approaches to their corresponding numerical computations. The main results are given in section 2.3 where we propose two effective algorithms

able to to compute general FMCI p-values (algorithm 1) or complementary p-value (algorithm 2). The theoretical background for these algorithms is given in the section 2.2.

2.1 Finite Markov Chain Imbedding

Let us consider $X = X_1,...,X_n$ a sequence of Bernoulli or Markov observations and E_n an event depending on the sequence X. We suppose that it is possible to build from X an order one Markov chain $Z = Z_1,...,Z_n$ on the finite state space $\mathcal{S}$ of size L. This space contains (in the order): k starting states denoted $s_1,...,s_k$, some intermediate states, and one final absorbing state f. The Markov chain is designed such as

$$(E_n|Z_1 = s_i) = (Z_n = f|Z_1 = s_i) = \prod^{n-1}(s_i, f) \quad (1)$$

where

$$\Pi = \begin{pmatrix} & R & & v \\ 0 & \dots & 0 & 1 \end{pmatrix} \quad (2)$$

is the transition matrix of Z.

If μ is the starting distribution of Z_1, we hence get

$$\mathbb{P}(E_n) = \sum_{i=1}^{k} \mu(s_i)\Pi^{n-1}(s_i, f) \quad (3)$$

Using this approach (and a binary decomposition of n - 1), it is possible to compute the p-value with $O(\log_2(n) \times L^2)$ memory complexity and $O(\log_2(n) \times L^3)$ time complexity. As L usually grows very fast when we consider more complex events E_n, these complexities are a huge drawback of the method. Moreover, numerical precision considerations prevent this approach to give accurate results when using the relation $(E_n^c) = 1 - (E_n)$ to compute the p-value of the complementary event (as the absolute error is then equal to the relative precision of the computations).

2.2 Effective computations

Proposition 1. For all $n \geq 1$ we have

$$\Pi^n = \begin{pmatrix} & R^n & & \gamma^{n-1} \\ 0 & \dots & 0 & 1 \end{pmatrix} \text{ with } \gamma^{n-1} = \sum_{i=0}^{n-1} R^i v \quad (4)$$

Proof. This trivial to establish by recurrence using matrix block multiplications. □

We hence get the

Corollary 2 (direct p-value). For all $n \geq 1$ we have

for all $1 \leq i \leq k$ $(E_n|X_1 = s_i) = \gamma_i^{n-2}$ and

$$\mathbb{P}(E_n) = \sum_{i=1}^{k} \mu_i \gamma_i^{n-2} \quad (5)$$

with γ^{n-2} computable through the following recurrence relations:

$x^0 = \gamma^0 = v$ and, for all $j \geq 0$ $x^{j+1} = Rx^j$ and $\gamma^{j+1} = \gamma^j + x^j$ (6)

Proof. Simply use proposition 1 to rewrite equations (1) and (3). Recurrence relations are then obvious to establish. □

And we also get the

Corollary 3 (complementary p-value). For all $n \geq 1$ we have

for all $1 \leq i \leq k$ $(E_n^c|X_1 = s_i) = x_i^{n-1}$ and

$$\mathbb{P}(E_n^c) = \sum_{i=1}^{k} \mu_i x_i^{n-1} \quad (7)$$

with x^0 is a size L - 1 column vector filled with ones and with $x^{n-1} = R^{n-1}x^0$ which is computable through the following recurrence relation:

for all $j \geq 0$ $x^{j+1} = Rx^j$ (8)

Proof. $\prod$ being a stochastic matrix, $\prod^{n-1}$ is also stochastic, it is therefore clear that the sum of R^{n-1} over the columns gives 1 - γ^{n-2} and the corollary is proved. □

Using these two corollaries, it is therefore possible to accurately compute the p-value of the event or of its complementary with a complexity $O(L + \zeta)$ in memory and $O(n \times \zeta)$ in time where ζ is the number of non zero terms in the matrix R. In the worst case, $\zeta = (L - 1)^2$ but the technique of FMCI usually leads to a very sparse structure for R. One should note that these dramatic improvements from the naive approach could even get better by considering the structure of R itself, but this have to be done specifically for each considered problem. We will give detailed examples of this in both our application parts but, for the moment, we focus on the general case for which we give algorithms.

2.3 Algorithms

Using with the corollary 2 we get a simple algorithm to compute $p = (E_n)$

algorithm 1: direct p-value

x is a real column vector of size L - 1 and y a real column vector of size k

initialization $x = (v_1,...,v_{L-1})'$ and $y = (v_1,...,v_k)'$

main loop for $i = 1...n$ - 2 do

- $x = R \times x$ (sparse product)
- $y = y + (x_1,...,x_k)'$

end return $p = \sum_{i=1}^{k} \mu_i y_i$

and using the corollary 3 we get an even simpler algorithm to compute the $q = 1 - p = (E_n^c)$

algorithm 2: complementary p-value

x is a real column vector of size L - 1

initialization $x = (1,...,1)'$

main loop for $i = 1...n$ - 1 do

- $x = R \times x$ (sparse product)

end return $q = \sum_{i=1}^{k} \mu_i x_i$

The more critical stage of both these algorithms is the sparse product of the matrix R by a column vector which can be efficiently done with ζ operations.

It is interesting to point out the fact that these algorithms do not require the stationarity of the underlying Markov chain. More surprisingly, it is also possible to relax the random sequence homogeneity assumption. Indeed, if our transition matrix $\prod$ depends on the position i in the sequence, we simply have to replace R in the algorithms with the corresponding R_i (which may use a significant amount of additional memory depending on its expression as a function of i).

For complementary p-value, we require to compute $R_1R_2...R_{n-1}R_nx$ which is easily done recursively starting from the right. In the direct p-value case however, it seems more difficult since we need to compute $x + R_1x + R_1R_2x + ... + R_1R_2...R_{n-1}R_nx$. Fortunately this sum can be rewritten as $x + R_1(x + R_2\{... [x + R_{n-1}(x + R_nx)]...\})$ which is again easy to compute recursively starting from the right.

The resulting complexities in the heterogeneous case are hence the same than in the homogeneous one (assuming that the number of non zero terms in R_i remains approximately constant). This remarkable property of the FMCI should be remembered especially in the biological field where most sequences are known to have complex heterogeneous structures which are often difficult to take into account.

3 Application 1: local score

We propose in this part to apply our results to the computation of exact p-values for local score. We first recall the definition of the local score of one sequence (section 3.1) and design a FMCI allowing to compute p-value in the particular case of an integer and i.i.d. score (section 3.2). We explain in sections 3.5 and 3.6 how to relax these two restrictive assumptions to consider rational or Markovian scores. The main result of this part is given in section 3.4 where we propose an algorithm improving the simple application of the general ones by using a specific asymptotic behaviour presented in section 3.3. As numerical application, we propose finally in section 3.7 to find significant hydrophobic segments in the Swissprot database using the Kyte-Doolittle hydrophobic scale. Our exact results are compared to the classical Gumble asymptotic approximations and discussed both in terms of numerical performance and reliability.

3.1 Definition

We consider $S = S_1,...,S_n$ a sequence of real scores and we define the local score H_n of this sequence by

$$H_n = \max\left\{0, \max_{i,j}\left(\sum_{\ell=i}^{j} S_\ell\right)\right\} \qquad (9)$$

which is exactly the highest partial sum score of a subsequence of S.

This local score can be computed in $O(n)$ using the auxiliary process

$$U_0 = 0 \quad \text{and for } 1 \le j \le n \quad U_j = \max\left\{0, \max_{i}\left(\sum_{\ell=i}^{j} S_\ell\right)\right\}$$

$$= \max\{0, U_{j-1} + S_j\} \qquad (10)$$

because we then have $H_n = \max_j U_j$.

Assuming the sequence S is random (Bernoulli or Markov model), we want to compute p-values relative to the event $E_n = \{H_n \ge a\}$ where $a > 0$.

3.2 Integer score

In order to simplify, we will first consider the case of integer scores (and hence $a \in$) then we will extend the result to the case of rational scores.

In the Bernoulli case, [12] introduced the FMCI Z defined by

$$Z_0 = 0 \quad \text{and} \quad Z_j = \begin{cases} U_j & \text{if there is no } a \text{ in } U_0,\ldots,U_j \\ a & \text{else} \end{cases} \qquad (11)$$

(resulting with a sequence of length $n + 1$) with 0 as the only starting state and a as the final absorbing state. The transition matrix $\prod$ is given by

$$\Pi = \begin{pmatrix} f(0) & p(1) & \ldots & p(a-1) & g(a) \\ \vdots & \vdots & & \vdots & \vdots \\ f(-h) & p(1-h) & \ldots & p(a-h-1) & g(a-h) \\ \vdots & \vdots & & \vdots & \vdots \\ f(1-a) & p(2-a) & \ldots & p(0) & g(1) \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \qquad (12)$$

where

$$p(i) = (S_1 = i) \quad f(i) = (S_1 \le i) \quad g(i) = (S_1 \ge i) \quad \forall i \in \qquad (13)$$

It is possible to apply to this case the general algorithm 1 with $L = a + 1$ and $k = 1$ (please note that we have added Z_0 to the sequence and n must then be replaced by $n + 1$ in the algorithm to get correct computations) to compute the p-value we are looking for. In the worst case, R has $\zeta = a^2$ non zero terms and the resulting complexity is $O(a^2)$ in memory and $O(n \times a^2)$ in times. But in most cases, S_1 support is reduced to a small number of values and the complexities decrease accordingly.

3.3 Asymptotic development

Is it possible to compute this p-value faster ? In the case where R admits a diagonal form, simple linear algebra could help to cut off the computations and answer yes to this question.

Proposition 4. If R admits a diagonal form we have

$$\mathbb{P}(H_n \ge a) = \left[\sum_{i=0}^{\alpha-1} R^i v\right]_1 + \lambda^\alpha \frac{\left(1-\lambda^{n-\alpha}\right)}{(1-\lambda)}\left[R^\infty v\right]_1 + O\left(\nu^\alpha\right) \quad \forall n \ge \alpha \qquad (14)$$

where $[]_1$ denotes the first component of a vector, with $R^\infty = \lim_{i\to\infty} R^i/\lambda^i$, where $0 < \lambda < 1$ is the largest eigenvalue of R and ν is the magnitude of the second largest eigenvalue. We also have $v = [g(a),\ldots,g(1)]'$.

Proof. By using the corollary 15 (appendix A) we know that

$$R^i - \lambda^i R^\infty = O(\nu^i) \qquad (15)$$

uniformly in i so we finally get for all α

$$\sum_{i=0}^{n-1} R^i v - \sum_{i=0}^{\alpha-1} R^i v - \sum_{i=\alpha}^{n-1} \lambda^i R^\infty v = \sum_{i=\alpha}^{n-1} O\left(\nu^i\right) = O\left(\nu^\alpha \frac{\left(1-\nu^{n-\alpha}\right)}{(1-\nu)}\right) = O\left(\nu^\alpha\right) \qquad (16)$$

uniformly for all $n \ge \alpha$ and the proposition is then proved by considering the first component of equation (16). □

Corollary 5. We have

$$\lim_{n\to\infty} \frac{\mathbb{P}(H_{n+1} \ge a) - \mathbb{P}(H_n \ge a)}{\lambda^n} = \left[R^\infty v\right]_1 \qquad (17)$$

and

$$\lim_{n\to\infty} \frac{\mathbb{P}(H_{n+2} \ge a) - \mathbb{P}(H_{n+1} \ge a)}{\mathbb{P}(H_{n+1} \ge a) - \mathbb{P}(H_n \ge a)} = \lambda \qquad (18)$$

Proof. Simply replace the terms in (17) and (18) with equation (14) to get the results. □

3.4 Algorithm

The simplest way to compute $(H_n \ge a)$ is to use the algorithm 2 in our particular case. As the number of non zero terms in R is then a^2, the resulting complexity is $O(n \times a^2)$. Using the proposition 4, it possible to get the same result a bit faster on very long sequence by computing the first two largest eigenvalues magnitudes λ and ν (complexity in $O(a^2)$ with Arnoldi algorithms) and to use them to compute a p-value.

As the absolute error is in $O(\nu^\alpha)$ we obtain a require ε error level using a α proportional to $\log(\varepsilon)/\log(\nu)$ which results in a final complexity in $O(\log(\varepsilon)/\log(\nu) \times a^2)$. Unfortunately, this last method requires to use delicate linear algebra techniques and is therefore more difficult to implement. Another better possibility is to use the corollary 5 to get the following fast and easy to implement algorithm:

algorithm 3: local score p-value

x a real column vector of size a, $(p_i)_{i\ge1}$ and $(\lambda_i)_{i\ge3}$ to sequences of real and i an integer

initialization $x = [g(a),\ldots,g(1)]'$, $p_1 = g(a)$, and $i = 0$

main loop while ($i < n$ and (λ_i) has not yet converged towards λ)

- $i = i + 1$
- $x = R \times x$ (sparse product)

- $p_i = p_{i-1} + x_1$
- $\lambda_i = (p_i - p_{i-1})/(p_{i-1} - p_{i-2})$ (if defined)

end • $p = p_i$

- if $(i < n)$ then $p = p + (p_i - p_{i-1}) \dfrac{\left(1-\lambda^{n-i}\right)}{(1-\lambda)}$
- return p

At any step i of the main loop we have $p_i = (H_i \geq a)$ and the final value taken by i is the α of proposition 4. One should note that only the last three terms of $(p_i)_{i\geq 1}$ and (for a simple convergence testing) the last two terms of $(\lambda_i)_{i\geq 3}$ are required by the algorithm.

3.5 Rational scores

What if we consider now a rational score instead of an integer one ? If we denote by $\mathcal{S} \subset$ the support of S_1, let us define $M = \min_{i\in}\{i\,\mathcal{S} \subset\}$. Changing the scale of the problem by the factor M allows us to get back to the integer case:

$$(H_n \geq a) = (M\,H_n \geq M\,a) \quad (19)$$

This scale factor will obviously increase the complexity of the problem, but as the support cardinal (denoted η) is not changed during the process, the resulting complexities are $O(M \times a \times \eta)$ in memory and $O(M \times n \times a \times \eta)$ in time (n could vanish from the time complexity thanks to the faster algorithm presented above).

For example, if we consider the Kyte-Doolittle hydrophobicity score of the amino-acids (see [10] and table 1), it takes only $\eta = 20$ values and $M = 10$, the resulting complexity to compute $(H_n \geq a)$ is then $O(200 \times n \times a)$. If we consider now the more refined Chothia score ([4]), the scale factor increases from $M = 10$ to $M = 100$ and the resulting complexities are multiplied by 10.

3.6 Markov case

All these results can be extended to the Markov case but this require to define a new FMCI allowing us to trace the last score (in the case of an order one Markov chain for the sequence S, if a higher order m is considered, we just have to add the corresponding number of preceding scores to Z instead of one):

$$Z_j = \begin{cases} \left(S_{j-1}, U_j\right) & \text{if there is no } a \text{ in } U_0,\ldots,U_j \\ f & \text{else} \end{cases} \quad (20)$$

Table 1: Distribution of amino-acids estimated on Swissprot (release 47.8) database and Kyte-Doolittle hydrophobic scale. Mean score is -0.244.

a. a.	F	M	I	L	V	C	W	A	T	G
in %	4.0	2.4	5.9	9.6	6.7	1.5	1.2	7.9	5.4	6.9
score	2.8	1.9	4.5	3.8	4.2	2.5	-0.9	1.8	-0.7	-0.4
a. a.	S	P	Y	H	Q	N	E	K	D	R
in %	6.9	4.8	3.1	2.3	3.9	4.2	6.6	5.9	5.3	5.4
score	-0.8	-1.6	-1.3	-3.2	-3.5	-3.5	-3.5	-3.9	-3.5	-4.5

Doing this now we get $k = \eta$ (the cardinal of the score support) starting states instead of one so we need a starting distribution μ (which could be a Dirac) to compute the p-value.

We will not detail here the structure of the corresponding sparse transition matrix $\prod$ (see [13]) but we need to know its number ζ of non zero terms. If a is an integer value (we suppose here that the scale factor has been already included in it) then the order of R is $M \times a \times \eta$ and $\zeta = O(M \times a \times \eta^2)$ (and we get $O(M \times a \times \eta^{m+1})$ when an order m Markov model is considered).

3.7 Numerical results

In this section, we apply the results presented above to a practical local score study. We consider the complete protein database of Swissprot release 47.8 and the classical amino acid hydrophobic scale of Kyte-Doolittle given in table 1 ([10]). The database contains roughly 200 000 sequences of various lengths (empiric distribution given in figure 1).

Once the best scoring segment has been determined for each of these sequences, we need to compute the corresponding p-values. According to [9], the asymptotic distribution of H_n is given (if mean score is < 0, which is precisely the case here) by the following conservative approximation:

$$(H_n \geq a) \simeq 1 - \exp\left(-nKe^{-a\lambda}\right) \quad (21)$$

where constants λ and K depend on the scoring distribution.

With our hydrophobic scale and a distribution of amino-acids estimated on the entire database we get

$\lambda = 5.144775 \times 10^{-3}$ and $K = 1.614858 \times 10^{-2}$

(computation performed with a C function implemented by Altschul). Once the constants are computed we could get all the approximated p-values very quickly (a few seconds for the 200 000 p-values).

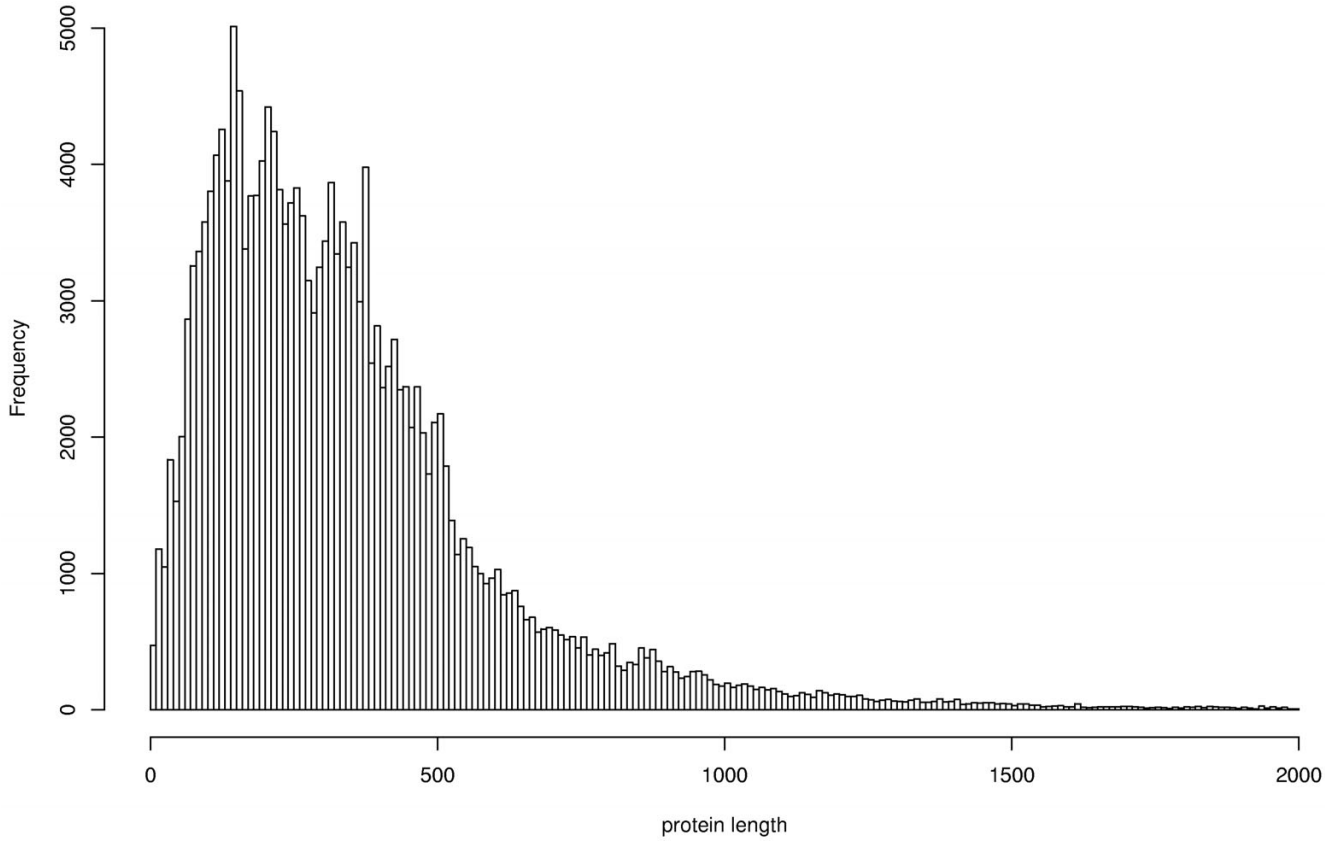

Figure 1
Empiric distribution of Swissprot (release 47.8) protein lengths. In order to improve readability, 0.5% of sequences with length $\in$ [2 000, 9 000] have been removed from this histogram.

On the other hand, our new algorithm allows to compute (for the very first time) the exact p-values for this example. As the chosen scoring function has a one digit precision level, we need to use a scale factor of $M = 10$ to fall back to the integer case. A C++ implementation (available on request) performed all the computations in roughly three hours on a Pentium 4 CPU 2.8 GHz (this means approximately 20 p-values computed by second).

We can see on figure 2 the comparison between exact values and Karlin's approximations. The conservative design of the approximations seems to be successful except for very short unsignificant sequences. While the approximations are rather close to perfection for sequences with more than 2 000 amino-acids, the smaller the sequence is, the worse the approximations get. This is obviously consistent with the asymptotic nature of Karlin's formula but seems to indicate that these approximations are not reliable for 99.5% of the sequence in the database (protein of length < 2 000).

One should object that it exists ([1,2]) a well known finite size correction to formula (21) that might be useful, especially when considering short sequences. Unfortunately in our case, this correction does not seems to improve the quality of the approximations (data not shown) and we hence make the choice to ignore it.

In table 2 we compare the number of sequences predicted to have a significant hydrophobic segment at a certain e-value level by the two approaches. If the Karlin's approximations are used, many proteins are considered unsignificant while they are. For example, with the classical database threshold of 10^{-5}, only few sequences (6%) are correctly identified by Karlin's approximations.

We have seen that Karlin's approximations are often far too conservative to give accurate results, but what about the ranking ? Table 3 proposes the Kendall's tau rank correlation (see [16] chapter 14.6 for more details) which is equal to 1.0 for a complete rank agreement and equal to -

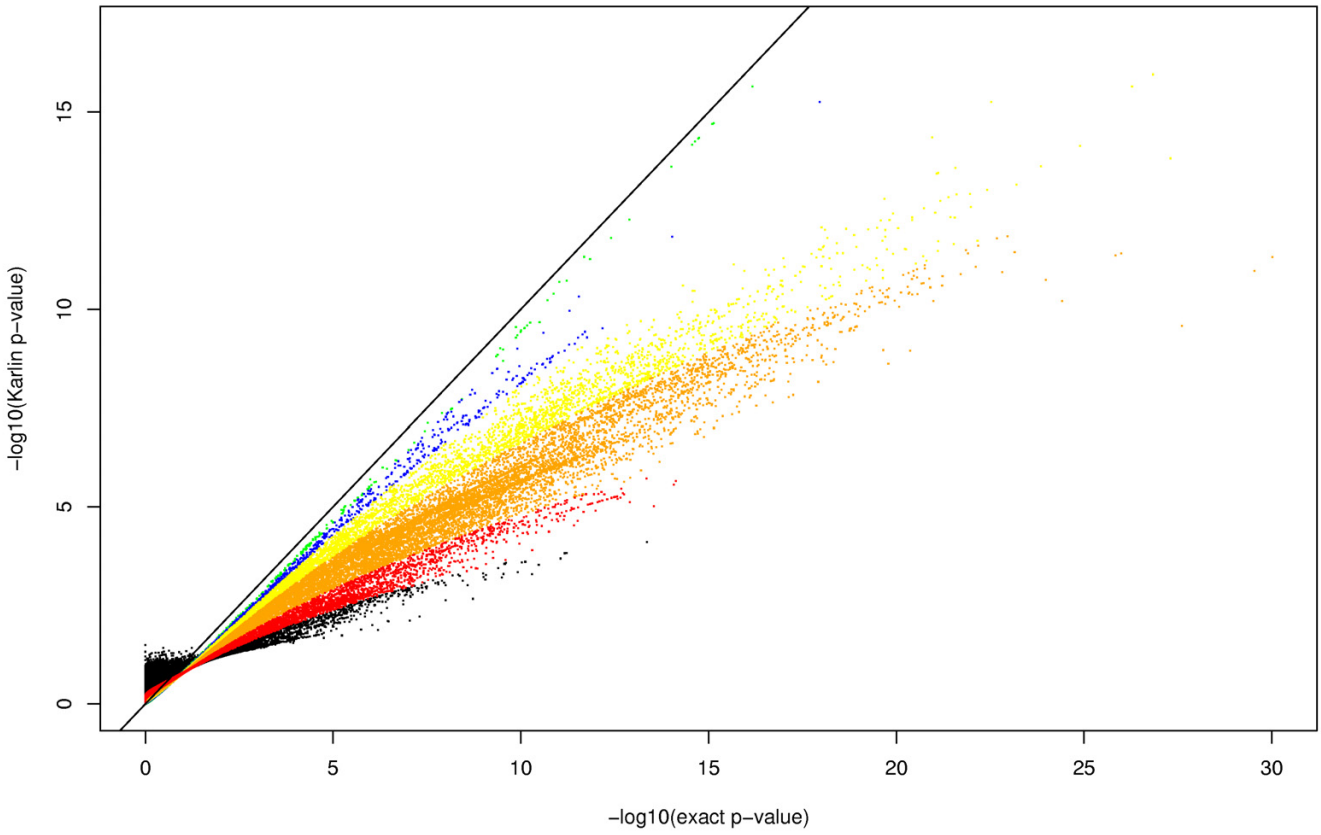

Figure 2
Exact p-value against Karlin ones (in log scale). Color refers to a range of sequence lengths: smaller than 100 in black (≃ 20 000 sequences), between 100 and 200 in red (≃ 40 000 sequences), between 200 and 500 in orange (≃ 90 000 sequences), between 500 and 1000 in yellow (≃ 30 000 sequences), between 1000 and 2 000 in blue (≃ 6 000 sequences) and greater than 2 000 in green (≃ 1 000 sequences). The solid line represents $y = x$. Range have been chosen for readability and few dots with exact p-value smaller than 10^{-30} are hence missing.

1.0 for a complete inverse rank agreement. As we will certainly be interested in the most significant sequences produced by our study, we compute our Kendall's tau only on these sequences. When all sequence lengths are considered, Karlin's approximations show their total irrelevance to give correct ranking for the first 10 or 50 most significant p-values. Even when the 100 first p-values are taken into account, relative ranks given by Karlin's approximations are wrong in 63% of the cases, which is huge. However, in the case where the approximations values are close to the exact ones (sequence lengths greater than 2 000, which correspond only to 0.5% of the database), p-values obtained with both methods are highly correlated.

4 Application 2: pattern statistics

In this part, we consider the application of FMCI to pattern statistics. After a short introduction of notations (section 4.1) we explain with an example in section 4.2 how to build through the tool of DFA a particular FMCI related to a given pattern. The block structure of this FMCI (section 4.3) is then used to get in section 4.4 two efficient algorithms for under- and over-represented patterns. We derive in section 4.5 some asymptotic developments but unlike with local score application, these results are not used to improve our algorithms. In the last section 4.6 we finally compare this new method to existing ones.

4.1 Definition

Let us consider a random order m homogeneous Markov sequence $X = X_1,...,X_n$ on the finite alphabet $\mathcal{A}$ (cardinal k). If N_i is the random variable counting the number of occurrences (overlapping or renewal) of a given pattern in $X_1...X_i$. We define the pattern statistic associated to any number $N_{\text{obs}} \in$ of observations by

Table 2: Number of e-value smaller than a threshold are given for exact computations (exact) and asymptotic Karlin's approximations (Karlin). The last row gives the accuracy of asymptotic predictions (accuracy = Karlin/exact).

e-value	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
exact	9473	7772	6271	4563	3232	2348
Karlin	3417	2047	1056	439	195	96
accuracy	34%	26%	17%	10%	6%	4%

$$S = \begin{cases} -\log_{10} \mathbb{P}(N_n \geq N_{obs}) & \text{if } N_n \geq \mathbb{E}[N_n] \\ +\log_{10} \mathbb{P}(N_n \leq N_{obs}) & \text{if } N_n < \mathbb{E}[N_n] \end{cases} \quad (22)$$

This way, a pattern has a positive statistic if it is seen more than expected, a negative statistic if seen less than expected and, in both cases, the corresponding p-value is given (in log scale) by the magnitude of the statistic.

The problem is: how to compute this statistic ?

4.2 DFA

We first need to construct a Deterministic Finite state Automaton (DFA) able to count our pattern occurrences. It is a finite oriented graph such as all vertexes have exactly k arcs starting from them each one tagged with a different letter of $\mathcal{A}$. One or more arcs are marked as counting ones. By processing a sequence X in the DFA, we get a sequence Y (of vertexes) in which the words of length 2 corresponding to the counting transitions occur each time a pattern occurs in X.

Example: If we consider the pattern aba.a (. means "any letter") on the binary alphabet $\mathcal{A} = \{a, b\}$. We define vertex set $\mathcal{V} = \{a, b, ab, aba, abaa, abab\}$ and then the structure of the DFA counting the overlapping occurrences (set of vertexes and structure would have been slightly different in the renewal case) of the pattern is given by

tag \ vertex	a	b	ab	aba	abaa	abab
a	a	ab	b	aba	abaa	abab*
b	ab	b	b	abab	ab	b

Table 3: Kendall's tau (rank correlation) comparing the most significant exact p-values (the reference) to the Karlin's approximations. The column "all" gives the result for all sequences while the R, give the results for a certain range of sequence lengths: smaller than 100 for R1, between 100 and 200 for R2, between 200 and 500 for R3, between 500 and 1 000 for R4, between 1 000 and 2 000 for R5 and greater than 2 000 for R6.

number of p-values	all	R1	R2	R3	R4	R5	R6
10	0.30	0.64	0.24	-0.20	0.58	0.64	0.97
50	0.14	0.73	0.50	0.46	0.56	0.78	0.97
100	0.37	0.70	0.67	0.62	0.61	0.80	0.98

(the counting arcs are denoted by a star). In the sequence

X = a a b b $\underline{\text{a b }\overline{\text{a b a}}\text{ a a}}$ b b $\underline{\text{a b a a a}}$ a b

of length n = 20, the pattern occurrences end in positions 9,11 and 18. Processing this sequence into the DFA gives

Y = a a ab b a ab aba $\underline{\text{abab aba}}$ $\overline{\text{abaa a}}$ ab b a ab aba $\underline{\text{abaa a}}$ a ab

which is a sequence of the same length as X, where occurrences of the pattern end exactly in the same positions.

If X is an homogeneous order one Markov chain, so is Y and its transition matrix is given by $P + Q$ where P contains the non counting transitions and Q the counting ones:

$$P = \begin{pmatrix} \mathbb{P}(a|a) & 0 & \mathbb{P}(b|a) & 0 & 0 & 0 \\ \mathbb{P}(a|b) & \mathbb{P}(b|b) & 0 & 0 & 0 & 0 \\ 0 & \mathbb{P}(b|b) & 0 & \mathbb{P}(a|b) & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbb{P}(a|a) & \mathbb{P}(b|a) \\ 0 & 0 & \mathbb{P}(b|a) & 0 & 0 & 0 \\ 0 & \mathbb{P}(b|b) & 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbb{P}(a|a) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbb{P}(a|b) & 0 & 0 \end{pmatrix}$$

It is therefore possible to work on Y rather than on X to compute the pattern statistics. In order to do that, it is very natural to use the large deviations (in this case, computations are closely related to the largest eigenvalue of the matrix $T_\theta = P + Qe^\theta$) but other methods can be used as well (binomial or compound Poisson approximations for example).

This method easily extends to cases where X is an order $m > 1$ Markov chain by modifying accordingly our vertex set. For example, if we consider an order $m = 2$ Markov model our vertex set becomes

$\mathcal{V}$ = {aa, ab, ba, bb, aba, abaa, abab}

In all cases, if we denote by L the cardinal of $\mathcal{V}$. In order to count overlapping occurrences of a non degenerate pattern of length h on a size k alphabet we get $L = k + h - 2$ when an order 1 Markov model is considered and $L = k^m + h - m - 1$ for an order $m > 1$ Markov model. For a degenerate pattern of length h, L is more difficult to know as it depends on the degeneracy of the patterns, in the worst case $L = k^{h-1}$, but L should be far smaller in most cases. One should note that L increases by the number of different words present in the pattern if we consider renewal occurrences instead of overlapping ones.

Although construction and properties of DFA are well known in the theory of language and automata ([8]), their connexions to pattern statistics have surprisingly not been extensively studied in the literature. In particular, the strong relation presented here between the FMCI technique for pattern and DFA appears to have never been highlighted before. If this interesting subject obviously need to (and will soon) be investigated more deeply, it is not really the purpose of this article which focus more on the algorithmic treatment of a built FMCI.

4.3 FMCI

Once a DFA and the corresponding matrices P and Q have been built, it is easy to get a FMCI allowing to compute the p-values we are looking for.

Let us consider

$$Z_i = \begin{cases} (Y_j, N_j) & \text{if } N_j < a \\ f & \text{if } N_j \geq a \end{cases} \qquad (23)$$

where Y_j is the sequence of vertexes, N_j is the number of pattern occurrences in the sequence $Y_1...Y_j$ (or $X = X_1...X_j$ as it is the same), where f is the final (absorbing state) and where $a \in$ is the observed number of occurrences N_{obs} if the pattern is over-represented and $N_{\text{obs}} + 1$ if it is under-represented.

The transition matrix of the Markov chain Z is then given by:

$$\Pi = \begin{pmatrix} & (R_{i,j}) & & (v_i) \\ 0 & \dots & 0 & 1 \end{pmatrix} \qquad (24)$$

where for all size L blocks i, j we have

$$R_{i,j} = \begin{cases} P & \text{if } j = i \\ Q & \text{if } j = i+1 \\ 0 & \text{else} \end{cases} \quad \text{and} \quad v_i = \begin{cases} \Sigma_Q & \text{if } i = a \\ 0 & \text{else} \end{cases} \qquad (25)$$

with Σ_Q, the column vector resulting from the sum of Q.

By plugin the structure of R and v in the corollaries 2 and 3 we get the following recurrences:

Proposition 6. For all $n \geq 1$ and $1 \leq i \leq k$ we have

$$\mathbb{P}(N_n < a \mid X_1 = i) = \left[u^{n-1}\right]_i \quad \text{and} \quad \mathbb{P}(N_n \geq a \mid X_1 = i) = \sum_{j=0}^{n-2} \left[v^j\right]_i \qquad (26)$$

where for $x = u$ or v we have $\forall j \geq 0$ the following size L block decomposition: $x^j = \left(x^j_{(a-1)}, \dots, x^j_0\right)'$ and we have the recurrence relations:

$$x_0^{j+1} = P x_0^j \quad \text{and} \quad \forall i \geq 1 \quad x_i^{j+1} = P x_i^j + Q x_{i-1}^j \qquad (27)$$

with $u^0 = (1...1)'$ and $v^0 = v$.

4.4 Algorithms

Using the proposition 6 it is possible to get an algorithm computing our pattern statistic for an under-represented pattern observed N_{obs} times:

algorithm 4under: exact statistics for under-represented pattern

$x_0, \dots, x_{N_{\text{obs}}}$ and $y_0, \dots, y_{N_{\text{obs}}}$ are $2 \times (N_{\text{obs}} + 1)$ real column vectors of size L

initialization for $j = 0...N_{\text{obs}}$ do $x_j = (1,...,1)'$

main loop for $i = 1...(n - 1)$ do

- for $j = 0...N_{\text{obs}}$ do $y_j = x_j$
- $x_0 = P \times y_0$
- for $j = 1...N_{\text{obs}}$ do $x_j = P \times y_j + Q \times y_{j-1}$

end • $q = \sum_{i=1}^{k} \mu_i \left[x_{N_{\text{obs}}}\right]_i$

- return $\log_{10}(q)$

If we consider now an over-represented pattern we get

algorithm 4over: exact statistics for over-represented pattern

$x_1,...,x_{N_{obs}}$, $y_1,...,y_{N_{obs}}$ and z are $2N_{obs} + 1$ real column vectors of size L

initialization $z = (0,...,0)'$, $x_1 = \Sigma_Q$ and for $j = 2...N_{obs}$ do $x_j = (0,...,0)'$

main loop for $i = 1...(n - 2)$ do

- for $j = 1...N_{obs}$ do $y_j = x_j$
- $x_1 = P \times y_1$
- for $j = 2...N_{obs}$ do $x_j = P \times y_j + Q \times y_{j-1}$
- $z = z + x_{N_{obs}}$

end • $p = \sum_{i=1}^{k} \mu_i [z]_i$

- return $-\log_{10}(p)$

As we have $O(k \times L)$ non zero terms in $P + Q$, the complexity of both of these algorithms is $O(k \times L + N_{obs} \times L)$ in memory and $O(k \times L \times n \times N_{obs})$ in time.

To compute p-values out of floating point range (ex: smaller than 10^{-300} with C double), it is necessary to use log computations in the algorithms (not detailed here). The resulting complexity stays the same but the empirical running time is obviously slower. That is why we advise to use log-computation only when it is necessary (for example by considering first a rough approximation).

4.5 Asymptotic developments

In this part we propose to derive asymptotic developments for pattern p-values from their recursive expressions. For under- (resp. over-) represented patterns, the main result is given in theorem 9 (resp. 12). In both cases, theses results are also presented in a simpler form (where only main terms are taken into account) in the following corollaries.

Proposition 7. For any $x = (x_{(a-1)},...,x_0)'$ and all $\beta \geq 0$ $x^\beta = R^\beta x$ is given by $x_0^\beta = P^\beta x_0^0$ and

$$\forall 1 \leq i < a \quad x_i^\beta = P^\beta x_i^0 + \sum_{j=1}^{\beta} P^{j-1} Q x_{i-1}^{\beta-j} = P^\beta x_i^0 + \sum_{j=1}^{\beta} P^{\beta-j} Q x_{i-1}^{j-1} \tag{28}$$

Proof. As $x_0^{j+1} = P x_0^j$ for all $j \leq 0$ it is trivial to get the expression of x_0^β. If we suppose now that the relation (28) is true for some i and β then, thanks to the relation (27) we have

$$x_i^{\beta+1} = P x_i^\beta + Q x_{i-1}^\beta \tag{29}$$

$$= P^{\beta+1} x_i^0 + P\left(\sum_{j=1}^{\beta} P^{\beta-j} Q x_{i-1}^j \right) + Q x_{i-1}^\beta \tag{30}$$

$$= P^{\beta+1} x_i^0 + \sum_{j=1}^{\beta+1} P^{\beta+1-j} Q x_{i-1}^j \tag{31}$$

and so the proposition is proved through the principle of recurrence. □

Lemma 8. For all $i \geq 0$ and $a \leq b \in$ and $r > 0$ we define

$$P_a^b(r,i) = \sum_{j=a}^{b} r^j (j-1)^i \tag{32}$$

If $r \neq 1$ we have for all $i \geq 0$ we have

$$(1-r) P_a^b(r,i) = r^a (a-1)^{i+1} - r^{b+1} b^{i+1} - r \sum_{d=0}^{i-1} C_i^d P_a^b(r,d) \tag{33}$$

and (case $r = 1$) for all $i \geq 0$ we have

$$(i+1) P_a^b(1,i) = b^{i+1} - (a-1)^{i+1} - \sum_{d=0}^{i-1} C_{i+1}^d P_a^b(1,d) \tag{34}$$

Proof. Easily derived from the following relation

$$P_a^b(r,i+1) = \sum_{j=a-1}^{b-1} r^{j+1} j^{i+1} = r^a (a-1)^{i+1} - r^{b+1} b^{i+1} + r \sum_{d=0}^{i+1} C_{i+1}^d P_a^b(r,d) \tag{35}$$

Theorem 9. If P is primitive and admits a diagonal form we denote by $\lambda > \nu$ the largest two eigenvalues magnitude of P by $P^\infty = \lim_{i \to +\infty} P^i / \lambda^i$ (a positive matrix) and we get for all $\alpha \geq 1$ and $i \geq 0$

$$x_i^\beta = \lambda^\beta D_i^\alpha(\beta) + O\left(\nu^\alpha \beta^i \lambda^\beta \right) \quad \forall \beta \geq (i+1)\alpha \tag{36}$$

uniformly in β and where D_i^α is a polynomial of degree i which is defined by $D_0^\alpha(\beta) = P^\infty x_0^0$ and for all $i \geq 1$ by the following recurrence relation:

$$D_i^\alpha(\beta) = P^\infty x_i^0 + \sum_{j=1}^{\alpha} \lambda^{-j} P^{j-1} Q D_{i-1}^\alpha(\beta - j) + \sum_{j=1}^{i\alpha} \lambda^{-j} P^\infty Q x_{i-1}^{j-1} + \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{-1} P^\infty Q D_{i-1}^\alpha(j-1) \tag{37}$$

Proof. See appendix B. □

Corollary 10. With the same assumptions than in the theorem 9, for all $\alpha \geq 1$ and $\beta \geq (i+1)\alpha$ we have

$$x_i^\beta = \beta^i \lambda^\beta \left(\lambda^\beta \frac{\left(P^\infty Q\right)^i P^\infty x_0^0}{i!\lambda^i} + O\left(\frac{1}{\beta}\right) + O\left(\nu^\alpha\right) \right) \qquad (38)$$

Proof. Equation (37) and the lemma 8 gives

$$\left[D_i^\alpha\right]_i = \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{-1} P^\infty Q \left[D_{i-1}^\alpha\right]_{i-1} (j-1)^{i-1} = \frac{P^\infty Q}{i\lambda}\left[D_{i-1}^\alpha\right]_{i-1} \qquad (39)$$

and the result is then proved by a simple recurrence. □

Proposition 11. For any $x = (x_{(a-1)},\ldots,x_0)'$ and all $\beta \geq 0$, $\gamma^\beta = \left(\gamma_{(a-1)}^\beta,\ldots,\gamma_0^\beta\right)' = \sum_{b=0}^{\beta} R^\beta x$ is given by $\gamma_0^\beta = \sum_{b=0}^{\beta} P^b x_0^0$ and

$$\forall 1 \leq i < a \quad \gamma_i^\beta = \sum_{b=0}^{\beta} P^b x_i^0 + \sum_{j=1}^{\beta} P^{j-1} Q \gamma_{i-1}^{\beta-j} = \sum_{b=0}^{\beta} P^b x_i^0 + \sum_{j=1}^{\beta} P^{\beta-j} Q \gamma_{i-1}^{j-1} \qquad (40)$$

Proof. Using equation (28) we get

$$\gamma_i^\beta = \sum_{b=0}^{\beta} P^b x_i^0 + \sum_{b=0}^{\beta}\sum_{j=1}^{b} P^{b-j} Q x_{i-1}^{j-1} \qquad (41)$$

$$= \sum_{b=0}^{\beta} P^b x_i^0 + \sum_{j=1}^{\beta}\sum_{b=0}^{\beta-j} P^{j-1} Q x_{i-1}^b \qquad (42)$$

which gives the proposition. □

From this result (very similar to proposition 7) it is possible to get a new theorem

Theorem 12. If P is primitive and admits a diagonal form we denote by $\lambda > \nu$ the largest two eigenvalues magnitude of P by $P^\infty = \lim_{i\to+\infty} P^i/\lambda^i$ (a positive matrix) and we get for all $\alpha \geq 1$ and $i \geq 0$

$$\gamma_i^\beta = A_i^\alpha + \lambda^\beta B_i^\alpha(\beta) + O\left(\nu^\alpha\right) + O\left(\nu^\alpha \beta^i \lambda^\beta\right) \quad \forall \beta \geq (i+1)\alpha \qquad (43)$$

uniformly in β and where A_i^α is a constant term defined by

$$A_0^\alpha = \sum_{b=0}^{\alpha-1} P^b x_0^0 + \frac{\lambda^\alpha}{1-\lambda} P^\infty x_0^0 \qquad (44)$$

and for all $i \geq 0$ by the following recurrence relation

$$A_i^\alpha = \sum_{b=0}^{\alpha-1} P^b x_i^0 + \frac{\lambda^\alpha}{1-\lambda} P^\infty x_i^0 + \sum_{j=1}^{\alpha} P^{j-1} Q A_{i-1}^\alpha + \frac{\lambda^\alpha}{1-\lambda} P^\infty Q A_{i-1}^\alpha \qquad (45)$$

and B_i^α is a polynomial of degree i which is defined by

$$B_0^\alpha(\beta) = -\frac{\lambda}{1-\lambda} P^\infty x_0^0 \qquad (46)$$

and for all $i \geq 1$ by the following recurrence relation:

$$B_i^\alpha(\beta) = -\frac{\lambda}{1-\lambda} P^\infty x_i^0 - \frac{\lambda^{-i\alpha}}{1-\lambda} P^\infty Q A_{i-1}^\alpha + \sum_{j=1}^{\alpha} \lambda^{-j} P^{j-1} Q B_{i-1}^\alpha(\beta - j) + \sum_{j=1}^{i\alpha} \lambda^{-j} P^\infty Q \gamma_{i-1}^{j-1} + \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{-1} P^\infty Q B_{i-1}^\alpha(j-1) \qquad (47)$$

Proof. Easy to derive from the proof of theorem 9. □

Corollary 13. We have the same assumptions than in the the theorem 12, for all $\alpha \geq 1$ and $\beta \geq (i+1)\alpha$ we have

$$\gamma_i^\beta = A_i^\alpha - \frac{\lambda}{1-\lambda} \beta^i \lambda^\beta \left(\frac{\left(P^\infty Q\right)^i P^\infty x_0^0}{i!\lambda^i} + O\left(\frac{1}{\beta}\right) + O\left(\nu^\alpha\right) \right) + O\left(\nu^\alpha\right) \qquad (48)$$

Proof. Easy to derive from the proof of corollary 10. □

4.6 Numerical results

We propose here to consider numerical applications of these new FMCI pattern statistics algorithms and to compare their results and performance to exact computations using Simple Recurrences ([17] and [18]) denoted SR from now.

All computations are performed using SPatt-1.2.0 package (see [14] or [15]) on a 2.7 Gz P4 with 512 Mo running the linux 2.6.8 system.

As we can see on table 4, the computational time to obtain the pattern statistics for all simple words of a given length are quite similar with both approaches. One exception: the Bernoulli case ($m = 0$) where SR are roughly 10 times faster than FMCI computations. This is due to the fact that the Bernoulli case uses simpler recurrences than the ones used for the true Markovian cases ($m \geq 1$). Similar simplifications in the DFA structures can reduce the computational time of FMCI approach in the independent case but they have not been implemented here (as their use is often marginal).

If we consider now degenerate patterns instead of simple words (see table 5), FMCI approach clearly outperforms the SR one. Nevertheless, as considering degenerated patterns roughly multiply their observed number of occurrences by the alphabet size for each inde-termination, the corresponding computational time grows in the same manner which usually limits the use of high degenerated patterns in practical cases.

Table 4: Computational time (in seconds) to get the statistics of all DNA words of length *h* in the HIV complete genome sequence (*n* = 9719) using either simple recurrences of finite Markov chain imbedding and in respect with an order *m* Markov model estimated on the genome.

Markov order word length	*m* = 0			*m* = 1			*m* = 2		
	h = 3	*h* = 4	*h* = 5	*h* = 3	*h* = 4	*h* = 5	*h* = 3	*h* = 4	*h* = 5
SR	3	4	5	61	59	62	104	102	106
FMCI	39	45	52	39	44	52	96	102	113

Another interesting point is the memory requirements of the two approaches. Exact computations using SR have a $O(n + \alpha \times k^{2m})$ memory complexity where n is the sequence length, k the alphabet size, m the Markov model order and α which depends on the convergence rate of the model towards its stationary distribution. As a consequence, SR is difficult to use in practice with $m > 3$ for DNA words or $m > 1$ for protein ones. For FMCI computations, the memory requirements remain very cheap and in practice, any Markov model that fit in memory can be considered.

What about the reliability of the two methods. Once the pattern DFA has been computed, the FMCI algorithms are very simple to implement and have a high numerical stability. On the other hand, SR algorithms are quite more complicated (especially for degenerated patterns) to implement and require to approximate the iterate power of the Markov transition by the stationary distribution for large iterates. Classical convergence issues could result then to some numerical instability when high Markov orders are considered. As a consequence, FMCI results are taken as references from this point.

In table 6 we can see that for p-values larger than 10^{-300} the results given by both methods are exactly the same when we consider order 0 Markov models. As smaller p-values are not well managed by C double precision computation (the exact limit depends on the system), we get wrong results unless log computations are used. Such computations have been implemented for FMCI algorithms (they are quite simple) but not for SR ones (where it is quite more complicated) which explain the differences for patterns at and tcgatc.

When we consider order $m > 0$ Markov models, the numerical approximations done on the iterate power of the transition matrix lead to some errors. For order 1 Markov model, these errors remain quite small, but when order 2 model are considered it is more sensitive. In both cases, the larger the statistic to compute is, the greater the errors made are.

5 Conclusion

We proposed in this paper two general algorithms allowing to compute quickly and in a stable numerical way any p-value that can be imbedded in a finite Markov chain. We used these algorithms in two applications: local score on one sequence and pattern statistics.

For local score, the resulting algorithms reduce dramatically the complexity of previously proposed naive ones allowing for the very first time to produce exact computations for practical biological studies (Kyte-Doolittle hydrophobic scale on the Swissprot database). Comparing the results to the classical and very popular Karlin's approximations, it appears that these approximations require long sequences (length greater than 2 000) which can dramatically reduce their range of applicability (only 0.5% of the data in our example). Of course, the exact computations require more time than the approximations, but are nevertheless fast enough (20 p-value per second in our example) to be used in most practicable cases. As a consequence we strongly advise to replace asymptotic approximations by these new exact ones whenever it is possible.

Concerning pattern statistics, the new FMCI algorithms appear to outperform the existing ones ([17]) in all possi-

Table 5: Computational time (in seconds) to get the statistics of degenerate patterns (the dot means "any letter") occurring 100 times in an order *m* = 1 Markovian sequence of length *n* = 9719 which parameters are estimated on the HIV complete genome sequence using either simple recurrences of finite Markov chain imbedding.

pattern	atgca	at.ca	at..a	a...a
SR	0.60	8.20	2438.43	105 209.12
FMCI	0.51	1.15	3.01	91.80

Table 6: Reliability of pattern statistics. They are computed in respect with an order *m* Markovian sequence of length *n* = 9719 which parameters are estimated on the HIV complete genome. Relative error uses FMCI statistics as reference.

pattern	order	observed	expected	FMCI	SR	relative error
acta	0	106	48.63	+12.208	+12.208	0.0
acta	1	106	47.01	+13.090	+13.079	8.7×10^{-4}
acta	0	26	48.63	-3.567	-3.567	0.0
acta	1	26	47.01	-3.231	-3.230	4.8×10^{-4}
acta	0	6	48.63	-13.856	-13.850	3.7×10^{-4}
acta	1	6	47.01	-13.237	-13.237	3.0×10^{-5}
at	0	50	759.48	-291.610	-291.610	0.0
at	0	25	759.48	-327.214	-318.192	2.8×10^{-2}
at	0	0	759.48	-377.009	-319.607	1.5×10^{-1}
tcgatc	0	185	1.37	+294.997	+294.997	0.0
tcgatc	0	195	1.37	+314.388	+314.388	5.7×10^{-8}
tcgatc	0	205	1.37	+333.931	na	na
acacaa	2	10	6.66	+0.865	+0.855	1.1×10^{-2}
acacaa	2	20	6.66	+4.669	+4.520	3.2×10^{-2}
acacaa	2	60	6.66	+35.751	+33.532	6.2×10^{-2}
acacaa	2	100	6.66	+79.736	+73.451	7.8×10^{-2}

ble ways: far easier to implement, more numerical stability, less memory requirements, as fast as SR for simple words (except in the M0 case, but this is due to a poor implementation of this particular case in FMCI approach) and dramatically faster (up to 1 000 times and more) for degenerated patterns. Even if the SR algorithms remain available in the SPatt package, FMCI ones are now used by default for exact computations.

Appendix A: power of a sub-stochastic matrix

Proposition 14. If P is an order L irreducible sub-stochastic matrix admitting a row-eigenvector basis $(e_1,...,e_L)$ where each e_j, is associated to the eigenvalue λ_j and $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_L|$ then we have

$$P^i = \lambda^i P^\infty + \lambda_2^i \sum_{j=2}^{L} \left(\frac{\lambda_j}{\lambda_2} \right)^i P_j^\infty \quad \forall i \geq 0 \qquad (49)$$

where $\lambda = |\lambda_1| = \lambda_1$, $P^\infty = P_1^\infty$ and $\forall j$

$$P_j^\infty = \begin{pmatrix} \left(I_1 e_j'\right) e_j \\ \vdots \\ \left(I_L e_j'\right) e_j \end{pmatrix} \text{ with } I_\ell = \left(\mathbb{I}_{1=\ell}, \ldots, \mathbb{I}_{L=\ell}\right) \ \forall 1 \leq \ell \leq L \qquad (50)$$

Proof. For any vector $x = \sum_{j=1}^{L} x_j e_j$ we have

$$x \times P^i = \sum_{j=1}^{L} x_j \left(e_j \times R^i \right) = \sum_{j=1}^{L} x_j \lambda_j^i e_j = \lambda_1^i x_1 e_1 + \lambda_2^i \sum_{j=1}^{L} \left(\frac{\lambda_j}{\lambda_2} \right)^i x_j e_j \qquad (51)$$

As P is an irreducible, Perron-Frobénius theorem (see [3] for example) assure that λ is real and the sub-stochastic property, that $\lambda \leq 1$ (in the particular case where P is also primitive *i.e.* $\exists m$, $P^m > 0$ then $|\lambda_2| < \lambda$) Replacing x by $I\ell$ equation (51) gives the expression of the row ℓ of P^i and the proposition is then proved. □

Corollary 15. If P is an order ≥ 2 irreducible sub-stochastic matrix admitting a diagonal form then there exists a matrix P^∞ such as

$$P^i - \lambda^i P^\infty = O(\nu^i) \qquad (52)$$

uniformly in i and where $1 \geq \lambda \geq \nu$ are the two largest eigenvalues magnitudes. In the special case where P is primitive, then $\lambda > \nu$ and $P^\infty = \lim_{i \to +\infty} P^i / \lambda^i$ is a positive matrix.

Proof. Using proposition 14 we get

$$\left| P^i - \lambda^i P^\infty \right| \leq |\lambda_2|^i \sum_{j=2}^{L} \left| \frac{\lambda_j}{\lambda_2} \right|^i \left| P_j^\infty \right| \leq \nu^i \underbrace{\sum_{j=2}^{L} \left| P_j^\infty \right|}_{C} \qquad (53)$$

Table 7:

tag\vertex	a	b	ab	aba	abaa	abab
a	a	a	aba	abaa	a*	aba*
b	ab	b	b	abab	ab	b

and the corollary is proved. □

Appendix B: proof of theorem 9

Using propositions 7 and 14 we get the result for $i = 0$. Let us prove the theorem by the principle of recurrence. We assume now that the result is true until rank i - 1. Computing x_i^β for all $\beta \geq (i+1)\alpha$ we get

$$x_1^\beta = P^\beta x_i^0 + \underbrace{\sum_{j=1}^{i\alpha} P^{\beta-j} Q x_{i-1}^{j-1}}_{A} + \underbrace{\sum_{j=i\alpha+1}^{\beta-\alpha} P^{\beta-j} Q x_{i-1}^{j-1}}_{B} + \underbrace{\sum_{j=\beta-\alpha+1}^{\beta} P^{\beta-j} Q x_{i-1}^{j-1}}_{C} \quad (54)$$

For all $1 \leq j \leq i\alpha$ we have $\beta - j \geq \beta - i\alpha \geq \alpha$ so we get

$$A = \sum_{j=1}^{i\alpha} \left(\lambda^{\beta-j} P^\infty + O\left(v^{\beta-j}\right) \right) Q x_{i-1}^{j-1} \quad (55)$$

$$= \lambda^\beta \sum_{j=1}^{i\alpha} \lambda^{-j} P^\infty Q x_{i-1}^{j-1} + O\left(v^{\beta-i\alpha}\right) \quad (56)$$

For all $i\alpha + 1 \leq j \leq \beta - \alpha$ we have $j - 1 \geq i\alpha$ and $\beta - j \geq \beta - i\alpha \geq \alpha$ and so, with the help of lemma 8 we get

$$B = \sum_{j=i\alpha+1}^{\beta-\alpha} \left[\lambda^{\beta-j} P^\infty + O\left(v^{\beta-j}\right) \right] Q \left[\lambda^{j-1} D_{i-1}^\alpha (j-1) + O\left(v^\alpha \lambda^{j-1} (j-1)^{i-1}\right) \right] \quad (57)$$

$$= \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{\beta-1} P^\infty Q D_{i-1}^\alpha (j-1) + \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{\beta-j} O\left(v^\alpha \lambda^{j-1} (j-1)^{i-1}\right) \quad (58)$$

$$+ \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{j-1} O\left(v^{\beta-j}\right) + \sum_{j=\alpha+1}^{\beta-\alpha} v^{\beta-j} O\left(v^\alpha \lambda^{j-1} (j-1)^{i-1}\right) \quad (59)$$

$$= \lambda^\beta \sum_{j=i\alpha+1}^{\beta-\alpha} \lambda^{-1} P^\infty Q D_{i-1}^\alpha (j-1) + O\left(v^\alpha \lambda^\beta \beta^i\right) \quad (60)$$

thanks to the recurrence assumption, it is easy to see than B contribute with polynomial terms of degree i (as we have a sum on $O(\beta)$ terms).

For all $\beta - \alpha + 1 \leq j \leq \beta$ we have $j - 1 \geq i\alpha$ so we get

$$C = \sum_{j=\beta-\alpha+1}^{\beta} P^{\beta-j} Q \left[\lambda^{j-1} D_{i-1}^\alpha (j-1) + O\left(v^\alpha \lambda^{j-1} (j-1)^{i-1}\right) \right] \quad (61)$$

$$= \sum_{j=1}^{\alpha} P^{j-1} Q \left[\lambda^{\beta-j} D_{i-1}^\alpha (\beta-j) + O\left(v^\alpha \lambda^{\beta-j} (\beta-j)^{i-1}\right) \right] \quad (62)$$

$$= \lambda^\beta \sum_{j=1}^{\alpha} \lambda^{-j} P^{j-1} Q D_{i-1}^\alpha (\beta-j) + O\left(v^\alpha \lambda^\beta \beta^{i-1}\right) \quad (63)$$

C contributing with polynomial terms of degree i - 1 (as we have a sum on α terms) Summing up all terms we get the result at rank i and the theorem is proved.

Acknowledgements

I would like to warmly thank Sabine Mercier for her kind help and her enthusiastic feedback during the writing of this article.

References

1. Altschul SF, Gish W: **Local alignment statistics.** In *Methods in Enzymology Volume 266*. Edited by: Doolittle RF. Academic Press, London; 1996:460-480.
2. Altschul SF, Bundschuh R, Olsen R, Hwa T: **The estimation of statistical parameters for local score alignment score distributions.** *Nucleic Acids Research* 2001, **29(2):**351-361.
3. Bapat RB, Raghavan TES: **Nonegative matrices and applications.** Cambridge University Press; 1997.
4. Chothia C: **The Nature of the accessible and buried surfaces in proteins.** *J Mol Biol* 1976, **105:**1-14.
5. Fu JC, Koutras MV: **Distribution theory of runs: a Markov chain approach.** *J Amer Statist Assoc* 1994, **89:**1050-1058.
6. Fu JC: **Distribution theory of runs and patterns associated with a sequence of multi-state trials.** *Statistica Sinica* 1996, **6(4):**957-974.
7. Fu JC, Lou WYW: **Distribution Theory of Runs and Patterns and Its Applications: A Finite Markov Chain Approach.** World Scientific, Singapore; 2003.
8. Hopcroft JE, Motwani R, Ullman JD: **Introduction to Automata Theory, Languages, and Computation (second edition).** Addison-Wesley; 2001.
9. Karlin S, Altschul SF: **Methods for assessing the statistical significance of molecular sequence features by using general.** *Proc Nat Acad Sci USA* 1990, **87:**2264-2268.
10. Kyte J, Doolittle RF: **A simple method for displaying the hydropathic character of a protein.** *J Mol Biol* 1982, **157(1):**105-132.
11. Lou WYW: **On runs and longest run tests: A method of nite Markov chain imbedding.** *J Am Statist Assoc* 1996, **91(436):**1595-1601.
12. Mercier S, Daudin J-J: **Exact Distribution for the Local Score of One i.i.d. Random Sequence.** *J Comp Bio* 2001, **8(4):**373-380.
13. Mercier S, Hassenforder C: **Exact distribution for the local score of a Markov chain.** *C R Acad Sci Paris* 2003, **336(10):**863-868.
14. Nuel G: **LD-SPatt: Large Deviations Statististic for Pattern on Markov chains.** *J Comp Biol* 2004, **11(6):**1023-1033.
15. Nuel G: **S-SPatt: Simple Statististic for Pattern on Markov chains.** *Bioinformatics* 2005, **21(13):**3051-3052.
16. Press WH, Teukolsky SA, Vetterling WT, Flannery BP: **Numerical recipes in C.** Cambridge University Press; 1992.
17. Robin S, Daudin JJ: **Exact distribution of word occurrences in a random sequence of letters.** *J App Prob* 1999, **36:**179-193.
18. Robin S, Daudin J-J, Richard H, Sagot M-F, Schbath S: **Occurrence probability of structured motifs in random sequences.** *J Comp Biol* 2002, **9:**761-773.