Motif Statistics

EXTENDED ABSTRACT

Pierre Nicodème
DKFZ Theoretische Bioinformatik
Im Neuenheimer Feld 280
69120 Heidelberg
Germany
p.nicodeme@dkfz-heidelberg.de

Bruno Salvy
Algorithms Project
Inria Rocquencourt
78153 Le Chesnay Cedex
France
Bruno.Salvy@inria.fr

Philippe Flajolet
Algorithms Project
Inria Rocquencourt
78153 Le Chesnay Cedex
France
Philippe.Flajolet@inria.fr

Abstract

We present a complete analysis of the statistics of number of occurrences of a regular expression pattern in a random text. This covers "motifs" widely used in computational biology. Our approach is based on: *(i)* a constructive approach to classical results in theoretical computer science (automata and formal language theory), in particular, the rationality of generating functions of regular languages; *(ii)* analytic combinatorics that is used for deriving asymptotic properties from generating functions; *(iii)* computer algebra for determining generating functions explicitly, analysing generating functions and extracting coefficients efficiently. We provide constructions for overlapping or non-overlapping matches of a regular expression. A companion implementation produces multivariate generating functions for the statistics under study. A fast computation of Taylor coefficients of the generating functions then yields exact values of the moments with typical application to random texts of size 30,000 while precise symptotic formulæ allow predictions in texts of arbitrarily large sizes. Our implementation was tested by comparing predictions of the number of occurrences of motifs against the 7 megabytes amino acid database PRODOM. We handled more than 88% of the standard collection of PROSITE motifs with our programs. Such comparisons help detect which motifs are observed in real biological data more or less frequently than theoretically predicted.

1 Introduction

The purpose of molecular biology is to establish relations between chemical form and function in living organisms. From an abstract mathematical or computational standpoint, this gives rise to two different types of problems: processing problems that, broadly speaking, belong to the realm of pattern-matching algorithmics, and probabilistic problems aimed at distinguishing between what is statistically significant and what is not, at discerning "signal" from "noise". The present work belongs to the category of probabilistic studies originally motivated by molecular biology. As we shall see, however, the results are of a somewhat wider scope.

Fix a finite alphabet, and take a large random *text* (a sequence of letters from the alphabet), where randomness is defined by either a Bernoulli model (letters are drawn independently) or a Markov model. Here, a *pattern* is specified by an *unrestricted regular expression* R and occurrences anywhere in a text file are considered. (Some controlled dependency on the past is allowed). The problem is to quantify precisely what to expect as regards the *number of occurrences* of pattern R in a random text of size n. We are interested first of all in moments of the distributions—*what is the mean and the variance?*—, but also in asymptotic properties of the distribution—*does the distribution have a simple asymptotic form?*—, as well as in computational aspects—*are the characteristics of the distribution effectively accessible for problems of a "reasonable" size?*

We provide positive answers to these three questions. Namely, for all "non-degenerate" pattern specifications[1] R, we establish the following results:

- The number of occurrences has a mean of the form $\mu \cdot n + O(1)$, with a standard deviation that is of order $\sqrt{n}$; in particular, concentration of distribution holds.
- The number of occurrences, once normalized by the mean and standard deviation, obeys in the asymptotic limit a Gaussian law.
- The characteristics of the distribution are effectively computable, both exactly and asymptotically, given basic computer algebra routines. The resulting procedures are capable of treating fairly large "real-life" patterns in a reasonable amount of time.

Though initially motivated by computational biology considerations, these results are recognizably of a general nature. They should thus prove to be of use in other areas, most notably, the analysis of complex string matching algorithms, large finite state models of computer science and combinatorics, or natural language studies. (We do not however pursue these threads here and stay with the original motivation provided by computational biology.)

The basic mathematical objects around which the paper is built are counting *generating functions*. In its

[1] Technically, non-degeneracy is expressed by the "primitivity" condition of Theorem 2. All cases of interest can in fact be reduced to this case; see the discussion at the end of Section 4.

bivariate version, such a generating function encodes exactly all the information relative to the frequency of occurrence of a pattern in random texts of all sizes. We appeal to a combination of classical results from the theory of *regular expressions and languages* and from basic combinatorial analysis (an ingenious marking by auxiliary variables) in order to determine such generating functions systematically. Specifically, we use a chain from regular expression patterns to bivariate generating functions that goes through nondeterministic and deterministic finite automata. Not too unexpectedly, the generating functions turn out to be rational (Theorem 1), but also computable at a reasonable cost for most patterns of interest (Section 6). Since coefficients of univariate rational GF's are computable in $O(\log n)$ arithmetic operations, this provides the exact statistics of matches in texts of several thousands positions in a few seconds, typically. Also, asymptotic analysis of the coefficients of rational functions can be performed efficiently (Gourdon & Salvy 1996). Regarding multivariate asymptotics, a perturbation method from analytic combinatorics then yields the Gaussian law (Theorem 2).

In the combinatorial world, the literature on pattern statistics is vast. It originates largely with the introduction of correlation polynomials by (Guibas & Odlyzko 1981) in the case of patterns defined by one word. The case of several words was studied by many authors, including (Guibas & Odlyzko 1981), (Flajolet, Kirschenhofer & Tichy 1988), and (Bender & Kochman 1993). Finite sets of words in Bernoulli or Markov texts are further considered by (Régnier 1998; Régnier & Szpankowski 1998). As a result of these works, the number of occurrences of any *finite set of patterns* in a random Bernoulli or Markov text is known to be asymptotically normal; see also the review in (Waterman 1995, Chap. 12). Several other works are motivated by computational biology considerations. For instance, the paper (Pevzner, Borodovski & Mironov 1989) handles a pattern allowing fixed length gaps of don't care symbols and determines the statistics of number of occurrences of these words in a random text; (Schbath, Prum & de Turckheim 1995; Prum, Rodolphe & de Turckheim 1995; Reinert & Schbath 1998) study by probabilistic methods words with unexpected frequencies and multiple words in texts generated by a Markov chain. (Sewell & Durbin 1995) compute algorithmically bounds on the probability of a match in random strings of length 1000. (Atteson 1998) evaluates numerically the probability of a match when the text is generated by a Markov chain for texts of size 2000. Our distributional results that deal with arbitrary regular expression patterns, including *infinite word sets*, thus extend the works of these authors.

The effective character of our results is confirmed by a *complete implementation* based on symbolic computation, the Maple system in our case. Our implementation has been tested against real-life data provided by a collection of patterns, the frequently used PROSITE collection[2] (Bairoch, Bucher & Hofman 1997), and a database of sequences, the PRODOM database[3] that constitutes the text. We apply our results for computing the statistics of matches and compare with what is observed in the PRODOM database. In its most basic version, string-matching considers one or a few strings that are searched for in the text. *Motifs* appear in molecular biology as signatures for families of similar sequences and they characterize structural functionalities of sequences derived from a common ancestor. For instance, a typical motif of PROSITE is

[LIVM](2)-x-D-D-x(2,4)-D-x(4)-R-R-[GH],

where the capital letters represent amino acids, 'x' stands for any letter, brackets denote a choice and parentheses a repetition. Thus x(2,4) means two to four consecutive arbitrary amino acids, while [LIVM](2) means two consecutive elements of the set {L,I,V,M}. Put otherwise, a motif is a regular expression of a restricted form that may be expanded, in principle at least, into a *finite* set of words. Our analysis that addresses general regular expression patterns, including a wide class of *infinite* sets of words, encompasses the class of all motives.

On the practical side, it is worthwhile to remark that the automaton description for a motif tends to be much more compact than what would result from the expansion of the language described by the motif, allowing for an exponential reduction of size in many cases. For instance, for motif PS00844 from PROSITE our program builds an automaton which has 946 states while the number of words of the finite language generated by the motif is about 2×10^{26}. In addition, regular expressions are able to capture long range dependencies, so that their domain of application goes far beyond that of standard motifs.

Contributions of the paper. This work started when we realized that computational biology was commonly restricting attention to what seemed to be an unnecessarily constrained class of patterns. Furthermore, even on this restricted class, the existing literature often had to rely on approximate probabilistic models. This led to the present work that demonstrates, both theoretically and practically, that a more general framework is fully workable. On the theory side, we view Theorem 2 as our main result, since it appears to generalize virtually everything that is known regarding probabilities of pattern occurrences. On the practical side, the feasability of a complete chain based on algorithms, some old and some new, and on the principles of Section 3 is demonstrated in Section 6. The fact that we can handle *in an exact way* close to 90% of the motifs of a standard col-

[2]At the moment, Prosite comprises some 1,200 different patterns, called "motifs", that are regular expressions of a restricted form and varying structural complexity.

[3]Prodom is a compilation of "homologous" domains of proteins in SWISS-PROT, and we use it as a sequence of length 6,700,000 over the alphabet of amino acids that has cardinality 20.

lection that is of common use in biological applications probably constitutes the most striking contribution of the paper.

2 Main statements

We consider the number of occurrences of a pattern (represented by a regular expression) in a text under two different situations: in the *overlapping* case, all the positions in the text where a match with the regular expression can occur are counted (once); in the *non-overlapping* case, the text is scanned from left to right, and every time a match is found, the count is incremented and the search starts afresh at this position. These cases give rise to two different statistics for the number X_n of matches in a random text of size n, and we handle both of them. Without loss of generality, we assume throughout that R does not contain the empty word ε.

In each context, the method we describe gives an algorithm for computing the bivariate probability generating function

$$P(z,u) = \sum_{n,k\geq 0} p_{n,k} u^k z^n, \tag{1}$$

where $p_{n,k} = \Pr\{X_n = k\}$. This generating function specializes in various ways:

— $P(z,0)$ is the probability generating function of texts that don't match against the motif, while

$$R(z) = 1/(1-z) - P(z,0)$$

is the probability generating function of texts with at least one occurrence. More generally, the coefficient $[u^k]P(z,u)$ is the generating function of texts with k occurrences.

— Partial derivatives

$$M_1(z) = \frac{\partial F}{\partial u}(z,1) \text{ and } M_2(z) = \frac{\partial}{\partial u} u \frac{\partial F}{\partial u}(z,u)\bigg|_{u=1},$$

are generating functions of the first and second moments of the number of occurrences in a random text of length n.

Our first result characterizes these generating functions as effectively computable rational functions.

Theorem 1. *Let R be a regular expression, X_n the number of occurrences of R in a random text of size n, and $p_{n,k} = \Pr\{X_n = k\}$ the corresponding probability distribution.*

Then, in the overlapping or in the non-overlapping case, and under either the Bernoulli model or the Markov model, the generating functions

$$P(z,u), \quad R(z), \quad M_1(z), \quad M_2(z),$$

corresponding to probabilities of number of occurrences, existence of a match, and first and second moment of number of occurrences, are rational and can be computed explicitly.

Our second result provides the corresponding asymptotics. Its statement relies on the fundamental matrix $T(u)$ defined in Section 4, as well as the notion of primitivity, a technical but nonrestrictive condition, that is defined there.

Theorem 2. *Under the conditions of Theorem 1, assume that the "fundamental matrix" $T(1)$ defined by (8) is primitive. Then, the mean and variance of X_n grow linearly,*

$$\begin{cases} \mathrm{E}(X_n) & = \mu n + c_1 + O(A^n), \\ \mathrm{Var}(X_n) & = \sigma^2 n + c_2 + O(A^n), \end{cases}$$

where $\mu \neq 0$, $\sigma \neq 0$, c_1, c_2 are computable constants.

The normalized variable, $(X_n - \mu n)/(\sigma\sqrt{n})$, converges with speed $O(1/\sqrt{n})$ to a Gaussian law:

$$\Pr\left(\frac{X_n - \mu n}{\sigma\sqrt{n}} \leq x\right) \to \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt.$$

A local limit and large deviation bounds also hold.

The constants that appear in the statement are related to spectral properties of a transition matrix $T(u)$, in particular to its dominant eigenvalue $\lambda(u)$. Their form is given in Eqs. (5) and (11).

3 Algorithmic Chain

In order to compute the probability generating function of the number of occurrences of a regular expression, we use classical constructions on non-deterministic and deterministic finite automata. For completeness, we state all the algorithms, old and new, leading to the probability generating functions of Theorem 1. References for this section are (Kozen 1997; Kelley 1995; Hopcroft & Ullman 1979; Rayward-Smith 1983) among numerous textbooks describing regular languages and automata.

3.1 Regular Languages

We consider a *finite alphabet* $\Sigma = \{\ell_1, \ldots, \ell_r\}$. A *word* over Σ is a finite sequence of *letters*, that is, elements of Σ. A *language* over Σ is a set of words. The *product* $A = A_1 \cdot A_2$ of two languages A_1 and A_2 is $A = \{w_1 w_2, w_1 \in A_1, w_2 \in A_2\}$, where $w_1 w_2$ is the concatenation of words w_1 and w_2. Let A^n be the set of products of n words belonging to A, then the *star closure* $A^\star$ of a language A is the infinite union $A^\star = \cup_{n\geq 0} A^n$. The language $\Sigma^\star$ is thus the collection of all possible words over Σ.

Regular languages over Σ are defined inductively. Such a language is either the empty word, or it reduces to a single letter, or it is obtained by union, product or star closure of simpler regular languages. The formula expressing a regular language in terms of these operations and letters is called a *regular expression*. As notational convenience, ℓ denotes the singleton language $\{\ell\}$, $+$ represents a union, and $\cdot$ is freely omitted. The order of precedence for the operators is $\star, \cdot, +$.

3.2 Nondeterministic Finite Automata

A *Nondeterministic Finite Automaton* (or NFA) is formally specified by five elements. (1) An input alphabet Σ; (2) A finite collection of states Q; (3) A start state $s \in \Sigma$; (4) A collection of final states $F \subset Q$; (5) A (possibly partial) transition function δ from $Q \times \Sigma$ to $\mathcal{S}_Q$ the set of subsets of Q. There exists a *transition* from state q_i to state q_j if there is a letter $\ell \in \Sigma$ such that $q_j \in \delta(q_i, \ell)$. A word $w = w_1 w_2 \cdots w_n \in \Sigma^\star$ is *accepted* or *recognized* by an NFA $A = (\Sigma, Q, s, F, \delta)$ if there exists a sequence of states $q_{i_0}, q_{i_1}, q_{i_2}, \ldots, q_{i_n}$ such that $q_{i_0} = s$, $q_{i_j} \in \delta(q_{i_{j-1}}, w_j)$ and $q_{i_n} \in F$.

Kleene's theorem states that a language is regular if and only if it is recognized by an NFA. Several algorithms are known to construct such an NFA. We present below an algorithm due to (Berry & Sethi 1986) as improved by (Brüggemann-Klein 1993) that constructs an NFA called the Glushkov automaton.

Algorithm 1 (Berry & Sethi).
Input: a regular expression R over an alphabet Σ.
Output: an NFA recognizing the corresponding language.

1. Give increasing indices to the occurrences of each letter of Σ occurring in R. Let Σ' be the alphabet consisting of these indexed letters.
2. For each letter $\ell \in \Sigma'$, construct the subset follow(ℓ) of Σ' of letters that can follow ℓ in a word recognized by R.
3. Compute the sets first(R) and last(R) of letters of Σ' that can occur at the beginning and at the end of a word recognized by R.
4. The automaton has for states the elements of Σ' plus a start state. The transitions are obtained using follow and erasing the indices. The final states are the elements of last(R).

Steps 2 and 3 are performed by computing inductively four functions "first", "last", "follow" and "nullable". Given a regular expression r over Σ', first returns the set of letters that can occur at the beginning of a match; last returns those that can occur at the end of a match; nullable returns true if r recognizes the empty word and false otherwise; for each $\ell \in \Sigma'$ that occurs in r, follow returns the set of letters that can follow ℓ in a word recognized by r. The computation is a simple induction as follows:

nullable(r) If $r = \varepsilon$ or $r = a^\star$, return true; if r is a letter, return false; if $r = a + b$, return (nullable(a) or nullable(b)); if $r = ab$ return (nullable(a) and nullable(b)).

first(r) If r is a letter, the result is the singleton consisting of this letter; if $r = a + b$, return first(a) + first(b); if $r = ab$ return first(a) if a is not nullable and first(a) + first(b) otherwise; if $r = a^\star$, return first(a).

last(r) is similar.

follow(r,x) If $r = \ell$ return $\emptyset$; if $r = a + b$ then because of the indexing, ℓ occurs in only one of a and b and the result is that of follow on this regular expression; if $r = ab$ and ℓ occurs in b then return follow(b, x), otherwise return follow(a, x) if x does not belong to last(a) and follow(a, x) + first(b) otherwise; if $r = a^\star$, then if $\ell \in$ last(a), return first(a) + follow(a, x), otherwise return follow(a, x).

As observed by (Brüggemann-Klein 1993), an appropriate data-structure for unions yields a quadratic complexity for the algorithm, provided the union in the computation of follow($a^\star, x$) is disjoint. (This is guaranteed if the regular expression is in star-normal form, a property we do not define but which is directly satisfied in our biological applications. There is anyway a linear complexity algorithm for converting a regular expression into a star-normal form, see (Brüggemann-Klein 1993).)

3.3 Deterministic Finite Automata

Deterministic finite automata (or DFAs) are special cases of NFAs where the images of the transition function are singletons. By a classical theorem of Rabin & Scott, NFAs are equivalent to DFAs in the sense that they recognize the same class of languages. This is made effective by the powerset construction.

Algorithm 2 (Rabin & Scott).
Input: an NFA $A = (\Sigma, Q, s, F, \delta)$.
Output: a DFA recognizing the same language.

1. Define a transition function $\Delta : \mathcal{S}_Q \times \Sigma \to \mathcal{S}_Q$ by:
$$\forall V \in \mathcal{S}_Q, \forall \ell \in \Sigma, \quad \Delta(V, \ell) = \bigcup_{q \in V} \delta(q, \ell).$$
2. Define $\mathcal{Q}_F$ as the set of subsets of Q that contain at least one element of F.
3. Return the automaton $(\Sigma, \mathcal{S}_Q, \{s\}, \mathcal{Q}_F, \Delta)$.

One needs only consider in the DFA the states reachable from the start state $\{s\}$. The number of states of the DFA constructed in this way is not necessarily minimal. In the worst case, the construction is of exponential complexity in the number of states of the NFA. For applications to motifs however, this construction is done in reasonable time in most cases (see Section 6).

3.4 Generating Functions

Let $\mathcal{A}$ be a language over Σ. The generating function of the language is obtained by summing formally all the words of $\mathcal{A}$ and collecting the resulting monomials with the letters being allowed to commute. The *generating function* of the language $\mathcal{A}$ is then defined as the formal sum
$$A(\ell_1, \ldots, \ell_r) = \sum_{w \in \mathcal{A}} \mathrm{com}(w),$$
with $\mathrm{com}(w) = w_1 w_2 \cdots w_n$ the monomial associated to $w = w_1 w_2 \cdots w_n \in \mathcal{A}$. We use the classical notation $[\ell_1^{i_1} \cdots \ell_r^{i_r}]A$ to denote the coefficient of $\ell_1^{i_1} \cdots \ell_r^{i_r}$ in the generating function A.

The generating function of a regular language is rational (Chomsky & Schützenberger 1963). This results from the following construction.

Algorithm 3 (Chomsky & Schützenberger).
Input: A regular expression.
Output: Its generating function.

1. Construct the DFA recognizing the language. For each state q, let $\mathcal{L}_q$ be the language of words recognized by the automaton with q as start state. These languages are connected by linear relations,
$$\mathcal{L}_q = (\varepsilon+) \bigcup_{\ell\in\Sigma} \ell\mathcal{L}_{\delta(q,\ell)},$$
where ε is present when q is a final state. The automaton being deterministic, the unions in this system are disjoint.
2. Translate this system into a system of equations for the associated generating functions:
$$L_q = (1+) \sum_{\ell\in\Sigma} \ell L_{\delta(q,\ell)}.$$
3. Solve the system and get the generating function $F = L_s$, where s is the start state.

The resulting generating is rational, as it is produced as a solution of a linear system. Naturally, the algorithm specializes in various ways when numerical weights (probabilities) are assigned to letters of the alphabet.

3.5 Regular Expression Matches

We first consider the Bernoulli model. The letters of the text are drawn independently at random, each letter ℓ_i of the alphabet having a fixed probability p_i, and $\sum p_i = 1$. The uniform case is the special case when $p_i = 1/|\Sigma|$, for $i = 1, \ldots, |\Sigma|$. The basis of the proof of theorem 1 is the following construction.

Algorithm 4 (Marked automaton).
Input: A regular expression R over the alphabet Σ.
Output: A DFA recognizing the (regular) language of words over $\Sigma \cup \{m\}$ where each match of the regular expression R is followed by the letter $m \notin \Sigma$, which occurs only there.

1. Construct a DFA $A = (Q, s, F, \Sigma, \delta)$ recognizing $\Sigma^\star R$.
2. Initialize the resulting automaton: set
$$A' = (Q', s, Q, \Sigma + m, \delta')$$
with initial values $\delta' = \delta$ and $Q' = Q$.
3. Mark the matches of R: for all $q \in Q$ and all $\ell \in \Sigma$ such that $\delta(q,\ell) = f \in F$, create a new state q_ℓ in Q', set $\delta'(q,\ell) := q_\ell$ and $\delta'(q_\ell, m) := f$.
4. Restart after match (non-overlap case only): for all $f \in F$, and all $\ell \in \Sigma$ set $\delta'(f,\ell) := \delta(s,\ell)$.
5. Return A'.

We note that the automaton constructed in this way is deterministic since all the transitions that have been added are either copies of transitions in A, or start from a new state, or were missing.

This automaton recognizes the desired language. Indeed, the words of $\Sigma^\star R$ are all the words of $\Sigma^\star$ ending with a match of R. Thus the final states of A are reached only at the end of a match of R. Conversely, since no letter is read in advance, every time a match of R has just been read by A, the state which has been reached is a final state. Thus inserting a non-final state and a marked transition "before" each final state corresponds to reading words with the mark m at each position where a match of R ends. Then by making all the states final except those intermediate ones, we allow the words to end without it being the end of a match of R. In the non-overlapping case, the automaton is modified in step 4 to start afresh after each match. (This construction can produce states that are not reachable. While this does not affect the correctness of the rest of the computation, suppressing these states saves time.)

The proof of Theorem 1 is concluded by the following algorithm in the Bernoulli model.

Algorithm 5 (Number of matches—Bernoulli).
Input: A regular expression R over an alphabet Σ and the probabilities p_i of occurrence of each letter $\ell_i \in \Sigma$.
Output: The bivariate generating function for the number of occurrences of R in a random text according to the Bernoulli model.

1. Construct the marked automaton for R.
2. Return the generating function $F(p_1 z, \ldots, p_r z, u)$ of the corresponding language, as given by the Chomsky-Schützenberger Algorithm.

The proof of Theorem 1 in the Markov model follows along similar lines. It is based on an automaton that keeps track of the letter most recently read.

Algorithm 6 (Markov automaton).
Input: A DFA A over an alphabet Σ.
Output: A DFA over the alphabet $(\ell_0+\Sigma)^2$, where $\ell_0 \notin \Sigma$. For each word $w_1 \cdots w_n$ recognized by A, this DFA recognizes the word $(\ell_0, w_1)(w_1, w_2) \cdots (w_{n-1}, w_n)$.

1. Duplicate the states of A until there are only input transitions with the same letter for each state. Let $(Q, s, F, \Sigma, \delta)$ be the resulting automaton.
2. Define a transition function $\Delta : Q \times (\ell_0 + \Sigma)^2 \to Q$ by $\Delta(\delta(q,\ell), (\ell,\ell')) = \delta(\delta(q,\ell),\ell')$ for all $q \in Q \setminus \{s\}$, and $\ell, \ell' \in \Sigma$; and $\Delta(\delta(s,\ell), (\ell_0,\ell)) = \delta(s,\ell)$ for all $\ell \in \Sigma$.
3. Return $(Q, s, F, (\ell_0 + \Sigma)^2, \Delta)$.

This construction then gives access to the bivariate generating function.

Algorithm 7 (Number of matches—Markov).
Input: A regular expression R over an alphabet Σ, the probabilities q_{ij} of transition from letter ℓ_i to ℓ_j and the probabilities q_{0j} of starting with letter ℓ_j for all $\ell_i, \ell_j \in \Sigma$.

Output: The bivariate generating function for the number of occurrences of R in a random text according to the Markov model.

1. Apply the algorithm "Marked automaton" with "Markov automaton" as an extra step between steps 1 and 2.
2. Return the generating function

$$F(q_{01}z, \ldots, q_{rr}z, u)$$

of the corresponding language.

This concludes the description of the algorithmic chain, hence the proof of Theorem 1, as regards the bivariate generating function $P(z,u)$ at least. The other generating functions then derive from P in a simple manner. □

4 Limiting Distribution

In this section, we establish the limiting behaviour of the probability distribution of the number of occurrences of a regular expression R in a random text of length n and prove that it is asymptotically Gaussian, thereby establishing Theorem 2. Although this fact could be alternatively deduced from limit theorems for Markov chains, the approach we adopt has the advantage of fitting nicely with the computational approach of the present paper. In this extended abstract, only a sketch of the proof is provided.

Streamlined proof. The strategy of proof is based on a general technique of singularity perturbation, as explained in (Flajolet & Sedgewick 1997) to which we refer for details. This technique relies on an analysis of the bivariate generating function

$$P(z,u) = \sum_{n,k\geq 0} p_{n,k} u^k z^n,$$

where $p_{n,k}$ is the probability that R has k matches in a random text of length n. The analysis reduces to establishing that in a fixed neighbourhood of $u = 1$, $P(z,u)$ behaves as

$$\frac{c(u)}{1 - z\lambda(u)} + g(z,u), \qquad (2)$$

with $c(1) \neq 0$, $c(u)$ and $\lambda(u)$ analytic in the neighbourhood of $u = 1$ and $g(z,u)$ analytic in $|z| > \delta$ for some $\delta > 1/\lambda(1)$ independent of u. Indeed, if this is granted, there follows

$$[z^n]P(z,u) = c(u)\lambda(u)^n(1 + O(A^n)), \qquad (3)$$

for some $A < 1$. The last equation says that X_n has a generating function that closely resembles a large power of a fixed function, that is, the probability generating function of a sum of independent random variables. Thus, we are close to a case of application of the central limit theorem and of Levy's continuity theorem for characteristic functions (Billingsley 1986). This part of our treatment is in line with the pioneering works of (Bender 1973; Bender, Richmond & Williamson 1983) concerning limit distributions in combinatorics. Technically, under the "variability condition", namely

$$\lambda''(1) + \lambda'(1) - \lambda'(1)^2 \neq 0, \qquad (4)$$

we may conveniently appeal to the *quasi-powers theorem* of (Hwang 1994) that condenses the consequences drawn from analyticity and the Berry-Esseen inequalities. This implies convergence to the Gaussian law with speed $O(1/\sqrt{n})$, the expectation and the variance being

$$\mathrm{E}(X_n) = n\lambda'(1) + c_1 + O(A^n), \qquad (5)$$
$$\mathrm{Var}(X_n) = n(\lambda''(1) + \lambda'(1) - \lambda'(1)^2) + c_2 + O(A^n),$$
$$c_1 = c'(1), \quad c_2 = c''(1) + c'(1) - c'(1)^2.$$

Linear structure. We now turn to the analysis leading to (2). Let A be the automaton recognizing $\Sigma^\star R$ and let m be its number of states. In accordance with the developments of Section 3, the matrix equation for the generating functions can be written

$$L = zT_0L + \epsilon, \qquad (6)$$

where ϵ is a vector whose ith entry is 1 if state i is final and zero otherwise. The matrix T_0 is a stochastic matrix (i.e., the entries in each of its lines add up to 1). The entry $t_{i,j}$ in T_0 for $i,j \in \{1,\ldots,m\}$, is the probability of reaching state j from state i of the automaton in one step. In the overlapping case, the construction of Section 3 produces a system equivalent to

$$L = zT_0 \operatorname{diag}(\phi_i)L + \mathbf{1}, \qquad \phi_i \in \{1,u\}, \qquad (7)$$

where $\mathbf{1}$ is a vector of ones since all the states of the new automaton are final, and $\phi_i = u$ when state i of A is final, and 1 otherwise. In the non-overlapping case, the system has the same shape; the transitions from the final states are the same as the transitions from the start state, which is obtained by replacing the rows corresponding to the final state by that corresponding to the start state.

Thus, up to a renumbering of states, the generating function $P(z,u)$ is obtained as the first component of the vector L in the vector equation

$$L = zT(u)L + \mathbf{1}, \qquad (8)$$

with $T(u) = T_0 \operatorname{diag}(1,\ldots,1,u,\ldots,u)$, the number of u's being the number of final states of A. Equation (8) implies

$$P(z,u) = (1,0,\ldots,0)L = \frac{B(z,u)}{\det(I - zT(u))}, \qquad (9)$$

for some polynomial $B(z,u)$, where I denotes the $m \times m$ identity matrix. The matrix $T(u)$ is called the *fundamental matrix* of the pattern R.

Perron-Frobenius properties. One can resort to results on matrices with nonnegative entries (Gantmacher 1959; Prasolov 1994) to obtain precise information on the location of the eigenvalue of $T(u)$ of largest modulus. Such eigenvalues determine dominant asymptotic behaviours and in particular they condition (2).

The Perron-Frobenius theorem states that if the matrix $T(u)$ $(u > 0)$ is *irreducible* and additionally *primitive*, then it has a unique eigenvalue $\lambda(u)$ of largest modulus, which is real positive. (For an $m \times m$–matrix A, irreducibility mean that $(I + A)^m \gg \mathbf{0}$ and primitivity means $A^e \gg \mathbf{0}$, for some e, where $X \gg \mathbf{0}$ iff all the entries of X are positive.) In the context of automata, irreducibility means that from any state, any other state can be reached (possibly in several steps); primitivity means that there is a large enough e such that for any pair (i, j) of states, the probability of reaching j from i in exactly e steps is positive. (Clearly, primitivity implies irreducibility.) In the irreducible case, if the matrix is not primitive, then there is a periodicity phenomenon and an integer $k \leq m$ such that $T(u)^k$ is "primitive by blocks". Irreducibility and primitivity are easily tested algorithmically.

Gaussian distribution. Consider the characteristic polynomial of the fundamental matrix,

$$Q(\lambda) \equiv Q(\lambda, u) = \det(\lambda I - T(u)),$$

where $T(u)$ is assumed to be primitive. By the Perron-Frobenius theorem, for each $u > 0$, there exists a unique root $\lambda(u)$ of $Q(\lambda)$ of maximal modulus that is a positive real number. The polynomial Q has roots that are algebraic in u and therefore continuous. Uniqueness of the largest eigenvalue of $T(u)$ then implies that $\lambda(u)$ is continuous and is actually an algebraic function of u for $u > 0$. Thus there exists a $\epsilon > 0$ and $\eta_1 > \eta_2$ two real numbers such that for u in a neighbourhood $(1-\epsilon, 1+\epsilon)$ of 1, $\lambda(u) > \eta_1 > \eta_2 > |\mu(u)|$, for any other eigenvalue $\mu(u)$.

The preceding discussion shows that in the neighbourhood $u \in (1 - \epsilon, 1 + \epsilon)$, (9) implies

$$P(z, u) = \frac{B(\lambda^{-1}(u), u)}{\lambda^{1-m}(u)Q'(\lambda(u))(1 - z\lambda(u))} + g(z, u),$$

where g is analytic in z with radius of convergence at least $1/\eta_2$. This proves (2). Then, the residue theorem applied to the integral

$$I_n(u) = \frac{1}{2i\pi} \oint_\gamma P(z, u) \frac{dz}{z^{n+1}},$$

where γ is a circle around the origin of radius $\delta = 2/(\eta_1 + \eta_2)$, yields (3).

Condition (4) is now derived by adapting an argument of (Vallée 1998) relative to analytic dynamic sources in information theory, which reduces in our case to using the Cauchy-Schwartz inequality. For the L_1 matrix norm, $||T(u)^n||$ is a polynomial in u with nonnegative coefficients. It follows that

$$||T^n(uv)|| \leq ||T^n(u^2)||^{1/2} ||T^n(v^2)||^{1/2}.$$

Since for any matrix T, the modulus of the largest eigenvalue of T is $\lim_{n\to\infty} ||T^n||^{1/n}$, we get

$$\lambda(uv) \leq \lambda(u^2)^{1/2} \lambda(v^2)^{1/2}, \qquad \forall u, v > 0.$$

This inequality reads as a concavity property for $\phi(t) := \log \lambda(e^t)$:

$$\phi\left(\frac{x+y}{2}\right) \leq \frac{\phi(x) + \phi(y)}{2}, \tag{10}$$

for any real x and y. If the inequality in (10) is strict in a neighbourhood of 0, then $\phi'' < 0$. (The case where $\phi''(0) = 0$ is discarded since $\lambda(u)$ is nondecreasing.) Otherwise, if there exist $x < 0$ and $y > 0$ such that the equality holds in relation (10), then necessarily equality also holds in the interval (x, y) and ϕ is actually affine in this interval. This in turn implies $\lambda(u) = au^b$ for some real a and b and u in an interval containing 1, and therefore equality holds for all $u > 0$ from the Perron-Frobenius theorem as already discussed. Since $\lambda(1) = 1$, necessarily $a = 1$. From the asymptotic behaviour (3) follows that $b \leq 1$. Now λ being a root of $Q(\lambda)$, if $\lambda(u) = u^b$ with $b < 1$, then b is a rational number p/q and the conjugates $e^{2ik\pi/q}\lambda$, $k = 1, \ldots, q - 1$ are also solutions of $Q(\lambda)$, which contradicts the Perron-Frobenius theorem. Thus the only possibility for b is 1. Now, u is an eigenvalue of $uT(1)$ and another property of nonnegative matrices (Prasolov 1994, Th. 37.2.2) shows that the only way u can be an eigenvalue of $T(u)$ is when $T(u) = uT(1)$, which can happen only when all the states of the automaton are final, i.e., $\Sigma^\star R = \Sigma^\star$, or, equivalently $\varepsilon \in R$. This concludes the proof of Theorem 2 in the Bernoulli case.

Markov model. The Markov case requires the tensor product construction of Section 3. This gives rise again to a linear system that is amenable to singularity perturbation. The condition of primitivity is again essential but it is for instance satisfied as soon as both the Markov model and the pattern automaton are primitive. (Details omitted in this abstract.) This discussion concludes the proof of Theorem 2. □

We observe that the quantities given in the statement are easily computable. In effect, from the characteristic polynomial Q of $T(u)$, the quantities involved in the expectation and variance of the statement of Theorem 2 are

$$\lambda'(1) = -\left.\frac{\frac{\partial Q}{\partial u}}{\frac{\partial Q}{\partial \lambda}}\right|_{z=\lambda=1} \tag{11}$$

$$\lambda''(1) = -\left.\frac{\frac{\partial^2 Q}{\partial u^2} + 2\lambda'(1)\frac{\partial^2 Q}{\partial u \partial \lambda} + \lambda'(1)^2 \frac{\partial^2 Q}{\partial \lambda^2}}{\frac{\partial Q}{\partial \lambda}}\right|_{z=\lambda=1}.$$

We end this section with a brief discussion showing how the "degenerate" cases in which $T(1)$ is not primitive are still reducible to the case when Theorem 2 applies.

Irreducibility. The first property we have used is the irreducibility of $T(1)$. It means that from any state of the automaton, any other state can be reached. In the non-overlapping case, this property is true except possibly for the start state, since after a final state each of the states following the start state can be reached. In the overlapping case, the property is not true in general, but since the generating function $P(z,u)$ does not depend on the choice of automaton recognizing $\Sigma^\star R$, we can assume that the automaton is minimal (has the minimum number of states), and then the property becomes true after a finite number of steps by an argument we omit in this abstract. Thus in both cases, $T(u)$ is either irreducible or decomposes as $\left(\begin{smallmatrix} P & L \\ 0 & A(u)\end{smallmatrix}\right)$ where $A(u)$ is irreducible and it can be checked that the largest eigenvalue arises from the A-block for u near 1. It is thus sufficient to consider the irreducible case.

Primitivity. When $T(u)$ is not primitive, there is an integer $k \le m$ such that $T^k(u)$ is primitive. Thus our theorem applies to each of the variables $X_n^{(i)}$ counting the number of matches of the regular expression R in a text of length $kn+i$ for $i = 0,\dots,k-1$. Then, the theorem still holds once n is restricted to any congruence class modulo k.

5 Processing Generating Functions

Once a bivariate generating function of probabilities has been obtained explicitly, several operations can be performed efficiently to retrieve information.

First, differentiating with respect to u and setting $u = 1$ yields univariate generating functions for the moments of the distribution as explained in Section 2. By construction, these generating functions are also rational.

5.1 Fast coefficient extraction

The following is classical and can be found in (Knuth 1981).

Algorithm 8 (Coefficient extraction).
Input: a rational function $f(z) = P(z)/Q(z)$ and an integer n.
Output: $u_n = [z^n]f(z)$ computed in $O(\log n)$ arithmetic operations.

1. Extract the coefficient of z^n in $Q(z)f(z) = P(z)$, which yields a linear recurrence with constant coefficients for the sequence u_n. The order m of this recurrence is the degree of Q.
2. Rewrite this recurrence as a linear recurrence of order 1 relating the vector $U_n = (u_n,\dots,u_{n-m+1})$ to U_{n-1} by $U_n = AU_{n-1}$ where A is a constant $m\times m$ matrix.
3. Use binary powering to compute the power of A in $U_n = A^{n-m}U_m$.

This operation is implemented in the Maple package `gfun` (Salvy & Zimmermann 1994).

As an example, Fig. 1 displays the probability that the pattern ACAGAC occurs exactly twice in a text over the alphabet {A,C,G,T} against the length n of the text. The probabilities assigned to each of the letters are taken from a viral DNA ($\phi X174$). The shape of the curve is typical of that expected in the non-asymptotic regime.

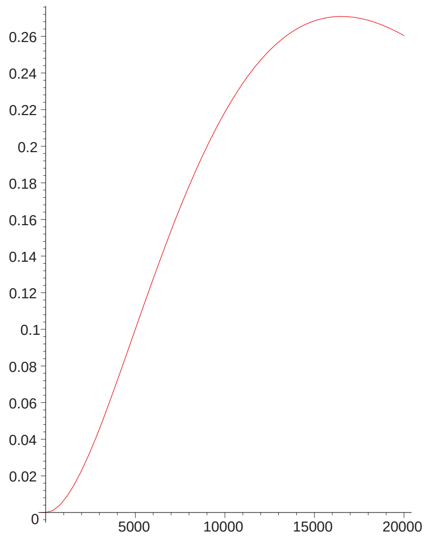

Figure 1: Probability of two occurrences of ACAGAC in a text of length up to 20,000

5.2 Asymptotics

Asymptotics of the coefficients of a rational function can be obtained directly. Since the recurrence satisfied by the coefficients is linear with constant coefficients, a solution can be found in the form of an exponential polynomial:

$$u_n = p_1(n)\lambda_1^n + \cdots + p_k(n)\lambda_k^n, \qquad (12)$$

where the λ_i's are roots of the polynomial $z^mQ(1/z)$ and the p_i's are polynomials. An asymptotic expression follows from sorting the λ_i's by decreasing modulus. When the degree of Q is large, it is possible to avoid part of the computation, this is described in (Gourdon & Salvy 1996). The idea is to isolate only those elements of the partial fraction decomposition which involve the largest λ_i's.

Equation (12) explains the important numerical instability of the computation when the largest eigenvalue of the matrix (corresponding to the largest λ) is 1, which Theorem 2 shows to be the case in applications: if the probabilities of the transitions do not add up exactly to 1, this error is magnified exponentially when computing moments for large values of n. This is another motivation for using computer algebra in such applications, and, indeed, numerical stability problems problems are encountered by colleagues working with conventional programming languages.

The solution of linear systems is the bottleneck of our algorithmic chain. In the special case when one is interested only in expectation and variance of the number of occurrences of a pattern, it is possible to save time by computing only the local behaviour of the generating function. The bivariate system $(I - zT(u))L + \mathbf{1} = 0$ from (8) is satisfied when $u = 1$ by $S(1,z) = \mathbf{1}/(1-z)$. Letting $A = 1 - zT(u)$ and differentiating the system yields a new system for the generating functions of the expectations:

$$A(1,z)\frac{\partial S}{\partial u}(1,z)+\frac{\partial A}{\partial u}(1,z)S(1,z)=0. \tag{13}$$

The matrix A being of degree 1 in z, one has $A(1,z) = A_0 + A_1(1-z)$ and $\frac{\partial A}{\partial u}(1,z)\mathbf{1} = C_0 - C_0(1-z)$. The unknown vector $\frac{\partial S}{\partial u}(1,z)$ can be expanded locally as $X_0(1-z)^{-2}+X_1(1-z)^{-1}+X_2+O(1-z)$. Extracting coefficients of powers of $(1-z)$ in (13) yields

$$A_0X_0 = 0, \qquad A_0X_1 + A_1X_0 + C_0 = 0,$$
$$A_0X_2 + A_1X_1 - C_0 = 0.$$

The first equation is solved by $X_0 = \alpha\mathbf{1}$ for some constant α. Solving the second one for α and the vector X_1 yields α and X_1 up to a constant multiple of X_0. The constant is obtained by solving the third equation. The same process applies to the generating function of second moments after differentiating (8) twice with respect to u, using for unknown the truncated expansion

$$\frac{\partial^2 S}{\partial u^2}(1,z)=\frac{Y_0}{(1-z)^3}+\frac{Y_1}{(1-z)^2}+\frac{Y_2}{1-z}+O(1).$$

We give only the algorithm for the expectation, the variance is similar.

Algorithm 9 (Asymptotic Expectation).
Input: the bivariate system $(I - zT(u))L + \mathbf{1} = 0$ from (8).
Output: first two terms of the asymptotic behaviour of the expectation of the number of occurrences of the corresponding regular expression.

1. Let $A_1 = T(1)$, $A_0 = I - T(1)$, $C_0 = -\frac{\partial T}{\partial u}(1)$.
2. Solve the system $A_0X_1 + \alpha\mathbf{1} = -C_0$. This yields a value for α and a line $\tilde{X}_1 + \beta\mathbf{1}$ for X_1.
3. Solve the system $A_0X_2 + \beta\mathbf{1} = C_0 - A_1\tilde{X}_1$ for β. The expectation is asymptotically

   $$E = \alpha n + \alpha - x + O(A^n)$$

   for some $A < 1$ and x the coordinate of X_1 corresponding to the start state of the automaton.

Algorithm 9 reduces the computation of asymptotic expectation to the solution of a few linear systems with constant entries instead of one linear system with polynomial entries. This leads to a significant speed-up of the computation. Moreover, with due care, the systems could be solved using floating-point arithmetic. (This last improvement will be tested in the future; the current implementation relies on safe rational arithmetics.)

As can be seen from (12) a nice feature of the expansion of the expectation to two terms is that the remainder is exponentially small.

6 Implementation

The theory underlying the present paper has been implemented principally as a collection of routines in the Maple computer algebra system. Currently, only the Bernoulli model and the non-overlapping case have been implemented. The implementation is based mainly on the package **`combstruct`** (developed at Inria and a component of the Maple V.5 standard distribution) devoted to general manipulations of combinatorial specifications and generating functions. Use is also made of the companion Maple library **`gfun`** which provides various procedures for dealing with generating functions and recurrences. About 1100 lines of dedicated Maple routines have been developed by one of us (P. N.) on top of **`combstruct`** and **`gfun`**, including a new Maple function named **`regexp`** which effects the conversion of regular expressions describing motifs into deterministic finite automata[4].

This raw analysis chain does not include optimizations and it has been assembled with the sole purpose of testing the methodology we propose. It has been tested on a collection of 1118 patterns described below and whose processing took about 10 hours when distributed over 10 workstations. The computation necessitates an average of 6 minutes per pattern, but this average is driven up by a few very complex patterns. In fact, *the median of the execution times is only 8 seconds.*

There are two main steps in the computation: construction of the automaton and asymptotic computation of expectation and variance. Let R be the pattern, D the finite automaton, and T the arithmetic complexity of the underlying linear algebra algorithms. Then, the general bounds available are:

$$|R| \le |D| \le 2^{|R|}, \qquad T = O(|D|^3), \tag{14}$$

as results from the previous sections. (Sizes of R and D are defined as number of states of the corresponding NFA or DFA.) Thus, the driving parameter is $|D|$ and, eventually, the computationally intensive phase is due to linear algebra.

In practice, the upper bounds (14) that are exponential appear to be *extremely* pessimistic. Statistical analysis of the 1118 experiments indicates that the automaton is constructed in time slightly worse than linear in $|D|$ and that $|D|$ is almost always between $|R|$ and $|R|^2$. The time taken by the second step behaves roughly quadratically (in $O(|D|^2)$), which demonstrates that the sparseness of the system is properly handled by our program. For most of the patterns, the overall "pragmatic" complexity T_{obs} thus lies somewhere around $|R|^3$ or $|R|^4$ (see Figure 2).

7 Experimentation

We now discuss a small campaign of experiments conducted on PROSITE motifs intended to test the soundness of the methodological approach of this paper. No immediate biological relevance is implied. Rather, our

[4] Recent updates of `combstruct` and `gfun` are available at the URL `http://algo.inria.fr/libraries`. The motif-specific procedures are to be found under Pierre Nicodème's home page, at `http://www.dkfz.de/tbi/people/nicodeme`.

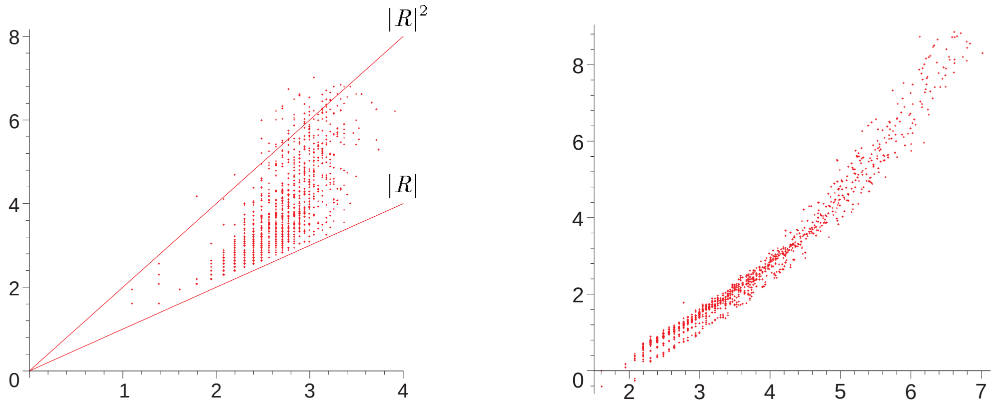

Figure 2: The correlations between $|R|, |D|$ (left) and $|D|, T_{\text{obs}}$ (right) in logarithmic scales.

aim is to check whether the various quantities computed do appear to have statistical relevance.

The biological target database, the "text", is built from the consensus sequences of the multi-alignments of PRODOM34.2. This database has 6.75 million positions, each occupied by one of 20 amino acids, so that it is long enough to provide matches for rare motifs. Discarding a few motifs constrained to occur at the beginning or at the end of a sequence (a question that we do not address here) leaves 1260 unconstrained motifs. For 1118 of these motifs (about 88% of the total) our implementation produces complete results. With the current time-out parameter, the largest automaton treated has 946 states. It is on this set of 1118 motifs that our experiments have been conducted.

For each motif, we have computed exactly the (theoretical) *expectation* E and *standard deviation* σ of the statistics of number of matches. The letter frequencies that we use in the mathematical and the computational model are the empirical frequencies in the database, and the quantities E, σ are determined by the computer algebra tools of the previous section: we use all the information coming from the pole at 1, which yields the first two terms of the asymptotic behaviour as given by Theorem 2. Each quantity E, σ is then compared to the corresponding number of observed matches (also called observables), denoted by O, that is obtained by a straight scan of the 6.75 million position PRODOM data base[5].

7.1 Expectations

First, we discuss expectations E versus observables O. For our reference list of 1118 motifs, the theoretical expectations E range from 10^{-23} to 10^5. The observed occurrences O range from 0 to 100,934, with a median at 1, while 0 is observed in about 12% of cases. Globally, we thus have a collection of motifs with fairly low expected occurrence numbers, though a few do have high

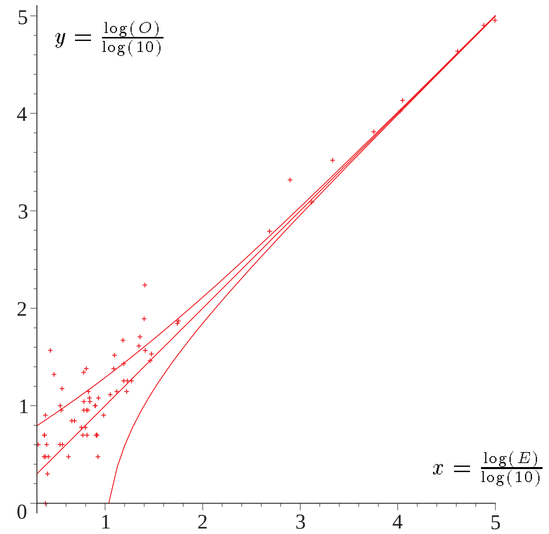

Figure 3: Motifs with theoretical expectation $E \geq 2$. Each point corresponds to a motif with coordinates (E, O) plotted on a log-log scale. The two curves represent an approximation of ± 3 standard deviations.

expected occurrences. Consider a motif to be "frequent" if $E \geq 2$. Figure 3 is our main figure: it displays in log-log scale points that represent the 71 pairs (E, O) for the frequent motifs, $E \geq 2$. The figure shows a good agreement between the *orders of growths* of predicted E and observed O values: (i) the average value of $\log_{10} O / \log_{10} E$ is 1.23 for these 71 motifs; (ii) the two curves representing 3 standard deviations enclose most of the data.

Figure 4 focusses on the classes of motifs observed $O = 1, 2, 3$ times in PRODOM. For each such class, a histogram of the frequency of observation versus $\log_{10} E$ is displayed. These histograms illustrate the fact that some motifs with very small expectation are still observed in the database. However, there is a clear tendency for motifs with smaller (computed) expectations E to occur less often: for instance, no motif whose ex-

[5]The observed quantities were determined by the PROSITE tools contained in the IRSEC motif toolbox `http://www.isrec.isb-sib.ch/ftp-server/`.

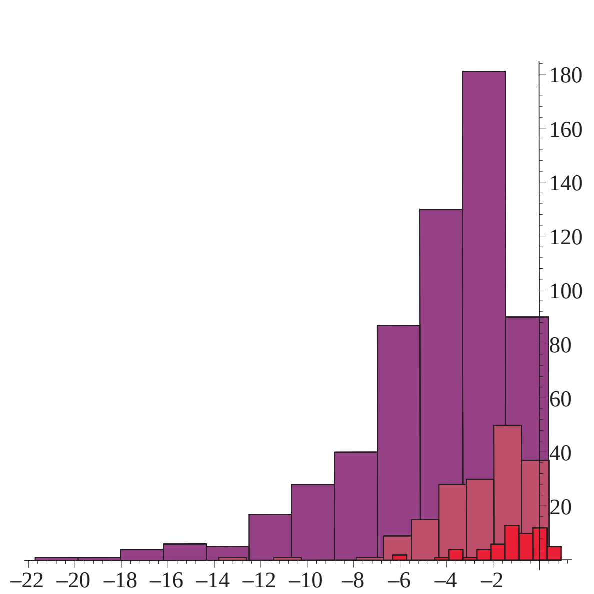

Figure 4: Histograms of motifs with 1 (dark gray), 2 (medium gray) and 3 (white) observed matches. Coordinates: $x = \log_{10} E, y =$ number of motifs.

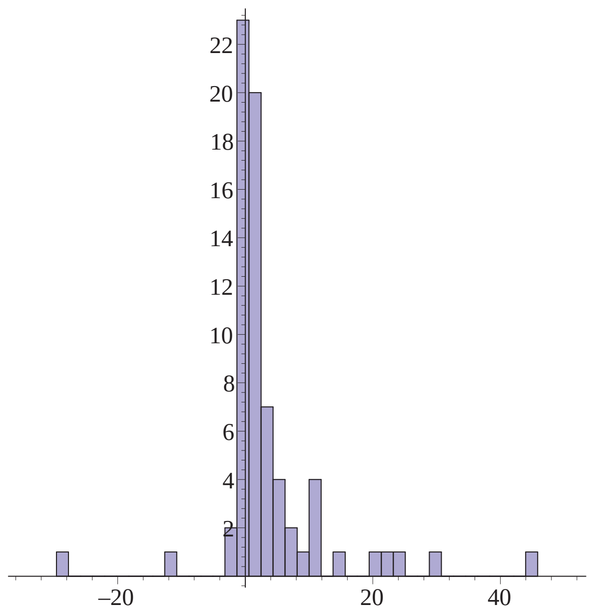

Figure 5: Motifs with theoretical expectation $E \geq 2$: Histogram of the Z-scores $Z = \frac{O-E}{\sigma}$.

pectation is less than 10^{-6} occurs 3 times.

7.2 Z-scores

Another way to quantify the discrepancy between the expected and the observed is by means of the Z-score that is defined as

$$Z = \frac{O - E}{\sigma}.$$

Consider again motifs that are frequent, namely those whose expectation satisfies $E \geq 2$. Histograms of the Z-scores for this class of motifs should converge to a Gaussian curve if the Bernoulli model would apply strictly and if there would be a sufficient number of data corresponding to large values of E. None of these conditions is satisfied here, but nonetheless, the histogram displays a sharply peaked profile tempered by a small number of exceptional points.

7.3 Standard deviations

We now turn to a curious property of the Bernoulli model regarding standard deviations. At this stage this appears to be a property of the model alone. It would be of interest to know whether it says something meaningful about the way occurrences tend to fluctuate in a large number of observations.

Theoretical calculations show that when the expectation of the length between two matches for a pattern is large, then

$$\sigma \approx \sqrt{E}$$

is an excellent approximation of the standard deviation. Strikingly enough, computation shows that for the 71 "frequent" patterns, we have $0.4944 \leq \log(\sigma)/\log(E) \leq 0.4999$. (Use has been made of this approximation when plotting (rough) confidence intervals of 3 standard deviations in Fig. 3.)

7.4 Discussion

The first blatant conclusion is that predictions (the expectation E) tend to underestimate systematically what is observed (O). This was to be expected since the Prosite patterns do have an *a priori* biological significance. A clearer discussion of this point can be illustrated by an analogy with words in a large *corpus* of natural language, such as observed with Altavista on the Web. The number of occurrences of a word such as 'deoxyribonucleic' is very large (about 7000) compared to the probability (perhaps 10^{-15}) assigned to it in the Bernoulli model. Thus, predictions on the category of patterns that contain long (hence unlikely) words that can occur in the *corpus* are expected to be gross underestimations. However, statistics for a pattern like "A ⟨ any_word ⟩ IS IN" (590,000 matches) are more likely to be realistic, not for reasons of laws of large numbers alone.

This naive observation is consistent with the fact that Fig. 3 is more accurate for frequent patterns than for others, and it explains why we have restricted most of our discussion to patterns such that $E \geq 2$. In addition, we see that the scores computed are meaningful as regards orders of growth, at least. This is supported by the fact that $\log O/\log E$ is about 1.23 (for the data of Fig. 3), and by the strongly peaked shape of Fig. 5.

Finally we discuss the patterns that are "exceptional" according to some measure.

— The largest automaton computed has 946 states and represents the expression $\Sigma^* R$ for the motif PS00844 ("[LIV]-x(3)-[GA]-x-[GSAIV]-R-[LIVCA]-D-[LIVMF](2)-x(7,9)-[LI]-x-E-[LIVA]-N-[STP]-x-P-

Index	Pattern	E	O	Z	$\frac{O-E}{E}$
2	S-G-x-G	2149	3302	25	0.54
4	[RK](2)-x-[ST]	11209	13575	22	0.21
13	DERK(6)-[LIVMFWSTAG](2)-[LIVMFYSTAGCQ]-[AGS]-C	788	2073	46	1.63
36	[KR]-x(1,3)-[RKSAQ]-N-x(2)-[SAQ](2)-x-[RKTAENQ]-x-R-x-[RK]	2.75	37	20	12.45
190	C-CPWHF-CPWR-C-H-CFYW	25	173	29	5.86
5	[ST]-x-[RK]	99171	90192	-30	-0.09

Table 1: Motifs with large Z-scores

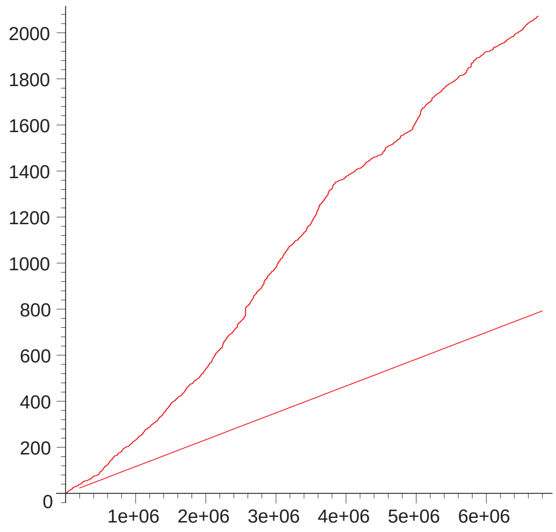

Figure 6: Scanning PRODOM with motif PS00013. Observed matches versus expectation.

[GA]", DALA_DALA_LIGASE_2). Expectation for this motif is 1.87×10^{-6}, standard-deviation 0.00136, while $O = 0$. This automaton corresponds to a finite set of patterns whose cardinality is about 1.9×10^{26}.

— The pattern with largest expectation is PS0006 ("[ST]-x(2)-[DE]", CK2_PHOSPHO_SITE) for which $E = 104633$ (and $O = 100934$) and the renewal time between two occurrences is as low as 64 positions.

— The motifs with very exceptional behaviours $|Z| > 19$ are listed in Table 1. The motif PS00005 ("[ST]-x-[RK]", PKC_PHOSPHO_SITE) is the only motif that is clearly observed significantly less than expected.

We plot in Fig. 6 the number of observed and expected matches of PS00013 against the number of characters of PRODOM that have been scanned. The systematic deviation from what is expected is the type of indication on the possible biological significance of this motif that our approach can give.

8 Directions for Future Research

There are several directions for further study: advancing the study of the Markov model; enlarging the class of problems in this range that are guaranteed to lead to Gaussian laws; conducting sensitivity analysis of Bernoulli or Markov models. We briefly address each question in turn.

The Markov model. Although the Markov model on letters is in principle analytically and computationally tractable, the brute-force method given by algorithm "Markov automaton" probably leaves room for improvements. We wish to avoid having to deal with finite-state models of size the product $|\Sigma| \times |Q|$, with $|\Sigma|$ the alphabet cardinality and $|Q|$ the number of states of the automaton. This issue appears to be closely related to the areas of Markov chain decomposability and of Markov modulated models.

Gaussian Laws. Our main theoretical result, Theorem 2, is of wide applicability in all situations where the regular expression under consideration is "nondegenerate". Roughly, as explained in Section 4, the overwhelming majority of regular expression patterns of interest in biological applications are expected to be nondegenerate. (Such is for instance the case for *all* the motifs that we have processed.) Additional work is called for regarding sufficient structural conditions for nondegeneracy in the case of Markov models. It is at any rate the case that the conditions of Theorem 2 can be tested easily in any specific instance.

Model sensitivity and robustness. An inspection of Table 1 suggests that the exceptional motifs in the classification of Z-scores cover very different situations. While a ratio O/E of about 3 and an observable O that is > 2000 is certainly significant, some doubt may arise for other situations. For instance, is a discrepancy of 5% only on a motif that is observed about 10^5 times equally meaningful? To answer this question it would be useful to investigate the way in which small changes in probabilities may affect predictions regarding pattern occurrences. Our algebraic approach supported by symbolic computation algorithms constitutes an ideal framework for investigating model sensitivity, that is, the way predictions are affected by small changes in letter or transition probabilities.

Acknowledgement

This work has been partially supported by the Long Term Research Project Alcom-IT (#20244) of the European Union.

References

Atteson, K. 1998. Calculating the exact probability of language-like patterns in biomolecular sequences. In *Sixth International Conference on Intelligent Systems for Molecular Biology*, 17–24. AAAI Press.

Bairoch, A.; Bucher, P.; and Hofman, K. 1997. The PROSITE database, its status in 1997. *Nucleic Acids Res.* 25:217–221. MEDLINE: 97169396, `http://expasy.hcuge.ch/sprot/prosite.html`.

Bender, E. A., and Kochman, F. 1993. The distribution of subword counts is usually normal. *European Journal of Combinatorics* 14:265–275.

Bender, E. A.; Richmond, L. B.; and Williamson, S. G. 1983. Central and local limit theorems applied to asymptotic enumeration. III. Matrix recursions. *Journal of Combinatorial Theory* 35(3):264–278.

Bender, E. A. 1973. Central and local limit theorems applied to asymptotic enumeration. *Journal of Combinatorial Theory* 15:91–111.

Berry, G., and Sethi, R. 1986. From regular expressions to deterministic automata. *Theoretical Computer Science* 48(1):117–126.

Billingsley, P. 1986. *Probability and Measure.* John Wiley & Sons, 2nd edition.

Brüggemann-Klein, A. 1993. Regular expressions into finite automata. *Theoretical Computer Science* 120(2):197–213.

Chomsky, N., and Schützenberger, M. P. 1963. The algebraic theory of context-free languages. In *Computer programming and formal systems.* Amsterdam: North-Holland. 118–161.

Flajolet, P., and Sedgewick, R. 1997. The average case analysis of algorithms: Multivariate asymptotics and limit distributions. Research Report 3162, Institut National de Recherche en Informatique et en Automatique. 123 pages.

Flajolet, P.; Kirschenhofer, P.; and Tichy, R. F. 1988. Deviations from uniformity in random strings. *Probability Theory and Related Fields* 80:139–150.

Gantmacher, F. R. 1959. *The theory of matrices. Vols. 1, 2.* New York: Chelsea Publishing Co. Translated by K. A. Hirsch.

Gourdon, X., and Salvy, B. 1996. Effective asymptotics of linear recurrences with rational coefficients. *Discrete Mathematics* 153(1–3):145–163.

Guibas, L. J., and Odlyzko, A. M. 1981. String overlaps, pattern matching, and nontransitive games. *Journal of Combinatorial Theory. Series A* 30(2):183–208.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to automata theory, languages, and computation.* Addison-Wesley Publishing Co., Reading, Mass. Addison-Wesley Series in Computer Science.

Hwang, H. K. 1994. *Théorèmes limites pour les structures combinatoires et les fonctions arithmétiques.* Ph.D. Dissertation, École polytechnique, Palaiseau, France.

Kelley, D. 1995. *Automata and formal languages.* Englewood Cliffs, NJ: Prentice Hall Inc. An introduction.

Knuth, D. E. 1981. *The art of computer programming. Vol. 2. Seminumerical algorithms.* Computer Science and Information Processing. Reading, Mass.: Addison-Wesley Publishing Co., second edition.

Kozen, D. C. 1997. *Automata and computability.* New York: Springer-Verlag.

Pevzner, P. A.; Borodovski, M. Y.; and Mironov, A. A. 1989. Linguistic of nucleotide sequences: The significance of deviation from mean statistical characteristics and prediction of the frequencies of occurrence of words. *Journal of Biomolecular Structure Dyn.* 6:1013–1026.

Prasolov, V. V. 1994. *Problems and theorems in linear algebra.* Providence, RI: American Mathematical Society. Translated from the Russian manuscript by D. A. Leĭtes.

Prum, B.; Rodolphe, F.; and de Turckheim, É. 1995. Finding words with unexpected frequencies in deoxyribonucleic acid sequences. *Journal of the Royal Statistical Society. Series B* 57(1):205–220.

Rayward-Smith, V. J. 1983. *A first course in formal language theory.* Oxford: Blackwell Scientific Publications Ltd.

Régnier, M., and Szpankowski, W. 1998. On pattern frequency occurrences in a Markovian sequence. *Algoritmica.* To appear.

Régnier, M. 1998. A unified approach to words statistics. In *Second Annual International Conference on Computational Molecular Biology*, 207–213. New-York: ACM Press.

Reinert, G., and Schbath, S. 1998. Compound Poisson approximations for occurrences of multiple words in Markov chains. *Journal of Computational Biology* 5(2):223–253.

Salvy, B., and Zimmermann, P. 1994. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software* 20(2):163–177.

Schbath, S.; Prum, B.; and de Turckheim, É. 1995. Exceptional motifs in different Markov chain models for a statistical analysis of DNA sequences. *Journal of Computational Biology* 2(3):417–437.

Sewell, R. F., and Durbin, R. 1995. Method for calculation of probability of matching a bounded regular expression in a random data string. *Journal of Computational Biology* 2(1):25–31.

Vallée, B. 1998. Dynamical sources in information theory: Fundamental intervals and word prefixes. Les cahiers du GREYC, Université de Caen. 32p.

Waterman, M. S. 1995. *Introduction to Computational Biology: Maps, sequences and genomes.* Chapman & Hall.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105,
78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS
Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
(France)
http://www.inria.fr
ISSN 0249-6399