Fundamenta Informaticae LVI (2003) 71–87
IOS Press

Regexpcount, a symbolic package for counting problems on regular expressions and words

Pierre Nicodème[*]
LIX - École polytechnique
and
Algorithm Project, INRIA-Rocquencourt

Abstract. In previous work [10], we considered algorithms related to the statistics of matches with words and regular expressions in texts generated by Bernoulli or Markov sources. In this work these algorithms are extended for two purposes: to determine the statistics of simultaneous counting of different motifs, and to compute the waiting time for the first match with a motif in a model which may be constrained. This extension also handles matches with errors. The package is fully implemented and gives access to high and low level commands. We also consider an example corresponding to a practical biological problem: getting the statistics for the number of matches of words of size 8 in a genome (a Markovian sequence), knowing that an (overrepresented DNA protecting) pattern named Chi occurs a given number of times.

Keywords: marked automata, generating functions, computer algebra.

1. Introduction

An important class of computational biology problems is related to counting. When considering words, one is interested in the probability of match in a sequence, the statistics of occurrences in a genome, and the selection of words with unusual counts. These problems have been studied by several authors using combinatorics or probabilistic (Poisson approximation) methods, in the Bernoulli and Markov case [11, 12, 13, 15, 14, 16, 18]. When considering regular expressions or motifs such as Prosite motifs,

[*]An extended abstract of this article has appeared in the Proceedings of the fifteenth german conference on bioinformatics GCB00 [8].
Address for corespondence: LIX - École polytechnique, 91128 Palaiseau cedex, France.
E-mail: `nicodeme@lix.polytechnique.fr`

Table 1. Examples of computations (see notations in Section 2)

pat.	stat.	md.	$\mu(n)$	$\sigma^2(n)$	$\mu(1000)$	$\sigma^2(1000)$
$abab$	n.m.o.	B_0	$\frac{1}{16}n - \frac{3}{16} + \mathrm{O}(\alpha^n)$	$\frac{17}{256}n - \frac{55}{256} + \mathrm{O}(\alpha^n)$	62.31	66.10
$aaab$	n.m.o	B_0	$\frac{1}{16}n - \frac{3}{16} + \mathrm{O}(\alpha^n)$	$\frac{9}{256}n - \frac{15}{256} + \mathrm{O}(\alpha^n)$	62.31	35.05
$abab$	n.m.r.	B_0	$\frac{1}{20}n - \frac{13}{100} + \mathrm{O}(\alpha^n)$	$\frac{69}{2000}n - \frac{139}{2000} + \mathrm{O}(\alpha^n)$	49.87	34.34
$abab$	n.m.o.	B_1	$\frac{9}{256}n - \frac{27}{256} + \mathrm{O}(\alpha^n)$	$\frac{2601}{65536}n - \frac{8559}{65536} + \mathrm{O}(\alpha^n)$	35.05	39.44
$abab$	n.m.o.	M_1	$\frac{9}{80}n - \frac{297}{800} + \mathrm{O}(\alpha^n)$	$\frac{3627}{32000}n - \frac{52353}{128000} + \mathrm{O}(\alpha^n)$	112.12	112.78
$\Delta(abab, 1)$	n.m.o.	B_0	$\frac{5}{8}n - \frac{11}{8} + \mathrm{O}(\alpha^n)$	$\frac{19}{64}n - \frac{23}{64} + \mathrm{O}(\alpha^n)$	623.62	296.18
$\Delta(abab, 1)$	n.m.o.	B_1	$\frac{63}{128}n - \frac{141}{128} + \mathrm{O}(\alpha^n)$	$\frac{6957}{16384}n - \frac{12663}{16384} + \mathrm{O}(\alpha^n)$	491.08	423.54
$\Delta(abab, 1)$	n.m.o.	M_1	$\frac{57}{80}n - \frac{1311}{800} + \mathrm{O}(\alpha^n)$	$\frac{7323}{32000}n - \frac{17601}{128000} + \mathrm{O}(\alpha^n)$	710.86	228.61
			$\lvert\alpha\rvert < 1$			
pat.	stat.	md.			μ	σ^2
$\epsilon, abab$	wait	B_0			20	275.89
$\epsilon, aaab$	wait	B_0			16	144
$abab, aaab$	wait	B_0			16	144
$aaab, abab$	wait	B_0			16	271.92
$\epsilon, abab$	wait	B_1			33.77	925.38
$\epsilon, abab$	wait	M_1			12.72	82.26

one is also interested in probability of match in a sequence [17], or, once again, in the statistics of number of matches [10]. In this article, we consider regular expressions, with words and patterns (finite set of words) as a subclass of these, and texts generated by either a (uniform or non-uniform) Bernoulli source or by a Markovian source. In [10], we addressed the counting problem for matches of one motif in these texts. Our method was based on construction of automata and the analysis of their generating functions (with one variable and one parameter). We extend this approach to handle simultaneous counting of several motifs, which corresponds to generating functions with one variable and several parameters and gives access to covariance statistics. We also consider a new statistic, the waiting time for the first match with a regular expression RE_2; this last problem may be constrained by knowing that a match just occurred with a regular expression RE_1, either with rematch with RE_1 allowed, or forbidden. Another new feature of the package is the possibility of handling statistics for matches with errors, for all the problems considered. In Section 2 we give some examples of our computations. In Section 3 we provide the necessary definitions. We describe our methods and algorithms in Section 4 and several extensions in Section 5. We give in Section 6 an application to a biological problem.

2. Examples

Table 1 contains some examples of the possible computations, with the following notations:

pat.: pattern. $\Delta(regexp, k)$: language of words at edit distance (substitution, insertion, deletion) $\leq k$ from $regexp$. In the case of the wait statistics, two patterns are given (see *stat.*).

stat.: considered statistics; (1) n.m.o : number of matches in the overlapping case (overlapping matches are accepted); (2) n.m.r. : number of matches in the renewal case (restart after each match, or non-overlapping case); (3) wait (pattern p_1, p_2): wait time for the first match with pattern p_2, knowing that pattern p_1 has been found (rematch with p_1 is allowed during the wait). If $p_1 = \epsilon$, wait time for the first match with p_2 from the beginning of the text.

md.: model; (1) B_0: Bernoulli uniform; (2) B_1: Bernoulli non-uniform (probability of letters $\pi_a = 1/4, \pi_b = 3/4$); (3) M_1: Markov model of order 1; (probability associated to the Markov chain: $\pi_a = 1/4, \pi_b = 3/4, \pi_{a,a} = 1/4, \pi_{a,b} = 3/4, \pi_{b,a} = 1/2, \pi_{b,b} = 1/2$, where π_x is the initial probability of letter x and $\pi_{x,y}$ is the probability that the letter x is followed by the letter y in the text).

$\mu(n), \sigma^2(n)$: asymptotic value of the expectation and variance for texts of size n.

$\mu(1000), \sigma^2(1000)$: numerical evaluation of the expectation and variance for texts of size 1000.

μ, σ^2 (wait statistics): numerical evaluation of the expectation and variance for the wait statistics.

See Section A of the appendix for a Maple session which computes lines $\Delta(abab, 1)$, n.m.o, B_1 and M_1 of Table 1.

3. Definitions

We recall in this section the classical definitions for languages, generating functions and finite automata.

Languages. An *alphabet* is a (finite) set of letters. A *word* is a concatenation of letters of the alphabet. A *language* over an alphabet Σ is a subset of the set $\Sigma^\star$ of all words over the alphabet. *Concatenation* of languages is denoted by a product ($A_1A_2 = \{w_1w_2, w_1 \in A_1, w_2 \in A_2\}$). *Union* of languages is the ordinary set union. The *empty word* is denoted by ϵ and the *Kleene star operator* "$\star$" is understood as $A^\star = \epsilon + A + A^2 + A^3 + \dots$, where $A^2 = AA$ and so on. A *regular language* over an alphabet Σ is built by recursively applying Union, Concatenation and Kleene-Star operators to the singleton languages $\{\epsilon\}$ and $\{\sigma\}$ ($\forall\sigma \in \Sigma$). A *regular expression* is a short-hand description of a regular language (most commonly using symbols "$., +, \star$" and brackets).

Generating functions. We give the definitions for an alphabet of 2 letters $\Sigma = \{a, b\}$ and consider generating functions for a language L. They generalize immediately to alphabets of higher cardinality. Define the *counting generating function* of a language L as $F(a, b) = \sum c_{i,j}a^ib^j$ where $c_{i,j}$ is the number of words of L with i letters a and j letters b; this is equivalent to $F(a, b) = \sum_{w\in L} \mathrm{com}(w)$, where the operator "com", applied to a word w, produces the monomial obtained by letting the letters of w commute. For example, if $L = \{aab, aba\}$ then $F(a, b) = 2a^2b$. Remark that if d_n is defined by $F(z, z) = \sum d_nz^n$ then d_n counts the number of words of size n in the language. Similarly the *"probability" generating function* of L is $P(z) = \sum \omega_n z^n$, where ω_n is the probability that a word of size n belongs to L. In the Bernoulli models B_0 and B_1 this is $F(z/2, z/2)$ and $F(\pi_a z, \pi_b z)$ where π_a, π_b are the probabilities of occurrences of letters a and b respectively. For example with $L = \{aab, aba\}$, we have $P(z) = z^3/4$ in the model B_0. In the Markov model M_1 the probability of each word is computed by using the probability of occurrences of the first letters and the transitions probabilities (this generalizes easily to higher Markov models).

As an example, the bivariate generating function of number of matches of the pattern $abab$ in the Markov model M_1 of Section 2 is

$$F(z,u) = \sum f_{kn} u^k z^n = \frac{1}{2} \times \frac{128 + 32z + (1-u)(48z^2 + 12z^3 + 9z^4)}{64 - 48z - 16z^2 + (1-u)(24z^2 - 18z^3 + 3z^4)},$$

where f_{kn} is the probability that there are k matches in a text of size n.

Translation rules. If unions are disjoint and concatenations are unambiguous, unions of languages translate to sums of generating functions and concatenations translate to products of generating functions; with the same conditions, the counting and probability generating functions of $L^\star$ are the quasi-inverses $F_{L^\star}(a,b) = 1/(1-F_L(a,b))$ and $P_{L^\star}(z) = 1/(1 - P_L(z))$.

Automata. A *Nondeterministic Finite Automaton* (or NFA) is a 5-tuple $(\Sigma, Q, s, F, \delta)$ such that: Σ is the input alphabet; Q is a finite collection of states; $s \in Q$ is the start state; $F \subset Q$ is the collection of final states; δ is a (possibly partial) transition function 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. Given an input regular language there are several algorithms to construct an NFA that recognizes it. See [6] among numerous text books for the construction using ϵ-transitions or [2] for the Berry-Sethy algorithm.
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 and Scott, NFAs are equivalent to DFAs in the sense that they recognize the same class of languages.
An important theorem of Chomsky and Schützenberger states that the generating function $F(z)$ of a regular language R is rational [3]. The proof uses a DFA recognizing the language R as an intermediate step; a system of linear equations is then deduced from the DFA; one of the unknowns of the system is $F(z)$.

4. Methods and algorithms

4.1. Marked automata

The process is best illustrated with an example. Consider the pattern aba and counting the number of matches of this pattern in random texts over the alphabet $\Sigma = \{a, b\}$. This may be done by considering marked texts where a mark m not belonging to Σ is inserted in the texts after each match. If we consider the text $t = aababaaabab$, we get $\mathrm{mark}(t) = aabambamaabamb$ if overlapping matches are allowed and $\mathrm{mark}(t) = aabambaaabamb$ otherwise.

A *marked automaton* is an automaton $\{\Sigma, Q, s, F, \delta, M_1, \ldots, M_k\}$ where $M_1, \ldots, M_k$ are subsets of the set Q of states that are distinguished. For instance, consider the DFA $A = \{\Sigma\ (=\{a, b\}), Q, s, F, \delta\}$ recognizing the regular expression $\Sigma^* aba$. When running this automaton against a text, each state reached after reading an aba is a final state. From there, define a marked automaton A_m with $M_1 = F$ as marked subset and all states final. Therefore $A_m = (\Sigma, Q, s, Q, \delta, F)$. We consider now a DFA $A' = (\Sigma' = \{a, am, b, bm\}, Q, s, Q, \delta_m)$ deduced from A_m by transforming δ to a marked transition function δ_m as follows: $\forall s \in Q, \forall l \in \Sigma$, if $\delta(s, l) \notin M_1$ then $\delta_m(s, l) = \delta(s, l)$, else $\delta_m(s, lm) = \delta(s, l)$ and $\delta_m(s, l)$ is not defined. Then A' recognizes all the marked texts where the mark m has been inserted after each match. This method has been developed in [10] and we extend it to the case where r patterns are considered. In the latter case, we define r marked sets $M_1, \ldots, M_r$ corresponding to the matches with each pattern, and a suitable rule of marking the transitions is chosen.
Since the marked automata that we construct in this paragraph are deterministic, the Chomsky and Schützenberger method is available. When a single motif is considered, it produces, in the Bernoulli case, a generating function $F(a, b, m)$. From $F(a, b, m)$ we get the multivariate probability generating function $P(z, u) = F(\pi_a z, \pi_b z, u) = \sum p_{n,k} u^k z^n$, where $p_{n,k}$ is the probability that a text of size n contains k occurrences of the pattern. In what follows, we consider that the insertion of marks is done during the Chomsky-Schützenberger translation into generating functions.

4.2. Algorithms

We consider a problem for one or several motifs $R_1, \ldots, R_i$ $(i \geq 1)$. The algorithmic chain is as follows.

1. If the match(es) with the motif R_i is (are) considered with a function of error Δ, build first an automaton recognizing R_i and next an "error" automaton recognizing $\Delta(R_i)$. Use classical ϵ-transitions constructions to connect this automaton to the automata built during the following step.
2. Construct a deterministic Bernoulli automaton corresponding to the problem.
3. If the model is Markov, transform the last automaton constructed to a Markov automaton.
4. Compute the (eventually multivariate) counting or probability generating function of the language recognized by the automaton.
5. Use computer algebra methods to extract the Taylor coefficients of order n of the generating functions, which provide expectation(s), moment(s) of order 2, or covariance(s) of the statistical parameters under study. See [10] for details.

Remarks about implementation: steps 1-3 are implemented in the package `regexpcount`; step 4 is performed either by use of the package `combstruct` (combinatorial structures), or by combination of functions of the packages `regexpcount` and `combstruct` and by the standard solver of Maple; step 5 is performed either by the package `gfun` (generating functions) when computing exact coefficients or by the packages `equivalent` (asymptotic expansions of coefficients of generating functions) and `gdev` (general asymptotic expansions) for asymptotic results.

Properties of marked states: For a given mark, the corresponding set of marked states shares the properties of the set of final states for the determinization or minimization of automata. This follows from the fact that, at a given time of the construction of a marked automata, the set of marked states has been (or has been equivalent to) the set of final states of an automaton. Similarly, the marking process behaves nicely during the cartesian product of automata.

NFA determinization. It uses the subset construction adapted to the case of marked automata. Consider the marked NFA $A_N = \{\Sigma, Q, s, F, \delta, M\}$ where M is the subset of states marked with the letter m (the extension to a sequence of sets $M_1, \ldots, M_r$ where M_j is the subset of states marked by the letter m_j is immediate). Forgetting M, the subset construction on A_N gives a DFA $A_D = \{\Sigma, \mathcal{Q}, \{s\}, \mathcal{F}, \Delta\}$ where $\mathcal{Q}$ (the set of states) is the subset of 2^Q used during the construction, Δ is the transition function induced on this set and $\mathcal{F} = \{S \in \mathcal{Q} \,|\, S \cap F \neq \emptyset\}$. This automaton is marked by adjoining the set of marked states $\mathcal{M} = \{S \in \mathcal{Q} \,|\, S \cap M \neq \emptyset\}$ to A_D.

Automaton minimization. It is performed when necessary during the algorithmic chain.

A marked automaton may be considered as an automaton that simultaneously recognizes several regular languages. For a given automaton A_m, a minimized automaton is an automaton with minimum number of states that simultaneously recognizes the same languages as A_m.

We consider first the classical minimization of non-marked deterministic automata. See [7] for details and proofs. Consider a deterministic automaton $A = \{\Sigma, Q, s, F, \delta\}$. We define inductively on the length of the strings the function $\widehat{\delta} : Q \times \Sigma^\star \to Q$ by

$$\widehat{\delta}(q, \epsilon) = q$$
$$\widehat{\delta}(q, xa) = \delta(\widehat{\delta}(q, x), a)$$

An equivalence relation $\equiv_A$ is defined on the set of states Q by

$$q_i \equiv_A q_j \quad \Leftrightarrow \quad \{\forall x \in \Sigma^\star, \quad \widehat{\delta}(q_i, x) \in F \Leftrightarrow \widehat{\delta}(q_j, x) \in F\}$$

The automaton naturally defined on the equivalence classes of the relation $\equiv_A$ on Q is one instance of minimal automaton equivalent to A.

Let us consider now a marked automaton $A_m = \{Q, s, F, \delta, M_1, \ldots, M_r\}$, where (after eventually renaming the marked sets) the index of the marked sets ranges from 1 to r. We identify the set F with a set M_0 and therefore the index set I ranges from 0 to r. The function $\widehat{\delta}$ is defined as previously. We consider now the set $\mathcal{I} = 2^I$ of subsets of I and for each $J \in \mathcal{I}$ the sets

$$N_J = \bigcap_{j \in J} M_j. \tag{1}$$

In many cases, the set N_J is empty. We denote by $\overline{\mathcal{I}}$ the set of indices J for which N_J is not empty and not duplicated that is defined by

$$\overline{\mathcal{I}} = \{J \in 2^I, \qquad N_J \neq \emptyset \quad \text{and} \quad \nexists J',\ J \subset J' \text{ and } N_J = N_{J'}\}.$$

We can describe this partitioning in terms of coloring by assigning a color to the set of terminal states and to each set of marked states; equivalently, we assign a color to each integer from 0 to r. The set of states is partitioned along the subsets of colors assigned to each state. The cardinality of the partition is upper bounded by $|Q|$ and by 2^{r+1}. We extend now the definition of the equivalence relation given precedently to the case of marked automata. We consider the marked automaton A_m defined above and consider the relation $\equiv_{A_m}$ on the set of states Q defined by

$$q_1 \equiv_{A_m} q_2 \quad \Leftrightarrow \quad \{\forall x \in \Sigma^\star, \forall J \in \overline{\mathcal{I}}, \quad \widehat{\delta}(q_1, x) \in N_J \Leftrightarrow \widehat{\delta}(q_2, x) \in N_J\}. \tag{2}$$

It is easy to prove that the relation $\equiv_{A_m}$ is an equivalence relation. The automaton induced by δ on the classes of equivalence of $Q/{\equiv_{A_m}}$ is an instance of minimal marked automaton equivalent to the original automaton A_m. All the required proofs are identical to the proofs done for classical automata.

The minimization algorithm for handling marked automata uses a slight generalization of the Hopcroft $n\log(n)$ algorithm [5]. Instead of beginning with a partition of the set of states with 2 subsets, the set of final states F and its complement $\overline{F}$, we begin with the partition $\{N_J, J \in \overline{\mathcal{I}}\}$ (see Equation 1). The algorithm then further refines the partition as does Hopcroft's one, and the subset created at each step has size at most half of the splitted subset. The algorithm has therefore complexity $O(n \log n)$ as Hopcroft's algorithm does.

Product automaton. An automaton is complete if for any state and any letter the transition function is defined. Let $[.,.]$ denote an ordered list. Given two complete DFAs $A_i = (\Sigma, Q_i, s_i, F_i, \delta_i, M_i),\ i = 1, 2$, we construct the marked product automaton $B = A_1 \times A_2 = (\Sigma, Q_B, [s_1, s_2], F_B, \delta_B, M_{B,1}, M_{B,2})$ as follows: Q_B is a subset of $\{[x, y], x \in Q_1, y \in Q_2\}$; if $[x, y] \in Q_B$, then $\delta_B([x, y], l) = [\delta_1(x, l), \delta_2(y, l)]$ belongs to Q_B. The start state $[s_1, s_2]$ belongs to Q_B. $F_B = \{[x, y], x \in F_1, y \in F_2\}$. The distinguished subsets are $M_{B,1} = \{[x, y], x \in M_1\}, M_{B,2} = \{[x, y], y \in M_2\}$. The construction is classical, but completed to handle the case of marked automata. This construction generalizes immediately to the product of more than 2 automata.

5. Extensions

5.1. Number of pairs of occurrences.

The automaton construction for counting the number of occurrences of a pattern in a text has been described in [10] and in Section 4. To simultaneously consider the occurrences of 2 patterns p_1 and p_2, construct two marked automata A_1 and A_2 recognizing the regular expressions $\Sigma^* p_1$ and $\Sigma^* p_2$ respectively and compute the product $B = A_1 \times A_2 = \{\Sigma, Q, s, Q, M_1, M_2\}$ of the two automata, where the set of final states is made equal to Q. This is the required automaton. The corresponding probability generating function is of the form $P(z, u, v) = \sum p_{n,i,j} u^i v^j z^n$ where $p_{n,i,j}$ is the probability that a text of size n has i occurrences of p_1 and j occurrences of p_2. The construction generalizes immediately to the case where more than two patterns are simultaneously considered.

5.2. Waiting times

We consider now two patterns $p_1 = ab$ and $p_2 = ba$. In this case, we are interested in the waiting time for a match with p_2, knowing that a match with p_1 occurred. (See the corresponding next paragraph for an algorithm). Let us exemplify the marking process on the text $t = abbbba$. This text begins with a match with p_1, finishes with a match with p_2 and has no other matches with p_2. The marked text is $\mathrm{mark}(t) = abbmbmbmam$; this text contains 4 marks m, corresponding to the number of letters read after the match with p_1 until the match with p_2 occurred. We assume that we know how to construct a marked automaton $B_{m,1} = (\Sigma, Q, s, F, \delta, M_1, M_2)$ verifying the following conditions: $B_{m,1}$ recognizes the texts that begin with a match with p_1, end with a match with p_2 and have no other match with p_2 (except possibly within the first match with p_1); M_1 is the set of states reached when matching p_1 and M_2 is the set of states crossed after a match with p_1 until a first match with p_2; furthermore we assume that no path from s to M_1 intersects a path from M_1 to F. Let Q_{p_1} be the subset of states of Q that are accessible and coaccessible in $B_{p_1} = (\Sigma, Q, s, M_1, \delta)$; note that B_{p_1} recognizes p_1. Then $M_2 = Q - Q_{p_1}$. Said in different words, the constructed automaton splits into a first part where p_1 is recognized and into a second part where the first match with p_2 is recognized, and the marked set M_2 is the set of states of the second part. Forget M_1 in $B_{m,1}$ to produce $B_{m,2} = (\Sigma, Q, s, F, \delta, M_2)$. Translate into a generating function $F(a, b, m)$ which enumerates the marked texts. In the Bernoulli model, this gives the probability generating function $P(u) = F(\pi_a, \pi_b, u) = \sum s_n u^n$ where s_n is the probability that the waiting time is n. From there, the expectation and second moment of the statistics are immediately obtained.

Waiting time, first match with a pattern p_1. Construct the automaton for $\Sigma^* p_1$ and erase all transitions from final states.

Waiting time $p_1 \to p_2$ for a match with p_2 after a match with p_1. Construct the marked automata $A^{(1)}$ recognizing p_1 and $A^{(2)}$ recognizing $\Sigma^* p_2$, where $A^{(i)} = (\Sigma, Q^{(i)}, s^{(i)}, Q^{(i)}, \delta^{(i)}, M^{(i)})$. Construct the automaton $B = A^{(1)} \times A^{(2)} = (\Sigma, Q, s, F, \delta,\ M_1, M_2)$. Construct an automaton C by duplicating the states of B by a bijective copy function ψ of Q onto $\psi(Q)$ such that $Q \cap \psi(Q) = \emptyset$. Then the required automaton is $C = (\Sigma, Q \cup \psi(Q),\ s, \psi(M_2), \delta_C, \psi(Q))$ where δ_C is defined as follows: if $s \in Q - M_1$, then $\delta_C(s, l) = \delta(s, l)$; if $s \in M_1$, then $\delta_C(s, l) = \psi(\delta(s, l))$; and if $s \in \psi(Q) - \psi(M_2)$, then $\delta_C(s, l) = \psi(\delta(\psi^{-1}(s), l))$; the transition function δ_C is not defined for the states of $\psi(M_2)$. Note that the marked states are all the copied states.

Remark: if rematch with p_1 is forbidden, remove all transitions $\delta_C(s, l)$ such that $\delta_C(s, l) = t$ and $t \in \psi(M_1)$.

5.3. "Error" automaton

We consider successively the cases of an error function Δ with (1) 1 substitution (2) 1 insertion (3) 1 deletion allowed for a motif R_i.

Let $A_i = (\Sigma, Q, s, F, \delta)$ be a complete DFA recognizing R_i (or $\Sigma^\star R_i$ if necessary). We use a copy function ψ as in the preceding paragraph. The NFA error automaton (an ϵ-NFA in the case of one deletion) is $B_i = (\Sigma, Q + \psi(Q), s, F + \psi(F), \delta_C)$. We have for δ_C respectively.

1. One substitution:

$$\forall s \in Q, \quad \delta_C(\psi(s), l) = \psi(\delta(s, l)), \quad \delta_C(s, l) = \{\delta(s, l)\} \cup \{\psi(\delta(s, l_i)),\ l_i \in \Sigma \backslash \{l\}\}.$$

2. One insertion:

$$\forall s \in Q, \quad \delta_C(\psi(s), l) = \psi(\delta(s, l)), \quad \delta_C(s, l) = \{\delta(s, l)\} \cup \{\psi(s)\}.$$

3. One deletion:

$$\forall s \in Q,\ \delta_C(\psi(s), l) = \psi(\delta(s, l)),\ \delta_C(s, l) = \{\delta(s, l)\},\ \delta_C(s, \epsilon) = \{\psi(\delta(s, l)), l \in \Sigma\}.$$

Use B_i as input in the algorithm corresponding to the problem to produce an ϵ-NFA. Determinize and minimize it, and translate into a generating function. When k errors are allowed, repeat the construction $k - 1$ times. Combinations of substitutions, insertions, and deletions are handled in the like.

5.4. Markov automaton

The construction from a Bernoulli automaton into a Markov automaton of order 1 has been given in [10]. The extension to a Markov automaton of order k is immediate: Let s be a state of the Bernoulli automaton, and let $W = \{w_i, w_i$ is a path of length k ending in $s\}$. Create a state s_{w_i} for each $w_i \in W$ in the Markov automaton and add the necessary transitions.

Remark that, in the case of a Markov model of order k, if the letter ρ is used for naming the transitions, a transition $\rho_{x_1,\ldots,x_k,x_{k+1}}$ from a state s means that the state has been entered by a transition of the form $\rho_{y,x_1,\ldots,x_k}$ and that the letter currently read is x_{k+1}. This translates to $\pi_{x_1,\ldots,x_k,x_{k+1}}z$ when computing generating functions.

6. Occurrences of words under constraint

Although the full power of the package `regexpcount` is best demonstrated on regular expressions, we give an application of our computations to a biological problem over words. This application is fully automatized. See Section B of the appendix for a Maple session computing the conditional expectation and the conditional standard deviation of the word considered in this section for the genome of *H. influenzae*.

6.1. Statistics of number of occurrences pairs

We make some preliminary remarks about the statistics of the number of occurrences of pairs of words. Let w_1 and w_2 be the words and $F(z, u, v) = \sum f_{n,i,j} u^i v^j z^n$ be the multivariate generating function counting the number of pairs of matches. Here, $f_{n,i,j}$ is the probability that a random sequence of size n has i matches with w_1 and j matches with w_2. Bender and Kochman [1] prove that the counts of generalized words (finite sets of words) verify a multivariate normal distribution. They also prove that this remains true when conditioning by the count of one of the generalized words. It seems reasonable to conjecture that these results also hold when considering counts of matches with regular expressions, but there are so far no proofs for this.

The "section" $[u^k z^n]f(z, u, v)$ of the generating function, up to a normalization coefficient, counts the number of matches of the word w_2 in texts of size n, knowing that exactly k matches with w_1 occurred. It is computationally expensive to compute the k-th Taylor coefficient with respect to the variable u of $F(z, u, v)$ when k is not small and practically impossible as soon as k is larger than 100. To circumvent this problem, we use the following heuristic: we shift the distribution so that the *expectation* of the number of matches with w_1 is k, and we compute the moments of matches with w_2 from the shifted distribution. The shift is obtained by giving to the parameter u the appropriate positive value α.

The methods developed in Nicodeme *et al.* [10] apply here. When α and v are real positive, $F(z, \alpha, v)$ is part of a system involving a matrix with non-negative entries. The Perron-Frobenius theory induces that there exist a dominant singularity. Suitable Cauchy integration and application of the large powers theorem of Hwang then imply that the distribution of number of matches of w_2 remains normal when the distribution is shifted.

6.2. Biological application

We consider now the Chi sequence $\chi = gxtggtgg$ of the 1830140 base pairs long genome of *H. influenzae* read in the transcription direction (x stands for any letter of the alphabet $\Sigma = \{a, c, g, t\}$). The variables $c_{i,j}$, with $i, j \in \Sigma$ count the number of bases i followed by the base j in the genome. From this, we deduce the Markov probability $\pi_{i,j} = c_{i,j}/(c_{i,a}+c_{i,c}+c_{i,g}+c_{i,t})$. With this Markov model of order 1, the expectation and standard deviation of number of occurrences of χ in the genome are 56.26 and 7.59. However, the χ sequence is highly overrepresented in the genome, and found 223 times. When looking for exceptional words in *H. influenzae*, we must condition the statistics by the observed number of occurrences of the χ sequence. We compute the constrained statistics of the word $w = tggtgggc$ as follows. Construct a marked automaton A_m for $\Sigma^*\chi$ and a marked automaton A_p for $\Sigma^* w$. Compute the product $B = \mathit{Markov}(A_m \times A_p) = \{\Sigma, Q, s, F, \delta, M, P\}$, where M and P are the set of states corresponding to matches with χ and with w. Set $F = Q$ in B to recognize all the marked texts $mark_{m,p}(\Sigma^*)$, where m and p are respectively inserted after a match with χ and w. Compute the generating function $F(z, u, v) = \sum f_{n,i,j} u^i v^j z^n$ where $f_{n,i,j}$ is the probability that a text of size n distributed as *H. influenzae* contains i occurrences of χ and j occurrences of w. We use the mean-shifting method[1] to get the constrained probability. Consider

$$\phi(n,\alpha) = \frac{[z^n]\left.\dfrac{\partial F(z,\alpha u,1)}{\partial u}\right|_{u=1}}{[z^n]F(z,\alpha,1)} \qquad \text{and} \qquad \mu_c = \frac{[z^n]\left.\dfrac{\partial F(z,\alpha_0,v)}{\partial v}\right|_{v=1}}{[z^n]F(z,\alpha_0,1)}. \tag{3}$$

Remark that $\phi(n, \alpha)$ is the shifted mean of occurrences of χ for parameter α. Solving $\phi(1830140, \alpha) = 223$ numerically provides $\alpha_0 = 3.715$. The expectation μ_c of the constrained statistics is given in Eq 3. More specifically, let

$$\kappa^n = [z^n]F(z,\alpha_0,1) \qquad \text{and} \qquad H(z) = \frac{P(z)}{Q(z)} = \left.\frac{\partial F(z/\kappa,\alpha_0,v)}{\partial v}\right|_{v=1}$$

and let ξ be the dominant singularity of $H(z)$. Note that ξ is a pole of order 2 of $H(z)$. The Taylor expansions of $P(z)$ and $Q(z)$ at $z = \xi$ are

$$P(z) = P(\xi) + (z-\xi)\frac{\partial P}{\partial z}(\xi) + o(z-\xi),$$

$$\text{and} \quad Q(z) = \frac{1}{2}(z-\xi)^2\frac{\partial^{(2)}Q}{\partial z^2}(\xi) + \frac{1}{6}(z-\xi)^3\frac{\partial^{(3)}Q}{\partial z^3}(\xi) + o(z-\xi)^3.$$

$$\text{Let } p_i = \frac{\partial^{(i)}P}{\partial z^i}(\xi) \text{ and } q_i = \frac{\partial^{(i)}Q}{\partial z^i}.$$

We get by Laurent expansion of $H(z)$ at ξ the asymptotic equivalent of the conditioned expectation,

$$\mu_c = \frac{2p_0}{q_2}(n+1)\xi^{-n-2} - \frac{2}{3}\frac{3p_1q_2 - p_0q_3}{q_2^2}\xi^{-n-1} + o(1)$$

[1]Greene and Knuth [4] give a nice account of this method. They want to compute the asymptotic value of the coefficient of z^n in the binomial distribution $(1/2 + z/2)^{3n}$. A direct use of the saddle point method is hard, the values considered being out of the domain of application of the central limit theorem. Greene and Knuth make the shift $z \to z/2$. This transforms the distribution to $(2/3 + z/3)^{3n}$, and a saddle point method applies within the domain of the central limit theorem. See also [19] for an application of this method to large deviations.

The moment of order 2 is computed similarly, and from there, the standard deviation σ_c follows. A 2.5 seconds computation gives the numerical values $\mu_c = 23.52$ and $\sigma_c = 4.85$, to be compared to the unconditioned values $\mu = 15.21$ and $\sigma = 3.9$. Theoretical [9] and numerical [10] considerations indicate that $\mu \approx \sigma^2$ and $\mu_c \approx \sigma_c^2$ when the words are not self-overlapping (or have low probability of self-overlapping matches); in this case (that covers most of the words and can be easily pre-checked by a pattern-matching algorithm) computation of μ_c alone that requires only 0.48 seconds provides a good estimation of σ_c. For the words with strong self-overlap structure, the full computation is necessary. This makes a total time shorter than 15 hours for computing all the words of size 8. As a comparison, the package `R'MES` at `http://www-bia.inra.fr/J/AB/genome/` computes the unconstrained statistics for all the words of size 8 in the Markov case in a few seconds, but fails to compute the constrained case.

7. Conclusion

We provide here a general purpose symbolic package for statistical properties of occurrences of words and regular expressions. Our method goes through the construction of automata and translations into generating function. Although determinization could lead to an explosion of the size of the automata constructed, previous work shows that this is not the case when considering exact matches and a biological application such as calibrating Prosite motifs. The package should be able to cope with matches with errors of DNA or RNA motifs in reasonable time. Complete biological applications and a push-button interface are part of future work.

Availability. The `regexpcount` package is written in Maple. It is fully documented and available at `http://algo.inria.fr/libraries/software.html`. The packages `combstruct`, `gfun`, `equivalent` and `gdev` are also available at this address.

Acknowledgments. The `regexpcount` package has been developed in close collaboration with Bruno Salvy. We thank Philippe Flajolet for precious hints and an anynomous referee for very constructive suggestions.

References

[1] Bender, E. A., Kochman, F.: The Distribution of Subword Counts is Usually Normal, *European Journal of Combinatorics*, **14**, 1993, 265–275.

[2] Berry, G., Sethi, R.: From regular expressions to deterministic automata, *Theoretical Computer Science*, **48**, 1986, 117–126.

[3] Chomsky, N., Schützenberger, M. P.: The algebraic theory of context-free languages, *Computer Programming and Formal Languages,*, 1963, 118–161, P. Braffort and D. Hirschberg, eds, North Holland.

[4] Greene, D. H., Knuth, D. E.: *Mathematics for the Analysis of Algorithms*, Birkhäuser, 1981.

[5] Hopcroft, J. E.: An $n \log n$ Algorithm for Minimizing States in a Finite Automaton, *Theory of Machines and Computation*, Kohavi ed., Academic Press, 1971.

[6] Kelley, D.: *Automata and Formal Languages, an Introduction*, Prentice Hall, 1995.

[7] Kozen, D. C.: *Automata and Computability*, Springer Verlag. Undergraduate texts in Computer Science, 1997.

[8] Nicodème, P.: Regexpcount, a symbolic package for counting problems on regular expressions and words, *Proceedings of the German Conference on Bioinformatics GCB00, Heidelberg*, 2000.

[9] Nicodème, P.: Fast Approximate Motif Statistics, *J. Comp. Biol.*, **8**(3), 2001, 235–248.

[10] Nicodème, P., Salvy, B., Flajolet, P.: Motif Statistics, *Theoretical Computer Science*, **287**(2), 2002, 593–618, Extended version of an article published in the proceedings of 7th Annual European Symposium on Algorithms ESA'99, Prague, July 1999.

[11] Pevzner, P. A., Borodovski, M. Y., Mironov, A. A.: Linguistic of nucleotide sequences: The significance of deviation from mean statistical characteristics and prediction of the frequencies of occurrence of words, *J. Biomol. Struct. Dyn*, **6**, 1989, 1013–1026.

[12] Prum, B., Rodolphe, F., de Turckheim, E.: Finding words with unexpected frequencies in deoxyribonucleic acid sequences, *J. R. statist. Soc. B*, **57**(1), 1995, 205–220.

[13] Régnier, M.: A unified approach to words statistics, *Second Annual International Conference on Computational Molecular Biology,*, ACM Press, New-York, 1998.

[14] Régnier, M.: A Unified Approach to Word Occurrences Probabilities, *Discrete Applied Mathematics*, **104**(1), 2000, 259–280, Special issue on Computational Biology.

[15] Régnier, M., Szpankowski, W.: On Pattern Frequency Occurrences in a Markovian Sequence, *Algorithmica*, **22**(4), 1998, 631–649.

[16] Reinert, G., Schbath, S.: Compound Poisson Approximations for Occurrences of Multiple Words in Markov Chains, *J. Comp. Biol.*, **5**(2), 1998, 223–253.

[17] Sewell, R. F., Durbin, R.: Method for calculation of probability of matching a bounded regular expression in a random data string, *J. Comp. Biol.*, **2**(1), 1995, 25–31.

[18] Sinha, S., Tompa, M.: A Statistical Method for Finding Transcription Factor Binding Sites, *Eigth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, 2000.

[19] Szpankowski, W.: *Average Case Analysis of Algorithms on Sequences*, Wiley-Interscience, 2001.

Appendix: maple sessions

A. Occurrences of $\Delta(abab)$

We consider here the number of occurrences with overlap and one possible error (insertion, substitution, deletion) of the pattern $abab$ in the non-uniform Bernoulli and Markov models. The asymptotic expectation and variance for these statistics are computed (see lines $\Delta(abab, 1)$, n.m.o, B_1 and M_1 of Table 1).

```
> Bwg:=[[1/4,a],[3/4,b]]:   # Bernoulli weights
> # Maple syntax necessitates a dummy letter (here rho) for Markov transitions.
> Mwg:=[[1/4,rho[a]],[3/4,rho[b]], # Markov weights
        [1/4,rho[a,a]],[3/4,rho[a,b]],[1/2,rho[b,a]],[1/2,rho[b,b]]]:

> GR:={abab=Prod(a,b,a,b), a=Atom, b=Atom}: # grammar for word abab
> t1:=time():
> marked_automat:=regexpcount[regexptomatchesgram](GR,ABAB,
                 [[abab,m,'overlap','error'[1,{'subst','ins','del'}]]]);
> time()-t1;
```

$$\begin{aligned}
&\mathit{marked\_automat} := \{a = \mathit{Atom},\ b = \mathit{Atom},\\
&\quad \mathit{w6} = \mathrm{Union}(E,\ \mathrm{Prod}(b,\ \mathit{w5}),\ \mathrm{Prod}(a,\ \mathit{w1})),\\
&\quad \mathit{w1} = \mathrm{Union}(E,\ \mathrm{Prod}(a,\ \mathit{w1}),\ \mathrm{Prod}(b,\ m,\ \mathit{w2})),\ m = E,\\
&\quad \mathit{w5} = \mathrm{Union}(E,\ \mathrm{Prod}(b,\ \mathit{w5}),\ \mathrm{Prod}(a,\ \mathit{w1})),\\
&\quad \mathit{w7} = \mathrm{Union}(E,\ \mathrm{Prod}(b,\ \mathit{w3}),\ \mathrm{Prod}(a,\ \mathit{w1})),\\
&\quad \mathit{w3} = \mathrm{Union}(E,\ \mathrm{Prod}(a,\ m,\ \mathit{w8}),\ \mathrm{Prod}(b,\ m,\ \mathit{w4})),\\
&\quad \mathit{ABAB} = \mathrm{Union}(E,\ \mathrm{Prod}(a,\ \mathit{w7}),\ \mathrm{Prod}(b,\ \mathit{w5})),\\
&\quad \mathit{w2} = \mathrm{Union}(E,\ \mathrm{Prod}(a,\ m,\ \mathit{w8}),\ \mathrm{Prod}(b,\ m,\ \mathit{w4})),\\
&\quad \mathit{w4} = \mathrm{Union}(E,\ \mathrm{Prod}(b,\ m,\ \mathit{w6}),\ \mathrm{Prod}(a,\ \mathit{w1})),\\
&\quad \mathit{w8} = \mathrm{Union}(E,\ \mathrm{Prod}(a,\ m,\ \mathit{w9}),\ \mathrm{Prod}(b,\ m,\ \mathit{w2})),\\
&\quad \mathit{w9} = \mathrm{Union}(E,\ \mathrm{Prod}(a,\ \mathit{w1}),\ \mathrm{Prod}(b,\ m,\ \mathit{w2}))\}
\end{aligned}$$

$$.167$$

```
> # Input: a marked automaton for the number of matches problem
  # getvals prints the asymptotic expectation and variance of number of matches
> getvals:=proc(auto,init,weight)
     local gfeq, wauto, i, gfzu, gfz, var, eq;
     wauto:=regexpcount[gramweight](auto):
     gfeq:=combstruct[gfeqns](wauto,unlabeled,z,weight);
     # gfzu is the bivariate generating function for number of matches
     gfzu:=subs(solve({op(gfeq)},{seq(op(1,i),i=gfeq)}),init(z,u));
     printf("gfzu=%a\n",gfzu);
     gfz[1]:=subs(u=1,diff(gfzu,u));            # g.f for expect
     gfz[2]:=subs(u=1,diff(u*diff(gfzu,u),u)); # g.f. for second moment
     for i to 2 do eq[i]:=equivalent(gfz[i],z,n,6); od;
     print("expectation", eq[1]);
     var:=eq[2]-eq[1]^2;
     print("variance",gdev(var,n=infinity,10));
  end:

> t2:=time(): getvals(marked_automat,ABAB,[op(Bwg),[u,m]]); time()-t2;
```

$$\mathrm{gfzu} = -\frac{\begin{array}{c}1024 - 120z^4u^3 - 144z^3u^2 - 48z^3u^3 - 192z^2u^2 + 276z^4u^2 + 192z^2\\ +9z^5u^3 - 18z^5u^2 - 192z^3 - 192z^4u + 384z^3u + 36z^4 + 9z^5u\end{array}}{8(128z - 128 + 9z^4u^3 - 6z^3u^2 + 6z^3u^3 + 24z^2u^2 - 9z^4u^2 - 24z^2)}$$

$$\text{“expectation”},\ \frac{63}{128}\,n - \frac{141}{128} + \mathrm{O}(n^{(-\infty)})$$

$$\text{“variance”},\ \frac{6957}{16384}\,n - \frac{12663}{16384} + \mathrm{O}(n^{(-\infty)})$$

$$.519$$

```
> t3:=time():
> markov_marked_automat:=regexpcount[grammarkov](marked_automat,ABAB,1,rho,[m]);
> time()-t3;
```

$$\begin{aligned}&\mathit{markov\_marked\_automat} := \{\\
&\quad m = E, \mathit{w7}_a = \mathrm{Union}(\mathrm{Prod}(\rho_{a,\,a},\ \mathit{w1}_a),\ E,\ \mathrm{Prod}(\rho_{a,\,b},\ \mathit{w3}_b)),\\
&\quad \mathit{ABAB} = \mathrm{Union}(E,\ \mathrm{Prod}(\rho_a,\ \mathit{w7}_a),\ \mathrm{Prod}(\rho_b,\ \mathit{w5}_b)),\\
&\quad \mathit{w1}_a = \mathrm{Union}(\mathrm{Prod}(\rho_{a,\,a},\ \mathit{w1}_a),\ \mathrm{Prod}(\rho_{a,\,b},\ m,\ \mathit{w2}_b),\ E),\\
&\quad \mathit{w9}_a = \mathrm{Union}(\mathrm{Prod}(\rho_{a,\,a},\ \mathit{w1}_a),\ \mathrm{Prod}(\rho_{a,\,b},\ m,\ \mathit{w2}_b),\ E),\\
&\quad \mathit{w4}_b = \mathrm{Union}(E,\ \mathrm{Prod}(\rho_{b,\,b},\ m,\ \mathit{w6}_b),\ \mathrm{Prod}(\rho_{b,\,a},\ \mathit{w1}_a)),\\
&\quad \mathit{w6}_b = \mathrm{Union}(E,\ \mathrm{Prod}(\rho_{b,\,b},\ \mathit{w5}_b),\ \mathrm{Prod}(\rho_{b,\,a},\ \mathit{w1}_a)),\\
&\quad \mathit{w2}_b = \mathrm{Union}(E,\ \mathrm{Prod}(\rho_{b,\,a},\ m,\ \mathit{w8}_a),\ \mathrm{Prod}(\rho_{b,\,b},\ m,\ \mathit{w4}_b)),\\
&\quad \mathit{w3}_b = \mathrm{Union}(E,\ \mathrm{Prod}(\rho_{b,\,a},\ m,\ \mathit{w8}_a),\ \mathrm{Prod}(\rho_{b,\,b},\ m,\ \mathit{w4}_b)),\\
&\quad \mathit{w8}_a = \mathrm{Union}(\mathrm{Prod}(\rho_{a,\,b},\ m,\ \mathit{w2}_b),\ E,\ \mathrm{Prod}(\rho_{a,\,a},\ m,\ \mathit{w9}_a)),\\
&\quad \mathit{w5}_b = \mathrm{Union}(E,\ \mathrm{Prod}(\rho_{b,\,b},\ \mathit{w5}_b),\ \mathrm{Prod}(\rho_{b,\,a},\ \mathit{w1}_a)),\\
&\quad \rho_a = \mathit{Atom},\ \rho_b = \mathit{Atom},\ \rho_{a,\,b} = \mathit{Atom},\ \rho_{b,\,a} = \mathit{Atom},\\
&\quad \rho_{b,\,b} = \mathit{Atom},\ \rho_{a,\,a} = \mathit{Atom}\}\end{aligned}$$

$$.079$$

```
> t4:=time(): getvals(markov_marked_automat,ABAB,[op(Mwg),[u,m]]); time()-t4;
```

$$\mathrm{gfzu} = \frac{\begin{array}{c}-32z - 48z^2 + 12z^3u^3 + 18z^4u^3 - 128 + 36z^3u^2\\ +48z^2u^2 - 30z^4u^2 - 66z^3u + 18z^3 + 15z^4u - 3z^4\end{array}}{2(48z - 8z^2 + 6z^3u^3 + 3z^4u^3 - 64 - 6z^3u^2 + 24z^2u^2 - 3z^4u^2)}$$

$$\text{“expectation”},\ \frac{57}{80}\,n - \frac{1311}{800} - \frac{27}{25}\,(-4)^{(-n)} + \mathrm{O}(\frac{4^{(-n)}}{n^8})$$

$$\text{“variance”},\ \frac{7323}{32000}\,n - \frac{17601}{128000} + \mathrm{O}(n\,4^{(-n)})$$

$$.785$$

B. Occurrences of words on DNA under constraint

We compute the expectation and standard deviation of the word $tggtgggc$ for *H. influenzae* (Markov model of order 1). Constraint: the observed number of occurrences of the motif Chi $gxtggtgg$ is 223. See Section 6.

```
> with(regexpcount): with(combstruct): with(gfun):
> nuc:=[a,g,c,t]: # nucleotides of DNA
> # counts of binucleotides (Markov of order 1)
> cnt[a,a]:= 213715: cnt[a,g]:=  91504: cnt[a,c]:= 86381: cnt[a,t]:= 166838:
> cnt[g,a]:=  97395: cnt[g,g]:=  71409: cnt[g,c]:= 95530: cnt[g,t]:=  97342:
> cnt[c,a]:= 115361: cnt[c,g]:=  72522: cnt[c,c]:= 63061: cnt[c,t]:=  85502:
> cnt[t,a]:= 131957: cnt[t,g]:= 126249: cnt[t,c]:= 91471: cnt[t,t]:= 223670:
> for i in nuc do
   tot[i]:=add(cnt[i,j],j=nuc): for j in nuc do wm[i,j]:=cnt[i,j]/tot[i] od:
  od:
```

```
> total:=add(tot[i],i=nuc):
> Hgen:={chi=Prod(g,x,t,g,g,t,g,g),x=Union(a,c,g,t),a=Atom,c=Atom,g=Atom,t=Atom}:
> word:={W=Prod(t,g,g,t,g,g,g,c)}:
> adnw:= [[tot[a]/total,k[a]],[tot[c]/total,k[c]],
        [tot[g]/total,k[g]],[tot[t]/total,k[t]],
        seq(seq([wm[i,j],k[i,j]],i=nuc),j=nuc),[u,m],[v,p]]:
> berw:=[seq([1/4,k[i]],i=nuc),seq(seq([1/4,k[i,j]],i=nuc),j=nuc),[u,m],[v,p]]:
> mkeq:=proc(auto::set,adw)     local WA, i, listat, eq;
    WA:=gramweight(auto);
    WA:= WA minus  select(type,WA,anything={identical('Epsilon')}):
    eq:={seq(eval(subs(Prod=`*`,Union=`+`,Epsilon=1,Atom=z,m=u,p=v,i)),i=op(WA))}:
    eq:=subs(seq(i[2]=i[1],i=adw),eq);
    eq;
  end:

> # n is given numerically to the function expstd
  # vu: numerical value of u; sc: sum of coefficients for u=vu and length n
  # xi is a pole of order 2 for EXGF and of order 3 for M2GF
> expstd:=proc(MVGFUV,vu,sc,n)
   local kappa, GF, xi, EXGF, M2GF,H1,H2, DN3, p0, p1, p2, q2, q3, q4, q5;
   if vu=1 then kappa:=1
   else kappa:=sc^(1/n) fi;
   GF:=subs(u=vu,z=z/kappa,MVGFUV);
   EXGF:=subs(v=1,diff(GF,v));
   xi:=op(1,[fsolve(denom(EXGF),z,0.8..1.2)]);
   p0:=subs(z=xi,numer(EXGF));
   p1:=subs(z=xi,diff(numer(EXGF),z));
   q2:=subs(z=xi,diff(denom(EXGF),z,z));
   q3:=subs(z=xi,diff(denom(EXGF),z,z,z));
   H1:=2*p0*(n+1)/q2/xi^(n+2)-2*(p1*q2-p0*q3/3)/q2^2/xi^(n+1);
   if vu=1 then print("Expectation",H1);
   else         print("Conditioned Expectation",H1); fi;
   M2GF:=subs(v=1,diff(v*diff(GF,v),v));
   p0:=subs(z=xi,numer(M2GF));
   p1:=subs(z=xi,diff(numer(M2GF),z));
   p2:=subs(z=xi,diff(numer(M2GF),z,z));
   q3:=subs(z=xi,diff(denom(M2GF),z,z,z));
   q4:=subs(z=xi,diff(denom(M2GF),z,z,z,z));
   q5:=subs(z=xi,diff(denom(M2GF),z,z,z,z,z));
   H2:=-6*p0*(n^2/2+3*n/2+1)/q3/xi^(n+3)+6*(p1*q3-p0*q4/4)*(n+1)/q3^2/xi^(n+2)
       -6*(p2*q3^2/2-p0*q3*q5/20-(4*p1*q3-p0*q4)*q4/16)/q3^3/xi^(n+1);
   if vu=1 then print("Standard Deviation", evalf(sqrt(H2-H1^2)));
   else         print("Conditioned Standard Deviation", evalf(sqrt(H2-H1^2)));
   fi
  end:


> # numerical evaluation of the Taylor coefficient of order n for alpha=v_alpha
> evalcoeff:=proc(f,v_alpha,n,k)     local sing, H;
   sing:=op(1,[fsolve(subs(alpha=v_alpha,denom(f)),z,.8..1.2)]);
   if k=1 then
       H:=-subs(alpha=v_alpha,z=sing,numer(f)/diff(denom(f),z))*sing^(-n-1)
    else # k=2
        H:=(n+1)*sing^(-n-2)*subs(alpha=v_alpha,z=sing, numer(f)*2/diff(denom(f),z,z))
    fi;
    H
> end:
> T:=time():  auto:=regexptomatchesgram(Hgen union word,CHI,
                                        [[chi,m,'overlap'],[W,p,'overlap']]):
> markovauto:=grammarkov(auto,CHI,1,'k'):   tauto:=time()-T; T:=time():
```

$$tauto := .315$$

```
> neq:=mkeq(markovauto,adnw):
> mvgfuv:=subs(solve({op(neq)},{seq(op(1,i),i=neq)}),CHI):  tsolve:=time()-T;T:=time():
```

$$tsolve := 1.910$$

```
> # when numerically solving phi with fsolve, only one argument is allowed
> # for phi; therefore L, Chi_obs, numphi and denphi are defined as global variables

> L:=1830140: Chi_obs:=223:

> mvgfu:=subs(v=1,mvgfuv):
> numphi:=subs(u=1,diff(mvgfu,u)): denphi:=subs(u=1,mvgfu):

> phi:=proc(v_alpha)     local nm, dn;
     nm:=evalcoeff(numphi,v_alpha,L,2); dn:=evalcoeff(denphi,v_alpha,L,1);
     nm/dn - Chi_obs
> end:

> Digits:=30: T:=time():
  alpha0:=fsolve(phi,2..5); time()-T;
```

$$\alpha_0 := 3.71500260619846885584814600795$$

$$.964$$

sumofcoeff $= \kappa^n = [z^n]F(z, \alpha_0, 1)$;
$[z^n]F(z/\kappa, \alpha_0, v)$ is a probability generating function; coefficients sum up to 1

```
> sumofcoeff:=evalcoeff(denphi,alpha0,L,1);
```

$$sumofcoeff := .330376940523924778854588001899\, 10^{69}$$

```
> Digits:=10: adnw;
```

$$\left[\left[\frac{186146}{609969}, k_a\right], \left[\frac{336446}{1829907}, k_c\right], \left[\frac{361676}{1829907}, k_g\right], \left[\frac{573347}{1829907}, k_t\right], \left[\frac{213715}{558438}, k_{a,a}\right], \left[\frac{97395}{361676}, k_{g,a}\right],\right.$$
$$\left[\frac{115361}{336446}, k_{c,a}\right], \left[\frac{131957}{573347}, k_{t,a}\right], \left[\frac{45752}{279219}, k_{a,g}\right], \left[\frac{71409}{361676}, k_{g,g}\right], \left[\frac{36261}{168223}, k_{c,g}\right],$$
$$\left[\frac{126249}{573347}, k_{t,g}\right], \left[\frac{86381}{558438}, k_{a,c}\right], \left[\frac{47765}{180838}, k_{g,c}\right], \left[\frac{63061}{336446}, k_{c,c}\right], \left[\frac{91471}{573347}, k_{t,c}\right],$$
$$\left.\left[\frac{83419}{279219}, k_{a,t}\right], \left[\frac{6953}{25834}, k_{g,t}\right], \left[\frac{42751}{168223}, k_{c,t}\right], \left[\frac{223670}{573347}, k_{t,t}\right], [u, m], [v, p]\right]$$

```
> Digits:=30: expstd(mvgfuv,alpha0,sumofcoeff,1830146);
```

“Conditioned Expectation”, 23.5181552641957932394301363950
“Conditioned Standard Deviation”, 4.84912348849073588546892002761

```
> tstats:=time()-T;  ttotal:=tauto+tsolve+tstats;
```

$$tstats := .227$$
$$ttotal := 2.452$$

```
> std(mvgfuv,1,1,1830146);
```

“Expectation”, 15.2124052543474542605033150726
“Standard Deviation”, 3.90005838002946105422993294350

```
> # for sake of comparison, multivariate generating function in the uniform
> # Bernoulli model
> bern_mvgfuv:=subs(gfsolve(auto,unlabeled,z,[[u,m],[v,p]]),z=z/4,CHI(z,u,v));
```

$$\text{bern-mvgfuv} = \frac{\begin{array}{c}1+\frac{1}{4194304}z^{11}v-\frac{1}{4194304}z^{11}uv+\frac{1}{65536}z^{8}v-\frac{1}{65536}z^{8}uv-\frac{1}{4194304}z^{11}+\frac{1}{4194304}z^{11}u\\ -\frac{1}{65536}z^{8}+\frac{1}{65536}z^{8}u+\frac{1}{4096}z^{7}-\frac{1}{4096}z^{7}u+\frac{1}{4096}z^{6}-\frac{1}{4096}z^{6}u+\frac{1}{64}z^{3}-\frac{1}{64}z^{3}u\end{array}}{\begin{array}{c}1-z+\frac{1}{64}z^{4}u-\frac{1}{65536}z^{9}v-\frac{1}{262144}z^{10}+\frac{1}{262144}z^{10}v-\frac{1}{65536}z^{9}u-\frac{1}{64}z^{3}u+\frac{1}{64}z^{3}\\ -\frac{1}{65536}z^{8}uv-\frac{1}{262144}z^{10}uv-\frac{3}{16384}z^{8}+\frac{1}{65536}z^{9}-\frac{1}{64}z^{4}-\frac{1}{4096}z^{6}u+\frac{1}{65536}z^{9}uv\\ +\frac{13}{65536}z^{8}u-\frac{1}{268435456}z^{15}v+\frac{1}{4194304}z^{12}-\frac{1}{268435456}z^{15}u+\frac{1}{268435456}z^{15}uv\\ +\frac{1}{16777216}z^{13}v+\frac{1}{268435456}z^{15}-\frac{1}{4194304}z^{12}u+\frac{1}{4194304}z^{12}uv-\frac{1}{16777216}z^{13}-\frac{1}{4194304}z^{12}v\\ +\frac{1}{262144}z^{10}u-\frac{1}{16777216}z^{13}uv+\frac{1}{16777216}z^{13}u+\frac{1}{4096}z^{6}\end{array}}$$

```
> # First terms of the Taylor expansion in z
> expand(series(bern_mvgfuv,z=0,14));
```

$$\begin{aligned}&1+z+z^2+z^3+z^4+z^5+z^6+z^7+\left(\frac{65531}{65536}+\frac{1}{16384}u+\frac{1}{65536}v\right)z^8\\ &+\left(\frac{32763}{32768}+\frac{1}{8192}u+\frac{1}{32768}v\right)z^9+\left(\frac{11}{262144}v+\frac{1}{262144}uv+\frac{262085}{262144}+\frac{47}{262144}u\right)z^{10}\\ &+\left(\frac{1048265}{1048576}+\frac{123}{524288}u+\frac{1}{1048576}u^2+\frac{7}{131072}v+\frac{1}{131072}uv\right)z^{11}\\ &+\left(\frac{19}{65536}u+\frac{524095}{524288}+\frac{1}{524288}u^2+\frac{3}{262144}uv+\frac{17}{262144}v\right)z^{12}\\ &+\left(\frac{1048115}{1048576}+\frac{5793}{16777216}u+\frac{5}{65536}v+\frac{255}{16777216}uv+\frac{47}{16777216}u^2+\frac{1}{16777216}u^2v\right)z^{13}+O(z^{14})\end{aligned}$$