Variable Length Markov Chains: Methodology, Computing and Software

Article *in* Journal of Computational and Graphical Statistics · January 2012

DOI: 10.1198/1061860043524

CITATIONS
69

READS
962

2 authors:

Martin Mächler
ETH Zurich
65 PUBLICATIONS **127,729** CITATIONS
SEE PROFILE

Peter Bühlmann
ETH Zurich
395 PUBLICATIONS **38,210** CITATIONS
SEE PROFILE

Variable Length Markov Chains: Methodology, Computing, and Software

Martin MÄCHLER and Peter BÜHLMANN

This article presents a tutorial and new, publicly available computational tools for variable length Markov chains (VLMC). VLMCs are Markov chains with the additional attractive structure that their memories depend on a variable number of lagged values, depending on what the actual past (the lagged values) looks like. They build a very flexible class of tree-structured models for categorical time series. Fitting VLMCs from data is a nontrivial computational task. We provide an efficient implementation of the so-called context algorithm which requires only $O(n \log(n))$ operations. The implementation, which is publicly available, includes additional important new features and options: diagnostics, goodness of fit, simulation and bootstrap, residuals, and tuning the context algorithm. Our tutorial is presented with a version in R which is available from the Comprehensive R Archive Network (CRAN). The exposition is self-contained, gives rigorous and partly new mathematical descriptions, and is illustrated by analyzing a DNA sequence from the Epstein-Barr virus.

Key Words: AIC; Bootstrap; Categorical time series; Classification; Comprehensive R archive network; Context algorithm; Diagnostics; DNA sequence; Simulation.

1. INTRODUCTION

We present a tutorial and new, publicly available computational tools for so-called variable length Markov chains (VLMC). They build a very flexible class of tree-structured models for stationary categorical time series. Examples of such time series include DNA sequence data or binary sequences, for example, from information or computing technology. Throughout this article, we will demonstrate our new computational tools on a DNA sequence of the BNRF1 gene from the Epstein-Barr virus (see Figure 1): the data can be downloaded from http://www-stat.ucdavis.edu/~shumway/tsa.html#3 and was described by Shumway and Stoffer (2000).

The origin of VLMCs is in information theory (Rissanen 1983); related models and

Martin Mächler is Senior Scientist, and Peter Bühlmann is Professor, Seminar für Statistik, LEO C16, Eidgenössische Technische Hochschule (ETH) CH-8092 Zürich, Switzerland (E-mail: maechler@stat.math.ethz.ch *and* buhlmann@stat.math.ethz.ch).

Journal of Computational and Graphical Statistics, Volume 13, Number 2, Pages 1–21
DOI: 10.1198/1061860043524

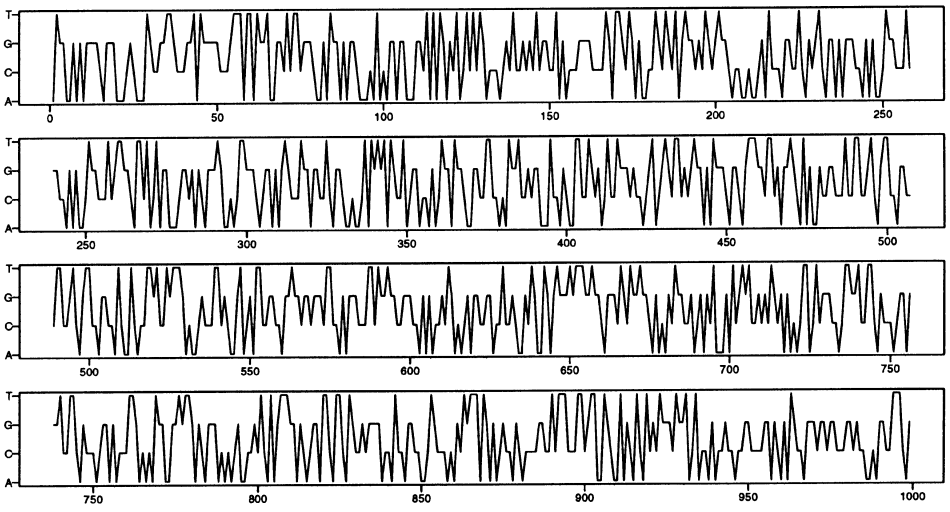

Figure 1. The first 999 (out of 3,954) bases of the BNRF1 gene section of the EB virus.

methods, from information theory and machine learning, are context tree weighting (Willems, Shtarkov, and Tjalkens 1996) or probabilistic suffix trees (Ron, Singer, and Tishby 1996). VLMCs were recently discussed from a more statistical view by Bühlmann and Wyner (1999). The aim here is to popularize the powerful VLMC model. Because computing and fitting such models is nontrivial, we provide a computational tutorial and publicly available, open-source software, covering the state of the art and new aspects of VLMC modeling. It is designed as a platform for further (computational) advances in this field.

Why should we use VLMCs ? One of the most general models for a stationary categorical process $(X_t)_{t\in\mathbb{Z}}$, taking values in a finite categorical space $\mathcal{X}$, is a full Markov chain of possibly high, but finite order. The only implicit assumption aside from stationarity is the finite memory of the process. We always refer to a stationary full Markov chain of order p, whenever the transition mechanism has no specific structure; that is, the state space is the entire $\mathcal{X}^p$. A nice model probabilistically, such full Markov chains, is not very appropriate from the statistical point of view. Let us illustrate two main problems. To be more specific, we momentarily take for illustrative purposes $\mathcal{X} = \{A, C, G, T\}$ for the letters of a DNA sequence as in the example in Figure 1 (but all the problems discussed below apply to any finite space $\mathcal{X}$).

Problem 1: The class of all finite order $\mathcal{X}$-valued full Markov chains is not structurally rich: there are not many members in the class. This structural poverty particularly implies that any kind of parsimonious representation of the state space is not possible. The table below additionally demonstrates such structural poverty in terms of the dimension of full Markov chain models (the number of free parameters) as a function of their orders p: dimension $= (\text{card}(\mathcal{X}) - 1)\text{card}(\mathcal{X})^p$ with cardinality $\text{card}(\mathcal{X}) = 4$. (The formula can be derived as follows: $\text{card}(\mathcal{X})^p$ possible states, and for every state $\text{card}(\mathcal{X}) - 1$ free parameters

for the transition probabilities which sum to one.)

order p	0	1	2	3	4	5	10
dimension	3	12	48	192	768	3072	$\approx 3.1 \cdot 10^6$

There are no models "in between," for example, it is impossible to fit a model with say 72 parameters.

Problem 2: As seen from the table above, the curse of dimensionality is particularly damaging when fitting high-order models, since the dimensionality increases exponentially with the order p. This then leads to highly variable estimates.

VLMCs address both problems and provide a natural and elegant way to avoid (some of) the difficulties mentioned. The idea is to allow the memory of the Markov chain to have a variable length, depending on the observed past values: hence the name variable length Markov chain.

Fitting a VLMC from data involves estimation of the structure of the variable length memory. As we will see in Section 2, the latter is a problem about estimating a tree. The so-called context algorithm is the key element for such tree estimation and also the algorithm then involves a tree structure. Thus, it can be implemented very efficiently. With an efficient implementation of the context algorithm at hand, we demonstrate and discuss goodness of fit and diagnostics, model selection and simulation from VLMCs , including a powerful bootstrap technique.

Our computational tutorial is presented with the implementation in the statistical software package R, publicly available at http://www.R-project.org/. But the basic routines are written in C and can be imported to other statistical software.

2. VARIABLE LENGTH MARKOV CHAINS

A variable length Markov chain is a potentially high-order Markov chain, taking values in a finite categorical space $\mathcal{X}$, with a natural parsimonious structure for the transition probabilities. In the sequel, capital letters X are usually used for random variables and small letters x for deterministic values.

Example A With DNA. Consider a time series $X_1, \ldots, X_n$ from a DNA sequence with $X_t \in \{A, C, G, T\}$. A model for such data could be a stationary Markov chain of order 2 with the following special structure for the time-homogeneous transition probabilities

$$\mathbb{P}[X_t = x_t \mid X_{t-1} = x_{t-1},\ X_{t-2} = x_{t-2}, \ldots]$$
$$= \begin{cases} \mathbb{P}[X_t = x_t \mid X_{t-1} = A], & \text{if } \ x_{t-1} = A \\ \mathbb{P}[X_t = x_t \mid X_{t-1} = C], & \text{if } \ x_{t-1} = C \\ \mathbb{P}[X_t = x_t \mid X_{t-1} = G], & \text{if } \ x_{t-1} = G \\ \mathbb{P}[X_t = x_t \mid X_{t-1} = T, X_{t-2} \in \{A, C, G\}], & \text{if } \ x_{t-1} = T,\ x_{t-2} \in \{A, C, G\} \\ \mathbb{P}[X_t = x_t \mid X_{t-1} = T, X_{t-2} = T], & \text{if } \ x_{t-1} = T,\ x_{t-2} = T. \end{cases}$$

It means that the transition probabilities are determined by looking back a variable number of lagged values, depending how such a lagged value history looks like. As mentioned, this model is a Markov chain of order 2 but with a special structure for the transition mechanism, described by the probabilities $\mathbb{P}[X_t = x_t \mid X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}]$:

x_{t-2}^{t-1}	$A*$	$C*$	$G*$	$T[ACG]$	TT
$\mathbb{P}[X_t \mid \cdot\,]$	π_1	π_2	π_3	π_4	π_5

(2.1)

where π_j are 4×1 vectors of transition probabilities (whose components sum up to one), and letter combinations denote the states: for example TT for $x_{t-1} = T, x_{t-2} = T$, $T[ACG]$ for $x_{t-1} = T, x_{t-2} \in \{A, C, G\}$, where "$[\cdots]$" is the regular expression notation for sets, and $A*$ denotes any pair starting with the letter A such as AC. Note that an ordinary full Markov chain of order 2 would have 16 different transition probability vectors, whereas here we have only 5 which shows the sparseness of the model.

This example is a special case of a variable length Markov chain model. Denote by $x_i^j = (x_j, x_{j-1}, \ldots, x_i)$, $i \le j$, a vector whose components are written in reverse order.

Definition 1. Variable length memory. *Let $(X_t)_{t\in\mathbb{Z}}$ be a stationary process with values $X_t \in \mathcal{X}$. Denote by $c_{\text{pre}} : \mathcal{X}^\infty \to \bigcup_{j=0}^{\infty} \mathcal{X}^j \cup \mathcal{X}^\infty$ ($\mathcal{X}^0 = \emptyset$) a preliminary function which maps an infinite sequence (the* infinite *past) to a possibly shorter string (the* relevant *past):*

$$\begin{aligned} c_{\text{pre}} \quad &: \quad x_{-\infty}^0 \mapsto x_{-\ell+1}^0, \ \textit{ where } \ell \textit{ is defined by} \\ \ell \quad &= \quad \ell(x_{-\infty}^0) = \min\{k; \mathbb{P}[X_1 = x_1 \mid X_{-\infty}^0 = x_{-\infty}^0] \\ &\qquad = \mathbb{P}[X_1 = x_1 \mid X_{-k+1}^0 = x_{-k+1}^0]\ \forall x_1 \in \mathcal{X}\}, \\ &\qquad \textit{and } \ell \equiv 0 \textit{ corresponds to independence}. \end{aligned}$$

Thus, the function $c_{\text{pre}}(\cdot)$ carries the information on which vectors are relevant from the infinite past of the process: $c_{\text{pre}}(\cdot)$ is called the preliminary context function and for any $t \in \mathbb{Z}$, $c_{\text{pre}}(x_{-\infty}^{t-1})$ is called the context (the relevant past) of the process at time t. Let $0 \le p \le \infty$ be the smallest integer such that

$$\text{card}(c_{\text{pre}}(x_{-\infty}^0)) = \ell(x_{-\infty}^0) \le p \quad \textit{for all} \quad x_{-\infty}^0 \in \mathcal{X}^\infty.$$

The number p is called the order of the preliminary context function $c_{\text{pre}}(\cdot)$, and if $p < \infty$, $(X_t)_{t\in\mathbb{Z}}$ is called a stationary variable length Markov chain (VLMC) of order p.

Due to stationarity of $(X_t)_{t\in\mathbb{Z}}$, transition probabilities are homogeneous in time and the restriction to indices $0, -1, \ldots$ in the definitions above is without loss of generality. Clearly, a VLMC of order p is a Markov chain of order p, with the additional structure of having a *memory of variable length ℓ*. Such a structure implies that some of the transition probabilities are the same for various states of (the embedding) Markov chain. If the preliminary context function $c_{\text{pre}}(\cdot)$ of order p is the full projection $x_{-\infty}^0 \mapsto x_{-p+1}^0$ for all $x_{-\infty}^0$, the VLMC is a full Markov chain of order p.

Example A (continued). The preliminary context function of the Markov chain of order 2 with transition probabilities in Equation (2.1) is

$$c_{\text{pre}}(x_{-\infty}^{0}) = \begin{cases} A, & \text{if} \quad x_0 = A,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ C, & \text{if} \quad x_0 = C,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ G, & \text{if} \quad x_0 = G,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ TA, & \text{if} \quad x_0 = T,\ x_{-1} = A,\ x_{-\infty}^{-2} \quad \text{arbitrary} \\ TC, & \text{if} \quad x_0 = T,\ x_{-1} = C,\ x_{-\infty}^{-2} \quad \text{arbitrary} \\ TG, & \text{if} \quad x_0 = T,\ x_{-1} = G,\ x_{-\infty}^{-2} \quad \text{arbitrary} \\ TT, & \text{if} \quad x_0 = T,\ x_{-1} = T,\ x_{-\infty}^{-2} \quad \text{arbitrary.} \end{cases}$$

The transition probabilities in Equation (2.1) for Example A suggest the *final* form of a context function $c(\cdot)$ which allows to lump *some* of the values of $c_{\text{pre}}(\cdot)$ whose second last symbols are the same.

Example A (continued). The (final form) context function of the Markov chain of order 2 with transition probabilities in Equation (2.1) is

$$c(x_{-\infty}^{0}) = \begin{cases} A, & \text{if} \quad x_0 = A,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ C, & \text{if} \quad x_0 = C,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ G, & \text{if} \quad x_0 = G,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ T[ACG], & \text{if} \quad x_0 = T,\ x_{-1} \in \{A, C, G\},\ x_{-\infty}^{-2} \quad \text{arbitrary} \\ TT, & \text{if} \quad x_0 = T,\ x_{-1} = T, \quad x_{-\infty}^{-2} \quad \text{arbitrary.} \end{cases}$$

The context function $c(\cdot)$ adds additional structure to the model and from now on we are exclusively interested in such final form context functions $c(\cdot)$ and their associated VLMC's. A VLMC has an important representation as a graphical tree model; see Figure 2.

Definition 2. Context tree. *Let $c(\cdot)$ be a context function of a stationary VLMC. The context tree τ is defined as*

$$\tau = \tau_c = \{w; w = c(x_{-\infty}^{0}),\ x_{-\infty}^{0} \in \mathcal{X}^{\infty}\}.$$

The context function $c(\cdot)$ can be reconstructed from the context tree τ_c which is nothing else than the minimal state space of the underlying VLMC. Note that the context tree does not need to be complete, for example, it can have less than $\text{card}(\mathcal{X})$ terminal offsprings from a parental node, whenever $c(\cdot)$ lumps together some values of $c_{\text{pre}}(\cdot)$.

Example A (continued). The context function $c(\cdot)$ above can be represented by the tree τ_c on the left side of Figure 2. It describes the minimal state space $\{A, C, G, T[ACG], TT\}$ (read top down).

Example B. $\mathcal{X} = \{0, 1\}$, order $p = 3$. The function

$$c(x_{-\infty}^{0}) = \begin{cases} 0, & \text{if} \quad x_0 = 0,\ x_{-\infty}^{-1} \quad \text{arbitrary} \\ 1,0,0, & \text{if} \quad x_0 = 1, x_{-1} = 0, x_{-2} = 0,\ x_{-\infty}^{-3} \quad \text{arbitrary} \\ 1,0,1, & \text{if} \quad x_0 = 1, x_{-1} = 0, x_{-2} = 1,\ x_{-\infty}^{-3} \quad \text{arbitrary} \\ 1,1, & \text{if} \quad x_0 = 1, x_{-1} = 1,\ x_{-\infty}^{-2} \quad \text{arbitrary} \end{cases}$$

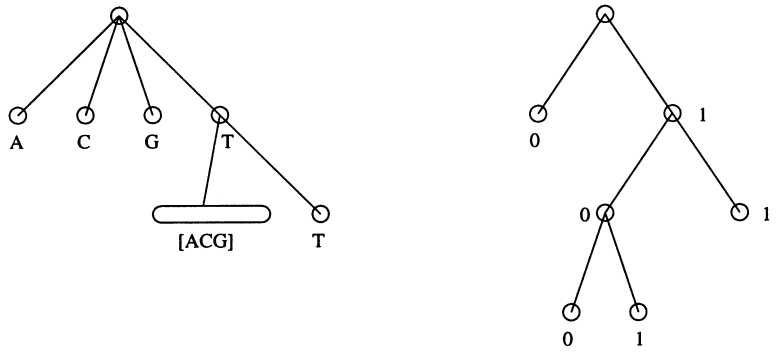

Figure 2. Tree representations of the variable length memories in Examples A (continued) and B.

can be represented by the tree τ_c on the right in Figure 2. The state space is given by the terminal nodes $\{0, 100, 101, 11\}$ of the tree (read top down).

An alternative representation of Example A's context tree (left side of Figure 2) is given by the incomplete tree without the round-edged terminal node `[ACG]` (Figure 3). In this incomplete tree, the internal node T then represents the state $T[ACG]$ of the underlying VLMC. This more economical representation will also be used in our algorithm for estimating a context tree.

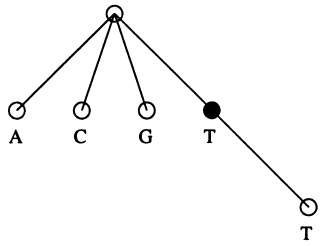

Figure 3. Alternative tree representation for Example A, where the filled circle "T" represents the state $T[ACG]$.

Particularly when thinking in terms of context trees, it becomes clear that VLMCs build a very flexible class ranging from full Markov chains (full tree) to parsimoniously parameterized transition models (sparse tree). With VLMCs , Problem 1 from Section 1 does not exist, and the models provide a way to deal intelligently with the curse of dimensionality (see Problem 2 in Section 1). The nontrivial issue of fitting a right-sized context tree from data is described in the next section.

3. FITTING VLMCs

Fitting a VLMC can be done with a version of the tree structured context algorithm (Rissanen 1983) which is detailed in Section 6.1. The variable length memory is usually represented with an estimated context tree. Thus, we are fitting tree-structured models

where every terminal (and some internal) node represents a state in the Markov chain and is equipped with corresponding transition probabilities. Our context algorithm grows a large tree and prunes it back subsequently. The pruning part requires specification of a tuning parameter, the so-called *cutoff*. The cutoff K is a threshold value when comparing a tree with its subtree by pruning away one terminal node; the comparison is made with respect to the difference of deviance from the two trees. A large cutoff has a stronger tendency for pruning and yields smaller estimated context trees, that is, a smaller dimension of the model.

The value of the cutoff K has a heuristic meaning on the scale of χ^2-quantiles. As a threshold for differences of deviances, an asymptotic $\frac{1}{2}\chi^2_\nu$ distribution with $\nu = \text{card}(\mathcal{X})-1$ is associated to every individual decision about pruning a terminal node (under the null-hypothesis of equal models, i.e., pruning). Therefore, we often specify the cutoff K on the scale of percentages (quantiles),

$$K = K(\alpha) = \frac{1}{2}\chi^2_{\nu,\alpha} = \frac{1}{2}\,\text{qchisq}(1-\alpha,\nu), \quad \nu = \text{card}(\mathcal{X}) - 1. \tag{3.1}$$

For the DNA case, $\nu = 3$, and hence $K(\alpha) = 3.91$ and 5.67 for $\alpha = 5\%$ or 1%, respectively. Note that we use the notation $\hat{\tau} \equiv \tau_{\hat{c}}$ for the fitted context tree.

3.1 IMPLEMENTATION IN R

We provide an R package VLMC, also available from the Comprehensive R Archive Network (CRAN 1997 ff.). After starting R, we load the package into the running R session, load the BNRF1 gene data (which comes with the VLMC package as well), and look at its first 50 values:

```
> library(VLMC)
> data(bnrf1)
> bnrf1EB [1:50]
 [1] a t g g a a g a g a g g g g c a g g g a a a c g c a a a t g
[31] c c g g t t g c c c g g t a t g g g g g
Levels:  a c g t
```

Now, fit a VLMC with cutoff $K = 5$ which gives a smaller than optimal tree with the advantage of using less space on paper. The resulting object, `vc5` gives basic information when printed:

```
> vc5 <- vlmc(bnrf1EB, cutoff = 5)
> vc5
'vlmc', a Variable Length Markov Chain;
         alphabet 'acgt', |alphabet| = 4, n = 3954.
Call: vlmc(dts = bnrf1EB, cutoff.prune = 5)
 -> extensions (=  $size ) :
   ord.MC   context nr.leaves     total
        4        18         9       154
AIC =  10580
```

```
> draw(vc5, cumul = FALSE)
[x]-( 0 0 0 0| 0)-F
 +--[a]-( 62 109 130 74| 375)
 |   +--[c]-( 43 47 70 37| 197)
 |   |   `--[c]-( 9 15 31 13| 68)
 |   |       `--[c]-( 2 12 4 0| 18)-T
 |   `--[t]-( 14 36 12 24| 86)-T
 +--[c]-( 144 186 127 152| 609)
 |   +--[a]-( 49 65 67 38| 219)-T
 |   `--[g]-( 23 59 29 39| 150)
 |       +--[a]-( 30 8 13 15| 66)-T
 |       `--[g]-( 35 43 17 22| 117)
 |           `--[c]-( 2 19 9 4| 34)-T
 +--[g]-( 244 367 417 203| 1231)-T
 `--[t]-( 32 97 120 61| 310)
     +--[c]-( 23 71 102 36| 232)
     |   `--[a]-( 13 7 6 12| 38)-T
     `--[g]-( 6 18 40 25| 89)
         +--[g]-( 8 21 29 6| 64)-T
         `--[t]-( 4 15 9 22| 50)-T
```

The context function is

$$c(x_{-\infty}^{0}) = \begin{cases} a[ag] & \text{if } x_0 = a,\ x_{-1} \in \{a, g\},\ x_{-\infty}^{-2} \text{ arbitrary} \\ ac[agt] & \text{if } x_0 = a,\ x_{-1} = c,\ x_{-2} \neq c,\ x_{-\infty}^{-3} \text{ arbitrary} \\ acc[agt] & \text{if } x_0 = a,\ x_{-1} = c,\ x_{-2} = c,\ x_{-3} \neq c, \ldots \\ accc & \text{if } x_0 = a,\ x_{-1} = c,\ x_{-2} = c,\ x_{-3} = c, \ldots \\ at & \text{if } x_0 = a,\ x_{-1} = t,\ x_{-\infty}^{-2} \text{ arbitrary} \\ c[ct] & \text{if } x_0 = c,\ x_{-1} \in \{c, t\},\ x_{-\infty}^{-2} \text{ arbitrary} \\ ca & \text{if } x_0 = c,\ x_{-1} = a,\ x_{-\infty}^{-2} \text{ arbitrary} \\ cg[ct] & \text{if } x_0 = c,\ x_{-1} = g,\ x_{-2} \in \{c, t\},\ x_{-\infty}^{-3} \text{ arbitr.} \\ cga & \text{if } x_0 = c,\ x_{-1} = g,\ x_{-2} = a,\ x_{-\infty}^{-3} \text{ arbitr.} \\ cgg[agt] & \text{if } x_0 = c,\ x_{-1} = g,\ x_{-2} = g,\ x_{-3} \neq c, \ldots \\ cggc & \text{if } x_0 = c,\ x_{-1} = g,\ x_{-2} = g,\ x_{-3} = c, \ldots \\ g & \text{if } x_0 = g,\ x_{-\infty}^{-1} \text{ arbitrary} \\ t[at] & \text{if } x_0 = t,\ x_{-1} \in \{a, t\},\ x_{-\infty}^{-2} \text{ arbitrary} \\ tc[cgt] & \text{if } x_0 = t,\ x_{-1} = c,\ x_{-2} \neq a,\ x_{-\infty}^{-3} \text{ arbitrary} \\ tca & \text{if } x_0 = t,\ x_{-1} = c,\ x_{-2} = a,\ x_{-\infty}^{-3} \text{ arbitrary} \\ tg[ac] & \text{if } x_0 = t,\ x_{-1} = g,\ x_{-2} \in \{a, c\},\ x_{-\infty}^{-3} \text{ arbitr.} \\ tgg & \text{if } x_0 = t,\ x_{-1} = g,\ x_{-2} = g,\ x_{-\infty}^{-3} \text{ arbitrary} \\ tgt & \text{if } x_0 = t,\ x_{-1} = g,\ x_{-2} = t,\ x_{-\infty}^{-3} \text{ arbitrary} \end{cases}$$

Figure 4. Fitted context tree $\hat{\tau}$ for `vc5`*. Note that "leaves" are marked with* `T` *("T"erminal), whereas intermediate nodes which do* not *belong to the context are marked* `F` *("F"ull) because they have the full number (*$\mathrm{card}(\mathcal{X}) = 4$*) of offsprings. The estimated transition probabilities $\hat{P}(x_t \mid \hat{c}(x_{t-1}, x_{t-2}, \ldots))$ can also directly be read from the above output of* `draw(*)`*: For each context, it is a probability vector of length 4, the fractions given between the "*`(...)`*", for example, $\hat{P}(X_t = u \mid \hat{c}(X_{t-1} = a, X_{t-2} \in \{a, g\}, \ldots))$ is 62/375, 109/375, 130/375, or 74/375, for $u = a, c, g$ or t, respectively.*

We see that the embedding Markov chain is of order $p = 4$, the context size $\mathrm{card}(\tau_{\hat{c}}) = 18$ with 9 leaves (terminal nodes). Figure 4 draws a "lying" context tree, one line per node, using the `draw()` method for `vlmc` objects.

The conditional probabilities $\hat{P}(x_t \mid \hat{c}(x_{t-1}, x_{t-2}, \ldots))$ (see Figure 4) can also be extracted from the fitted `vlmc` object using the `predict()` method. On one hand, the "in-sample" values for $t = 2, 3, \ldots, n$,

```
> p5 <- predict(vc5)
> dim(p5)                                          # n × 4 where p5[1,] === NA
[1] 3954    4
> p5[1:5,]
          a         c         g         t
a        NA        NA        NA        NA
t 0.1653333 0.2906667 0.3466667 0.1973333
g 0.1032258 0.3129032 0.3870968 0.1967742
g 0.1982128 0.2981316 0.3387490 0.1649066
a 0.1982128 0.2981316 0.3387490 0.1649066
```

or for a "new" observation $(x_7, x_6, \ldots, x_1) = (a, c, g, g, c, g, c)$ which in R is written in the usual forward time-order $(x_1, x_2, \ldots)$,

```
> predict(vc5, c("c","g","c","g","g","c","a"))[7,]
         a         c         g         t
0.05882353 0.55882353 0.26470588 0.11764706
```

gives the probabilities 2/34, 19/34, 9/34, and 4/34 corresponding to the (c g g c) context.

These probabilities (or `predict` directly) can also be used for predicting future observations or, put differently, as a classifier, given the past context by predicting $\arg\max_x P(x|.)$, for example,

```
> predict(vc5, c("c","g","c","g","g","c","a"), type="class")
[1] NA c  g  c  g  g  c
Levels:  a c g t
```

gives the one step ahead predictions for each but the first position, where, for example, the last `"c"` is the most probable value with probability 19/34. Finally, `predict(*, type = ..)` can be used to give the above plus more detailed information, including the context "at x_t", using `type = "ALL"`,

```
> predict(vc5, c("c","g","c","g","g","c","a"), type="ALL")
   fit Pr[X= a ] Pr[X= c ] Pr[X= g ] Pr[X= t ] id  flags ctxt
c  NA  NA        NA        NA        NA        NA  0     NA
g  c   48/203    62/203    127/609   152/609   5   55    c
c  g   244/1231  367/1231  417/1231  203/1231  6   0     g
g  c   23/150    59/150    29/150    13/50     22  0     cg
g  g   244/1231  367/1231  417/1231  203/1231  6   0     g
c  g   244/1231  367/1231  417/1231  203/1231  6   0     g
a  c   1/17      19/34     9/34      2/17      361 0     cggc
```

or the context depth "at x_t" from `predict(*, type = "depth")`.

3.2 TUNING I: CHOOSING THE CUTOFF PARAMETER WITH INFORMATION CRITERIA

The only tuning parameter for our fitting of a VLMC is the cutoff K, mentioned in the previous section. Because this is a one-dimensional parameter, optimization with respect to the cutoff is relatively easy (optimizing among all subtrees from a large tree is prohibitive).

One of the most popular methods for model selection are the AIC and BIC criterion: here, they become

$$-\,2\text{log-likelihood}_K + \gamma\,(\text{card}(\mathcal{X}) - 1)\text{card}(\tau_{\hat{c}_K}), \tag{3.2}$$

where $\gamma = 2$ or $\log(n)$ for AIC and BIC, respectively. We emphasize here the dependence on the cutoff K used in the context algorithm. Optimizing AIC or BIC is fast. An AIC estimated cutoff aims to minimize the Kullback-Leibler divergence between the true underlying process and the fitted VLMC model; see Shibata (1997).

Our VLMC package provides `logLik()` and `AIC()` methods for `"vlmc"` objects, for example,

```
> logLik(vc5)
'log Lik.' -5236.205 (df=54)
> c(AIC = AIC(vc5),  BIC = AIC(vc5, k = log(vc5$n)))
     AIC      BIC
10580.41 10919.66
```

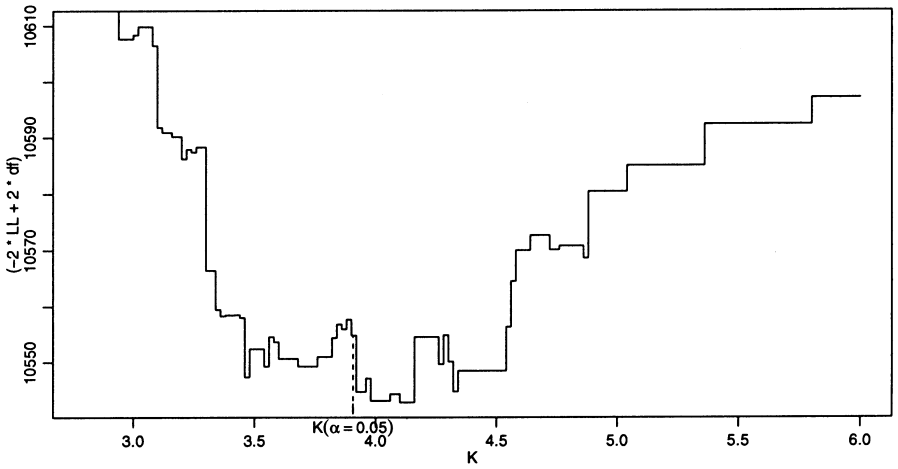

Figure 5. `AIC(vlmc(bnrf1EB, cutoff = K))` *as function of* K. *Note the local minima and the fact that* $K(\alpha = 5\%)$ *is close to optimal.*

Figure 5 shows the result of using `AIC(vlmc(*, cutoff = K))` for $K \in \{2.8, 2.82, \ldots, 6\}$ ("`for(K in seq(2.8, 6, by = .02))`" in R). Note that we often use AIC rather than BIC for the following reasons. Whereas AIC is known to overfit for choosing the correct finite-dimensional models, it can give smaller prediction squared error in the infinite case (such as $AR(\infty)$) and actually often *underfits* for minimal 0-1 prediction loss, for example, see Friedman, Hastie, and Tibshirani (2000, Rejoinder, figure 1). Moreover, we have tried to use BIC instead of AIC for VLMC in several real data examples, and BIC there tends to underfit severely, in some cases even choosing the independency model. On the other hand, if the data is generated by a VLMC, BIC has empirically identified the correct model for large n, that is, needing rather $n = 10{,}000$ instead of $n = 1{,}000$, even for a very simple VLMC. If the aim is not minimizing the Kullback-Leibler divergence, such simple criteria are not at hand. Estimation of the cutoff parameter can then be pursued with bootstrapping as described in Section 4.1.

4. BOOTSTRAPPING AND SIMULATING FROM A VLMC

Simulating a VLMC of order p is done via its transition probabilities $\{P(x_1 \mid c(x^0_{-p+1}); x^1_{-p+1} \in \mathcal{X}^{p+1}\}$: start with an initial p-vector $X^0_{-p+1} \in \mathcal{X}^p$, where p is the order of the VLMC and simulate

$$X_t \sim P(\cdot \mid c(X_{t-p}^{t-1})),\ t = 1, 2, \ldots \tag{4.1}$$

Under regularity conditions, the effect of the initialization gets forgotten exponentially fast as the simulated path gets longer, and the simulated values are becoming close to a sample from the stationary distribution of the VLMC. Thus, to simulate an n-dimensional sample

from the stationary distribution of the VLMC (assuming it exists), we proceed as follows:

$$\begin{aligned} &\text{simulate } X_1, X_2, \ldots, X_{m+n} \text{ as in (4.1);} \\ &\text{choose } X_{m+1}, \ldots, X_{m+n} \text{ as an approximately stationary sample of size } n. \end{aligned} \tag{4.2}$$

Here m is a large number such as 10^3 or 10^4 and the initial vector in (4.1) may be chosen as a p-vector whose elements are all equal to some $x \in \mathcal{X}$. Our R function `simulate.vlmc()` has an argument `n.start` for m taking $m = 64 \times (\text{context size}) = 64\text{card}(\tau_c)$ as default value, for example,

```
> simulate.vlmc(vc5, n = 17)
 [1] "a" "g" "t" "g" "a" "g" "c" "a" "c" "g" "a" "g" "g" "t" "c" "c" "c"
> simulate.vlmc(vc5, n = 17)
 [1] "g" "a" "g" "c" "g" "a" "t" "g" "c" "g" "g" "t" "c" "g" "t" "g" "t"
> simvc5 <- simulate.vlmc(vc5, n = 100000)
> table(simvc5)
    a     c     g     t
18822 30249 31010 19919
```

where the last simulation (of length 100,000) still takes only between 1 or 2 tenths of a second on a 900 MHz Pentium III Computer.

The VLMC bootstrap is nothing other than simulating from a fitted VLMC. The bootstrap sample of size n is constructed as in (4.2), using in (4.1) the estimated transition probabilities from the context algorithm $\hat{P}(\cdot \mid \hat{c}(X_{t-p}^{t-1}))$ with p the order of the estimated VLMC. The usual notation for such a bootstrap sample is then

$$X_1^*, \ldots, X_n^*. \tag{4.3}$$

We *could* (but have not done so in the VLMC package) define

```
> bootstrap.vlmc <- function(x, B)
sapply(1:B, function(i) simulate.vlmc(x, n = x$n, integer.ret = TRUE))
```

where the `integer.return = TRUE` argument results in series with values in $\{0, 1, \ldots, \text{card}(\mathcal{X}) - 1\}$ instead of characters, for example, `"a", "c", "g", "t"`. This is faster and needs less memory for the result. VLMC-bootstrapping of an estimator $T_n = h_n(X_1, \ldots, X_n)$, a function $h_n(\cdot)$ of the data, is constructed with the plug-in rule:

$$T_n^* = h_n(X_1^*, \ldots, X_n^*),$$

with the same function $h_n(\cdot)$. As an example, we take $h_n(x_1^n) = \#\{i \mid x_i = \mathtt{a},\ x_{i-1} = \mathtt{t}\}/(n-1)$, that is, the occurrence frequency of "TA" in the DNA sequence. To this end we simulate from a relatively large model,

```
> vc3 <- vlmc(bnrf1EB, cutoff= 3)                    # smaller K overfits
```

The above `bootstrap.vlmc` function allows things like

```
> bb <- bootstrap.vlmc(vc3, B=200)                    # needs around 1 sec
> dim(bb)                                             # n × B
[1] 3954  200
>  object.size(bb)
[1] 3163412
```

which produces relatively large results `bb`. Using a loop instead, `for(i in 1:B) { ri <- simulate.vlmc(x, ...); ......}` often makes more sense to avoid memory problems. Continuing our example, we now compute and plot the bootstrap distribution of $h_n()$ (which is `at.freq()` in the following):

```
>  at.freq <- function(x) {
                              # Frequency of "A T" (backwards) in 0:3 coded x:  "a" ^= 0, "t" ^= 3
      n <- length(x)
      sum(x[-1] == 0 & x[-n] == 3)/(n-1)
  }
> (at.f <- at.freq(alpha2int(bnrf1EB,"acgt")))
[1] 0.02175563
> at.f == 86 / 3953
[1] TRUE
>  at.st <- apply(bb,2, at.freq)
>  summary(at.st)
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01644 0.02074 0.02214 0.02214 0.02378 0.02859

> plot(density(at.st, bw = 5e-4), ylim = c(0, 190),
    main="VLMC-Bootstrap Distribution of 'T-A' Frequency in 'bnrf1EB'")
> boxplot(at.st, add = TRUE, horizontal = TRUE, at = -3.6, boxwex= 7)
> abline(v=at.f, col = "red", lty = 3)
> hst <- hist(at.st, nc = 12, plot = FALSE)
> lines(hst, freq = FALSE)
```

which produces Figure 6.

4.1 TUNING II: CHOOSING THE CUTOFF PARAMETER WITH THE BOOTSTRAP

We have discussed in Section 3.2 how information criteria can be used to estimate the cutoff parameter from the context algorithm. When using AIC, this amounts to tuning with respect to Kullback-Leibler divergence.

In many applications, other loss functions are of interest. For example, the classification error, or zero-one loss, is often of interest in practice. When restricting to one-step ahead predictions, the goal is to minimize

$$\mathbb{P}[\hat{X}_{n+1;K} \neq X_{n+1}], \tag{4.4}$$

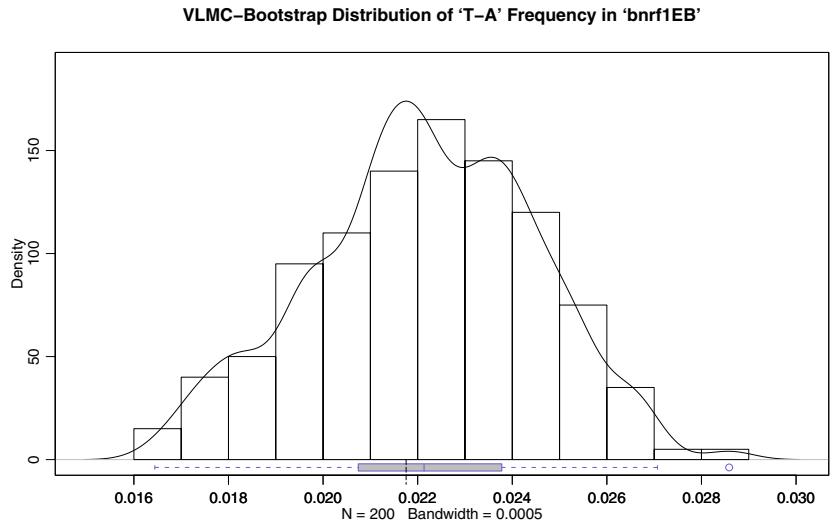

Figure 6. VLMC-bootstrap distribution ($B = 200$) of the number of occurrences of "T-A" (or "A-T" backwards) where the VLMC bootstrap is simulated from `vlmc(bnrf1EB, cutoff = 3)`. *The sample value 86/3,953 = 0.0218 is marked by a short vertical dashed line.*

for the classifier

$$\hat{X}_{n+1;K} = \operatorname*{arg\,max}_{x_{n+1}\in\mathcal{X}} \hat{P}(x_{n+1} \mid \hat{c}_K(x_1^n))$$

whose implementation has been demonstrated in Section 3.1. Optimality with respect to Kullback-Leibler divergence and misclassification in (4.4) do not coincide.

The VLMC bootstrap (depending on an "initial" cutoff K_0) has been proposed for estimating an optimal cutoff K for general loss functions; see Bühlmann (2000). Estimation of K aiming to minimize (4.4) can be done as follows:

Step 1. Choose an initial cutoff K_0 and simulate $X_1^*, \ldots, X_{n+1}^*$ as in (4.3) from the estimated transition probabilities $\hat{P}(\cdot \mid \hat{c}_{K_0}(X_{t-p_0}^{t-1}))$, where p_0 is the order of the VLMC, fitted with cutoff K_0.

Step 2. For a given cutoff K, estimate the one-step ahead predictor $\hat{X}_{n+1;K}^*$ from the context algorithm, based on $X_1^*, \ldots, X_n^*$. Evaluate the zero-one loss

$$\mathbf{1}_{[\hat{X}_{n+1;K}^* \neq X_{n+1}^*]}.$$

Repeat this B times (e.g., $B = 1{,}000$) and average the zero-one losses to obtain

$$\operatorname*{ave}_B \left[\mathbf{1}_{[\hat{X}_{n+1;K}^* \neq X_{n+1}^*]}\right] = B^{-1} \sum_{b=1}^{B} \mathbf{1}_{[\hat{X}_{n+1;K}^{*b} \neq X_{n+1}^{*b}]}$$

which is a Monte Carlo approximation for $\mathbb{P}^*[\hat{X}_{n+1;K}^* \neq X_{n+1}^*]$.

Step 3. Searching over candidate cutoffs K yields the estimate

$$\hat{K} = \arg\min_K \operatorname*{ave}_B \left[\mathbf{1}_{[\hat{X}^*_{n+1;K} \neq X^*_{n+1}]} \right].$$

Note that $\hat{K}$ depends on the initial cutoff K_0 in Step 1. Its choice has a minor effect (but see Figure 7). We advise to take the initial K_0 such that the estimated context tree is "large." For example, we could take K_0 from minimizing the criterion in (3.2) with $\gamma = 1$. For the gene data example, this gives $K_0 = 1.33$. The following R code (our real code additionally measured and saved the computing time used, and for monitoring purposes prints out status information for each sample) for estimating $\hat{K}$ by bootstrapping $B = 1{,}000$ on a grid of K values of size 89

```
> K0 <- 1.33                                   # 1.33 is the IC[gamma=1] optimum

> n <- length(x <- bnrf1EB)
> B <- 1000
> (nK <- length(Kset <- seq(1.6, 6, by = 0.05)))
[1] 89
> (vK0 <- vlmc(x, cut = K0))

'vlmc', a Variable Length Markov Chain;
         alphabet 'acgt', |alphabet| = 4, n = 3954.
Call: vlmc(dts = x, cutoff.prune = K0)
 -> extensions (= $size ) :
   ord.MC   context nr.leaves     total
        8      1041       605      9074
AIC =  12398
> set.seed(1521)                               # Random seed for reproducibility
> n.matches <- integer(nK)

> for(b in 1:B) {                              # for each bootstrap sample do
    cat(b,"")
    x0 <- simulate.vlmc(vK0, n = n + 1, n.start = 10000)
    for(iK in 1:nK) {                                                  # for each K
        K <- Kset[iK]
        v0 <- vlmc(x0[1:n], cut = K)
        p <- v0$size["ord.MC"]
        n.matches[iK] <- n.matches[iK] +
        (x0[n+1] == predict(v0, x0[(n-p+1):(n+1)], type = "class")[p+1])
   }
}
```

needs about 73 minutes on a Pentium III 933 ($\approx 4.7''$ per sample for all K's). Repeating these 89 bootstrap simulations (of $B = 1{,}000$ replications each) for three other values $K_0 =$

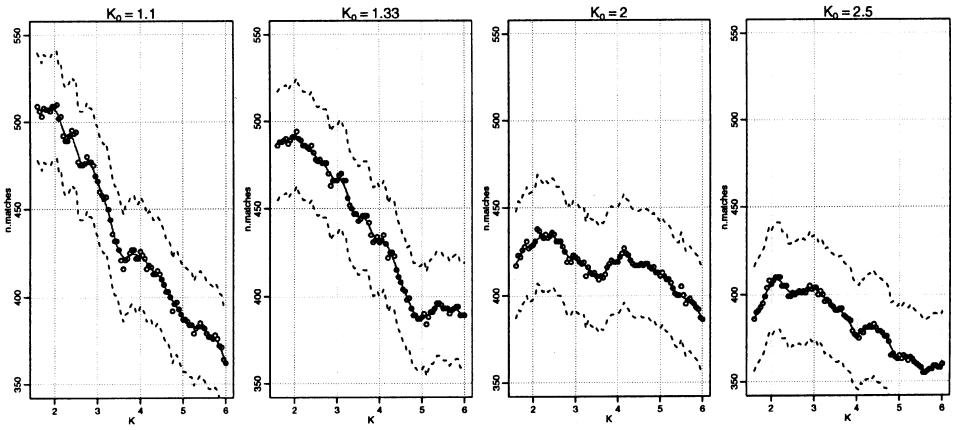

Figure 7. Four series of VLMC-bootstraps (in the BNRF1 gene dataset) of decreasing complexity determined by K_0 for determining an optimal "0-1 loss" cutoff K where optimality (to be maximized*) is measured as number of* correct *1-step ahead predictions. We used $B = 1{,}000$ bootstrap samples for each $K \in \{1.6, 1.65, \ldots, 6\}$. Note that random guessing would give 250 matches on average. Additionally, 95% pointwise confidence intervals (for binomial proportions) are drawn. In all cases, an optimal K is about 2–2.2, substantially smaller (corresponding to much larger models) than the AIC optimal $K \approx 4$, see Section 3.2 and Figure 5 above.*

1.1, 2, and 2.5 and plotting `n.matches` versus `Kset` for each produces Figure 7.

Generalizations to other classifiers respecting nonequal misclassification costs, to multistep ahead predictions and to other loss functions can be pursued analogously.

Alternatively, an out-of-sample estimate of a loss function could be minimized to obtain an estimate of a optimal cutoff K. Or instead, subsampling techniques for time series may be used, see Fukuchi (1999).

5. DIAGNOSTICS

Here, we describe graphical and numerical diagnostics about the quality and appropriateness of a fitted VLMC model. In addition to the log-likelihood or AIC value given in Section 3.2, the (in)famous R^2 diagnostic of regression would correspond to the inherent in-sample prediction quality of the model for the data. The first, a (often biased) estimator of $P(X_t = x_t \mid c(x_{t-1}, x_{t-2}, \ldots))$, is simply $\overset{n}{\underset{t=p}{\text{ave}}} \mathbf{1}_{\hat{X}_t = x_t}$, where $\hat{X}_t$ is using the fitted model and the data context $(x_{t-1}, x_{t-2}, \ldots)$. More generally, when $m = \text{card}(\mathcal{X})$, we want to see the $m \times m$ confusion matrix $c = (c_{ij})$ where $c_{i,j} = \#\{x_t = i, \hat{X}_t = j\} = \sum_{t=2}^{n} \mathbf{1}_{x_t = i} \mathbf{1}_{\hat{X}_t = j}$ for $i, j \in \{1, 2, \ldots m\}$. For the gene data example and the cutoff $K = 2$ (see Figure 7), we get

```
> vc2  <- vlmc(bnrf1EB, cutoff= 2)
> summary(vc2)                          # summary() is print() + more :

'vlmc', a Variable Length Markov Chain;
         alphabet 'acgt', |alphabet| = 4, n = 3954.
```

```
Call: vlmc(dts = bnrf1EB, cutoff.prune = 2)
 -> extensions (= $size ) :
  ord.MC   context nr.leaves     total
       7       481       251      4106
AIC =  11032.37
R^2 = % correctly predicted = 50.78%
Confusion matrix:
    predicted
data  a   c   g   t
   a 337 197 157  52
   c 144 701 268  82
   g 136 285 736  75
   t 116 209 224 234
Markov chain depths along the data:
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     1       4       4   4.126       5       7
```

More interesting than these numbers are time-dependent features of the fit. Apart from the fitted values (the most probable class by `predict(*, type = "class")`), we have already seen the (instantaneous) depth of the Markov chain or length of context, that is, `predict(*, type = "depth")`. Both of these might be used in plots, but have not been too much revealing in our example.

We have defined several kinds of *residuals* for VLMC models; the first kind (and default in our implementation of `residuals.vlmc()`), is *multivariate*, returning an $n \times m$ matrix (of the same dimension as `predict()`) of *"classwise"* residuals $r_{t,j} = \mathbf{1}_{x_t=j} - \hat{P}(j \mid \hat{c}(x_1^{t-1}))$ where for notational convenience, we assume the alphabet $\mathcal{X} = \{1, 2, \ldots, m\}$, that is, for each t, the only positive residual is the one corresponding to the observed value, and $\sum_{j=1}^{\text{card}(\mathcal{X})} r_{t,j} = 0$ for all t.

The other generally useful kind of residuals are the *deviance* residuals r_t, defined such that $\sum_t r_t^2 = \text{Deviance} = -2 \cdot \text{log-likelihood}$, that is, $r_t^2 = -2 \log \hat{P}(x_t \mid \hat{c}(x_1^{t-1}))$ and the *sign* of the deviance residuals is set to the sign of the *response* or *working* residuals. These are defined as the difference $x_t - \hat{x}_t$ something which makes sense in the binary case and whenever $\mathcal{X}$ can be interpreted as *ordered*.

We propose a novel useful graphic, `RCplot()`, plotting the squared deviance residuals, r_t^2 against the context $\hat{c}(x_1^{t-1})$, that is, summarizing the r_t^2 by a boxplot for each context, in Figure 8, where we also show the number of observations per context state, both literally as text, and by boxplot width proportional to the square root of the number. Designing and interpreting diagnostic plots in the present context of discrete time series fitting is still very much a subject of research and experimentation. We have provided the relevant quantities with the above variants of `predict()` and `residuals()` for fitted VLMC models.

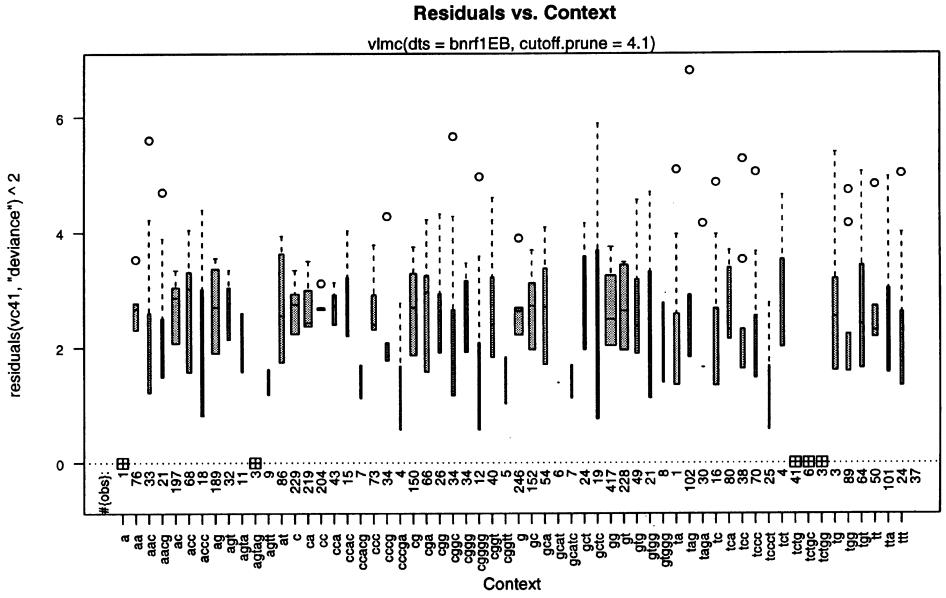

Figure 8. `RCplot(vc41)`, *a plot of* squared *deviance residuals vs. context, for a VLMC(with AIC tuned cutoff* $K = 4.1$*) fitted to the Epstein-Barr virus data (*$n = 3{,}954$*). Note that out of 60 different context states,* `"a"` *is not a proper one (but corresponds to* X_2*), and there are four other contexts (of only three or six observations) where all residuals are 0; in particular, a* `tctg` *past seems well predictable.*

6. ALGORITHMS AND SUPPORTING THEORY

6.1 THE CONTEXT ALGORITHM

Fitting a VLMC is done with a version of the tree-structured context algorithm. It uses the notion of terminal node context trees

$$\tau^T = \tau_c^T = \{w; w \in \tau_c \quad \text{and} \quad wu \notin \tau_c \quad \text{for all} \quad u \in \mathcal{X}\}.$$

Here and in the sequel $wu = (w_{\text{card}(w)}, \ldots, w_2, w_1, u_{\text{card}(u)}, \ldots, u_2, u_1)$ denotes the concatenation of the vectors w and u (components always written in reverse order). In Example A, $\tau^T = \{A, C, G, TT\}$ is the set of terminal nodes in the tree in Figures 2 and 3, whereas $\tau = \tau^T \cup \{T[ACG]\}$. In Example B, τ^T and τ from Figure 2 coincide. The information of τ^T is equivalent to the information in τ. Thus, the terminal node tree yields a more compact representation.

In the sequel, we often write for a probability distribution P on $\mathcal{X}^{\mathbb{Z}}$ (for stochastic process), $P(x) = \mathbb{P}_P[X_1^m = x]$ $(x \in \mathcal{X}^m)$ and $P(x \mid w) = P(xw)/P(w)$ $(x, w \in \bigcup_{j=1}^{\infty} \mathcal{X}^j)$. Denote by

$$N(w) = \sum_{t=1}^{n-\text{card}(w)+1} 1_{[X_t^{t+\text{card}(w)-1} = w]}, \qquad w \in \bigcup_{m=1}^{\infty} \mathcal{X}^m, \tag{6.1}$$

the number of occurrences of the string w in the sequence X_1^n, and let $N_{-1}(w)$ be the same as $N(w)$ but summing up to $t = n - \text{card}(w)$, dropping the last term. Moreover, let

$$\hat{P}(w) = N(w)/n, \;\; \hat{P}(x \mid w) = \frac{N(xw)}{N_{-1}(w)}, \quad x, w \in \bigcup_{m=1}^{\infty} \mathcal{X}^m. \tag{6.2}$$

(This corrects a small error in Bühlmann and Wyner (1999) which had $N(w)$ instead of $N_{-1}(w)$.) The following algorithm constructs the estimated context tree $\hat{\tau}$ as the biggest context tree (with respect to the order "$\preceq$" defined in Step 1 below) such that

$$\Delta_{wu} = \sum_{x \in \mathcal{X}} \hat{P}(x \mid wu) \log\left(\frac{\hat{P}(x \mid wu)}{\hat{P}(x \mid w)}\right) N(wu) \geq K \quad \text{for all} \quad wu \in \hat{\tau}^T \; (u \in \mathcal{X}) \tag{6.3}$$

where K is the user specified cutoff; see Sections 3.2 and 4.1.

Step 1. Given $\mathcal{X}$-valued data $X_1, \ldots, X_n$, fit a maximal context tree, that is, search for the context function $c_{\max}(\cdot)$ with terminal node context tree representation $\tau^T_{\max}$, where $\tau^T_{\max}$ is the biggest tree such that every element (terminal node) in $\tau^T_{\max}$ has been observed at least twice in the data. This can be formalized as follows:

- $\tau^T_{\max}$ is such that $w \in \tau^T_{\max}$ implies $N(w) \geq 2$, and such that for every τ^T, where $w \in \tau^T$ implies $N(w) \geq 2$, it holds that $\tau^T \preceq \tau^T_{\max}$. Here, $\tau_1 \preceq \tau_2$ means: $w \in \tau_1 \Rightarrow wu \in \tau_2$ for some $u \in \bigcup_{m=0}^{\infty} \mathcal{X}^m$ ($\mathcal{X}^0 = \emptyset$).
- Set $\tau^T_{(0)} = \tau^T_{\max}$.

Step 2. Examine every element (terminal node) of $\tau^T_{(0)}$ as follows (the order of examining is irrelevant). Let $c(\cdot)$ be the corresponding context function of $\tau^T_{(0)}$ and let

$$wu = x^0_{-\ell+1} = c(x^0_{-\infty}), \; u = x_{-\ell+1}, \; w = x^0_{-\ell+2},$$

where wu is an element (terminal node) of $\tau^T_{(0)}$, which we compare with its pruned version $w = x^0_{-\ell+2}$ (if $\ell = 1$, the pruned version is the empty branch $\emptyset$, that is, the root node).

Prune $wu = x^0_{-\ell+1}$ to $w = x^0_{-\ell+2}$ if

$$\Delta_{wu} = \sum_{x \in \mathcal{X}} \hat{P}(x \mid wu) \log\left(\frac{\hat{P}(x \mid wu)}{\hat{P}(x \mid w)}\right) N(wu) < K,$$

with K a cutoff parameter and $\hat{P}(\cdot \mid \cdot)$ as defined in (6.2). Decision about pruning for every terminal node in $\tau^T_{(0)}$ yields a (possibly) smaller tree $\tau_{(1)} \preceq \tau^T_{(0)}$. From $\tau_{(1)}$, construct the terminal node context tree $\tau^T_{(1)}$.

Step 3. Repeat Step 2 with, starting with $\tau^T_{(i)}$ yielding $\tau_{(i+1)}, \tau^T_{(i+1)}$ ($i = 1, 2, \ldots$) until no more pruning is possible. Denote this maximal pruned context tree (not necessarily of terminal node type) by $\hat{\tau} = \tau_{\hat{c}}$ and its corresponding context function by $\hat{c}(\cdot)$.

Step 4. If interested in probability distributions, estimate the underlying transition probabilities $P(x_1 \mid c(x^0_{-\infty}))$ by $\hat{P}(x_1 \mid \hat{c}(x^0_{-\infty}))$, where $\hat{P}(\cdot \mid \cdot)$ is defined as in (6.2).

More details and motivation can be found in Bühlmann and Wyner (1999).

6.1.1 Supporting Theory

Under regularity conditions for the data generating VLMC, the context algorithm consistently estimates the underlying context tree τ_c,

$$\mathbb{P}[\tau_{\hat{c}} = \tau_c] \to 1 \ (n \to \infty),$$

see Bühlmann and Wyner (1999). Having a consistent estimate of the structure of the memory of a VLMC, a consistent estimate of the true transition probabilities

$$P(x_1|c(x_{-\infty}^0)) \quad \text{for all} \quad x_{-\infty}^1 \in \mathcal{X}^\infty, \tag{6.4}$$

and hence of all finite-dimensional distribution

$$\mathbb{P}[(X_1, \ldots, X_m) = x] \quad \text{for all} \quad x \in \mathcal{X}^m \text{ and all } m \tag{6.5}$$

follows. Even more, the context algorithm yields efficient estimates of the transition probabilities in (6.4). It implies, in the sense of statistical efficiency, maximal estimation accuracy for parameters being smooth functions of the transition probabilities (Bühlmann 1999).

Moreover, the context algorithm has good approximation properties for categorical processes which are not necessarily a VLMC. For a general stationary $\mathcal{X}$-valued process, the corresponding context tree has infinite depth. Under suitable regularity conditions, the context algorithm still yields a consistent estimate in the sense that the finite-length contexts are consistently estimated and the algorithm grows longer contexts where the underlying context tree has infinite depth; see Ferrari and Wyner (2003). The result can be interpreted as follows: the context algorithm consistently estimates the finite memory parts and grows longer memory whenever the underlying true contexts are infinitely long. Also, all finite-dimensional distributions in (6.5), even for a (suitably regular) stationary process not being a VLMC, are consistently estimated. The accuracy of the VLMC approximation with nearly optimal rates has been established for the asymptotic variance and distribution of smooth function of means, see also Section 6.2.

Asymptotic theory tells that the cutoff K should depend on the sample size: a sufficient condition is $K = K_n = C\log(n)$ with $C > 2\text{card}(\mathcal{X}) + 4$. Since the condition seems far from necessary, we view a strict choice on the $\log(n)$ scale as dangerous for finite samples. This choice, $K = (2\text{card}(\mathcal{X}) + 4)\log(n)$, was often found to produce too large cutoffs yielding underfitted models when sample size was in the range of 1,000. This is one reason, we've kept the default (3.1), that is, $K = K(\alpha) = \frac{1}{2}\chi^2_{\nu,\alpha}$ for $\alpha = 5\%$. But the theoretical fact that long time series require larger cutoffs is certainly relevant for real applications.

6.2 SUPPORTING THEORY FOR THE VLMC BOOTSTRAP

The resampling scheme has also been called the VLMC-sieve bootstrap: because the fitted VLMC can be viewed as a sieve estimator (estimate in the approximating class of

VLMC models) for the underlying data generating process. In some or even many cases, it is more accurate than the block bootstrap (Künsch 1989) which is a general bootstrap technique for stationary time series. In Bühlmann (2002), the following result about bootstrap variance estimation is given. Consider estimators T_n which are smooth transformations of means. Examples include counts or relative frequencies of tuples of labels (i.e., words) from the categorical time series. If the underlying process is geometrically mixing (geometric decay of dependence of events as their separation distance grows) and satisfies additional regularity conditions, then

$$n\text{var}^*(T_n^*) - n\text{var}(T_n) = O_P\left(n^{-1/2+\epsilon}\right) \qquad \text{for any} \quad \epsilon > 0,$$

where $n\text{var}(T_n)$ converges to a nondegenerate limit. This rate should be compared to $O_P(n^{-1/3})$ for the block bootstrap.

Estimation of the cutoff tuning parameter with the VLMC bootstrap, as described in Section 4.1, has been shown to have some reasonable asymptotic properties for general loss functions (not only the zero-one loss in Section 4.1); see Bühlmann (2000).

7. CONCLUSIONS

We have provided methodology and a tutorial based on new software for fitting VLMCs. Our tutorial is self-contained, gives rigorous and partially new mathematical descriptions of important concepts, and emphasizes interesting aspects by examples, including an application with a DNA sequence from the Epstein-Barr virus. VLMCs are based on the idea of having a memory whose length is allowed to vary depending on how the immediate past (lagged values) looks like. Such a variable length memory structure adds enormous additional flexibility for modeling and fitting Markov chains for categorical time series. In particular, they allow for many possible models between a classical Markov chain of order p and $p+1$, they have interesting interpretation as a tree-structured model and they open an avenue to deal intelligently with the curse of dimensionality. Although versions of VLMCs are mainly used in information theory, mostly in appearing in the form of algorithms and not so much as a model, the statistics community is still largely unaware of the attractive VLMC model.

Estimating a VLMC is a nontrivial computational task. Our approach and implementation is based on an efficient, tree-structured context algorithm which requires $O(n \log(n))$ operations only. A number of additional options and tools complement the core algorithm, some of them being essential for statistically sound use of VLMCs : goodness-of-fit measures, residuals, diagnostics, tuning for the context algorithm and simulating and bootstrapping with VLMCs . Our tutorial, together with the publicly available software with many new options and features, aims to substantially support the use and future development of VLMC. Our routines are written in C and are directly available in R; the former makes it possible to import our computing methodology to other, individually favored software environments.

[Received March 2002. December 2002.]

REFERENCES

Bühlmann, P. (1999), "Efficient and Adaptive Post-Model-Selection Estimators," *Journal of Statistical Planning and Inference*, 79, 1–9.

——— (2000), "Model Selection for Variable Length Markov Chains and Tuning the Context Algorithm," *Annals of the Institute of Statistical Mathematics*, 52, 287–315.

——— (2002), "Sieve Bootstrap With Variable Length Markov Chains for Stationary Categorical Time Series" (with discussion) *Journal of the American Statistical Association*, 97, 443–471.

Bühlmann, P., and Wyner, A. J. (1999), "Variable Length Markov Chains," *The Annals of Statistics*, 27, 480–513.

CRAN (1997 ff.), *The Comprehensive R Archive Network*, http://cran.R-project.org/.

Ferrari, F., and Wyner, A. J. (2003), "Estimation of General Stationary Processes by Variable Length Markov Chains," *Scandinavian Journal of Statistics*, 30, 459–480.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2000), "Boosting Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, 28, 400–407; Rejoinder.

Fukuchi, J.-I. (1999), "Subsampling and Model Selection in Time Series Analysis," *Biometrika*, 86, 591–604.

Künsch, H. R. (1989), "The Jackknife and the Bootstrap for General Stationary Observations," *The Annals of Statistics*, 17, 1217–1241.

Rissanen, J. (1983), "A Universal Data Compression System," *IEEE Transactions on Information Theory*, IT-29, 656–664.

Ron, D., Singer, Y., and Tishby, N. (1996), "The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length," *Machine Learning*, 25, 117–150.

Shibata, R. (1997), "Bootstrap Estimate of Kullback-Leibler Information for Model Selection," *Statistica Sinica*, 7, 375–394.

Shumway, R. H., and Stoffer, D. S. (2000), *Time Series Analysis and Its Applications*, New York: Springer Texts in Statistics.

Willems, F., Shtarkov, Y., and Tjalkens, T. (1996), "Context Weighting for General Finite-Context Sources," *IEEE Transactions on Information Theory*, IT-42, 1514–1520.