

this idea, examples are provided using both simulated and real-world financial data. It is shown that some structural regularities, undetectable by classical statistical tests, can be revealed by this new approach.

Our method is based on the seminal works of Andrei (Kolmogorov 1965) who proposed a definition of random sequences in non-statistical terms. His definition, which is actually among the most general ones, is itself based on Turing's and Godel's contributions to the so-called "computability theory" or "recursion theory" (Turing 1937, Gödel 1931). Kolmogorov's ideas have been exploited by physicists for a renewed definition of entropy (Zurek 1989), by biologists for the classification of phylogenetic trees (Cilibrasi & Vitanyi 2005) and by psychologists to estimate randomness-controlling difficulties (Griffith & Tenenbaum 2001).

In finance, Kolmogorov complexity is often used to measure the possibility to predict future returns. For example, Chen & Tan (1996) or Chen & Tan (1999) estimated the stochastic complexity¹ of a stock market using the sum of the squared prediction errors obtained by econometric models. Azhar, et al. (1994) measured the complexity of stock markets with the highest successful prediction rate² (SPR) that one can achieve with different compression techniques. Shmilovici, et al. (2003) and Shmilovici, et al. (2009) used the Variable Order Markov model (VOM, a variant of context predicting compression tools) to predict the direction of financial returns. They found a significant difference between the SPR obtained from financial data and that obtained from random strings. To establish a formal link between this result and the Efficient Market Hypothesis (EMH), the authors also simulated VOM-based trading rules on Forex time series and concluded that there were no abnormal profits.

Exploiting another compression technique, Silva, et al. (2008) and ranked stock markets all over the world according to their LZ index, an indicator showing how well the compression algorithm proposed by Lempel & Ziv (1976) works on financial returns.

Despite the perspectives opened up by these pioneer works, two main limits in the aforementioned literature can be highlighted:

¹The notion of stochastic complexity is proposed by Rissanen (1986) in replacing the universal Turing machine in the definition of Kolmogorov complexity by a class of *probabilistic* models.

²To obtain this successful prediction rate, at each step, the author uses compression algorithms to predict the direction of the next return, and calculates the rate of successful predictions for the whole series.

1. From a theoretical point of view, the frontier between the probabilistic framework and the algorithmic one is not clearly established.

A strength of the algorithmic complexity is that it tackles *one given string* at a time, and not necessarily a *population of strings* with probabilities generated by a given stochastic process. Hence, no probabilistic assumption is needed when one uses algorithmic complexity.

The estimation of successful prediction rates seems to suggest that price motions follow a certain distribution law. Despite the use of compression tools, this technique reintroduces a probabilistic framework. The general and non-probabilistic framework proposed by the algorithmic complexity theory is then weakened.

2. This is not the case with Silva et al. (2008) and Giglio, et al. (2008). However, the discretization technique used in these papers remains open to discussion. Actually, financial returns are often expressed in real numbers³, while compression tools only deal with integer numbers. So, regardless the use of compression tools, a discretization process which transforms real-number series into discrete ones, is always necessary.

To fit this requirement, Shmilovici et al. (2003) or Shmilovici et al. (2009), as well as Silva et al. (2008) or Giglio et al. (2008), proposed to transform financial returns into 3 signals: "positive", "negative", or "stable" returns.

Undoubtedly, this radical change leads to a significant loss of information from the original financial series. As Shmilovici et al. (2009) remarked themselves, "*the main limitation of the VOM model is that it ignores the actual value of the expected returns (Shmilovici et al. 2009).*"

The introduction of algorithmic complexity in finance could have wider implications. For example, Dionisio, et al. (2007) have claimed that the notion of complexity could become a measure of financial risk - as an alternative to "value at risk" or "standard deviation" - which could have general implications for portfolio management (see Groth & Muntermann (2011) for an application in this sense).

Given these two points, it seems interesting to establish a general algorithmic framework for price motions.

³For example, continuous returns are computed as the logarithm of the ratio between consecutive prices : $r_t = \log(p_{t+1}) - \log(p_t)$.

We propose an empirical method that allows to treat financial time series with algorithmic tools, avoiding the over-discretisation problem. This method is based on the initial investigation of Brandouy & Delahaye (2005) whose idea is developed by Ma (2010) on tackling real-world financial data. In this same framework, Zenil & Delahaye (2011) fostered future applications of Kolmogorov complexity in finance.

Using the algorithmic approach, we show that daily returns of Dow Jones industrial index has a relatively high Komogorov complexity during the period from 01/02/1928 to 30/08/2010. To a certain degree, this result supports the Efficient Market Hypothesis (Fama 1970) that attests the impossibility of outperforming the market.

This paper is organized as follows:

A first section formally presents Kolmogorov complexity and provides an elementary illustration of

this concept. A second section proposes a generic methodology to estimate Kolmogorov complexity of numeric series. We then show, in a third section, that some regularities, difficult to detect with traditional statistic tests, can be identified by compression algorithms. The last section is dedicated to the interests and limits of our techniques for real financial data analysis.

1. Regularity, randomness and Kolmogorov complexity

We start this section with traditional interpretations of randomness in finance in order to contrast them with their counterpart in computability theory. We then explicitly present the notion of “Kolmogorov complexity” and illustrate its empirical applications with simulated data.

Before our formal presentation of Kolmogorov complexity, readers might be interested by the solution to our initial question: “how to identify, among sequences A , B , C , D , E and F , the randomly generated one?”

The first four sequences (or strings) are *simple* in the sense of Kolmogorov, since they can be described by the following rules:

- obviously, A is a 40 times repetition of “01” .
- B is the so-called “Thue-Morse” sequence (see Allouche & Cosnard (2000), which is iteratively generated as follows:
 1. the initial digit is 0.
 2. the negation of “0” (*resp.* “1”) is “1” (*resp.* “0”) .
 3. to generate B in a recursive way, take the existing digits, and add their negations on the right:
 - Step 1: “0” ← “1” ⇒ “01”
 - Step 2: “01” ← “10” ⇒ “0110”
 - Step 3: “0110” ← “1001” ⇒ “01101001”
 - Step 4: “01101001” ← “10010110” ⇒ “0110100110010110”
 - ...
- C is the concatenation of natural numbers written in base 2: 0, 1, 10, 11, 100, 101, 110, 11 1000 It is known as the “Champernowne (1933) constant” .
- D is generated by transforming the decimals of π into binary digits.

These generating rules make A , B , C and D compressible by the corresponding algorithms. For D , common compression tools may turn out to be inefficient, but it is easy to write a π -compressing algorithm, in exploiting the generating function(s) of the famous constant.

E has been drawn from an uniform law which delivers, with a probability of 50% for each, 1 or 0. The last sequence (F) corresponds to the daily variations of the Dow Jones industrial index observed at closing hours from 07/10/1987 to 10/27/1987, with “0” coding the drops and “1” the rises.

One can question here to which extent F is similar to the first four strings, and to which degree lossless compression algorithms can distinguish them. This question serves as a baseline in this paper which attempts to provide a new consideration of randomness in financial dynamics.

1.1. Probabilistic versus algorithmic randomness

Traditionally, financial price motions were modeled (and are still modeled) as random walks⁴ (see equation 1):

$$p_t = p_{t-1} + \varepsilon_t \quad (1)$$

In this equation, p_t stands for the price of one specific security at “ t ” and ε_t is a stochastic term drawn from a certain distribution law⁵. This stochastic term is often estimated by a Gaussian noise, though it fails to correctly represent some stylized facts such as the time dependence in absolute returns or the auto-correlation in volatility (see for example Mandelbrot & Ness (1968) or Fama (1965)). Several stylized facts are taken into account by volatility models (see, for instance GARCH models⁶), while others stay rarely exploited in financial engineering (for example, multi-scaling laws (Calvet & Fisher 2002), return seasonality . . .).

In this sense, financial engineering proposes an iterative process which, step by step, improves the quality of financial time series models, both in terms of describing and forecasting⁷. At least from a theoretical point of view, this iterative process should deliver, at the end, a perfectly deterministic model⁸. If this final stage symbolizes the perfect comprehension of price dynamics, each step in this direction can be considered as a progress of our knowledge on financial time series. With an imperfect understanding of this latter, the iterative process will stop somewhere before the determinist model. Meanwhile, wherever we stop in this process, the resulting statistical model is a “compressed expression” (some symbols and parameters) of the long financial time series (hundreds or thousands of observations).

“To understand is to compress” is actually the main idea in Kolmogorov complexity (Kolmogorov 1965) whose definition is formally introduced as follows:

⁴These processes correspond to the so-called “Brownian motions” in continuous contexts. In this paper, as real-world financial data are always discrete and algorithmic tools only handle discrete sequences, our theoretical developments will lie in a discrete framework.

⁵Up to now, there is no consensus in the scientific community of finance concerning the driving law behind ε_t series.

⁶Generalized Auto Regressive Conditional Heteroscedasticity Models, Bollerslev (1986)

⁷It is clear that “description” is the easier part of the work, with forecasting being far more complicated.

⁸Which is simply impossible for the moment, because of our imperfect comprehension of financial time series.

Definition 1.1. Let s denote a finite binary string consisting of digits from $\{0, 1\}$. Its Kolmogorov complexity $K(s)$ is the length of the shortest program P generating (or printing) s . Here, the length of P depends on its binary expression in a so-called “universal language”⁹.

As a general indicator of randomness, Kolmogorov complexity of a given string should be relatively stable to the choice of universal language. This stability is shown in the following theorem (see (Kolmogorov 1965) or (Li & Vitányi 1997) for a complete proof) according to which, a change in the universal language has always a bounded impact on a string’s Kolmogorov complexity.

Theorem 1.2. Let $K_{L_1}(s)$ and $K_{L_2}(s)$ denote the Kolmogorov complexities of a string s respectively measured in two different universal languages L_1 and L_2 . The difference between $K_{L_1}(s)$ and $K_{L_2}(s)$ is always a constant c which is completely independent of s :

$$\exists c \forall s |K_{L_1}(s) - K_{L_2}(s)| \leq c \quad (2)$$

According to this invariance theorem, one need not pay too much attention to the choice of universal language in Kolmogorov complexity estimations, since its intrinsic value (the part depending on s) remains stable with regard to programming technics in use.

Besides the invariance theorem, another property of Kolmogorov complexity, firstly proved in (Kolmogorov 1965), makes it a good randomness measure: a finite string’s Kolmogorov complexity always takes a finite value.

Theorem 1.3. If s is a sequence of length n then:

$$K(s) \leq n + O(\log(n)) \quad (3)$$

In equation 3, $O(\log(n))$ only depends on the universal language in use. This property comes from the fact that all finite strings can at least be generated by the program, “print (s)”, whose length is close to the size of s .

Remark that most n -digit strings (provided that n is big enough) have their Kolmogorov complexity $K(s)$ close to n . $K(s) \ll n$ implies the existence of

⁹A universal language is a programming language in which one can code all “calculable functions”. “Calculability” is often related to the notion of “effectiveness” in computer science (see for example Velupillai (2004)). Most modern programming languages are universal (for example C, Java, R, Lisp . . .).

a program P which can generate s and is much shorter than s . In this case, P is said to have found a structural regularity in s , or P has compressed s .

On the contrary,

Definition 1.4. If for an infinite binary string denoted by S :

$$\exists C \forall n K(S \upharpoonright n) \geq n - C$$

where $S \upharpoonright n$ denotes the first n digits of S and C a constant in the natural number set \mathbb{N} , then S is defined to be a “random string”.

This definition of random strings, proposed by Martin-Löf (1966), reformulated by Chaitin (1987) and reviewed in Downey & Hirschfeldt (2010), is irrelevant to any probability notion, and only depends on structural properties of S .

Following definition 1.4, the best compression rate one can get from the n first digits of a random string is calculated by

$$rate = \frac{n - K(S \upharpoonright n)}{n} = \frac{C}{n}$$

As $n \rightarrow \infty$, $rate \rightarrow 0$. In other terms, random strings are incompressible ones.

Contra-intuitively, compressible strings are relatively seldom. For example¹⁰, among all n -digit strings, the proportion of those verifying $K(s) < n - 10$ does not exceed $1/1000$.

To illustrate the relation between compressibility and regularity, let’s consider two compressible (regular) strings:

a : 01101110010111011110001001101010
 111100110111101111100001000110010
 100111010010101
 b : 111110011110011001111011100111011
 0101010111011110111101101111111
 11101001101110

The first string, known as Champernowne sequence (i.e. string C on page 159), is generated with a quite simple arithmetic rule: it is the juxtaposition of natural numbers written in base 2. The optimal compression rate on the first million digits of is above 99% (c.f.

annex B), which indicates that can be printed by a short program.

The second string was independently drawn from a random law (with the help of a physical procedure) delivering “1” with a $2/3$ probability and “0” with a $1/3$ probability. The computer program generating b can be much shorter than it. In fact, classical tools can obtain on b a compression rate calculated as follows¹¹:

$$-(1/3)\log_2(1/3) - (2/3)\log_2(2/3) = 91.8\% \quad (4)$$

While b is drawn from a random law, its compressibility remains perfectly compatible with statistical conjectures: composed by definitively more 1 than 0, b should not be considered as the path of a standard random walk.

Face to rare events, algorithmic and statistical approaches also deliver similar conclusions. Let c denote a 100-digit string exclusively composed of 0. To deduce whether c was generated by tossing a coin, one can make 2 kinds of conjectures:

1. according to computability theory, c has a relatively negligible Kolmogorov complexity¹² and can not be the outcome of a coin tossing procedure.
2. in statistical terms, as the probability of obtaining 100 successive “heads” (or “tails”) by tossing a coin is not more than $7.889e - 31$, c is probably not the result of such a procedure.

However, despite the two preceding arguments, c could always have been generated by tossing a coin, since even with a zero probability, rare events do take place. This phenomenon is often referred to as “black swans” by financial practitioners.

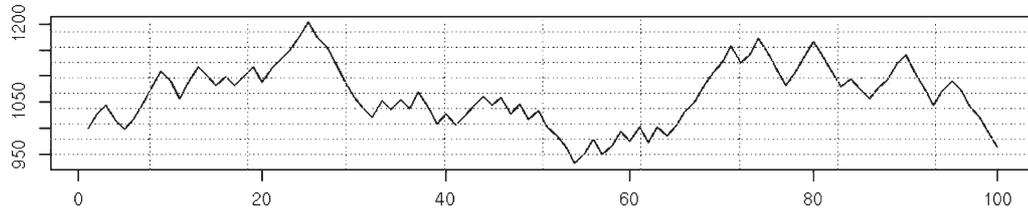
In this section, Kolmogorov complexity is presented by comparing it with the statistical framework in regard to randomness consideration. Two properties - invariance theorem and boundedness theorem - of the former concept are stressed in this theoretical introduction since they allow Kolmogorov complexity to be a good indicator of the distance between a given sequence and random ones.

Based on this theoretical development, we will show in the following sections how Kolmogorov complexity can be used in empirical data analysis.

¹⁰The probability to observe $K(s) < n - 20$ is less than $1/1000000$. Proof for this results are provided in appendix A.

¹¹This compression rate is related to the Shannon’s entropy of b .

¹²As c can be easily generated with a short command in most programming languages.

Fig. 1. Simulated series e_1 .

1.2. Kolmogorov complexity of price series: a basic example

After the theoretical introduction of Kolmogorov complexity, we illustrate with the following example how to search structural regularities in certain sequences behind their complex appearance. This example is developed with a simulated price series which is plotted in Figure 1. The first 14 values of the simulated series are:

$$e_{1,t} = 1000, 1028, 1044, 1015, 998, 1017, 1048, \\ 1079, 1110, 1090, 1058, 1089, 1117, 1100, \dots$$

One apparent regularity in $e_{1,t}$ is that all prices seem to be concentrated around 1000. One can “erase” this regularity on taking the first order difference of $e_{1,t}$ with equation: $e_{2,t} = p_t - p_{t-1}$. This process delivers the following sequence:

$$e_{2,t} = 28, 16, -29, -17, 19, 31, 31, 31, -20, \\ -32, 31, 28, -17, -17\dots$$

As positive series can be transformed into base 2 more easily, each element of $e_{2,t}$ is added by 32, which gives:

$$e_{3,t} = 60, 48, 3, 15, 51, 63, 63, 63, 12, 0, 63, 60, 15, 15, \dots$$

Then, $e_{3,t}$ is coded with binary numbers of 6 bits:

$$e_{4,t} = 111100, 110000, 000011, 001111, 110011, \\ 111111, 111111, 111111, 001100, 000000, \\ 111111, 111100, 001111, 001111, \dots$$

Without commas, $e_{4,t}$ becomes:

$$e_5 = 111100110000000011001111110011111 \\ 11111111111111001100000000111111\dots$$

Is e_5 compressible? In other terms, will there remain structures in e_5 after the preceding transformations? Before answering these questions, one must notice that

the binary expression of e_2 is already shorter than that of e_1 . Our first transformation actually corresponds to a first compression by exploiting the sequential, concentrated aspect of e_1 ¹³.

Were e_5 incompressible, our “regularity erasing procedure” (thereafter, REP) would have attained its final stage. However, it is not the case. In e_5 , there remains another structure which can be described as follows: if the $(2n - 1)$ th term is 1 (resp. 0), so will be the $(2n)$ th term. By exploiting this regularity, e_5 can be compressed to e_6 which only carries every second element of e_5 :

$$e_6 = 110100001011101111111111010000111 \\ 110011011\dots$$

At this step of REP, have we got an incompressible (random) string? Once again, this is not the case. e_6 is actually produced by a pseudo-random generator which theoretically, can be reduced to its seed! Thus, e_5 is compressible in principle. However, it should be admitted that this ultimate compression is almost infeasible in practice, given the available range of compression tools.

This basic example highlights how three important regularities¹⁴ can hide behind the rather complex appearance of $e_{1,t}$, and how these structures can be revealed by a mere deterministic procedure.

In the next sections, with the help of classical compression algorithms¹⁵, we show how REP can be implemented on both simulated (c.f. Section 3) and real-world financial time series (c.f. Section 3).

¹³ e_1 requires 56 decimal digits for the first 14 prices, whereas e_2 only needs 30. Of course, we have to store “1000” somewhere in e_2 to keep a trace of the first price. Measured in base 2, the length of e_1 turns from $4 \times 56 = 224$ bits into $4 \times 30 = 120$ bits, since each decimal digit ranging from 0 to 9, is coded with at least 4 bits.

¹⁴(i) Concentrated price values, (ii) particular structure in the binary digits and (iii) pseudo-random generator.

¹⁵We will use algorithms named Huffman, RLE and Paq808 which are presented in Annexe C.

In particular, we highlight that REP can be used to discover regularities actually undetectable by most statistical tests.

2. A methodological point

The punch-line of this paper is to show how lossless compression algorithms can be used to measure the randomness level of a numeric string. This requires a series of special transformations. Initial data must go through two consecutive operations: discretization and REP.

To concentrate on the relation between compression and regularity, in the last section, we show with a concrete example how REP can reveal structures in integer strings. However, to apply this latter procedure to real-world data, a discretization is necessary since financial returns are often considered to be Real numbers, while computer tools only work with discrete ones.

One can therefore query the effect of such discretization on financial series' randomness. To answer this question, we introduce the principle of lossless discretization.

2.1. The lossless discretization principle

In this section, we propose a generic methodology to estimate Kolmogorov complexity of a logarithmic price series¹⁶, as what one can observe in real-world financial markets. As mentioned in Section 1.2, a first "compression" consists of transforming the logarithmic price sequence into returns. This operation, familiar to financial researchers, actually delivers unnecessarily precise data: for example, no one would be interested by the 18th decimal place of a financial return. Thus, **without sensitive information loss**, one can transform real number series into integer ones by associating to each integer a certain range of real returns.

We also posit that integers used for discretization belong to a subset of \mathbb{N} whose length is arbitrarily fixed to be powers of 2. The total number of integers should be powers of 2, since this allows to re-code the discretization result in base 2 without bias¹⁷. Once written

¹⁶The reason why we use logarithmic price (p_t) here is that returns (r_t) in finance are usually defined as *continuously compounded returns*, i.e. $r_t = \ln(p_t) - \ln(p_{t-1})$

¹⁷Integer subsets fixed like this could be, for example, from 0 to 255 or from -128 to +127.

in binary base, financial returns become a sequence consisting of 0 and 1, which is perfectly suitable for compression tools.

After presenting the main idea of discretization, three points must be discussed concerning this latter procedure:

1. Why not compress real returns in base 10 directly?

Had price variations been coded in base 10 and directly written in a text file for compression, each decimal digit in the sequence would have occupied 8 bits (one byte) systematically. As decimal digits vary from 0 to 9 by definition, coding them with 8 bits would cause a suboptimal occupation of stocking space: only 10 different bytes would be used among the $2^8 = 256$ possibilities.¹⁸ Thus, even on random strings, spurious compressions would be observed because of the suboptimal coding.¹⁹ Obviously, this kind of compression is of no interest in financial series studies, and the binary coding system is necessary to introduce Kolmogorov complexity in finance.

2. Is there any information loss when transforming real-number innovations into integers? Obviously, on attaching the same integer, for instance "134", to two close variations (such as "1,25%" and "1,26%"), we reduce the precision of the initial data.

However, as mentioned above, the range of integers used in the discretization process can be chosen from $[0, 255]$, $[0, 511]$, \dots , $[0, 2^n - 1]$ to obtain the "right" precision level: neither illusorily precise nor grossly inaccurate.

This principle is shared by numerical photography: on choosing the right granularity of a picture, we stock relevant information and ignore some insignificant details or variations.

Following this principle, to preserve the fourth decimal place of each return, one needs to use integers ranging from 0 to 8192.

3. If each return is coded with one single byte²⁰, discretized financial series can be transformed into ASCII characters. Lossless compression algorithms such as Huffman, RLE and PAQ8o8 can

¹⁸Each byte can represent $2^8 = 256$ different combinations of 1 and 0. Thus, we can potentially get 256 different bytes in a text file.

¹⁹Low (used-bytes) / (possible-bytes) rate.

²⁰This implies that Real-number returns are discretized with integers ranging from 0 to 255.

be used to search patterns in the corresponding text files.

2.2. REP: an incremental procedure for pattern detection

As shown in Section 1.1, a generic methodology to capture financial dynamics requires to identify the underlying random process structuring ε_t (for example, *i.i.d.* Gaussian random innovations).

On introducing Kolmogorov complexity in finance, we want to consider whether it is possible or not to find, besides the well-documented stylized facts, other structures more subtly hidden in financial returns. In other terms, we try to find new and fainter regularities at the presence of more apparent ones.

If one uses compression methods on return series directly, algorithms would only catch the “main structures” in the data, and could potentially overlook weaker (but existing) patterns.

To expose these latters, we suggest an iterative process that, step by step, erases the most evident structures from the data. Resulting series are subject to compression tests for unknown structures. Due to this process, even if financial returns witness obvious patterns such as stylized facts, compression algorithms can always concentrate on unknown structures, since once erased, identified patterns would no longer be mixed with unknown structures, and any further compression could be related to the presence of new patterns.

To interpret the compression results from REP, we distinguish 3 situations:

1. The original series, denoted by s , is reduced to a perfectly determinist procedure that can be actually detected by compression tools. In this case, s is a regular series.
2. s is reduced to an incompressible “heart”. In this case, s is a random string.
3. s has a determinist kernel, but this latter is too complex to be exploited by compression tools in practice. In this case, s should be considered as random. For example, it is the case with normal series generated by a programming language like “R”²¹. REP’s first step is to erase the distribution law and to deliver a sequence produced by the programming language’s pseudo-random generator. This

generator, which is a perfectly determinist algorithm, is a theoretically detectable and erasable structure. Nevertheless, to our knowledge, no compression algorithm explores this kind of structures. So, in practice, a binary series drawn from an uniform law in “R” is considered as “perfectly random”, though this is not the case from a theoretical point of view.

Actually, given a finite time series, it is just impossible to attest whether it is a regular or random one with certainty. At least two arguments support this point:

1. As statistical conjectures are based on samples instead of populations, Kolmogorov complexity is estimated with finite strings. Thus, none can tell **in a certain way** if a finite string is a part of a random one.
2. The function attaching to s the true value of its Kolmogorov complexity cannot be calculated with a computer. In other terms, except in some very seldom case, one can never be sure if a given program P is the shortest expression of s .

So, compression tools only *estimate* Kolmogorov complexity, none of them should be considered as an ultimate solution that will detect all possible regularities in a finite sequence. This conclusion is closely related to the Gödel undecidability.

In this sense, compression algorithms share the same shortfall with statistical tests: both approaches are inept to deliver certain statements. Nevertheless, as statistical tests are making undeniable contributions to financial studies, compression algorithms can also be a powerful tool for pattern detection in return series.

To illustrate this point, we will compare algorithmic and statistical methods in the next section with regard to structure detection in numeric series.

3. Non-equivalence between algorithmic and statistical methods

In this section, we illustrate with simulated data how compression algorithms can be used to search regular structures. Statistical and algorithmic tools are compared in these illustrations to show how some patterns can be overlooked by standard statistical tests but identified by compression tools.

To be more precise, with simulated data we will show:

²¹R Development Core Team (2005)

Table 1
Unit root test for simulated returns

Test	Val. statistical	order	<i>p</i> -value
ADF	-32.7479***	31	0.01
PP	-178.52949***	16	0.01

H_0 : the simulated series has a unit root.

Table 2
Autocorrelation tests for simulated returns

Series	χ - square	deg. lib.	<i>p</i> -value
baseline	0.1838	1	0.668
	36.9157	36	0.4264

H_0 : the simulated series is *i.i.d.*.

Table 3
BDS test for simulated returns, $m = \{2, 3\}$

ε	0.5012	1.002	1.5035	2.0047
$m = 2$	-0.2082	-0.2987	-0.5232	-0.7221
<i>p</i> -value	0.8351	0.7651	0.6009	0.4702
$m = 3$	0.8351	0.7651	0.6009	0.4702
<i>p</i> -value	0.9503	0.9803	0.9344	0.8600

H_0 : the simulated series is *i.i.d.*.

1. how statistical and algorithmic methods perform on random strings,
2. how some statistically invisible structures can be detected by algorithmic tools,
3. and that compression tests have practical limits also.

3.1. Illustration 1: Incompressible randomness

In this section, we simulate a series of 32000 *i.i.d.* $N(0, 1)$ returns with the statistical language “R”, and examine the simulated data with statistical tests and compression algorithms respectively. We show that compression algorithms cannot compress random strings, and sometimes they even increase the length of the initial data.

The simulated series is quite similar to “consecutive financial returns” and can be used to generate an artificial “price sequence”. Figure 2 gives a general description of the data.

Statistical tests do not detect any specific structure in the normally distributed series, as we can witness in Tables 1, 2 and 3.

To check the performance of the algorithmic approach, we implement compression tools on the simulated data. As explained in the last section, before

using compression algorithms, we should firstly transform the simulated real-number returns into integers (discretization). In other terms, we must associate to each return an integer ranging from 0 to 255, with “0” and “255” corresponding to the lower and upper bounds of the simulated data respectively. Here, instead of a “one to one” correspondence, each integer from 0 to 255 must represent a range of real numbers. The size of the interval associated to each integer (denoted by e) is fixed as follows:

$$e = (M - m)/256 \quad (5)$$

where M and m represent the upper and lower bounds of the simulated returns.

Provided the value of e , we can divide the whole range $[m, M]$ into 256 intervals, and associate, to each return (denoted by x), an integer k satisfying:

$$x \in [m + (k - 1) \times e, m + k \times e[\quad (6)$$

In other terms, after sorting the 256 “ e -sized” intervals in ascending order, k is the rank of the one containing x . As k varies from 0 to 255, it should be coded with 8 bits, since $2^8 = 256$. Here, by dividing the real set $[m, M]$ into 256 identical subsets, we will obtain a series of regular bounds on X-axis under the standard normal distribution curve, as represented by Fig. 3(a).

Using this discretization method, we obtain normally distributed integers: values close to 128 are much more frequent than those close to 0 or 255. An efficient compression algorithm will detect this regularity and compress the discretized data.

However, this normal-law-based compression is of little interest for financial return analysis. According to REP, the normal distribution should be erased from the simulated data in order to expose more sequential structures.

To do this, we propose a second discretization process that will deliver uniformly distributed integers, instead of normally distributed ones.

In this second process, the main idea stays the same: real number returns can be discretized in dividing the whole interval $[m, M]$ into 256 subsets²², and then in associating to each return x the rank of the subset containing x .

²²Here, one can choose a power of 2 large enough to preserve all necessary information in the initial sequence. This is possible since the initial data have a limited precision.

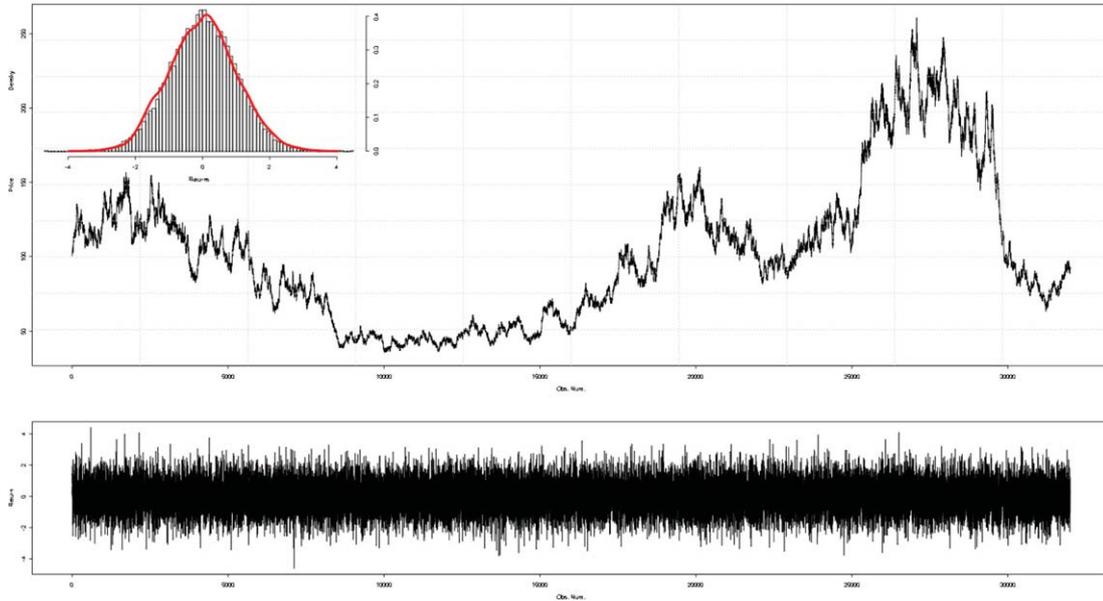


Fig. 2. Return series simulated from the standard normal distribution and the price motion corresponding to the simulated returns. Top-left: histogram of simulated returns. Middle: price sequence generated from simulated returns, with an initial price of 100. Bottom: time series plot of simulated returns.

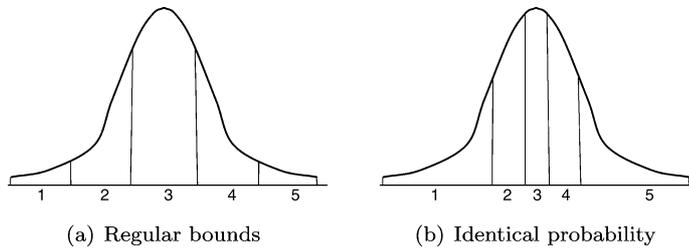


Fig. 3. Data discretization: how to place bounds?

However, this time, instead of getting equal-size subsets, we will fix the separating bounds in such a way that probability surface of the normal distribution would be divided into equal parts (c.f. Fig. 3(b)). To be more precise, we want to define 257 real-number bounds, denoted by $flag(0), flag(2), \dots, flag(256)$, to make sure that each value draw from $N(0, 1)$ has the same probability ($1/256$) to fall in each interval $[flag(i), flag(i + 1)]$. Subsets defined like this don't have the same size: as illustrated in Fig. 3(b), the more a subset is close to zero, the smaller it becomes.

The discretization process described in Fig. 3(b) is implemented to simulated returns. Discretized returns are presented in Fig 9. The i th point in this figure represents the integer associated to the i th return. One can notice that the plot of the discretized returns is rela-

tively homogeneous without any particularly dense or sparse area (see Figure in appendix D).²³ This rapid visual examination confirms the uniform distribution of discretized returns.

When lossless compression algorithms are used on the ascii text obtained from the uniformly discretized returns (c.f. Fig. 10 in appendix D), these algorithmic tools turn out to be inefficient (c.f. Table 4).

This example clearly shows that normally distributed returns can be transformed into an uniformly distributed integer string whose length is entirely irreducible by lossless compression tools. We can conjecture from this experiment, that the initial series has a Kolmogorov complexity close to its length.

²³It will be shown that the plot of discretized returns is not always as homogeny as this, especially when the initial data are not *i.i.d.*

Table 4
Compression tests

Algorithm	file size	compression rate
Huffman	32000	0%
Gzip	32502	-1.57%
PAQ8o8	32073	-0.23%
	32118	-0.37%

Interpretation: *discretized returns seem to be incompressible.*

With data simulated from the standard normal law, we show that statistical and algorithmic methods deliver the same conclusion on random strings.

In a further comparison between the 2 approaches, we will “hide” some structures in an uniformly distributed sequence, and show that the hidden regularity is detectable by compression algorithms but not by statistical tests.

3.2. Illustration 2: Statistically undetectable structures and compression algorithms

In this section, we make a further comparison between statistical and algorithmic methods with simulated series. More precisely, instead of random strings, we generate structured data to check the power of these two methods in pattern detection.

A return series can carry many types of structures. Some of them are easily detectable by standard statistical tests (for example, auto-regressive process or conditional variance process). However, to distinguish compression algorithms from statistical tests, this kind of regularities will not be the best choice. Here, we want to build statistically undetectable regularities that can be revealed by compression tools.

In this purpose, the return series is simulated as follows:

1. Draw 32000 integers from the uniform distribution $unif\{0, 255\}$ ²⁴, and denote by *text* the sequence containing these integers. *text* is then submitted to several transformations to “hide” statistically undetectable regularities behind its random appearance.

Let *text'* denote the biased series which should be distinguished from the uniformly distributed one, *text*.

text' is obtained from *text* by changing the last digit (*resp.* last 3 digits) of the binary expression of each integer in *text*.

²⁴ $unif\{0, 255\}$ denotes a discret uniform distribution on the integers 0,1,2,..., 255.

To be more precise, in case 1, *text'* exhibits an alternation between 0 and 1 on the last digit of each term. In case 2, elements from *text'* repeat the cycle 000, 001, 010, 011, 100, 101, 110, 111 on the last 3 digits.

For example, in coding each integer in *text'* with 8 bits, we could get, in case 1, a sequence as follows:

$$\{0000000\underline{1}, 0001101\underline{10}, 1110100\underline{1}, \\ 10000111\underline{0}, 10000111\underline{1}, \dots\}$$

This regularity is actually a “parity alternation”, since binary numbers ended by 1 (*resp.* 0) are always odd (*resp.* even).

In case 2, we could get a sequence like:

$$\{0101000\underline{0}, 1110100\underline{01}, 0110101\underline{0}, \\ 0110101\underline{1}, \dots, 1110011\underline{1}, 0001100\underline{0}\}$$

2. The biased integer sequence *text'* is what we want to obtain after the discretization of real number returns. So, the second step of our simulation process is to transform *text'* into return series.

In other terms, we should associate to each integer in *text'* a real number return. This association is based on the separating bounds calculated in the last section (c.f. $flag(0)$, $flag(2)$, ..., $flag(256)$ described in paragraph 1). To each integer in *text'*, denoted by $text'[i]$, we attach a real number that is independently drawn from the uniform law $unif\{flag(text'[i]), flag(text'[i] + 1)\}$. Let *chron* denote the return series obtained from this transformation.

By construction, *chron* has two properties:

- (a) globally, its terms follow a normal law;
- (b) if we discretize *chron* with the process described in paragraph 3.1, the resulted integer sequence will be exactly *text'*.

So, by construction, *chron* is a normally distributed series that will exhibit patterns after discretization. Are these structures detectable by both statistical and algorithmic methods? Or only one of these approaches can reveal the hidden regularities? That's what we want to see with the following tests.

Simulated returns are plotted in Fig. 4, with Fig. 4(a) corresponding to case 1, and Fig. 4(b) to case 2.

As shown in Tables 5, 6 and 7, statistical tests detect no structure in *chron*.

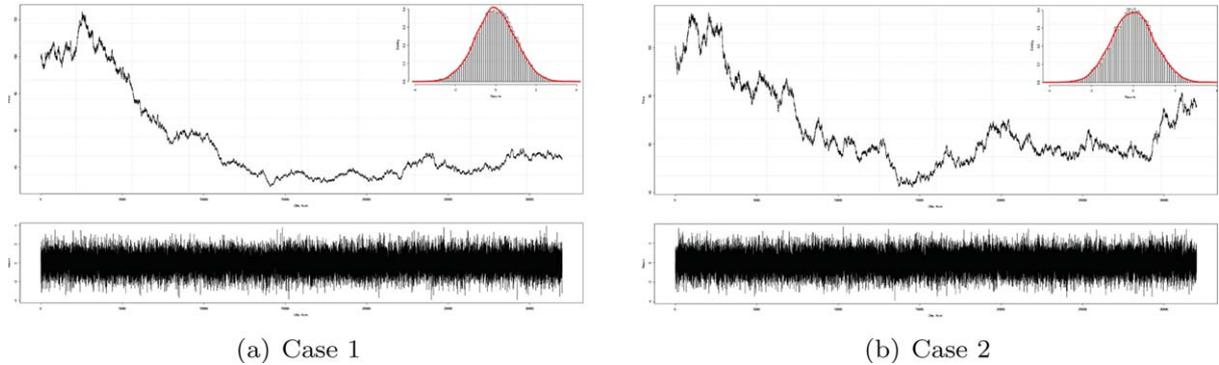


Fig. 4. Simulated returns with two important structures. On the top of each figure, we plotted the pseudo price series obtained from *chron* with an initial price of 100. At the bottom, we plotted the simulated return series *chron*.

Table 5
Unit root tests for *chron*

Test	case 1			case 2		
	Val. statistic	order	<i>p</i> -value			
ADF	-31.5598***	31	0.01	-31.2825***	31	0.01
PP	-178.797***	16	0.01	-179.9449***	16	0.01

H_0 : *chron* has a unit root.

Table 6
Autocorrelation tests for *chron*

case1			case2		
χ - square	deg. lib.	<i>p</i> -value	χ - square	deg. lib.	<i>p</i> -value
0.0096	1	0.9219	1.1169	1	0.2906
29.6655	36	0.7629	45.4802	36	0.1337

H_0 : *chron* is not autocorrelated.

After statistical tests, the REP is applied to *chron*. From a theoretical point of view, regularity hidden in case 1 implies a compression rate of 12.5%. This rate is calculated as follows: each integer in the discretized *chron* (which is actually *text'*) is coded with 8 bits, while only 7 of them are necessary. Actually, given the “parity alternation”, the last digit of each byte in *text'* is determinist. In other terms, we can save 1 bit on

Table 8
Compression tests

Algorithm	case 1		case 2	
	file size	compression rate	file size	compression size
	32000	0%	32000	0%
Huffman	31235	2.39%	23079	27.88%
Gzip	31322	2.12%	23160	27.63%
PAQ8o8	28296	11.58%	20974	34.46%

Interpretation in case 1 : *text'* is compressible. Interpretation in case 2 : *text'* is compressible.

every byte. Whence the theoretical compression rate $1/8 = 12.5\%$. Following the same principle, we can calculate the theoretical compression rate in case 2: $3/8 = 37.5\%$.

Table 8 presents compression results in the 2 cases. Notice that realized compression rates are close to theoretical ones, but they never attain their exact value. Among the three algorithms in use, *Paq8o8* offers the best estimator of theoretical rates.

In this illustration, we show that the algorithmic approach, essentially based on Kolmogorov complexity, can sometimes identify statistically-undetected structures in simulated data.

Table 7
BDS tests for *chron*, $m = \{2, 3\}$

case 1					case 2				
ε	0.5006	1.0012	1.5019	2.0025	ε	0.4988	0.9976	1.4964	1.9952
$m = 2$	0.1121	0.1790	0.3377	0.4318	$m = 2$	0.1886	0.1186	0.0627	0.1207
<i>p</i> -value	0.9108	0.8579	0.7356	0.6659	<i>p</i> -value	0.8504	0.9056	0.9500	0.9039
$m = 3$	0.1329	0.2233	0.3578	0.4662	$m = 3$	-0.0974	-0.0617	0.0336	0.2153
<i>p</i> -value	0.8943	0.8233	0.7205	0.6411	<i>p</i> -value	0.9224	0.9508	0.9732	0.8295

H_0 : *chron* is *i.i.d.*

However, as mentioned in the theoretical part, the ultimate algorithm calculating the true Kolmogorov complexity for all binary strings **does not** exist. Compression tools also have practical limits. In other terms, certain structures are undetectable by available compression tools. We show this point in the next section.

3.3. Practical limits of the algorithmic method: decimal digits of π

Besides the Euler numbers and the Fibonacci numbers, π is perhaps one of the most studied mathematic numbers. There are many methods to calculate π . The following two equations both deliver a big number of its decimal digits:

- Leibnitz-Madavar’s formula,

$$\pi = 4 \times \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \tag{7}$$

- And the second formula :

$$\pi = \sqrt{6 \times \left(1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots + \frac{1}{n^2}\right)} \tag{8}$$

π can be transformed into a return series in 3 steps:

1. Each decimal digit of π is coded in base 2 with 4 bits. For example, the first 4 decimals (c.f. 1, 4, 1, 5) become 0001, 0100, 0001, 0101. Following this principle, the first 50000 decimals of π correspond to 200000 bits of binary information.
2. The 200000-digit binary string obtained from the first step is then re-organized in bytes. For example, the first 4 decimals make two successive bytes: 00010100, 00010101. Each byte corresponds to a integer ranging from 0 and 255. Here, the first two bytes of π become 20 and 21. Denote by π' the integer sequence after this re-organization.
3. Finally, we associate a real-number return to each integer in π' . To do this, we follow the same principle as in the last section: to each term of π' , denoted by π'_i ($t \in [1, 25000]$), we associate a real number that is independently drawn from the uniform distribution $unif\{flag(\pi'_i), flag(\pi'_i + 1)\}$. Where $flag(i)$ ($i \in [0, 256]$) means the i^{th} separating bound obtained from the uniform discretization of the normally distributed return

Table 9
Compression test: π

Algorithm	case 1	
	file size	compression rate
Huffman	12500	100%
Gzip	12955	-3.64%
PAQ8o8	12566	-0.528%
	12587	-0.70%

Interpretation: We can't compress the π based return series.

Table 10
Unit root test for the series constructed by π

Test	Val. statistical	order	p-value
ADF	-23.3799***	23	0.01
PP	-110.1364***	13	0.01

H_0 : the π based series has a unit root.

Table 11
Autocorrelation tests

$\chi - square$	deg. lib.	p-value
2.7339	1	0.09824

H_0 : the π based series is not autocorrelated.

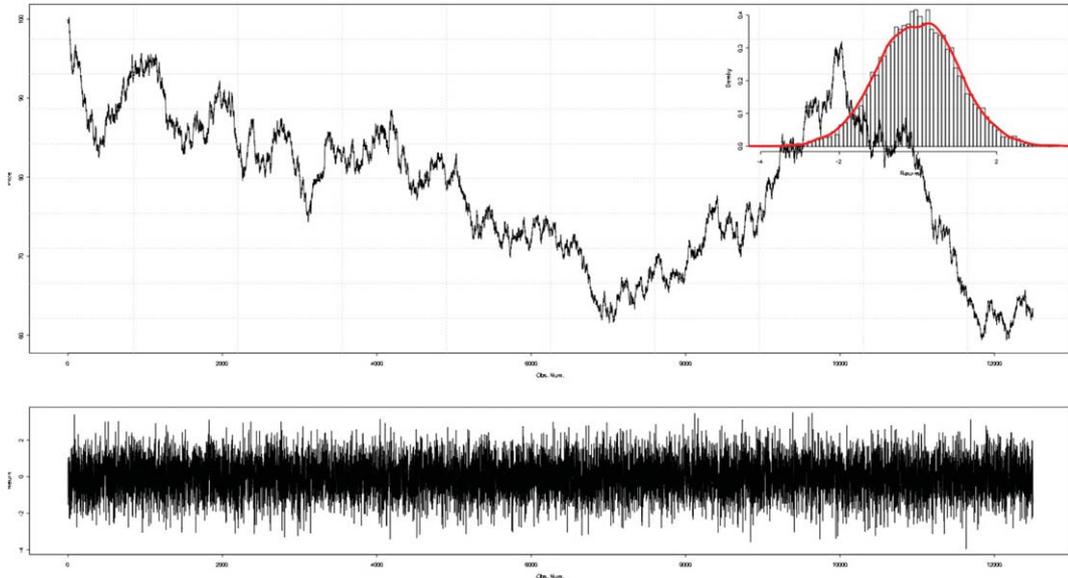
series in Section 3.1. After this step, we get a pseudo return series plotted in Fig. 5.

As exposed in Table 9, π -based return series is incompressible after discretization, since to our knowledge, no compression algorithm exploits decimals of π .

The π -based simulation is another example that shows the possibility to hide patterns behind a random appearance. It also witnesses that some theoretically compressible structures may be overlooked by available compression tools. These structures are perfectly compressible in theory, but not in practice yet.

To check the performance of statistical tools on decimals of π , we conducted the same tests as in the preceding illustrations. We notice in Tables 10, 11 and 12, that statistical tests do nothing better than compression tools: none of them can reject H_0 which implies the absence of regularity.

In this section, simulated data are used to illustrate the performance of compression tools in pattern detection. Two main results are supported by these illustrations: (1) Some statistically undetectable patterns can be traced by compression tools. (2) Certain structures stay undetectable by currently available compression tools.

Fig. 5. Pseudo financial time series generated from decimals of π .Table 12
BDS test for the π based series, $m = \{2, 3\}$

ε	0.5023	1.0046	1.5069	2.0092
$m = 2$	-0.0895	0.0468	0.0395	0.0129
p -value	0.9287	0.9627	0.9685	0.9897
$m = 3$	-0.4005	-0.1755	-0.2780	-0.3208
p -value	0.6888	0.8607	0.7810	0.7483

H_0 : the π based series is *i.i.d.*

In the next section, we test the algorithmic approach with real-world financial return series.

4. Kolmogorov complexity of real-world financial data: the case of Dow Jones industrial index

In this section, we estimate the Kolmogorov complexity of real-world financial returns with lossless compression algorithms. To do this, we use the logarithmic difference of Dow Jones daily closing prices observed from 01/02/1928 to 30/08/2010. Data used in this study are extracted from Datastream. Our sample containing 27423 observations is plotted in Fig. 6.

Following the REP, we uniformly discretize the real-number returns to prepare them for compression tests.

While separating bounds that are used to discretize simulated data in the preceding sections all come from the standard normal law, real-world returns cannot be

tackled in the same way, since it is well known that financial returns are not normally distributed. Actually, there is no consensus on the way financial returns are distributed in the financial literature. Therefore, to discretize Dow Jones daily returns, separating bounds ($flag(i)$) should be estimated from their empirical distribution.

Such an estimation can be realized in 3 steps:

1. sort the whole return series in ascending order,
2. divide the ascending sequence into 256 equally-sized subsets,
3. each return is represented by the rank of the subset containing it.

The advantage to estimate $flag(i)$ with this 3-step approach is that one can discretize a sample without making any hypothesis on the distribution law of the population.

Figure 7 is a plot of discretized Dow Jones daily returns.

One can notice in this figure that the uniform discretization fails to deliver a perfectly homogeneous image as in the case of normally distributed returns. Several areas seem to be sparser than the others. For instance, regions exposing returns from Time 9000 to Time 10000 and from Time 13000 to Time 20000 both present a great difference in point density.

This statement can be explained by the volatility clustering phenomenon, a well documented stylized

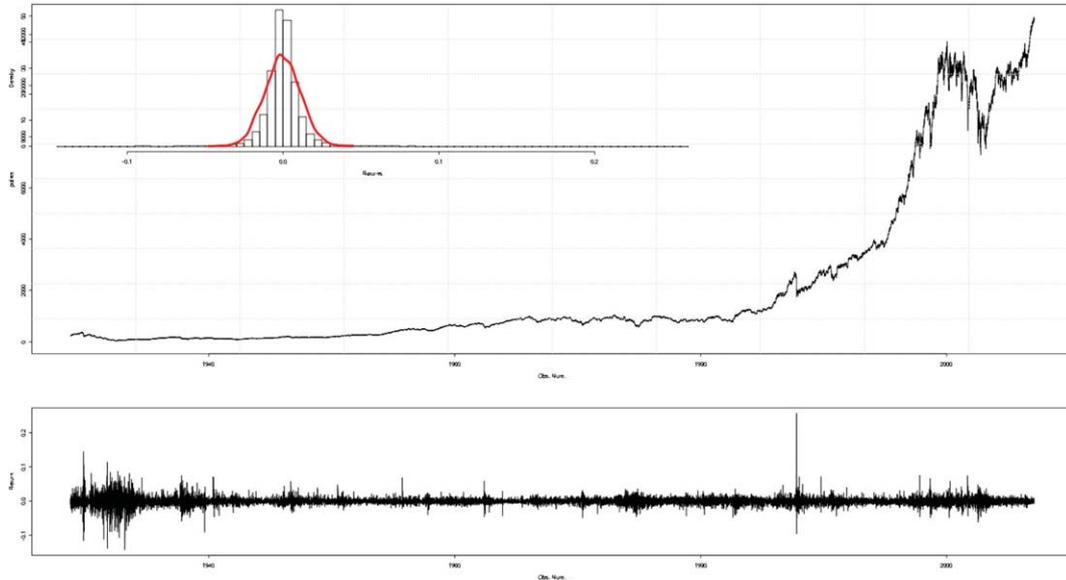


Fig. 6. Series constructed from Dow Jones daily closing prices. Top-left: histogram of logarithmic differences. Middle: Dow Jones daily closing prices observed from 01/02/1928 to 30/08/2010. Bottom: time series plot of the real-number returns.

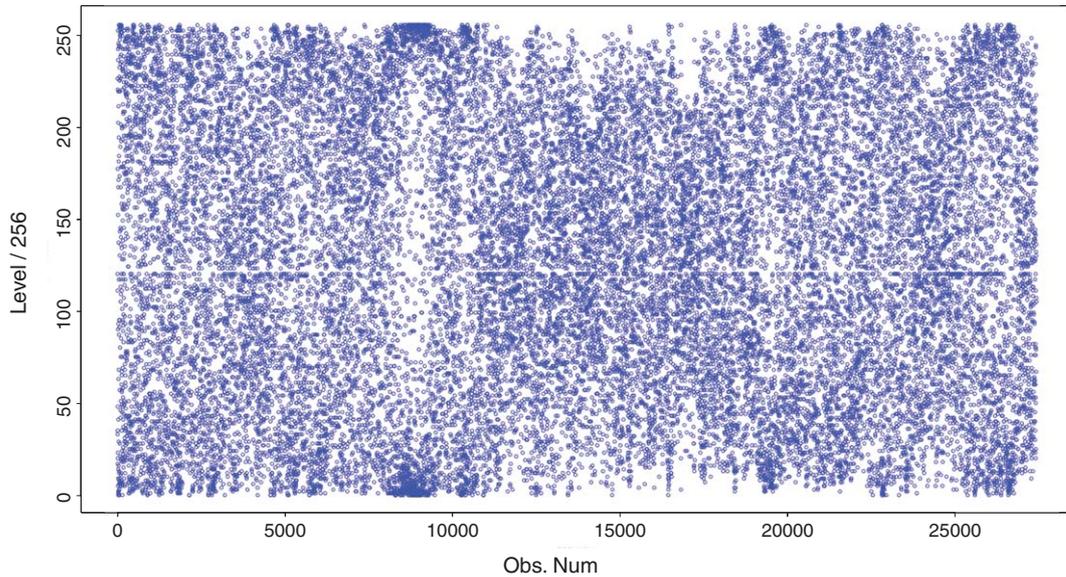


Fig. 7. Uniformly discretized Dow Jones daily returns.

fact in finance. For example, from Time 9000 to Time 10000, extreme values are obviously more frequent than those close to 0.

This eye-detectable structure is confirmed by compression tests. As we can witness in Table 13, algorithm PAQ8o8 obtains a 0.82% compression rate on the uniformly discretized series.

This compression rate appearing extremely weak, its robustness may appear doubtful. To test its significance, we simulate 100 integer sequences from an *i.i.d.unif*(0, 255) process, with each of them containing 27423 observations as the Dow Jones daily return series (thereafter DJ). These simulated sequences are tested by compression tools. PAQ8o8 delivers no

Table 13
Compression tests on discretized DJ

Algorithm	file size	compression rate
Huffman	27423	100%
Gzip	27456	-0.12%
PAQ8o8	27489	-0.24%
PAQ8o8	27198	0.82%

Interpretation: *Discretized Dow Jones is compressible by PAQ8o8.*

positive compression rate on none of the simulated series.

This result confirms the theoretical relation between regularity and compressibility: despite its weak level, the compression rate got from discretized DJ indicates the presence of volatility clusters in the data.

To advance another step in the REP and verify whether or not daily DJ returns contain other patterns than the witnessed stylized facts, we should remove the “volatility clustering” phenomenon with the help of reversible transformations.

In fact, in the uniform discretization process described above, each integer represents the same range of real returns. That is why highly (*resp.* lowly) volatile periods are marked by an over-presence (*resp.* a sub-presence) of extreme values in Fig. 7. To remove “volatility clusters” is to modify these heteronomous areas and to ensure that each part of Fig. 7 has the same point density.

The solution we propose here is to discretize DJ in a progressive way. To be more precise, instead of discretizing the entire series at once, we treat it block by block with an iterative procedure.

Denote by S_t the Dow Jones daily return series, the following 3-step procedure can erase clustering volatilities in S_t :

- To start up, a 512-return sliding window is placed at the beginning of S_t . Returns in the window (i.e. the first 512 ones) are transformed into integers with the above-described uniform discretization procedure. The integer associated to S_{512} is stocked as the first term of the discretized sequence.
- The sliding window moves one step to the right, returns in the window are discretized again, and the integer corresponding to S_{513} is stocked as the second term of the discretized sequence.
- Repeat the second step until the last return of S_t . The integer sequence obtained from this procedure doesn't reveal any volatility cluster. A comparison between Figures 8 and 7 illustrates this progressive procedure's impact.

How this iterative process erases volatility clusters? The main idea is to code each return in DJ with those appearing in its close proximity. In the above example, the sliding window's length is fixed to 512, this implies

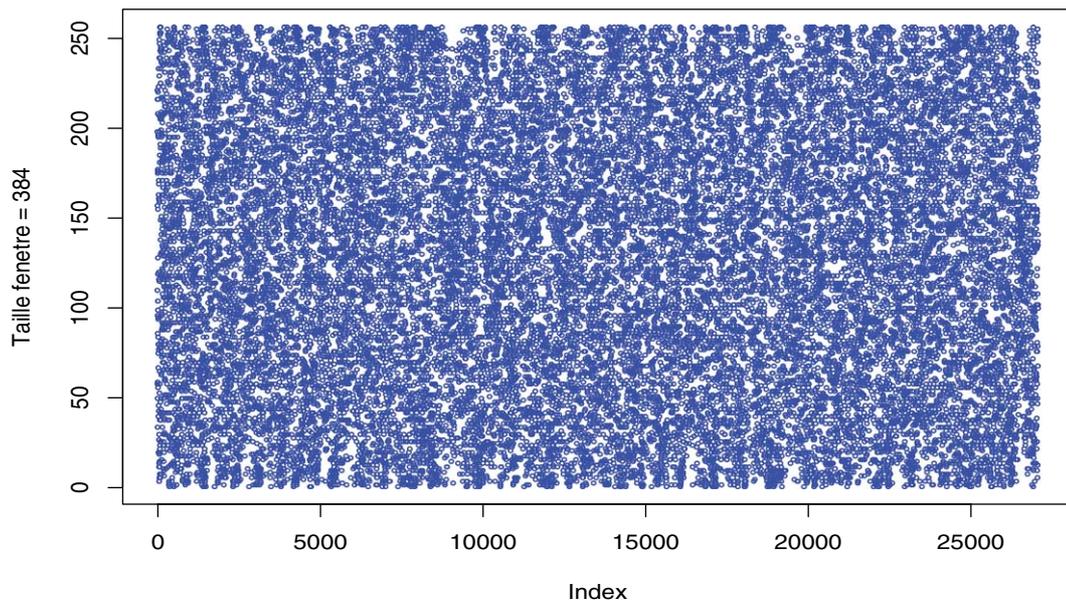


Fig. 8. DJ after the progressive discretization.

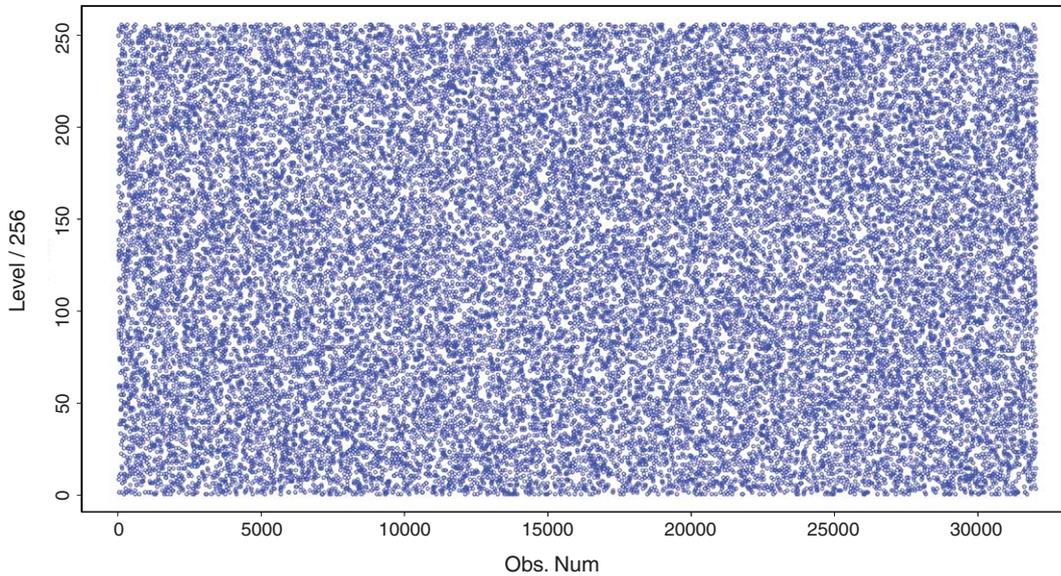


Fig. 9. Uniformly discretized returns in Section 3.1.



Fig. 10. Discretized returns expressed in ascii code.

that each return’s discretization only depends on the 511 preceding terms.

Therefore, each integer in the discretized sequence, for example 255, can represent a 10% rise during highly volatile periods as well as a 3% up during less volatile ones. Due to the progressive discretization, extreme values in volatile periods are “pulled” back to zero, and returns during less volatile periods will be “pushed” to extreme values.

Were the non-uniform distribution and volatility clusters the only regularities in DJ, the progressively discretized sequence (i.e. s_t) would be incompressible by algorithmic tools. Positive compression rates obtained on s_t would indicate the presence of unknown structures.

Table 14

Compression test: DJ after progressive discretization

Algorithm	file size	compression rate
	27039	100%
Huffman	27075	-0.12%
Gzip	27105	-0.24%
PAQ8o8	26913	0.27%

Interpretation: *Even after the progressive discretization process, DJ remains compressible by PAQ8o8.*

To check this, compression tests are conducted on s_t . Results are presented in Table 14:

We notice in this table that even after the progressive discretization process, DJ remains compressible by PAQ8o8. This result could indicate the presence

of unknown structures in financial returns. To better understand these structures, further research is necessary to identify their nature and tell how to remove them from s_t and advance once again in the REP.

However, as we can witness in Table 14, compression rate based on these unknown structures is extremely weak (c.f. 0.27%). This indicates a high similitude between s_t and a random string. In other terms, although not completely random, once stylized facts erased, Dow Jones daily returns have an extremely high Kolmogorov complexity.

In a certain degree, this result supports the EMH like most statistical works in finance (see for eg. Lo & Lee (2006)). Once unprofitable stylized facts are erased from DJ, the latter series is quite similar to a random string. The high Kolmogorov complexity observed in our study indicates to which extent it is difficult to find a practical trading rule that is outperforming the market in the long run.

5. Conclusion

In this paper, we propose a generic methodology to estimate the Kolmogorov complexity of financial returns. With this approach, the weak-form efficiency assumption proposed by Fama (1970) can be studied by compression tools.

We give examples with simulated data that illustrate the advantages of our algorithmic method : among others, some regularities that cannot be detected with statistical methods can be revealed by compression tools.

Applying compression algorithms to daily returns of the Dow Jones Industrial Average, we conclude on an extremely high Kolmogorov complexity and by doing so, we propose another empirical observation supporting the impossibility to outperform the market.

A limit of our methodology lies in the fact that currently available lossless compression tools, initially developed for text files, are obviously not particularly designed for financial data. They could consequently be restricted in their use for detecting patterns in financial motions. Future researches could therefore develop finance-oriented compression tools to establish a more direct link between a compression rate delivered from a return series and the possibility to use any possible hidden pattern to outperform a simple buy and hold strategy.

Another important point to be stressed in this paper lies in the fact that our methodology proposes an iterative process which will improve, step by step, our comprehension on the “stratified” structure of financial price motions (*i.e.* layers of mixed structures).

As illustrated in the empirical part of this research, even after removing some of the more evident stylized facts from real-world data, compression rates obtained from the Dow Jones daily returns seem to indicate the presence of unknown structures. Although these patterns could be difficult to exploit by strategy designers, at least from a theoretical point of view, it is challenging to understand the nature of these unknown structures. The next step could then consist in removing them from the initial data, if possible, and go one step further in the iterative process seeking at identifying new regularities. The ultimate goal of this process, even if it is probably a vast and perhaps quixotic project, could be to obtain an incompressible series, and, so to speak, to reveal layer after layer, the whole complexity of financial price motions.

A) Proportion of n -digit random strings

In base 2, each digit can be either 0 or 1. So, there are at most 2 different 1-digit binary strings. And more generally, at most 2^i i -digit binary strings. So, there are less than

$$2 + \dots + 2^{h-1} = 2^h - 2$$

different binary strings that are strictly shorter than h .

Then, the proportion of the binary strings, whose length can be reduced by more than k digits, cannot exceed

$$(2^{n-k} - 2)/2^n < 1/2^k.$$

With $k = 10$, at most $1/2^{10} = 1/1024$ of all n -digit binary strings can be compressed by more than 10 digits. With $k = 20$, this latter proportion cannot exceed $1/1048576$.

B) A generating program of the Champernowne’s constant

In this appendix, we present the program - written in “R”²⁵ - that generates digits of Champernowne’s constant.

²⁵<http://www.r-project.org/>

```

> c<-0; j<-0 ; d<-0
> for (n in 1:10000)
> {
> a<-n;k<-0
> while (a!=0) {k<-k+1;d[k]<-a%%2;a<-(a-d[k])/2}
> for (h in 1:k) {j<-j+1; c[j]<-d[k-h+1]}
> }

```

This program delivers the first 123631 digits of Champernowne's number, while it is written with $132 \times 8 = 1056$ digits. In other terms, this program realizes a 99.145% compression rate.

C) Lossless compression algorithms

Although the intrinsic value of Kolmogorov complexity remains stable to programming technics, the presence of the constant “ c ” in the invariance theorem (see page 5) can modify the compression rate one can obtain on a finite string. Thus, our choice of compression tools should be as large as possible to estimate the shortest expression of a given financial series.

From a technical point of view, there are 2 categories of lossless compression algorithms:

1. Entropy coding algorithms reduce file size on exploiting the statistical frequency of each symbol. Two technics are often used in this purpose:
 - Huffman coding: one of the most traditional text compression algorithms. According to this approach, the more frequent is a given symbol, the shorter will be its corresponding code in the compressed file. Following this principle, Huffman coding is particularly powerful on highly repetitive texts.
 - Dictionary coding: another statistical compression technic used by a big family of recent tools, such as Gzip, LZ77/78, LZW. This approach consists to construct a one-to-one correspondence between words in the initial text and their code in the compressed file, a so-called “dictionary”.

Then, according to the dictionary, each word is “translated” into a short expression. A positive compression rate can be observed if all words in the initial text don't have the same appearance frequency. Compared to Huffman coding algorithms, dictionary used in this approach can be modified during the compression procedure. Therefore, dictionary coding

algorithms exploit local properties with more efficiency.

2. Context-based predicting algorithms compress data by forecasting future terms with historical observations. With the development of artificial intelligence, this context-based approach become more and more performant on self-dependent data. A big number of technics can be used for data prediction, such as Prediction by Partial Marching (PPM), Dynamic Markov compression (DMC).

PAQ²⁶ is one of the most remarkable family of context-based compression algorithms. These algorithms are reputed by their exceptional compression rates. On choosing the PAQ version, one can attach more or less importance to the speed of a compression procedure.

PAQ808 is a recent version of PAQ which maximizes the compression rate at the expense of speed and memory. It's a predicting algorithm which, on analyzing the first t digits of a finite string, predicts the appearance probability of each possible symbol at the next digit. The $(n + 1)$ th digit is coded according to these conditional probabilities.

In this paper, we tested the performance of the 3 above-cited compression technics on simulated data as well as on Dow Jones daily returns, and reported the best compression rate obtained by each category of tools. PAQ808 delivers by far the best compression rate on all discretized series.

References

- Allouche, J.-P., Cosnard, M., 2000. The komornik-loreti constant is transcendental. *Amer Math Monthly* 107.
- Azhar, S., et al., 1994. Data compression techniques for stock market prediction. In *Data Compression Conference, 1994. DCC '94. Proceedings*, pp. 72–82.
- Bollerslev, T., 1986. Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 307–327.
- Brandouy, O., Delahaye, J.-P., 2005. Kolmogorov complexity of financial time series. Unpublished technical report, University of Lille 1.
- Calvet, L., Fisher, A., 2002. Multifractality in asset returns: Theory and evidence. *Review of Economics and Statistics* 84, 381–406.

²⁶Maintained by Matt Mahoney on the website <http://cs.fit.edu/mmahoney/compression/>

- Chaitin, G., 1987. *Algorithmic Information Theory*. Cambridge University Press. Cambridge Tracts in Theoretical Computer Science 1.
- Champernowne, D., 1933. The construction of decimal normal in the scale of ten. *J London Math Soc* 8.
- Chen, S.-H., Tan, C.-W., 1996. Measuring Randomness by Rissanen's Stochastic Complexity: Applications to the Financial Data, chap. Information, Statistics and Induction in Science, World Scientific, pp. 200–211.
- Chen, S.-H., Tan, C.-W., 1999. Estimating the complexity function of financial time series: An estimation based on predictive stochastic complexity, *Journal of Management and Economics* 3(3).
- Cilibrasi, R., Vitányi, P., 2005. Clustering by compression. *IEEE Transactions on Information Theory*.
- Dionisio, A., et al., 2007. Entropy and uncertainty analysis in financial markets. *Time* 1937(1937), 1–9.
- Downey, R.G., Hirschfeldt, D.R., 2010. *Algorithmic Randomness and Complexity*. Theory and Applications of Computability. Springer.
- Fama, E., 1965. Behavior of stock market prices. *Journal of Business* 38.
- Fama, E., 1970. Efficient capital markets: A review of theory and empirical work. *Journal of Finance* 25.
- Giglio, R., et al., 2008. Algorithmic complexity theory and the relative efficiency of financial markets. *EPL (Europhysics Letters)* 84(4), 48005.
- Gödel, K., 1931. On Formally Undecidable Propositions of The Principia Mathematica and Related Systems (I.). *Manstshefte für Mathematik und Physik*.
- Griffith, T., Tenenbaum, J., 2001. Randomness and Coincidences: Reconciling Intuition and Probability Theory. 23th Annual Conference of the Cognitive Science Society.
- Groth, S.S., Muntermann, J., 2011. An intraday market risk management approach based on textual analysis. *Decision Support Systems* 50(4), 680–691.
- Kolmogorov, A.N., 1965. Three approaches to the quantitative definition of information. *Prob Info Trans* 1(1), 3–11.
- Lempel, A., Ziv, J., 1976. On the complexity of finite sequences. *IEEE Transactions on Information Theory* 22(1), 75–81.
- Li, M., Vitányi P., 1997. *An Introduction of Kolmogorov Complexity and its Implications*. Springer-Verlag, 2 edn.
- Lo, M., Lee, C.-F., 2006. A reexamination of the market efficiency hypothesis: Evidence from an electronic intra-day, inter-dealer FX market. *The Quarterly Review of Economics and Finance* 46(4), 565–585.
- Ma, L., 2010. Structures et hasard en finance: une approche par la complexité algorithmique de l'information. Ph.d. thesis, University of Lille 1.
- Mandelbrot, B., Ness, J.W.V., 1968. Fractional Brownian Motions, Fractional Noises and Applications. *SIAM Review* October.
- Martin-Löf, P., 1966. On the Concept of a Random Sequence. *Theory Probability Appl.*
- R Development Core Team, 2005. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
- Rissanen, J., 1986. Stochastic complexity and modeling. *The Annals of Statistics* 14(3).
- Shmilovici, A., et al., 2003. Using a stochastic complexity measure to check the efficient market hypothesis. *Computational Economics* 22(2), 273–284.
- Shmilovici, A., et al., 2009. Measuring the efficiency of the intraday forex market with a universal data compression algorithm. *Computational Economics* 33, 131–154.
- Silva, S.D., et al., 2008. The relative efficiency of stockmarkets. *Economics Bulletin* 7(6), 1–12.
- Turing, A.M., 1937. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society Series* 2(42), 230–265.
- Velupillai, K.V., 2004. Economic dynamics and computation-resurrecting the icarus tradition. *Metroeconomica* 55(2-3), 239–264.
- Zenil, H., Delahaye, J.-P., 2011. An algorithmic information-theoretic approach to the behaviour of financial markets. *Journal of Economic Surveys* 25.
- Zurek, W.H., 1989. Algorithmic Randomness and the Physical Entropy. *Physical Review A*.