Ordo ab Chao

William F. Barnes

June 21, 2020

Contents

1 Introduction

The notion of *randomness* naturally arises in systems having high disorder or unpredictability. In the purest sense, true randomness is justified only by quantum mechanics, however this does not prohibit classical systems from exhibiting unpredictable behavior. We thus take the definition of 'random' to mean 'disorder' in the most general sense.

Any event that generates a random data point is called *stochastic* process. Repeated stochastic processes, such as rolling a dice or drawing a card from a shuffled deck, generate random sequences of data. When relying on a computer to generate random sequences, it's well-known that care must be taken to assure that a sequence is free of structure, repetition, bias, etc.

Various tools exist for checking the validity of random sequences. These include the average test, the deviation test, chi-square analysis, and so on. While each test thrives in its own domain, certain datasets can fool them. In this study we outline a general protocol for detecting order or repetition in sequences, and demonstrate the method on several certain special cases.

2 Sequences

2.1 Notation

Let us officially define a *sequence* $S = S^N(\phi_m)$ as an ordered list of N members, where each member belongs to a closed set ϕ_m consisting of m unique elements. For instance, the binary sequence

$$S^{10}(\phi_2) = (\ 0\,1\,1\,0\,0\,0\,1\,0\,1\,1\ )$$

has ten total members, each choosing from elements 0 or 1. The list of elements forms an element 'vector', denoted ϕ_2 such that

$$\phi_2 = (0, 1)\ .$$

For another example, a list of 16 of random integers from 0 to 9 may occur as

$$S^{16}(\phi_{10}) = (\ 7\,1\,1\,2\,4\,0\,1\,6\,1\,2\,9\,2\,5\,8\,0\,3\ )\ ,$$

with corresponding element vector

$$\phi_{10} = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) = (0, \ldots, 9)\ .$$

Individual Members

The jth member in the sequence $S^N(\phi_m)$ is denoted

$$S_j^N(\phi_m)\ ,$$

where $1 \leq j \leq N$.

Individual Elements

The jth element in the element vector ϕ_m is denoted ϕ_m^j such that $1 \leq j \leq m$.

For a concrete example, recall the ϕ_2 element vector has two elements $\phi_2 = (0, 1)$. Each element in ϕ_2 reads:

$$\phi_2^1 = 0 \qquad\qquad \phi_2^2 = 1$$

By virtue of the base-ten number system, it turns out that ϕ_{10} is extremely easy to work with, as the jth element is equal to $j - 1$:

$$\phi_{10}^1 = 0 \qquad \phi_{10}^2 = 1 \qquad \cdots \qquad \phi_{10}^{10} = 9$$

All ϕ Notation

In terms of ϕ-notation alone, a sequence $S^N(\phi_m)$ may be expressed as

$$S^N(\phi_m) = S_1^N(\phi_m)\ S_2^N(\phi_m)\ S_3^N(\phi_m)\ \cdots\ S_N^N(\phi_m)\ .$$

In this form, it's clear we could branch this study into an exploration of varying m in each ϕ-term, potentially leading to very wild sequences. For now we shall stick a mental flag here and otherwise stay on target by leaving m alone.

2.2 Element Regrouping

To proceed, define a modified element vector $\phi_{m,n}$ that lists all n-order arrangements of the m original elements, a list of size n^m. For example, the second-order regrouping of ϕ_2 is

$$\phi_{2,2} = (00, 01, 10, 11)\ ,$$

which has $2^2 = 4$ total elements. Similarly, the second-order regrouping of ϕ_{10} reads

$$\phi_{10,2} = (00, 01, 02, \ldots, 97, 98, 99)\ ,$$

containing $10^2 = 100$ elements. A third-order regrouping of ϕ_2 must have 2^3 elements, easily verified as

$$\phi_{2,3} = (000, 001, 010, 011, 100, 101, 110, 111)\ .$$

A third-order regrouping of ϕ_{10} contains 10^3 elements

$$\phi_{10,3} = (000, 001, 002, \ldots, 997, 998, 999)\ ,$$

and so on.

Regrouped Elements

The jth element in the regrouped element vector $\phi_{m,n}$ is denoted $\phi_{m,n}^j$ such that $1 \leq j \leq n^m$.

Continuing the ϕ_2 example, recall the ϕ_2 vector with second-order regrouping has four elements $\phi_{2,2} = (00, 01, 10, 11)$. Each element in $\phi_{2,2}$ reads:

$$\phi_{2,2}^1 = 00 \qquad \phi_{2,2}^2 = 01 \qquad \phi_{2,2}^3 = 10 \qquad \phi_{2,2}^4 = 11$$

Like the ϕ_{10} case, it turns out that elements $\phi_{10,2}$ also follow behind the j-index by one:

$$\phi_{10,2}^1 = 00 \qquad \phi_{10,2}^2 = 01 \qquad \cdots \qquad \phi_{10,2}^{100} = 99$$

3 Sequence Manipulation

3.1 Translation via Regrouping

We shall proceed by exploiting the observation that a sequence S^N can be expressed via regrouped element vectors without using losing information or changing the sequence.

Second-Order Regrouping

Starting with an easy example, a series having regrouped element vector $\phi_{m,2}$ can be written

$$S^N\left(\phi_m\right) = \left(S_1^{N/2}\left(\phi_{m,2}\right)\ S_2^{N/2}\left(\phi_{m,2}\right)\ S_3^{N/2}\left(\phi_{m,2}\right)\ \cdots\ S_{N/2}^{N/2}\left(\phi_{m,2}\right)\right) + \mathcal{R}^1\left(\phi_m\right)\ ,$$

where the parenthesized sequence contains half of the number of elements as the original sequence. The remaining $\mathcal{R}$-term equals zero of S^N contains an even number of terms, but contains one element from ϕ_m if N is odd (the last number has no match).

For a concrete example, consider a random binary series with ten terms S_2^{10} that admits a translation with no remainder:

$$\begin{aligned} S^{10}\left(\phi_2\right) &= (\ 0\,1\,1\,0\,0\,0\,1\,0\,1\,1\ ) \\ S^{10}\left(\phi_{2,2}\right) &= (\ B\,C\,A\,C\,D\ )\ , \end{aligned}$$

where the simplification

$$\phi_{2,2}^1 = 00 = A \qquad \phi_{2,2}^2 = 01 = B \qquad \phi_{2,2}^3 = 10 = C \qquad \phi_{2,2}^4 = 11 = D$$

has been used. In the regrouped form, the original series is surely shorter, but we have traded length for a kind of 'dimensionality'. The modified series is no longer binary, but lends itself to richer statistical analysis. *The heart of this study is convincing the reader that* $(\ B\,C\,A\,C\,D\ )$ *is richer than, but still equivalent to* $(\ 0\,1\,1\,0\,0\,0\,1\,0\,1\,1\ )$.

Generalized Regrouping

A sequence S_m^N can be grouped in terms of higher-order element vectors

$$S^N\left(\phi_m\right) = \left(S_1^{N/n}\left(\phi_{m,n}\right)\ S_2^{N/n}\left(\phi_{m,n}\right)\ S_3^{N/n}\left(\phi_{m,n}\right)\ \cdots\ S_{N/n}^{N/n}\left(\phi_{m,n}\right)\right) + \mathcal{R}^{N \bmod n}\left(\phi_m\right)\ ,$$

where the modulus operator calculates the cutoff where regrouping ceases. For very long sequences with $m \ll N$, the remainder becomes negligible.

The strange case $m > N$ applies to short sequences, or if the number of elements from which to sample is vast in size.

3.2 Hand-worked Example

Consider the sequence containing 24 elements

$$S^{24}\left(\phi_2\right) = (\ 0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\ )\ .$$

The question on hand is, does S^{24} contain any detectable order, and if so, how much? To proceed, let's write out the order-two and order-four regrouping of the element vector $\phi_2 = (0, 1)$:

$$\phi_{2,2} = (00, 01, 10, 11) = (A, B, C, D)$$

$$\phi_{2,4} = \begin{bmatrix} 0000 & 0001 & 0010 & 0011 \\ 0100 & 0101 & 0110 & 0111 \\ 1000 & 1001 & 1010 & 1011 \\ 1100 & 1101 & 1110 & 1111 \end{bmatrix} = \begin{bmatrix} AA & AB & AC & AD \\ BA & BB & BC & BD \\ CA & CB & CC & CD \\ DA & DB & DC & DD \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

Translating the 24-member sequence using ϕ_2 and $\phi_{2,2}$, we find

$$\begin{aligned} S^{24}(\phi_2) &= (\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,) \\ S^{24}(\phi_{2,2}) &= (\,B\,A\,D\,A\,D\,C\,B\,A\,D\,A\,D\,C\,) \\ S^{24}(\phi_{2,4}) &= (\,e\,m\,o\,e\,m\,o\,) \ . \end{aligned}$$

Interestingly, the regrouped sequence ($e\,m\,o\,e\,m\,o$) exhibits discernible order that would be tedious to pick out from the original sequence.

Driving further, we may treat ($e\,m\,o\,e\,m\,o$) as a new sequence with $\phi_{2,4}$ as its element vector, corresponding to the 'short list' case discussed above. Performing a third-order regrouping on $\phi_{2,4}$ and denoting the result $\psi_{16,3}$, we have

$$\psi_{16,3} = (aaa, aab, aac, \ldots, ppn, ppo, ppp) \ ,$$

containing $16^3 = 4096$ elements. Somewhere in $\psi_{16,3}$ is the arrangement *eno*, which we shall denote as its kth element. In this case, our original sequence becomes

$$S^{24}(\psi_{16,3}) = \left(\, \psi^k_{16,3}\, \psi^k_{16,3} \,\right) \ .$$

In this form, it's perfectly evident that S^{24}_2 is an ordered sequence, namely, it was contrived from two identical copies of

$$S^{12}(\phi_2) = (\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,) \ .$$

In order to put a number to just 'how' improbable it is that S^{24} was generated randomly, we make analogy to the so-called birthday problem (see Appendix). From elementary arguments, it can be shown that the probability of any two people out of population N sharing a birthday is given by

$$P(N) = 1 - \frac{365!}{365^N\,(365 - N)!} \ .$$

In our case, the population is analogous to the size of the regrouped sequence, in which case we have $N = 2$ elements. The number 365 is replaced by the number of total elements in $\phi_{16,3}$, namely 4096. Putting this all together, we find that the probability that $\psi^k_{16,3}$ should show up twice is

$$P(2) = 1 - \frac{4096!}{4096^2\,(4096 - 2)!} = \frac{1}{4096} \ .$$

Of course for $N = 2$ the answer could have been written down quickly, but the $P(N)$-equation handles any N.

3.3 Translation via Element Shifting

Translation via regrouping is sensitive to the leading elements in a sequence. For instance, the sequence

$$S^9 = (\ 3\,1\,2\,3\,1\,2\,3\,1\,2\ )\ ,$$

using a third-order regrouping of the element vector $\phi_3 = (1, 2, 3)$, resolves to three instances of the same element $\phi^k_{3,3} = 321$. If we create a slightly modified version, namely ${}^1S^9$, with the first element bumped to the last place

$${}^1S^9 = (\ 1\,2\,3\,1\,2\,3\,1\,2\,3\ )\ ,$$

then the regrouped sequence is three copies of a different element $\phi^q_{3,3} = 123$.

A shifted sequence shall be denoted ${}^qS^N$, where q is the number of elements moved to the end of the sequence, preserving order. That is,

$${}^qS^N(\phi_m) = S^N_{1+q}(\phi_m)\ S^N_{2+q}(\phi_m)\ S^N_{3+q}(\phi_m)\ \cdots\ S^N_N(\phi_m)\ S^N_1(\phi_m)\, S^N_2(\phi_m)\ \cdots S^N_q(\phi_m)\ .$$

The shift factor q is also allowed to be negative, in which case the tailing sequence terms are translated to the front.

For an infinite sequence, the 'last' terms are generally undefined, in which case ${}^qS^N$ effectively deletes the first q elements in S^N. Negative q are undefined in this case.

4 Histograms

With the full 'translation via regrouping' mechanism established, we quickly review a key analysis tool that commonly used to validate the 'randomness' of a sequence, called the *histogram*.

A histogram is a two-dimensional graph that portrays the number of occurrences of a particular element ϕ^j_m in a sequence S^N. Histograms aid in visualizing the *average* value among the elements, along with the *deviation* of each element around the average. (See any text on probability and statistics for elaboration on these terms.)

To see a histogram in action, let us create a random sequence $S^{10000}(\phi_{10})$ using $QB64$:

```
OPEN "randoms.txt" FOR OUTPUT AS #1
FOR k = 1 TO 10000
    PRINT #1, INT(RND * 10)
NEXT
CLOSE #1
END
```

Next, we open Gnuplot and prepare the viewing canvas to display the contents of 'randoms.txt':

```
binwidth=1
bin(x,width)=width*floor(x/width)
plot 'randoms.txt' using (bin($1,binwidth)):(1.0) smooth freq with boxes
```

According to the Gnuplot histogram, each element occurs roughly 1000 times in the random sequence, as expected. Each element's deviation from 1000 'hits' is consistent with N being 'only somewhat' large.

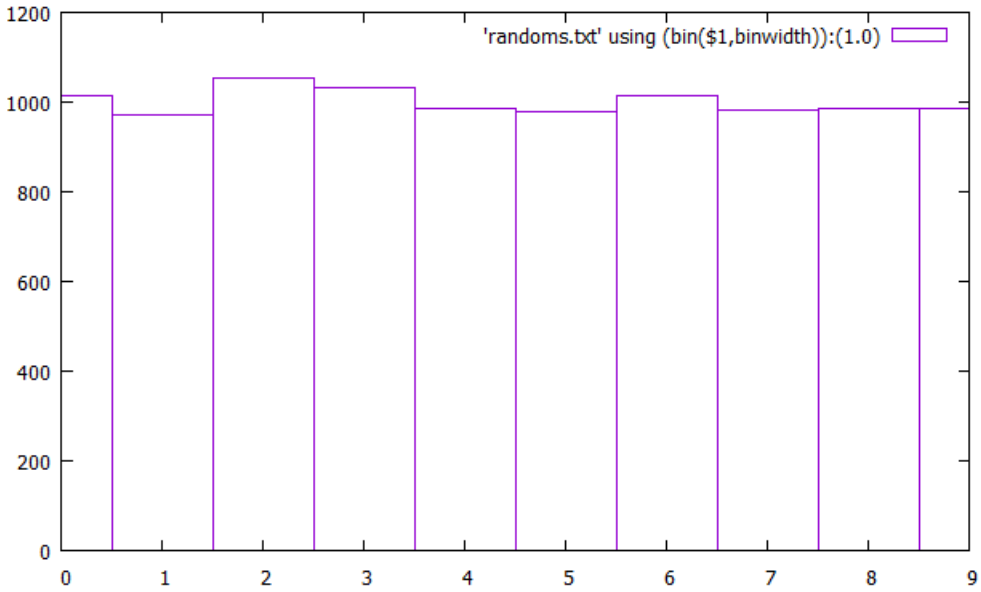

Figure 1: Histogram of 10000 random integers between 0 and 9, inclusive.

5 Numeric Example

5.1 Problem Setup

With all of the tools laying around, let us cook up an example of a mostly-random sequence, with little chunks of order sprinkled in. For example, the QB64 code

```
n = 10000
x1$ = "4136865152"
OPEN "seq_10_0.txt" FOR OUTPUT AS #1
FOR k = 1 TO n
    IF (RND > .02) THEN
        PRINT #1, LTRIM$(RTRIM$(STR$(INT(RND * 10))))
    ELSE
        FOR j = 1 TO LEN(x1$)
            PRINT #1, MID$(x1$, j, 1)
        NEXT
    END IF
NEXT
CLOSE #1
```

will write at least 10000 random integers to a file, however there is an intentional 'bug' in the random function that occasionally inserts 10 ordered digits (my incorrect phone number) into the sequence. The 'game' we play here is, pretend we didn't know the exact bug in the program. That is, if you were handed just the data file, can you find the bug?

Reaching first for histogram analysis, use the same Gnuplot code to generate the following plot:

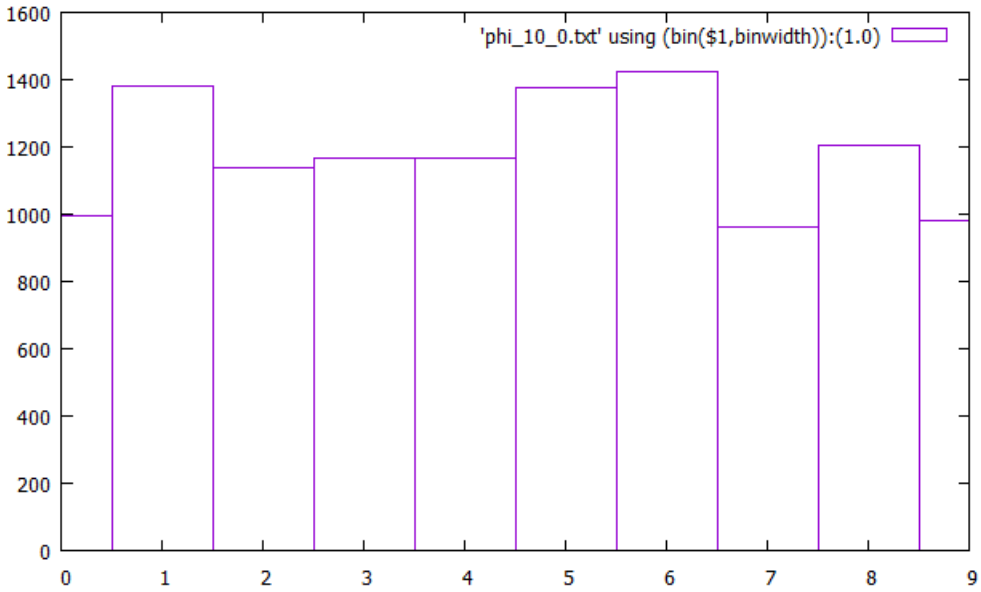

Figure 2: Histogram of > 10000 random integers between 0 and 9, inclusive, using a buggy RND function.

Of course, the histogram isn't as 'flat looking' as the one generated using the pure RND function. Clearly *something* is causing this, but we're stuck staring at this histogram without a better idea.

5.2 Data Preparation: Regrouping and Shifting

We don't know ahead of time what kind of order we're looking for, so the most robust way forward is to translate the with respect to multiple re-groupings $\phi_{10,n}$. Moreover, it's worth keeping track of each shifted series ${}^{q}S^{N}\left(\phi_{10,n}\right)$, where q does not exceed a given n. For instance, a regrouping based on $\phi_{10,2}$ can start with

$$ {}^{0}S^{10000}\left(\phi_{10,2}\right) \qquad \text{or} \qquad {}^{1}S^{10000}\left(\phi_{10,2}\right) $$

whereas a regrouping based on $\phi_{10,3}$ can start with

$$ {}^{0}S^{10000}\left(\phi_{10,3}\right) \qquad {}^{1}S^{10000}\left(\phi_{10,3}\right) \qquad {}^{2}S^{10000}\left(\phi_{10,3}\right) \ , $$

and so on.

Given a raw data file containing the base sequence, the following QB64 code generates a new sequence ${}^{q}S^{10000}\left(\phi_{10,n}\right)$ using all reasonable q for several choices of n:

```
FOR q = 2 TO 8 STEP 1
    FOR f = 0 TO q - 1
        OPEN "seq_10_0.txt" FOR INPUT AS #1
        OPEN "seq_10_" + LTRIM$(RTRIM$(STR$(q))) +
          "_" + LTRIM$(RTRIM$(STR$(f))) + ".txt" FOR OUTPUT AS #2
```

```
        y$ = ""
        qcount = 0
        ff = f
        q$ = ""
        DO WHILE NOT EOF(1)
            LINE INPUT #1, x$
            IF (ff > 0) THEN
                q$ = q$ + x$
                ff = ff - 1
            ELSE
                y$ = y$ + x$
                qcount = qcount + 1
                IF (qcount = q) THEN
                    PRINT #2, y$
                    y$ = ""
                    qcount = 0
                END IF
            END IF
        LOOP
        z$ = y$ + q$
        IF (LEN(z$) > 0) THEN
            IF (LEN(z$) < q) THEN
                PRINT #2, z$
            ELSE
                y$ = LEFT$(z$, q)
                z$ = RIGHT$(z$, LEN(z$) - LEN(y$))
                PRINT #2, y$
                PRINT #2, z$
            END IF
        END IF
        CLOSE #2
        CLOSE #1
    NEXT
NEXT
```

5.3 Processing

Since any shifted sequence contains the same number of data points as the un-shifted sequence, we gain efficiency by plotting all histograms of the same element vector $\phi_{m,n}$ on the same graph. Typing

```
set key off
binwidth=1
bin(x,width)=width*floor(x/width)
j=2
```

```
plot for[i = 0:j] "phi_10_".j."_".i.".txt" using (bin($1,binwidth)):(1.0)
  smooth freq with boxes
```

into Gnuplot, simultaneous histograms for the $\phi_{10,2}$ regrouping are stacked into the corresponding plot. Also let $j = 4$ to generate simultaneous histograms for the $\phi_{10,4}$ regrouping. Repeat for $j = 6$ and $j = 8$ (see below).

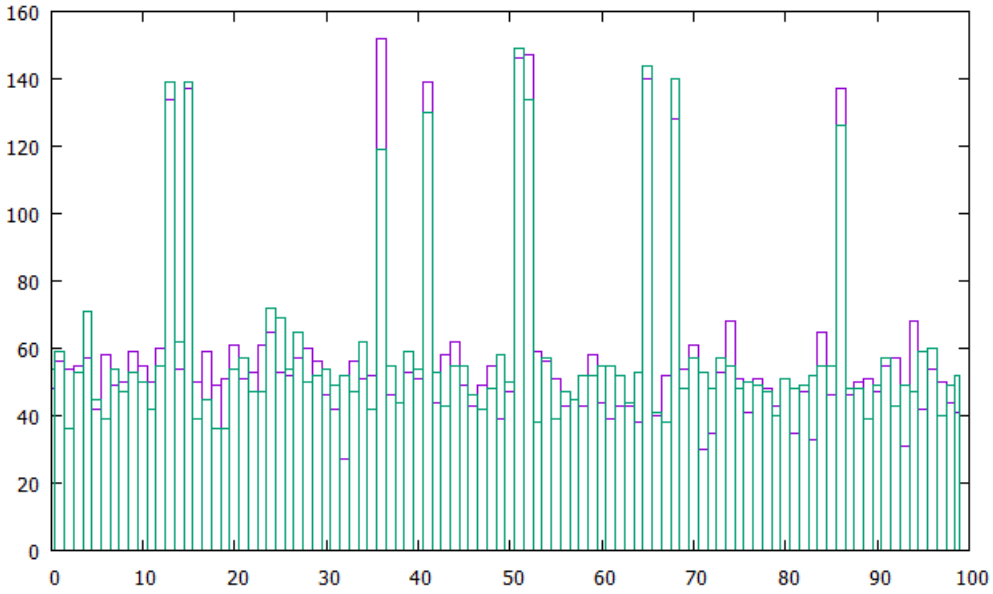

Figure 3: Regrouping of a pseudo-random sequence in terms of the $\phi_{10,2}$ element vector.

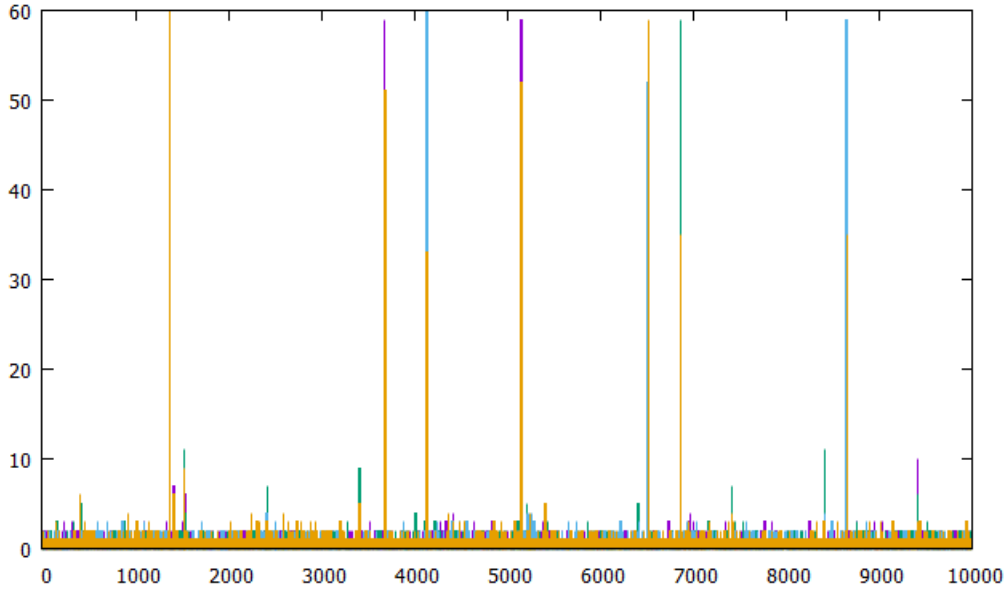

Figure 4: Regrouping of a pseudo-random sequence in terms of the $\phi_{10,4}$ element vector.

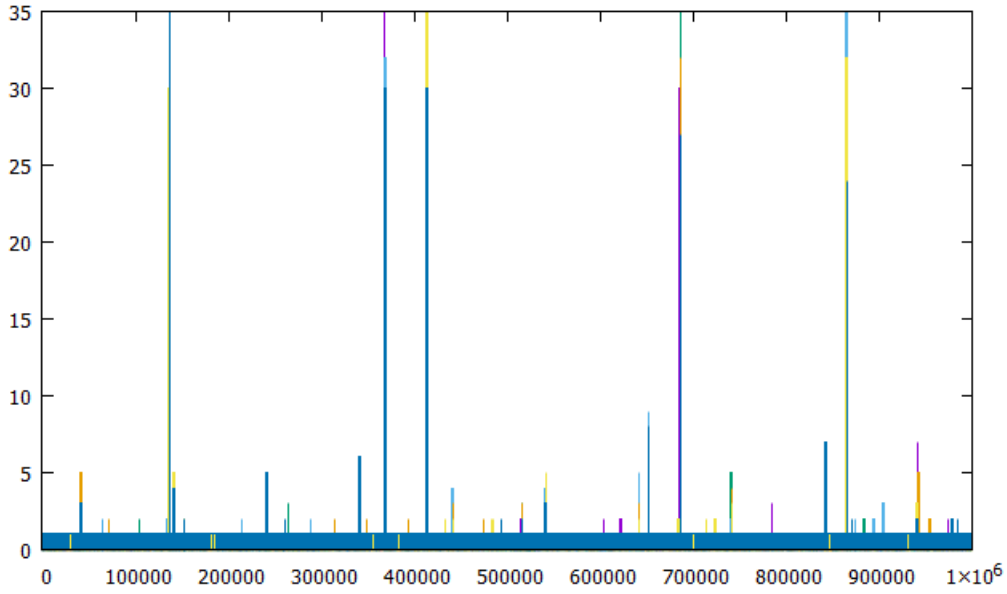

Figure 5: Regrouping of a pseudo-random sequence in terms of the $\phi_{10,6}$ element vector.

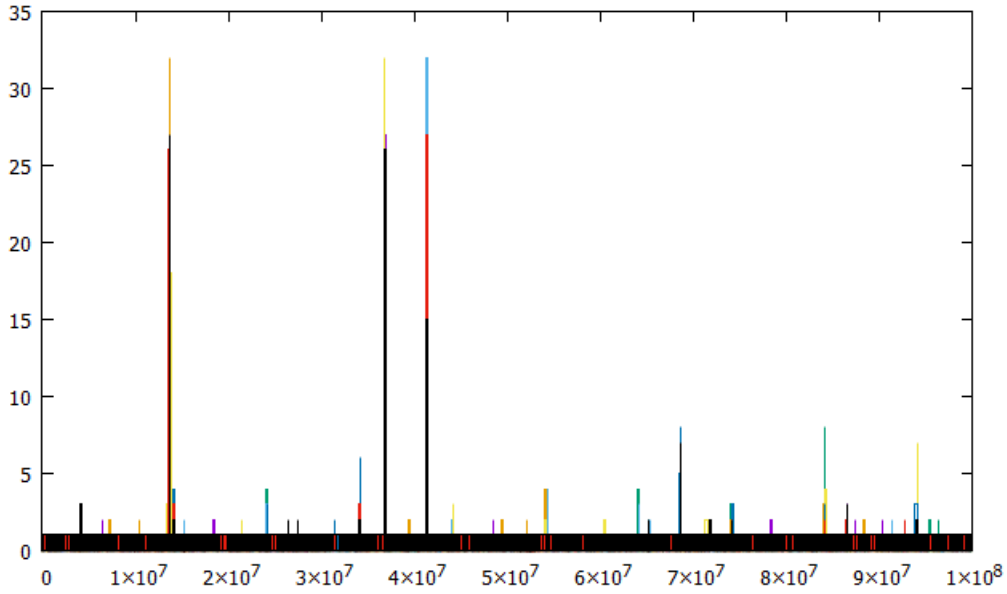

Figure 6: Regrouping of a pseudo-random sequence in terms of the $\phi_{10,8}$ element vector.

5.4 Results

We've come a long way since the non-regrouped histogram. In each of the results produced, several entries tower above the rest. Looking closely at each peak, we can extract the x-axis value ω_m^j under it. (This could be automated.)

Order Two

The peaks in the order-two histogram occur roughly at

$$\omega_2 = (13, 15, 35, 41, 51, 52, 65, 68, 86)$$

Order Four

Scaled by 100, The peaks in the order-four histogram occur roughly at

$$\omega_4 = (136, 368, 413, 515, 651, 686, 865)$$

Order Six

Scaled by 100, the peaks in the order-six histogram occur roughly at

$$\omega_6 = (1368, 3686, 4136, 6865, 8651)$$

5.5 Analysis

While we may keep stepping through $\phi_{10,n}$ to grow our confidence, there's enough evidence to figure out the bug in the RND function that generated the original series. Note that each ω_m turns up often-repeated sub-sequences that occur in the base sequence, meaning each member of ω_m may contain repeated elements that can be stitched together like a jigsaw puzzle.

Looking at the ω_6 result, observe that the structure 136 occurs in ω_6^1 and ω_6^3, and meanwhile 686 occurs in both ω_6^2 and ω_6^4. Taking all of this at once, we deduce that the subsequence

$$4136865152$$

occurs too often in the series, and we have found the intentional bug in the RND function without looking for it in any particular way.

6 ASCII Example

6.1 Problem Setup

For a more tricky example, let us take a highly-random sequence of printable characters from the ASCII set. This set contains all letters, numbers, brackets, punctuation, and so on - with the total of number of characters denoted $\tilde{m}$, implying a vector $\phi_{\tilde{m}}$. For example, the QB64 code

```
n = 100000
asc1 = 32
asc2 = 126
ascd = asc2 - asc1
x1$ = "@ll w0rk_&_n0~pl@y m@ke$.j@ck^@^Dull b0y"
OPEN "seq_abc_0.txt" FOR OUTPUT AS #1
FOR i = 1 TO n
    IF (RND > 10 / n) THEN
        p = INT(RND * (ascd + 1)) + asc1
        PRINT #1, CHR$(p)
    ELSE
        FOR j = 1 TO LEN(x1$)
            PRINT #1, MID$(x1$, j, 1)
        NEXT
    END IF
NEXT
CLOSE #1
```

will write at least 100000 random characters to a file, however there is an intentional 'bug' in the random function that occasionally inserts an ordered string into the sequence. Again, pretend we didn't know the exact bug in the program. That is, if you were handed just the data file, can you find the repeated substring?

6.2 Data Preparation: Regrouping and Shifting

As in the previous example, we don't know ahead of time what kind of order we're looking for. A conservative approach involves translating the sequence with respect to a splay of re-groupings. For no particular reason, we shall take:

$$
\begin{array}{llll}
{}^{0}S^{100000}\left(\phi_{\tilde{m},12}\right) & {}^{1}S^{100000}\left(\phi_{\tilde{m},12}\right) & \cdots & {}^{11}S^{100000}\left(\phi_{\tilde{m},12}\right) \\
{}^{0}S^{100000}\left(\phi_{\tilde{m},16}\right) & {}^{1}S^{100000}\left(\phi_{\tilde{m},16}\right) & \cdots & {}^{15}S^{100000}\left(\phi_{\tilde{m},16}\right) \\
{}^{0}S^{100000}\left(\phi_{\tilde{m},20}\right) & {}^{1}S^{100000}\left(\phi_{\tilde{m},20}\right) & \cdots & {}^{19}S^{100000}\left(\phi_{\tilde{m},20}\right) \\
{}^{0}S^{100000}\left(\phi_{\tilde{m},24}\right) & {}^{1}S^{100000}\left(\phi_{\tilde{m},24}\right) & \cdots & {}^{23}S^{100000}\left(\phi_{\tilde{m},24}\right)
\end{array}
$$

Given a raw data file containing the base sequence, the following QB64 code generates a new sequence ${}^{q}S^{100000}\left(\phi_{\tilde{m},n}\right)$ using all reasonable q for several choices of n:

```
FOR q = 12 TO 24 STEP 4
    FOR f = 0 TO q - 1
        OPEN "seq_abc_0.txt" FOR INPUT AS #1
        OPEN "seq_abc_" + LTRIM$(RTRIM$(STR$(q))) +
          "_" + LTRIM$(RTRIM$(STR$(f))) + ".txt" FOR OUTPUT AS #2
        y$ = ""
        qcount = 0
        ff = f
```

```
        q$ = ""
        DO WHILE NOT EOF(1)
            LINE INPUT #1, x$
            IF (ff > 0) THEN
                q$ = q$ + x$
                ff = ff - 1
            ELSE
                y$ = y$ + x$
                qcount = qcount + 1
                IF (qcount = q) THEN
                    PRINT #2, y$
                    y$ = ""
                    qcount = 0
                END IF
            END IF
        LOOP
        z$ = y$ + q$
        IF (LEN(z$) > 0) THEN
            IF (LEN(z$) < q) THEN
                PRINT #2, z$
            ELSE
                y$ = LEFT$(z$, q)
                z$ = RIGHT$(z$, LEN(z$) - LEN(y$))
                PRINT #2, y$
                PRINT #2, z$
            END IF
        END IF
        CLOSE #2
        CLOSE #1
    NEXT
NEXT
```

6.3 Processing

Since $\phi_{\tilde{m}}$ is already so large, let alone any regrouping $\phi_{\tilde{m},n}$, we dispense with plotting histograms and turn to an automated text-only process by using standard Unix tools. The following shell scripts analyze each sequence generated by the above and report the 20 most-common occurrences of the elements $\phi_{\tilde{m},n}$ in the file *result.txt*.

main.sh

```
#!/bin/bash

for k in 12 26 20 24; do
    ./combine.sh $k
```

```
done

for filename in data/*.dat; do
    ./core.sh $filename
done

./analyze.sh
```

combine.sh

```
#!/bin/bash

touch data/data.tmp

j=$1
for filename in data/seq_abc_"$j"_*.txt; do
    cat data/data.tmp $filename >
      data/tmpf && cat data/tmpf > data/data.tmp
done
rm data/tmpf

newfile=data/seq_abc_"$j"_all.dat
mv data/data.tmp $newfile
```

core.sh

```
#!/bin/bash

cp "$1" data_t.tmp

# Sort and count unique entries.
while read -r
do
    echo "$REPLY"
done < data_t.tmp > tmpf
sort tmpf | uniq -c | sort -nr > data_t.tmp

# Move the first column to the end.
awk '{first = $1; $1=""; print $0, "\t" first}'
  data_t.tmp > tmpf && cat tmpf > data_t.tmp

# Clean up.
rm tmpf

# Save sorted file.
```

```
mv data_t.tmp $(echo "$1" | cut -f 1 -d '.').res
```

analyze.sh

```
#!/bin/bash

for i in data/*.res; do
    awk 'NR>20{exit} {print $0}' $i
done > result.txt
```

6.4 Results

It turns out that only the $\phi_{\tilde{m},24}$ regrouping is needed to pick out the ordered substring in the original sequence. The relevant part of *result.txt* comes out to:

```
w0rk_&_n0~pl@y m@ke$.j@c  14
rk_&_n0~pl@y m@ke$.j@ck^  14
pl@y m@ke$.j@ck^@^Dull b  14
n0~pl@y m@ke$.j@ck^@^Dul  14
ll w0rk_&_n0~pl@y m@ke$.  14
l@y m@ke$.j@ck^@^Dull b0  14
l w0rk_&_n0~pl@y m@ke$.j  14
k_&_n0~pl@y m@ke$.j@ck^@  14
0rk_&_n0~pl@y m@ke$.j@ck  14
0~pl@y m@ke$.j@ck^@^Dull  14
~pl@y m@ke$.j@ck^@^Dull   14
_n0~pl@y m@ke$.j@ck^@^Du  14
_&_n0~pl@y m@ke$.j@ck^@^  14
@y m@ke$.j@ck^@^Dull b0y  14
@ll w0rk_&_n0~pl@y m@ke$  14
&_n0~pl@y m@ke$.j@ck^@^D  14
w0rk_&_n0~pl@y m@ke$.j@   14
)@ll w0rk_&_n0~pl@y m@ke   2
```

This file is essentially just the high peaks that would occur in a histogram.

6.5 Analysis

Evidently, the code has detected at least 10 instances of an ordered substring in the original sequence. Aligning each row to have the same character in each column (this is the only part done by hand), we find:

```
 w0rk_&_n0~pl@y m@ke$.j@c
   rk_&_n0~pl@y m@ke$.j@ck^
           pl@y m@ke$.j@ck^@^Dull b
        n0~pl@y m@ke$.j@ck^@^Dul
```

```
  ll w0rk_&_n0~pl@y m@ke$.
                l@y m@ke$.j@ck^@^Dull b0
   l w0rk_&_n0~pl@y m@ke$.j
        k_&_n0~pl@y m@ke$.j@ck^@
       0rk_&_n0~pl@y m@ke$.j@ck
              0~pl@y m@ke$.j@ck^@^Dull
               ~pl@y m@ke$.j@ck^@^Dull
            _n0~pl@y m@ke$.j@ck^@^Du
          _&_n0~pl@y m@ke$.j@ck^@^
                  @y m@ke$.j@ck^@^Dull b0y
 @ll w0rk_&_n0~pl@y m@ke$
           &_n0~pl@y m@ke$.j@ck^@^D
     w0rk_&_n0~pl@y m@ke$.j@
)@ll w0rk_&_n0~pl@y m@ke
----------------------------------------
 @ll w0rk_&_n0~pl@y m@ke$.j@ck^@^Dull b0y
```

Finally, our efforts distill down to discovering a single substring. The random sequence generator has about a 1/10000 chance of inserting the message:

```
@ll w0rk_&_n0~pl@y m@ke$.j@ck^@^Dull b0y
```

7 Conclusion

The 'translation via regrouping' method avails a means for detecting order embedded in a pseudo-random sequence that would not be detectable using averages, deviations, or other standard or statistical tests.

8 Appendix

Certain steps in this study draw analogy from well-known problems, namely the two-state system, along with the so-called 'birthday problem'.

8.1 Balanced Coin

One common generator of stochastic events is the *two-state* system, with the most common example being a two-sided coin. For instance, ten tosses of a balanced coin may generate the sequence

$$S_{\text{coin}} = (\text{H T H H T T T H H T}) \ ,$$

where 'H' denotes Heads, and 'T' denotes Tails. Of course, certain outcomes are more probable than others. If S_{coin} contained nothing but H, or similarly nothing but T, you would be right to suspect the coin is unbalanced.

For a balanced coin, it's easy to show that the probability of attaining a 'score' of m total heads among n trials is

$$P\left(m,n\right) = \frac{1}{2^n}\frac{n!}{m!\left(n-m\right)!} \ .$$

A game of 10 tosses should most likely result 5 Heads and 5 tails. Plugging $m = 5$ and $n = 10$ into the above, the exact probability of this is

$$P\left(5,10\right) = \frac{1}{2^{10}}\frac{10!}{5!\left(10-5\right)!} = \frac{1}{2^{10}}\frac{10\cdot 9\cdot 8\cdot 7\cdot 6}{5\cdot 4\cdot 3\cdot 2\cdot 1} \approx 25\% \ .$$

A different outcome would have 6 Heads and 4 Tails, or vice-versa (the probability is the same in each case):

$$P\left(6,10\right) = P\left(4,10\right) = \frac{1}{2^{10}}\frac{10!}{6!4!} = \frac{1}{2^{10}}\frac{10\cdot 9\cdot 8\cdot 7}{4\cdot 3\cdot 2\cdot 1} \approx 21\%$$

Continuing this for $m = 7$, $m = 8$, $m = 9$, $m = 10$, we find

$$P\left(7,10\right) = P\left(3,10\right) \approx 12\% \qquad P\left(8,10\right) = P\left(2,10\right) \approx 4\%$$

$$P\left(9,10\right) = P\left(1,10\right) = \frac{10}{2^{10}} < 1\% \qquad P\left(10,10\right) = P\left(0,10\right) = \frac{1}{2^{10}} \approx 0\% \ ,$$

which also calculates for the cases $m = 3$, $m = 2$, $m = 1$, $m = 0$.

With the probability of each possible score (0 to 10) written down, we verify that the sum of all probabilities must equal one:

$$\sum_{m=0}^{10} P\left(m,10\right) = 25\% + 2\cdot\left(21\% + 12\% + 4\%\right) + O\left(1\%\right) \approx 100\%$$

That is, if you play, you will always get *some* score.

8.2 Birthday Problem

Consider a classroom of total population N. What is the probability that any two people were born on the same day?

Begin with the trivial case $N = 2$, in where there is a $1/365$ chance of a common birthday:

$$P(2) = \frac{1}{365} = 1 - \frac{364}{365}$$

The result is written in the form $1 - X$ so we may focus on X, the probability of *no* common birthday.

A third person entering the system, making $N = 3$, has $365 - 2 = 363$ available days to avoid a common birthday. The probability becomes

$$P(3) = 1 - \frac{364}{365} \cdot \frac{363}{365},$$

and the pattern is now obvious. For total population N, the probability that some pair of people share a birthday is:

$$P(N) = 1 - \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{(365 - N + 1)}{365} = 1 - \frac{365!}{365^N (365 - N)!}$$

Note that X has been expressed as a recursion of conditional probabilities

$$X(n|n-1) = \frac{365 - (n-1)}{365} \qquad \rightarrow \qquad X(N) = \prod_{n=2}^{N} X(n|n-1),$$

which could also have been written directly by the permutation formula

$$X(N) = \frac{P_{365}^N}{365^N} = \frac{365!}{365^N (365 - N)!} \ .$$

Following is a list of various populations N with their corresponding $P(N)$:

N	P(N)
5	2.71%
10	11.7%
20	41.1%
23	50.7%
30	70.6%
50	97.0%

Remarkably, the population need only be 23 in order for there to be a 50% chance that any two people share a birthday.