Fake data, lies and traps in paper *Towards practical and robust DNA-based data archiving using the yin–yang codec system*

Yiyi HUANG

Apr 11 2024

Contents

Abstract

The paper "Towards practical and robust DNA-based data archiving using the yin–yang codec system" contains deadly fake data, lies and traps.

Computer programs for this post are held on my github page `https://github.com/tom123jack321/test_fake_in_paper_Towards`

Keywords: DNA storage; genetics; encode algorithm; decode algorithm; fake data; scientific misconduct; academic fraud; lies; robustness test

1 Cited papers

Two papers are mentioned in this post, Paper YYC and Paper Erlich et al.

Paper YYC: "Towards practical and robust DNA-based data archiving using the yin–yang codec system" (doi: 10.1038/s43588-022-00231-2) by (1) Zhi Ping, (2) Shihong Chen, (3) Guangyu Zhou, (4) Xiaoluo Huang, Sha Joe Zhu, Haoling Zhang, Henry H. Lee, Zhaojun Lan, Jie Cui, Tai Chen, Wenwei Zhang, Huanming Yang, Xun Xu, (14) George M. Church, (15) Yue Shen from (1)BGI-Shenzhen, Shenzhen, China, (2)Guangdong Provincial Key Laboratory of Genome Read and Write, BGI-Shenzhen, Shenzhen, China, (3)George Church Institute of Regenesis, BGI-Shenzhen, Shenzhen, China, (4)Shenzhen Institute of Synthetic Biology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, (5)China National GeneBank, BGI-Shenzhen, Shenzhen, China, (6)Department of Genetics, Harvard Medical School, Boston, MA, USA, (7)Department of Molecular and Cellular Biology, Harvard University, Cambridge, MA, USA, (8)Big Data Institute, Li Ka Shing Centre for Health Information and Discovery, University of Oxford, Oxford, UK, (9)School of Mathematical Science, Capital Normal University, Beijing, China,(10)Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA, USA published on Nature Computational Science `https://www.nature.com/articles/s43588-022-00231-2#peer-review`.

Paper Erlich et al.: Erlich Y, Zielinski D. DNA Fountain enables a robust and efficient storage architecture. Science. 2017 Mar 3;355(6328):950-954. doi: 10.1126/science.aaj2038. PMID: 28254941. The paper is listed as reference [22] in Paper YYC

2 Two fake critical statistics

Paper YYC advertises "high" *information density* which is defined to be the number of bits of input information divided by the number of nucleotides generated by coding schemes (in paper Erlich et al., the definition is named "net information density"; paper YYC equates the two names). Table 1 (Figure 1) states information density of that work is 1.95 bits/nt. However, the text later asserts the information density ranges "from 1.75 to 1.78 bits per base (Methods and Supplementary Fig. 2)". Differences between the two statements imply at least one (or both) statistic is fake. How did the authors obtain these statistics?

Supplementary Section 2 (Quantitative analysis; page 4-7) of supplementary information presents some specious analysis. Page 7 indicates the "1.95 bits/nt" is computed theoretically by removing index (Figure 2). Indices acts as location of each sequence, like the page number of a book. Without indices, it's impossible to decode DNA sequences; the original information can not be recovered, and hence the encoded DNA sequences as well as the

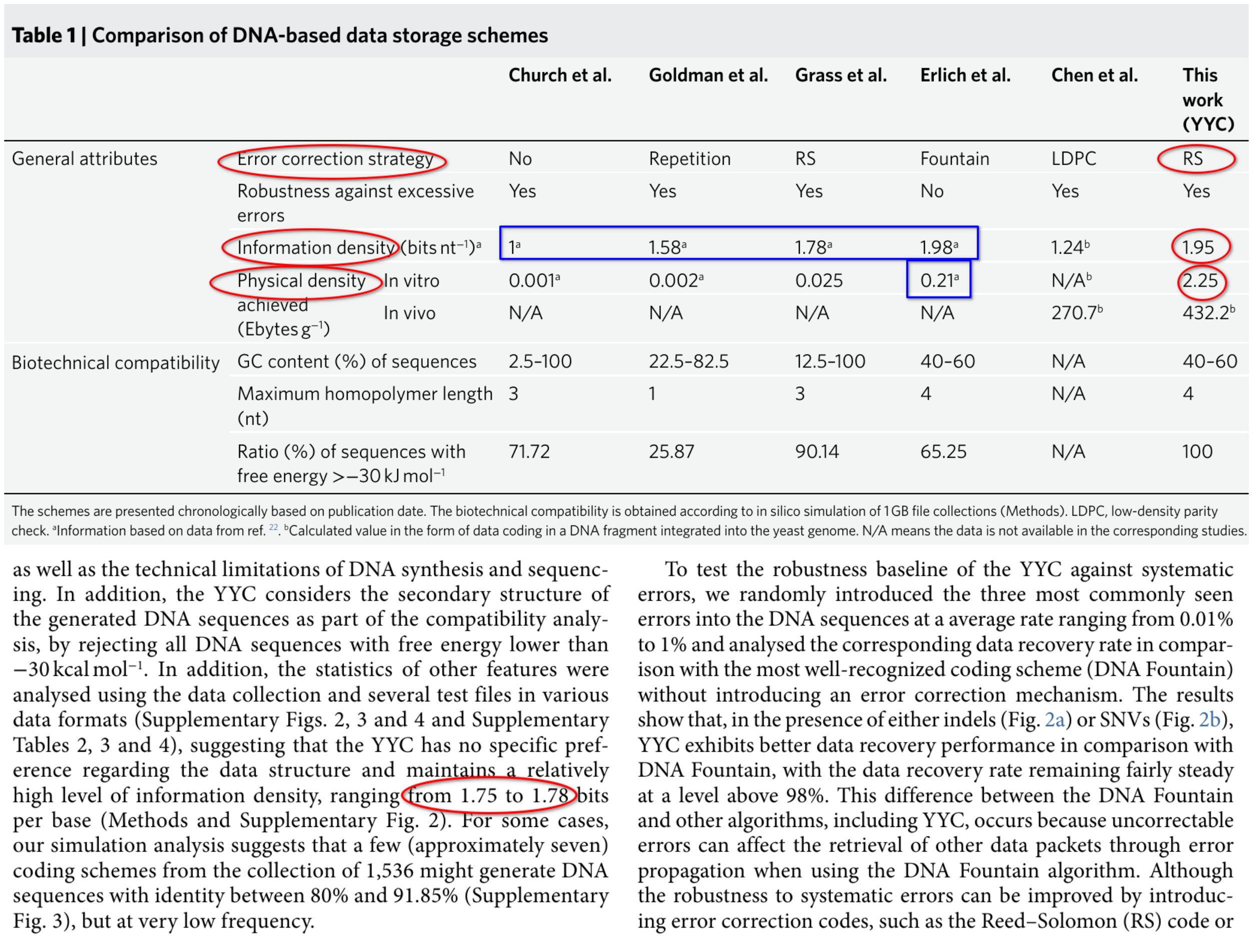


Table 1 | Comparison of DNA-based data storage schemes

			Church et al.	Goldman et al.	Grass et al.	Erlich et al.	Chen et al.	This work (YYC)
General attributes	Error correction strategy		No	Repetition	RS	Fountain	LDPC	RS
	Robustness against excessive errors		Yes	Yes	Yes	No	Yes	Yes
	Information density (bits nt^{-1})[a]		1[a]	1.58[a]	1.78[a]	1.98[a]	1.24[b]	1.95
	Physical density achieved (Ebytes g^{-1})	In vitro	0.001[a]	0.002[a]	0.025	0.21[a]	N/A[b]	2.25
		In vivo	N/A	N/A	N/A	N/A	270.7[b]	432.2[b]
Biotechnical compatibility	GC content (%) of sequences		2.5–100	22.5–82.5	12.5–100	40–60	N/A	40–60
	Maximum homopolymer length (nt)		3	1	3	4	N/A	4
	Ratio (%) of sequences with free energy >−30 kJ mol^{-1}		71.72	25.87	90.14	65.25	N/A	100

The schemes are presented chronologically based on publication date. The biotechnical compatibility is obtained according to in silico simulation of 1 GB file collections (Methods). LDPC, low-density parity check. [a]Information based on data from ref. [22]. [b]Calculated value in the form of data coding in a DNA fragment integrated into the yeast genome. N/A means the data is not available in the corresponding studies.

as well as the technical limitations of DNA synthesis and sequencing. In addition, the YYC considers the secondary structure of the generated DNA sequences as part of the compatibility analysis, by rejecting all DNA sequences with free energy lower than −30 kcal mol^{-1}. In addition, the statistics of other features were analysed using the data collection and several test files in various data formats (Supplementary Figs. 2, 3 and 4 and Supplementary Tables 2, 3 and 4), suggesting that the YYC has no specific preference regarding the data structure and maintains a relatively high level of information density, ranging from 1.75 to 1.78 bits per base (Methods and Supplementary Fig. 2). For some cases, our simulation analysis suggests that a few (approximately seven) coding schemes from the collection of 1,536 might generate DNA sequences with identity between 80% and 91.85% (Supplementary Fig. 3), but at very low frequency.

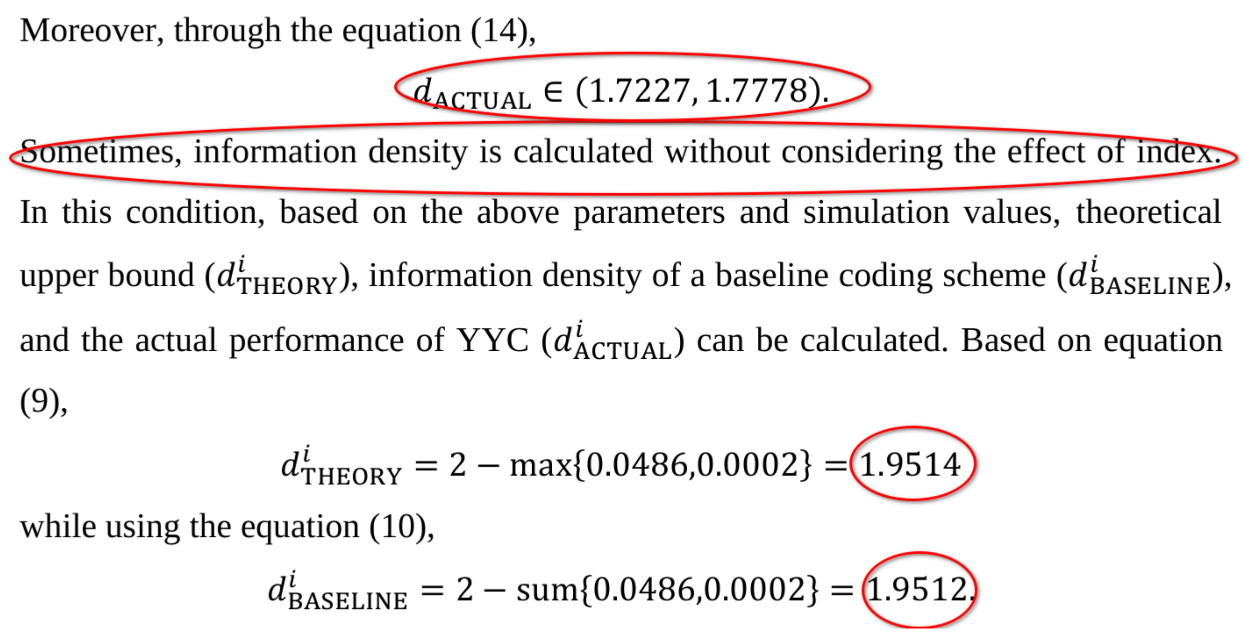
To test the robustness baseline of the YYC against systematic errors, we randomly introduced the three most commonly seen errors into the DNA sequences at a average rate ranging from 0.01% to 1% and analysed the corresponding data recovery rate in comparison with the most well-recognized coding scheme (DNA Fountain) without introducing an error correction mechanism. The results show that, in the presence of either indels (Fig. 2a) or SNVs (Fig. 2b), YYC exhibits better data recovery performance in comparison with DNA Fountain, with the data recovery rate remaining fairly steady at a level above 98%. This difference between the DNA Fountain and other algorithms, including YYC, occurs because uncorrectable errors can affect the retrieval of other data packets through error propagation when using the DNA Fountain algorithm. Although the robustness to systematic errors can be improved by introducing error correction codes, such as the Reed–Solomon (RS) code or

Figure 1: Paper YYC Table 1 and text after

entire coding system become useless. That's why Table 1 of paper Erlich et al. calls such quantity "coding potential" (meaning theoretical upper bound of coding capacity) and defines another concept "net information density" (meaning actual performance of coding capacity) (Figure 3). Paper YYC compares itself against paper Erlich et al. across the entire text and cites its statistics provided under "coding potential", but falsifies them to be "information density".

It should also be noticed the quantity d_{ACTUAL} should be in range (1.72265, 1.875) instead of (1.7227, 1.7778) in page 7 of supplementary information, by conducting elementary arithmetic according to equation (14) therein. This quantity is defined earlier in page 6 as the actual information density of YYC (Figure 6). However, Supplementary Table 3 (page 17 of supplementary information) (Figure 4) and Supplementary Table 4 (page 18 of supplementary information) (Figure 5) exhibits actual results below and above that range. Therefore, the analysis in Supplementary Section 2 is wrong.

Supplementary Fig. 2 (page 10) in supplementary information claims the statistic "from 1.75 to 1.78 bits per base" (Figure 7). All dots in the figure are incredibly aligned between lines 1.75 and 1.78 in a weird way that no one lies beyond that range, conflicting with data from Supplementary Table 3 (Figure 4) and Supplementary Table 4 (Figure 5), where information density of YYC coding scheme No.496 takes values 1.4480, 1.7250, 1.7265, ..., 1.8037, information density of YYC coding scheme No.1 takes values 1.440, 1.719, 1.726, ..., 1.804, and so on. **According to the two supplementary tables (Figure 4, 5), the true information density of YYC should reside**


Moreover, through the equation (14),

$$d_{\text{ACTUAL}} \in (1.7227, 1.7778).$$

Sometimes, information density is calculated without considering the effect of index. In this condition, based on the above parameters and simulation values, theoretical upper bound (d^i_{THEORY}), information density of a baseline coding scheme (d^i_{BASELINE}), and the actual performance of YYC (d^i_{ACTUAL}) can be calculated. Based on equation (9),

$$d^i_{\text{THEORY}} = 2 - \max\{0.0486, 0.0002\} = 1.9514$$

while using the equation (10),

$$d^i_{\text{BASELINE}} = 2 - \text{sum}\{0.0486, 0.0002\} = 1.9512.$$

Figure 2: Page 7 of supplementary information of Paper YYC

in a range containing interval [1.440, 1.805].

The explanation and terms provided in Supplementary Table 3 (Figure 4) and Supplementary Table 4 (Figure 5) demonstrates paper YYC equates the concepts "information density" and "net information density".

The conclusion is, both statistics "1.95 bits/nt" and "1.75 to 1.78 bits per base" are fake.

3 A third fake critical number

Supplementary Table 4 (Figure 5) shows, the differences on information density in the 10 testing cases between YYC No.1 scheme and DNA Fountain are 0.204, 0.855, 0.122, 0.066, 0.016, 0.278, 0.015, 0.001, -0.239, 0.004 respectively. The relative differences with respect to DNA Fountain are then 12.75%, 92.9348%, 7.37606%, 3.91924%, 0.928613%, 19.2922%, 0.874636%, 0.057971%, -14.2347%, 0.232288%, of which the median is 2.42393%, implying **YYC No.1 scheme does not achieve significantly higher information density. The other YYC schemes yield close data.**

Physical density is proportional to information density. Now that YYC does not differ from DNA fountain significantly regarding information density, it's impossible YYC achieves physical density 2.25 Ebytes/g whereas DNA fountain only 0.21 Ebytes/g which is less than $\frac{1}{10}$ of the former (Figure 1). **The statistic 2.25 Ebytes/g is fake.**

4 Tests on encode process

As the authors of Paper YYC falsified 3 critical statistics, it raises a question whether other data (Figure 4, 5) is fake too.

Download computer programs provided by authors of Paper YYC from `https://github.com/ntpz870817/DNA-storage-YYC` (as of Apr 3 2024). Copy "yyc" to under "examples" so that programs in "examples" are able to

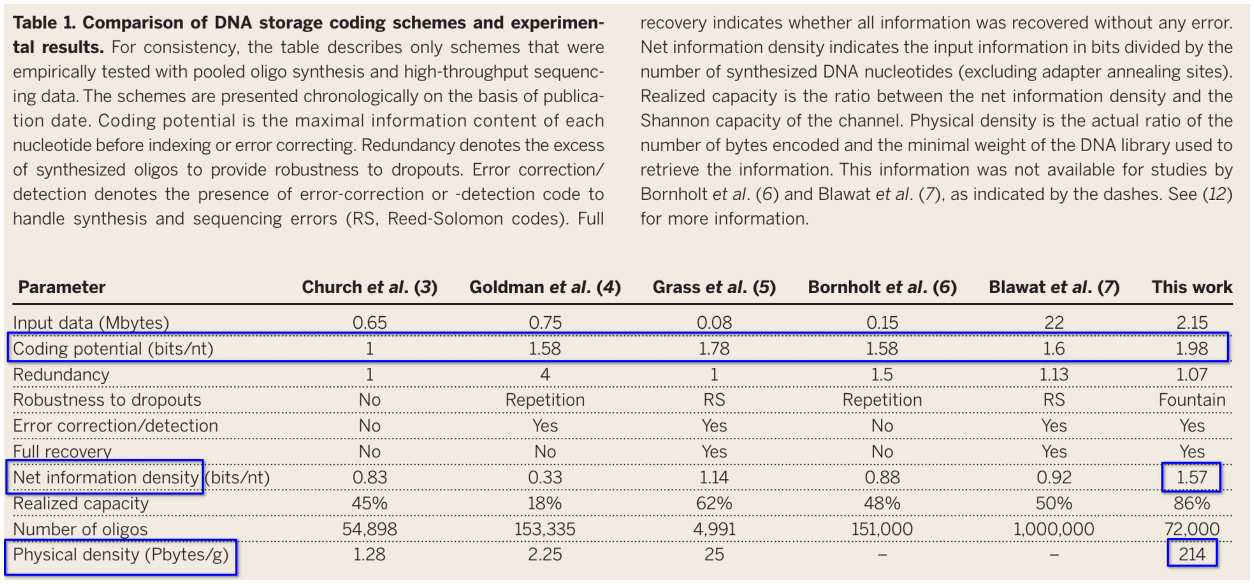


Table 1. Comparison of DNA storage coding schemes and experimental results. For consistency, the table describes only schemes that were empirically tested with pooled oligo synthesis and high-throughput sequencing data. The schemes are presented chronologically on the basis of publication date. Coding potential is the maximal information content of each nucleotide before indexing or error correcting. Redundancy denotes the excess of synthesized oligos to provide robustness to dropouts. Error correction/detection denotes the presence of error-correction or -detection code to handle synthesis and sequencing errors (RS, Reed-Solomon codes). Full recovery indicates whether all information was recovered without any error. Net information density indicates the input information in bits divided by the number of synthesized DNA nucleotides (excluding adapter annealing sites). Realized capacity is the ratio between the net information density and the Shannon capacity of the channel. Physical density is the actual ratio of the number of bytes encoded and the minimal weight of the DNA library used to retrieve the information. This information was not available for studies by Bornholt *et al.* (*6*) and Blawat *et al.* (*7*), as indicated by the dashes. See (*12*) for more information.

Parameter	Church *et al.* (*3*)	Goldman *et al.* (*4*)	Grass *et al.* (*5*)	Bornholt *et al.* (*6*)	Blawat *et al.* (*7*)	This work
Input data (Mbytes)	0.65	0.75	0.08	0.15	22	2.15
Coding potential (bits/nt)	1	1.58	1.78	1.58	1.6	1.98
Redundancy	1	4	1	1.5	1.13	1.07
Robustness to dropouts	No	Repetition	RS	Repetition	RS	Fountain
Error correction/detection	No	Yes	Yes	No	Yes	Yes
Full recovery	No	No	Yes	No	Yes	Yes
Net information density (bits/nt)	0.83	0.33	1.14	0.88	0.92	1.57
Realized capacity	45%	18%	62%	48%	50%	86%
Number of oligos	54,898	153,335	4,991	151,000	1,000,000	72,000
Physical density (Pbytes/g)	1.28	2.25	25	–	–	214

Figure 3: Paper Erlich et al. Table 1

find the module "yyc". Change directory into "examples". There holds 2 example programs, "test_mona_lisa.py" and "test_united_nations_flag.py", for tests on files "files/Mona Lisa.jpg" and "files/United Nations Flag.bmp", respectively. Authors did not provid the other 8 testing files. Fortunately, a program library called Chamaeleo (`https://github.com/ntpz870817/Chamaeleo` as of Apr 3 2024) that shares authors with Paper YYC provides a file named "Exiting the Factory.flv", which according to its test result later is hopefully the same file used in Paper YYC (Figure 4, 5).

Paper YYC repeats advertising again and again its "1536 schemes" or "1536 rules", but never document how to set a specific scheme. Readers have to search each file and read source codes for that. Nevertheless, only 4 YYC schemes are found. They are defined in comments of source code "yyc/utils/rule_set.py" (from line 19 to 22):

```
#      1: ["A", [0, 0, 1, 1], [[0, 1, 0, 1], [0, 1, 0, 1], [0, 1, 0, 1], [0, 1, 0, 1]]]
#    496: ["A", [0, 1, 0, 1], [[1, 1, 0, 0], [1, 0, 0, 1], [1, 1, 0, 0], [1, 1, 0, 0]]]
#    888: ["A", [1, 0, 0, 1], [[0, 1, 0, 1], [1, 1, 0, 0], [1, 0, 1, 0], [0, 0, 1, 1]]]
#   1536: ["A", [1, 1, 0, 0], [[1, 0, 1, 0], [1, 0, 1, 0], [1, 0, 1, 0], [1, 0, 1, 0]]]
```

Accordingly, the values

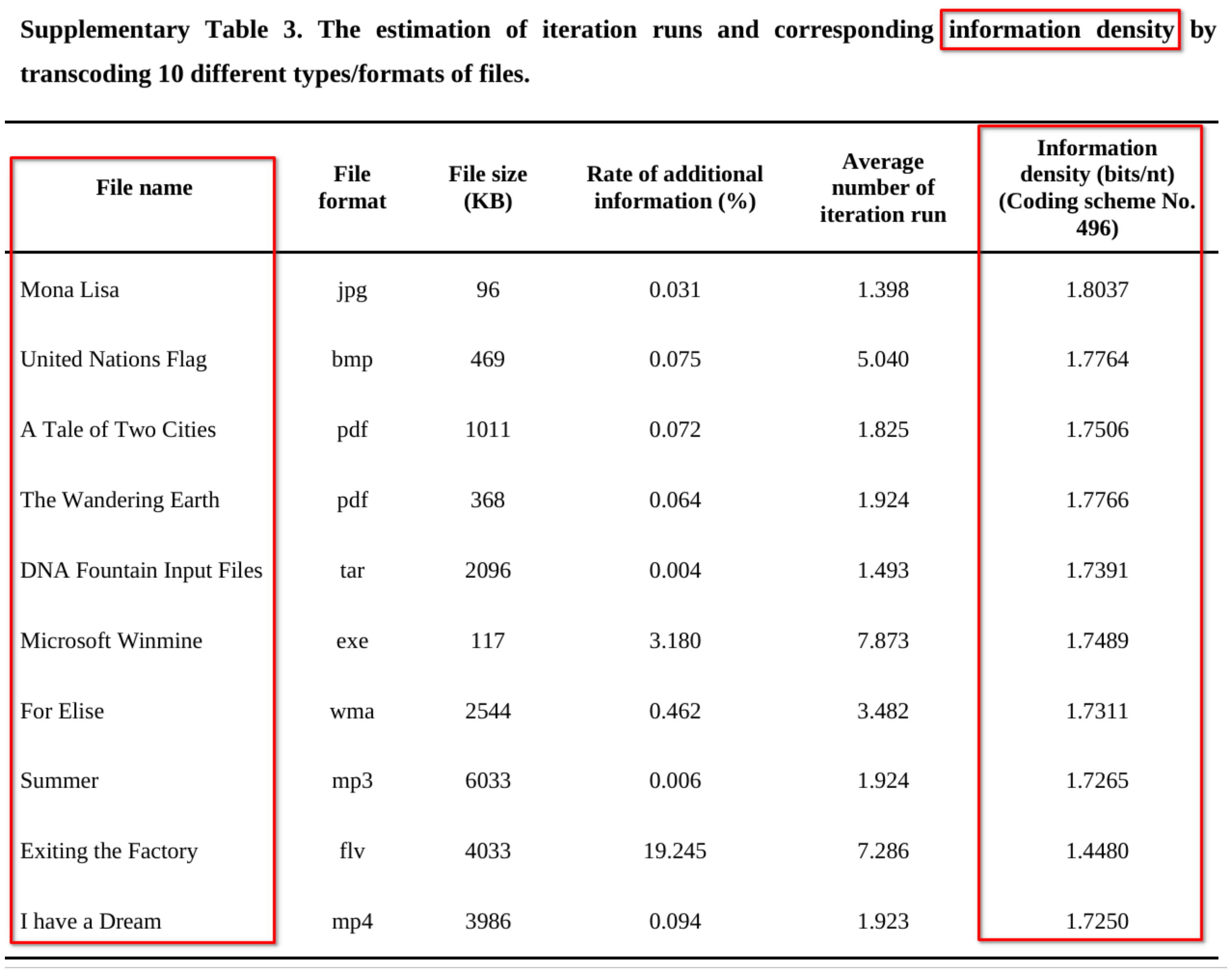
[support_base, rule1, rule2] = ["A", [0, 1, 0, 1], [[1, 1, 0, 0], [1, 0, 0, 1], [1, 1, 0, 0], [1, 1, 0, 0]]]

in "test_mona_lisa.py" define YYC scheme No.496, while

[support_base, rule1, rule2] = ["A", [1, 0, 0, 1], [[0, 1, 0, 1], [1, 1, 0, 0], [1, 0, 1, 0], [0, 0, 1, 1]]]

in "test_united_nations_flag.py" define YYC scheme No.888.

However, due to their randomness nature, these YYC schemes do not always work, although Paper YYC keeps boasting of them. To encode a file, users have to run every possible YYC scheme for multiple times until it succeeds. For instance, YYC scheme No.496 fails to encode the file "files/United Nations Flag.bmp" many times before it succeeds. By "fail", no DNA sequences are generated; users have to try the same scheme again or switch to a different scheme from all the asserted 1536 schemes that are never documented though. Paper YYC never mentions the failure of YYC schemes on encoding. **What a filthy hidden trap**!



Supplementary Table 3. The estimation of iteration runs and corresponding information density by transcoding 10 different types/formats of files.

File name	File format	File size (KB)	Rate of additional information (%)	Average number of iteration run	Information density (bits/nt) (Coding scheme No. 496)
Mona Lisa	jpg	96	0.031	1.398	1.8037
United Nations Flag	bmp	469	0.075	5.040	1.7764
A Tale of Two Cities	pdf	1011	0.072	1.825	1.7506
The Wandering Earth	pdf	368	0.064	1.924	1.7766
DNA Fountain Input Files	tar	2096	0.004	1.493	1.7391
Microsoft Winmine	exe	117	3.180	7.873	1.7489
For Elise	wma	2544	0.462	3.482	1.7311
Summer	mp3	6033	0.006	1.924	1.7265
Exiting the Factory	flv	4033	19.245	7.286	1.4480
I have a Dream	mp4	3986	0.094	1.923	1.7250

Figure 4: Supplementary Table 3 of supplementary information of Paper YYC

4.1 Test 1: YYC scheme No.496 on file "./files/Mona Lisa.jpg"

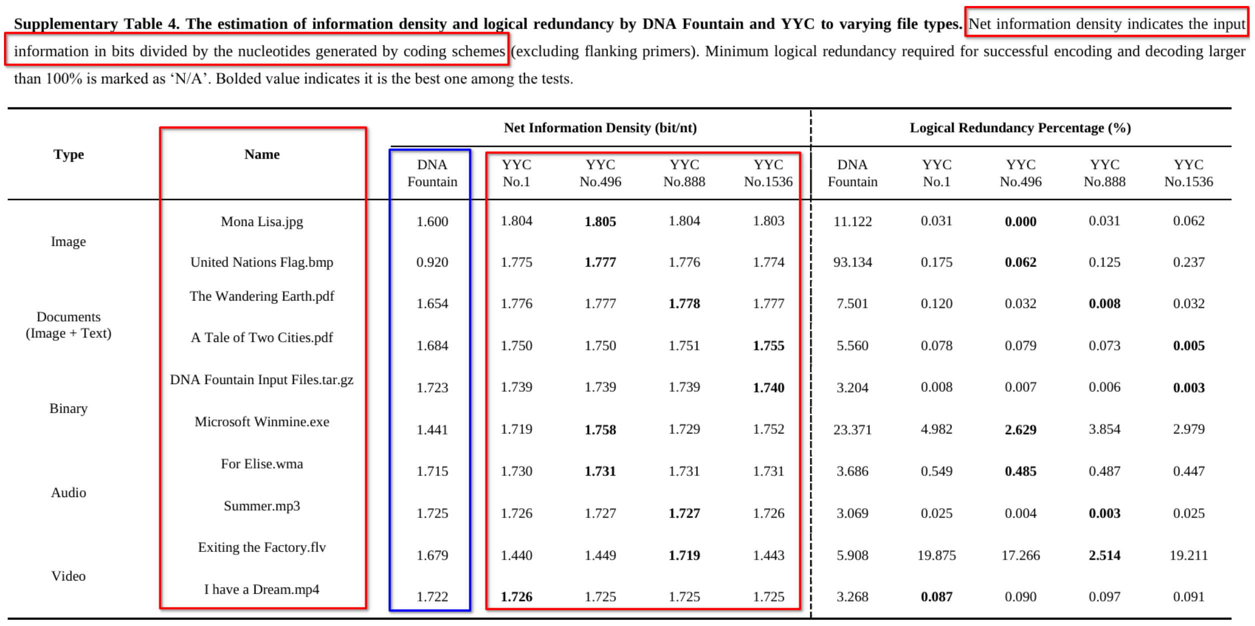
See "test_1" on my github page for related computer programs, input and output. Run "test_mona_lisa.py" multiple times.

The input "./files/Mona Lisa.jpg" has 780240 bits. Due to randomness nature of YYC schemes, the output "output/mona_lisa.dna" can record different numbers of nucleotides, such as 432516, 432383 and 432782, resulting in information density 1.80396 bits/nt, 1.80451 bits/nt, 1.80285 bits/nt, respectively. The corresponding data in Figure 4 and 5 is quite close to these, although not the same.

4.2 Test 2: YYC scheme No.888 on file "./files/Mona Lisa.jpg"

See "test_2" on my github page for related computer programs, input and output. Run "test_mona_lisa.py" multiple times.

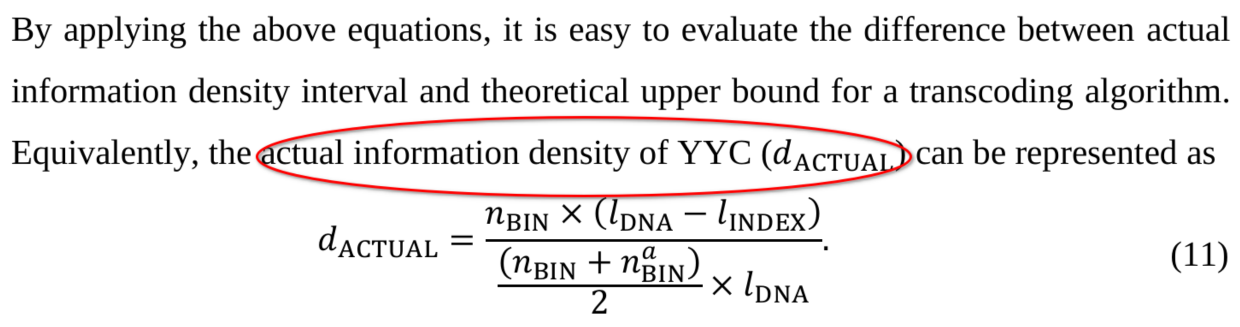
The input "./files/Mona Lisa.jpg" has 780240 bits. The output "output/mona_lisa.dna" records 432383 nucleotides, resulting in information density 1.80451 bits/nt. The corresponding data in Figure 5 is quite close.



Supplementary Table 4. The estimation of information density and logical redundancy by DNA Fountain and YYC to varying file types. Net information density indicates the input information in bits divided by the nucleotides generated by coding schemes (excluding flanking primers). Minimum logical redundancy required for successful encoding and decoding larger than 100% is marked as 'N/A'. Bolded value indicates it is the best one among the tests.

Type	**Name**	**Net Information Density (bit/nt)**					**Logical Redundancy Percentage (%)**				
		DNA Fountain	YYC No.1	YYC No.496	YYC No.888	YYC No.1536	DNA Fountain	YYC No.1	YYC No.496	YYC No.888	YYC No.1536
Image	Mona Lisa.jpg	1.600	1.804	**1.805**	1.804	1.803	11.122	0.031	**0.000**	0.031	0.062
	United Nations Flag.bmp	0.920	1.775	**1.777**	1.776	1.774	93.134	0.175	**0.062**	0.125	0.237
Documents (Image + Text)	The Wandering Earth.pdf	1.654	1.776	1.777	**1.778**	1.777	7.501	0.120	0.032	**0.008**	0.032
	A Tale of Two Cities.pdf	1.684	1.750	1.750	1.751	**1.755**	5.560	0.078	0.079	0.073	**0.005**
Binary	DNA Fountain Input Files.tar.gz	1.723	1.739	1.739	1.739	**1.740**	3.204	0.008	0.007	0.006	**0.003**
	Microsoft Winmine.exe	1.441	1.719	**1.758**	1.729	1.752	23.371	4.982	**2.629**	3.854	2.979
Audio	For Elise.wma	1.715	1.730	**1.731**	1.731	1.731	3.686	0.549	**0.485**	0.487	0.447
	Summer.mp3	1.725	1.726	1.727	**1.727**	1.726	3.069	0.025	0.004	**0.003**	0.025
Video	Exiting the Factory.flv	1.679	1.440	1.449	**1.719**	1.443	5.908	19.875	17.266	**2.514**	19.211
	I have a Dream.mp4	1.722	**1.726**	1.725	1.725	1.725	3.268	**0.087**	0.090	0.097	0.091

Figure 5: Supplementary Table 4 of supplementary information of Paper YYC



By applying the above equations, it is easy to evaluate the difference between actual information density interval and theoretical upper bound for a transcoding algorithm. Equivalently, the actual information density of YYC (d_{ACTUAL}) can be represented as

$$d_{\text{ACTUAL}} = \frac{n_{\text{BIN}} \times (l_{\text{DNA}} - l_{\text{INDEX}})}{\frac{(n_{\text{BIN}} + n_{\text{BIN}}^{a})}{2} \times l_{\text{DNA}}}. \tag{11}$$

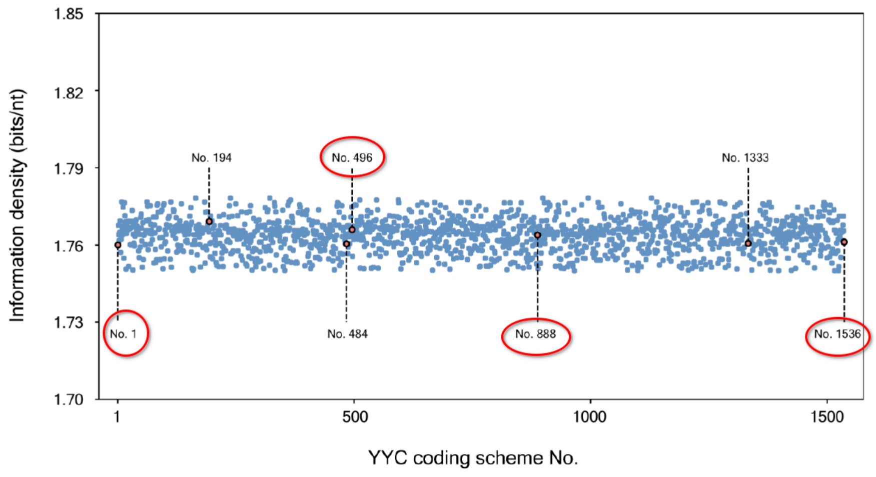
Figure 6: Page 6 of supplementary information of Paper YYC

4.3 Test 3: YYC scheme No.496 on file "./files/United Nations Flag.bmp"

See "test_3" on my github page for related computer programs, input and output.

Run "test_united_nations_flag_manual.py" manually for 30 times or "test_united_nations_flag_automatic.py" for 1 time. The file "test_united_nations_flag_automatic.py" automatically calls encode function of YYC scheme No.496 for 30 times within a "while" loop. Most calls failed. Due to python's incompetent exception handling, the process halts on those failed calls and users have to press Ctrl+C to terminate the halted loop so as to proceed to next loop. **The scheme succeeded in encoding only 4 times upon 30 times calls.**

The input "./files/United Nations Flag.bmp" has 3840448 bits. The output files of DNA sequences, which have been moved to "output_old", record 2161755, 2161485, 2161620 nucleotides, resulting in information density 1.77654, 1.77676, 1.77665 (bits/nt), respectively. The corresponding data in Figure 4 and 5 is quite close, although not the same.



Supplementary Figure 2: Information density evaluation of 1536 YYC coding schemes using 1G data collection. Constraint parameters applied: maximum homopolymer run as 4, GC content 40%-60%, index length as 16, free energy of secondary structure > -30 kj/mol. YYC coding schemes NO. 1, 194, 484, 496, 888, 1333 and 1536 used in the simulation analysis and experimental validation are highlighted as red dots.

Figure 7: Supplementary Figure 2 of supplementary information of Paper YYC

4.4 Test 4: YYC scheme No.888 on file "./files/United Nations Flag.bmp"

See "test_4" on my github page for related computer programs, input and output. Run "test_united_nations_flag_manua manually for 30 times or "test_united_nations_flag_automatic.py" for 1 time. The file "test_united_nations_flag_automati automatically calls encode function of YYC scheme No.888 for 30 times within a "while" loop. All calls succeeded. The generated 30 files of DNA sequences have been moved to "output_old". It turns out the corresponding information density data in Figure 5 is quite close to that of these tests.

4.5 Test 5: YYC scheme No.496 on file "./files/Exiting the Factory.flv"

See "test_5.py" on my github page for related computer programs, input and output. The information density observed is 1.44761. It turns out the corresponding information density data in Figure 4 and 5 is quite close to that of this test.

Besides, it has to be noticed that while encoding "./files/Mona Lisa.jpg" takes 5 seconds with YYC scheme No.496 on my computer, encoding "./files/Exiting the Factory.flv" with the same scheme takes 2320 seconds, which is 464 times of the former, although size of the latter file (4036K) is only 42 times of the former file (96K). **This implies highly non-linearity of time complexity and that YYC is extremely slow and indeed useless in practice.**

4.6 Test 6: YYC scheme No.888 on file "./files/Exiting the Factory.flv"

See "test_6" on my github page for related computer programs, input and output. It turns out the corresponding information density data in Figure 5 is quite close to that of this test.

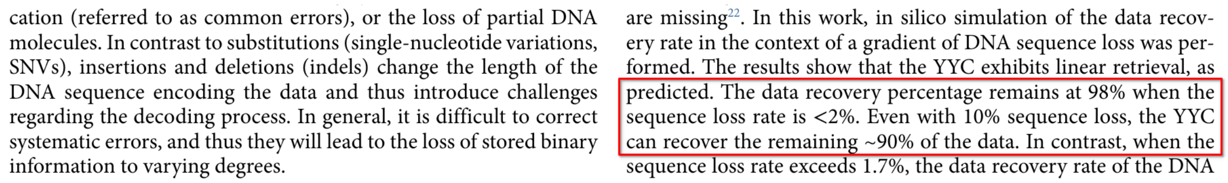
4.7 Conclusion

The encode results of YYC schemes are random, in the sense that the encode process can fail or succeed and that if it succeeded, information density can vary. The tested YYC schemes on

3 files produce quite close information density to those of Supplementary Table 3 (Figure 4) and Supplementary Table 4 (Figure 5) of supplementary information of paper YYC. These tests also confirm the conclusion that information density of YYC should reside in a range containing interval [1.440, 1.805]

5 Robustness tests

So far, only the encode process of YYC is tested. This section focuses on decode process. During synthesis, replication and sequencing, DNA sequences are prone to errors of substitution, deletion and insertion. A DNA storage system must utilise certain error detection and correction methods to protect its decode process from errors. Paper YYC asserts RS method was used as error correction strategy, which should be able to detect and correct multiple errors. The text and Fig.2 of the article asserts 98% or above data recovery rate under various errors of rates from 0.01% to 1.00% (Figure 8). The authors did not provide computer programs reproducing these results.



cation (referred to as common errors), or the loss of partial DNA molecules. In contrast to substitutions (single-nucleotide variations, SNVs), insertions and deletions (indels) change the length of the DNA sequence encoding the data and thus introduce challenges regarding the decoding process. In general, it is difficult to correct systematic errors, and thus they will lead to the loss of stored binary information to varying degrees.

are missing[22]. In this work, in silico simulation of the data recovery rate in the context of a gradient of DNA sequence loss was performed. The results show that the YYC exhibits linear retrieval, as predicted. The data recovery percentage remains at 98% when the sequence loss rate is <2%. Even with 10% sequence loss, the YYC can recover the remaining ~90% of the data. In contrast, when the sequence loss rate exceeds 1.7%, the data recovery rate of the DNA

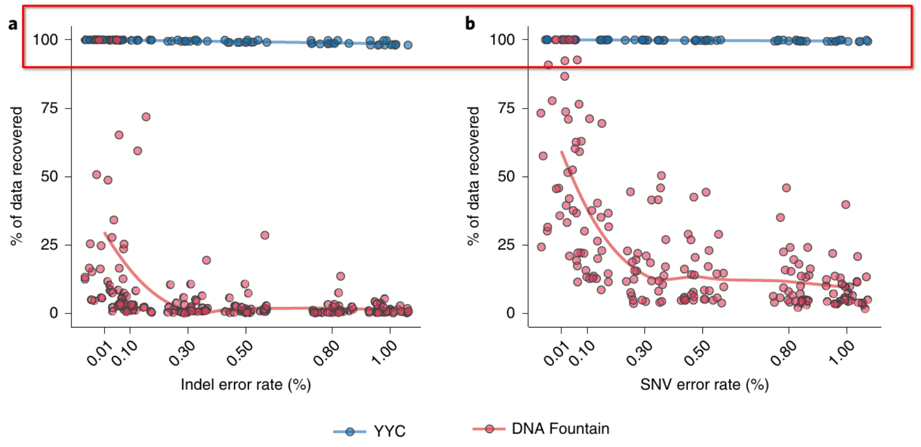


Fig. 2 | Robustness analysis for the YYC and DNA Fountain coding schemes. a,**b**, The binary data recovery rate of the YYC (blue) and DNA Fountain (red) coding strategies without any error-correction algorithm for indels (**a**) and SNVs (**b**) introduced randomly with an error rate of 0.01%, 0.1%, 0.3% 0.5%, 0.8% or 1%.

Figure 8: Paper YYC text and Fig. 2 on robustness

In what follows, both manual tests and automatic tests on robustness of YYC are performed. In manual tests, the image “files/Mona Lisa.jpg” was encoded into DNA sequences recorded in “output/mona_lisa.dna” with YYC No.496 scheme. Then made a copy of the file and introduced some errors, generating a new file “output/mona_lisa.dna_new”. Each manual test substituted, deleted or inserted only **one** nucleotide. Finally the new file was decoded. If it succeeded, compare the decoded image “output/output_mona_lisa.jpg” with the original image “files/Mona Lisa.jpg”. The programs used for encoding, decoding and comparison are all provided by authors of paper YYC.

In automatic tests, likewise, the image “./files/mon_lisa.jpg” and “./files/unite_nation_flag.bmp” were encoded into DNA sequences recorded in “output/mona_lisa.dna” and “output/united_nations_flag.dna” with YYC No.888

scheme. Then made copies of the files and introduced some errors, generating new files "output/mona_lisa.dna_new" and "output/united_nations_flag.dna_new" (names of the new files in automatic Test 14 are extended to contain information about error rates, error types and sub-test numbers). Each automatic test or sub-test **randomly** substitute, delete or insert (or randomly do any of these operations) certain number of nucleotides depending on error rates. Finally the new file was decoded. If it succeeded, compare the decoded image with the original image. The programs used for encoding, decoding and comparison are all provided by authors of paper YYC.

5.1 (Manual) Test 7: substitute the first nucleotide

See "test_7" on my github page for related computer programs, input and output. Run "test_mona_lisa_encode.py". Make a copy of output "output/mona_lisa.dna" with name "output/mona_lisa.dna_new". The first nucleotide, in this test, was "G". Due to randomness nature of YYC, the nucleotide can vary on different instances of encode process. Now, change "G" to "T". Run "test_mona_lisa_decode.py".

The decode process generated a 285-byte file "output/output_mona_lisa.jpg" that is not a valid image. Note that the original image "files/Mona Lisa.jpg" has 97530 bytes. The comparison program printed "false" meaning they were not the same file. **Thus, YYC failed to recover original information.**

5.2 (Manual) Test 8: substitute the last nucleotide

See "test_8" on my github page for related computer programs, input and output. Run "test_mona_lisa_encode.py". Make a copy of output "output/mona_lisa.dna" with name "output/mona_lisa.dna_new". The last nucleotide, in this test, was "T". Due to randomness nature of YYC, the nucleotide can vary on different instances of encode process. Now, change "T" to "A". Run "test_mona_lisa_decode.py".

The decode process generated a file "output/output_mona_lisa.jpg" of which approximately merely one-quarter looks similar to the original image (Figure 9), whereas the other areas become nothing. The comparison program printed "false" meaning they were not the same file. **Thus, YYC failed to recover original information.**

5.3 (Manual) Test 9: delete the first nucleotide

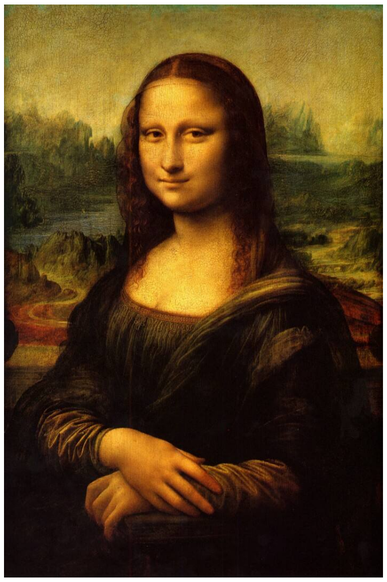
See "test_9" on my github page for related computer programs, input and output. Run "test_mona_lisa_encode.py". Make a copy of output "output/mona_lisa.dna" with name "output/mona_lisa.dna_new". The first nucleotide, in this test, was "G". Due to randomness nature of YYC, the nucleotide can vary on different instances of encode process. Now, delete "G". Run "test_mona_lisa_decode.py".

The decode process failed to finish. Nothing was generated. **Thus, YYC failed to recover original information.**

5.4 (Manual) Test 10: delete the last nucleotide

See "test_10" on my github page for related computer programs, input and output. Run "test_mona_lisa_encode.py". Make a copy of output "output/mona_lisa.dna" with name "output/mona_lisa.dna_new". The last nucleotide, in this test, was "T". Due to randomness nature of YYC, the nucleotide can vary on different instances of encode process. Now, delete "T". Run "test_mona_lisa_decode.py".

The decode process failed to finish (as a result, the comparison program was not executed), but still generated a file "output/output_mona_lisa.jpg" of which approximately merely $\frac{1}{6}$ looks similar to the original image (Figure 10), whereas the other areas become nothing. **Thus, YYC failed to recover original information.**


(a) original image

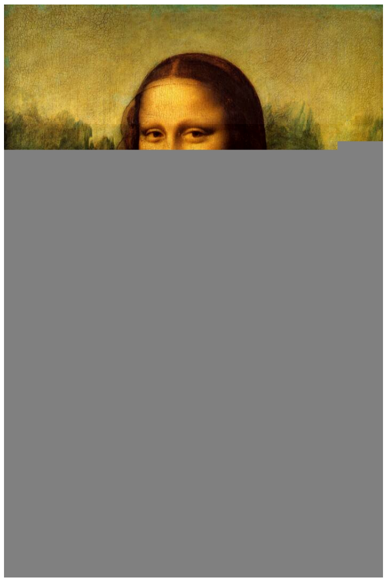
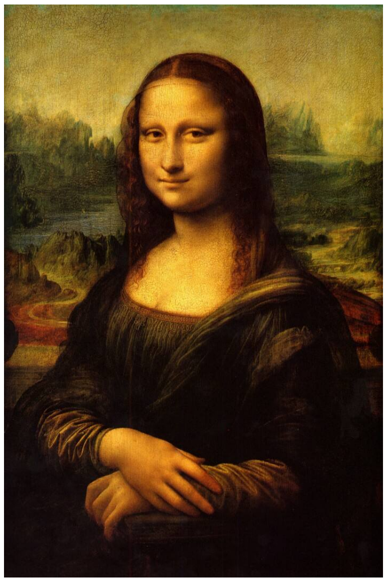

(b) decoded image

Figure 9: Comparison of original image with decoded in Test 8


(a) original image

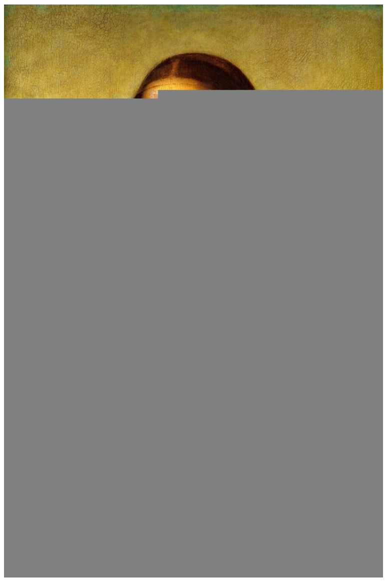

(b) decoded image

Figure 10: Comparison of original image with decoded in Test 10

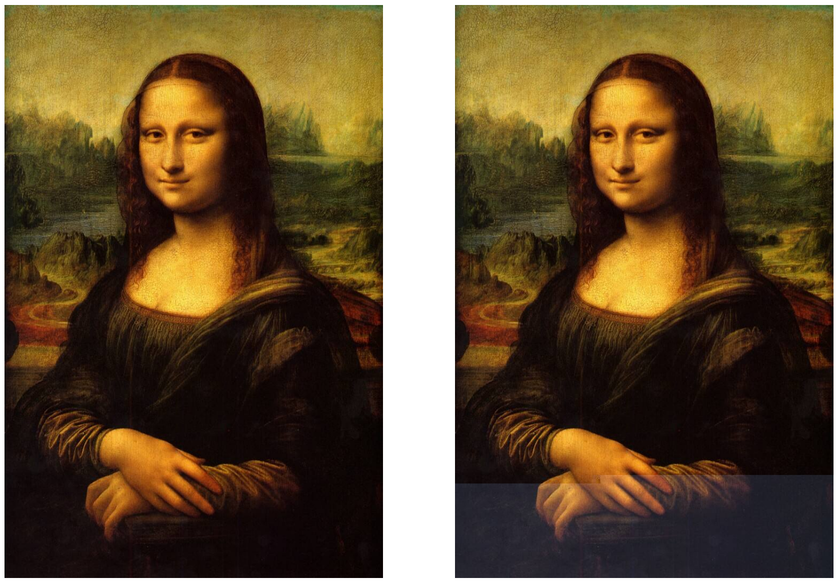

(a) original image (b) decoded image

Figure 11: Comparison of original image with decoded in Test 12

5.5 (Manual) Test 11: insert one nucleotide to before the first one

See "test_11" on my github page for related computer programs, input and output. Run "test_mona_lisa_encode.py". Make a copy of output "output/mona_lisa.dna" with name "output/mona_lisa.dna_new". Insert a nucleotide "C" to before the first one. Run "test_mona_lisa_decode.py".

The decode process generated a 285-byte file "output/output_mona_lisa.jpg" that is not a valid image. Note that the original image "files/Mona Lisa.jpg" has 97530 bytes. The comparison program printed "false" meaning they were not the same file. **Thus, YYC failed to recover original information.**

5.6 (Manual) Test 12: insert one nucleotide to before the last one

See "test_12" on my github page for related computer programs, input and output. Run "test_mona_lisa_encode.py". Make a copy of output "output/mona_lisa.dna" with name "output/mona_lisa.dna_new". Insert a nucleotide "C" to before the last one. Run "test_mona_lisa_decode.py".

The decode process generated a file "output/output_mona_lisa.jpg" of which at least $\frac{1}{6}$ has different color to the original image (Figure 11). The comparison program printed "false" meaning they were not the same file. **Thus, YYC failed to recover original information.**

5.7 (Automatic) Test 13: massive random tests on different error types, error rates and files

See "test_13" on my github page for related computer programs, input and output. Run "automatic_test_robustness.py"

For each error type, namely substitution, deletion, insertion or mix (meaning random choice of any of the other 3 types), at different error rates 0.0001, 0.0010, 0.0030, 0.0050, 0.0080 or 0.0100, **1000** many tests were performed on each file of "./files/mon_lisa.jpg" and "./files/unite_nation_flag.bmp" with YYC scheme No.888 (rather than No.496 which fails randomly on "./files/unite_nation_flag.bmp"). To avoid ambiguity, these tests are called *sub-tests*. Thus, 48 (=4*6*2) groups of sub-tests were carried out; each group contained 1000 sub-tests. Totally there were **48000** sub-tests. The test results were written in "robustness.txt" group by group. In each group, the number of successful recoveries was counted; the success rate defined as quotient of the number of successful recoveries divided by 1000 was calculated.

The file "robustness.txt" shows, in all groups, the number of successful recoveries were all 0, so success rates were all 0 too. **YYC failed to recover original information in any sub-test.**

5.8 (Automatic) Test 14: 48 random tests on different error types, error rates and files

See "test_14" on my github page for related computer programs, input and output. Run "automatic_test_robustness.py"

For each error type, namely substitution, deletion, insertion or mix (meaning random choice of any of the other 3 types), at different error rates 0.0001, 0.0010, 0.0030, 0.0050, 0.0080 or 0.0100, **1** test were performed on each file of "./files/mon_lisa.jpg" and "./files/unite_nation_flag.bmp" with YYC scheme No.888 (rather than No.496 which fails randomly on "./files/unite_nation_flag.bmp"). To avoid ambiguity, these tests are called *sub-tests*. Thus, 48 (=4*6*2) groups of sub-tests were carried out; each group contained 1 sub-tests. Totally there were **48** sub-tests. The test results were written in "robustness.txt" group by group. In each group, the number of successful recoveries was counted (it's 0 or 1); the success rate defined as quotient of the number of successful recoveries divided by 1 was calculated. All files of DNA sequences, randomly modified DNA sequences, decoded images were saved in "./output".

(Automatic) Test 14 differs from (Automatic) Test 13 in the number of sub-tests and that all output files were saved for the sake of examination rather than deleted for the sake of disk space.

43 decoded files were generated in "./output", which implies 5 of the 48 sub-tests did not even manage to finish decode process. Among the 43 decoded files, 21 files are decoded "mona_lisa" with file names in the form of "output/output_mona_lisa_Error_Rate_Error_Type_subTest_Number.jpg". The size of these 21 files are all less or equal to 314 bytes, which is less than the size of original image "files/mona_lisa.jpg" (97530 bytes). What's worse, none of the 21 files are valid images.

Among the 43 decoded files, 22 files are decoded "united_nations_flag" with file names in the form of "output/output_united_nations_flag_Error_Rate_Error_Type_subTest_Number.jpg". 10 of the 22 files are of size tens of bytes; 10 of the 22 files are of size hundreds of bytes; the other 2 of the 22 files have 6180 and 25845 bytes respectively. They are all smaller than the original image "files/united_nations_flag.bmp" (480056 bytes). What's worse, none of the 22 files are valid images.

The file "robustness.txt" shows, in all groups, the number of successful recoveries were all 0, so success rates were all 0 too. **YYC failed to recover original information in any sub-test.**

5.9 Conclusions

5.9.1 Conclusions to manual tests

YYC failed to recover original information in any test. The robustness results of paper YYC (Figure 8) are lies. How despicable the fraud is!

Among 3 of the 6 tests, nothing was recovered at all. Imagine someone pay an immense amount of money to encode some important files into DNA powder or solution and is told later nothing can be restored merely because one of those millions or billions of nucleotides was distorted, missing or superfluous. What a funny joke!

Among the other 3 tests, seemingly $\frac{1}{4}$, $\frac{1}{6}$ or $\frac{5}{6}$ of the original image was retrieved, which might look positive. However, in practice, users are unlikely to feel pleased under such situations. What's worse, shoud the original information be replaced with other types of files such as compressed archive (which should be the case of practical applications for the sake of money) or executable programs, getting some "debris" still means nothing (because such debris does not have valid format, so can not be opened or used).

In these tests, YYC can not correct even one error, which clearly conflicts with the capability of RS error correction method. It is likely paper YYC did not use RS method so as to improve information density.

5.9.2 Conclusions to automatic tests

YYC failed to recover original information in any of the 48048 sub-tests. The robustness results of paper YYC (Figure 8) are lies. How despicable the fraud is!

In some sub-tests, YYC decode process did not even manage to finish. Nothing was generated at all. In other sub-tests, files that are not even valid images were generated.

6 How a filthy garbage paper like YYC got published

Such a filthy garbage paper should be thrown to thermal power station, where garbage gets burned so as to contribute its pathetic meagre value to human society. How was such a despicable garbage paper full of fake data, lies and traps published?

The peer reviewer reports available in the same page of article YYC show both reviewer #3 and reviewer #4 found various deadly issues which were never addressed and instead neglected. Questions on real performances of YYC were also raised. However, they received answers consisting of unrelated vacuous information generated in a way ancient malfunctioned robots prior to ChatGPT would do.

It's surprising 15 authors worked together to create such crap. What a brilliant effort!

References