RaCAF ANR-15-CE40-0016-01: Dépasser les frontières de l'aléatoire et du calculable (Randomness and Computability: Advancing the Frontiers)

Alexander Shen,
LIRMM CNRS & Univ. Montpellier

March 22, 2018

Random objects from different perspectives

Random objects from different perspectives

- Classical probability theory: random variables

Random objects from different perspectives

- Classical probability theory: random variables
- Algorithmic randomness: random bit sequence, real number in $(0, 1)$, bit string

Random objects from different perspectives

- Classical probability theory: random variables
- Algorithmic randomness: random bit sequence, real number in $(0,1)$, bit string
- Numerical computations: pseudorandom number generators and statistical tests

Random objects from different perspectives

- Classical probability theory: random variables
- Algorithmic randomness: random bit sequence, real number in $(0, 1)$, bit string
- Numerical computations: pseudorandom number generators and statistical tests
- Complexity theory and computational cryptography: pseudorandom number generators (Blum, Micali)

Random objects from different perspectives

- Classical probability theory: random variables
- Algorithmic randomness: random bit sequence, real number in $(0, 1)$, bit string
- Numerical computations: pseudorandom number generators and statistical tests
- Complexity theory and computational cryptography: pseudorandom number generators (Blum, Micali)
- Combinatorics: randomness extractors

Classical probability theory

Classical probability theory

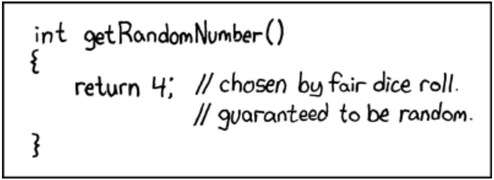
- Random variable: mapping defined on a probability space

Classical probability theory

- Random variable: mapping defined on a probability space
- No such thing as an individual random object

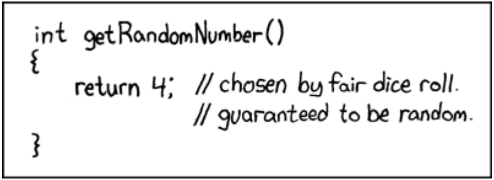
Classical probability theory

- Random variable: mapping defined on a probability space
- No such thing as an individual random object



Classical probability theory

- Random variable: mapping defined on a probability space
- No such thing as an individual random object



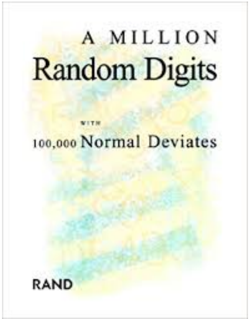
- of course, we usually speak about sequences, not individual digits

Tables of random numbers

. . . but the question remains:

Tables of random numbers

... but the question remains:



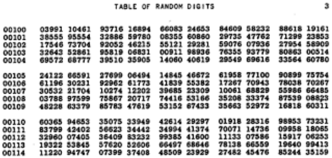


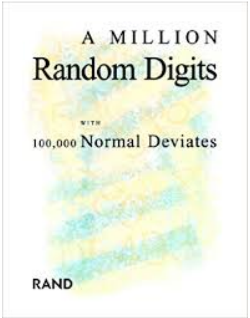
TABLE OF RANDOM DIGITS 3

00100	03991	10461	93716	16894	66083	24653	84609	58232	88618	19161
00101	38555	95554	32886	59780	08355	60860	29735	47762	71299	23853
00102	17546	73704	92052	46215	55121	29281	59076	07936	27954	58909
00103	32643	52861	95819	06831	00911	98936	76355	93779	80863	00514
00104	69572	68777	39510	35905	14060	40619	29549	69616	33564	60780
00105	24122	66591	27699	06494	14845	46672	61958	77100	90899	75754
00106	61196	30231	92962	61773	41839	55382	17267	70943	78038	70267
00107	30532	21704	10274	12202	39685	23309	10061	68829	55986	66485
00108	03788	97599	75867	20717	74416	53166	35208	33374	87539	08823
00109	48228	63379	85783	47619	53152	67433	35663	52972	16818	60311
00110	60365	94653	35075	33949	42614	29297	01918	28316	98953	73231
00111	83799	42402	56623	34442	34994	41374	70071	14736	09958	18065
00112	32960	07405	36409	83232	99385	41600	11133	07586	15917	06253
00113	19322	53845	57620	52606	66497	68646	78138	66559	19640	99413
00114	11220	94747	07399	37408	48509	23929	27482	45476	85244	35159

($600 hardcover, $41 paperback, $9 digital)

Tables of random numbers

. . . but the question remains:



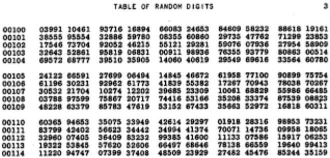


TABLE OF RANDOM DIGITS 3

00100	03991	10461	93716	16894	66083	24653	84609	58232	88618	19161
00101	38555	95554	32886	59780	08355	60860	29735	47762	71299	23853
00102	17546	73704	92052	46215	55121	29281	59076	07936	27954	58909
00103	32643	52861	95819	06831	00911	98936	76355	93779	80863	00514
00104	69572	68777	39510	35905	14060	40619	29549	69616	33564	60780
00105	24122	66591	27699	06494	14845	46672	61958	77100	90899	75754
00106	61196	30231	92962	61773	41839	55382	17267	70943	78038	70267
00107	30532	21704	10274	12202	39685	23309	10061	68829	55986	66485
00108	03788	97599	75867	20717	74416	53166	35208	33374	87539	08823
00109	48228	63379	85783	47619	53152	67433	35663	52972	16818	60311
00110	60365	94653	35075	33949	42614	29297	01918	28316	98953	73231
00111	83799	42402	56623	34442	34994	41374	70071	14736	09958	18065
00112	32960	07405	36409	83232	99385	41600	11133	07586	15917	06253
00113	19322	53845	57620	52606	66497	68646	78138	66559	19640	99413
00114	11220	94747	07399	37408	48509	23929	27482	45476	85244	35159

($600 hardcover, $41 paperback, $9 digital)
can we complain to amazon if “non-random”?

Algorithmic randomness

Algorithmic randomness

- finite or infinite objects

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- “randomness deficiency” for finite

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- “randomness deficiency” for finite
- randomness=incompressibility

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- “randomness deficiency” for finite
- randomness=incompressibility
- Kolmogorov complexity = length of the generating program

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- "randomness deficiency" for finite
- randomness=incompressibility
- Kolmogorov complexity = length of the generating program
- randomness deficiency=length−complexity

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- “randomness deficiency” for finite
- randomness=incompressibility
- Kolmogorov complexity = length of the generating program
- randomness deficiency=length−complexity
- non-computable

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- “randomness deficiency” for finite
- randomness=incompressibility
- Kolmogorov complexity = length of the generating program
- randomness deficiency=length−complexity
- non-computable
- dependence on the programming language

Algorithmic randomness

- finite or infinite objects
- sharp boundary for infinite (Martin-Löf)
- "randomness deficiency" for finite
- randomness=incompressibility
- Kolmogorov complexity = length of the generating program
- randomness deficiency=length−complexity
- non-computable
- dependence on the programming language
- (our main field of expertise)

Pseudo-random number generators

Pseudo-random number generators

- $x_{n+1} = (1103515245 \cdot x_n + 12345) \bmod 2^{32}$

Pseudo-random number generators

- $x_{n+1} = (1103515245 \cdot x_n + 12345) \bmod 2^{32}$
- many other generators (not only linear)

Pseudo-random number generators

- $x_{n+1} = (1103515245 \cdot x_n + 12345) \bmod 2^{32}$
- many other generators (not only linear)
- used for simulations

Pseudo-random number generators

- $x_{n+1} = (1103515245 \cdot x_n + 12345) \bmod 2^{32}$
- many other generators (not only linear)
- used for simulations
- Monte-Carlo method

Pseudo-random number generators

- $x_{n+1} = (1103515245 \cdot x_n + 12345) \bmod 2^{32}$
- many other generators (not only linear)
- used for simulations
- Monte-Carlo method
- easily computable and predictable

Pseudo-random number generators

- $x_{n+1} = (1103515245 \cdot x_n + 12345) \bmod 2^{32}$
- many other generators (not only linear)
- used for simulations
- Monte-Carlo method
- easily computable and predictable
- why better than $x_{n+1} = x_n + 1 \bmod 2^{32}$?

Statistical tests

Statistical tests

- diehard (G. Marsaglia, originally available at `http://stat.fsu.edu/pub/diehard`)

Statistical tests

- diehard (G. Marsaglia, originally available at `http://stat.fsu.edu/pub/diehard`)
- `dieharder` linux package (R. Brown, `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`

Statistical tests

- diehard (G. Marsaglia, originally available at `http://stat.fsu.edu/pub/diehard`)
- `dieharder` linux package (R. Brown, `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`
- NIST, `https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software`

Statistical tests

- diehard (G. Marsaglia, originally available at `http://stat.fsu.edu/pub/diehard`)
- `dieharder` linux package (R. Brown, `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`
- NIST, `https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software`
- TestU01, `https://www.iro.umontreal.ca/~lecuyer/myftp/papers/testu01.pdf`

What is a statistical test?

What is a statistical test?

- input: bit sequence r (typical length: 10^7)

What is a statistical test?

- input: bit sequence r (typical length: 10^7)
- output: not just “pass” or “fail”, but some real $p(r) \in (0, 1)$

What is a statistical test?

- input: bit sequence r (typical length: 10^7)
- output: not just "pass" or "fail", but some real $p(r) \in (0, 1)$
- "p-value": $\Pr_r[p(r) \leqslant \varepsilon] = \varepsilon$

What is a statistical test?

- input: bit sequence r (typical length: 10^7)
- output: not just "pass" or "fail", but some real $p(r) \in (0, 1)$
- "p-value": $\Pr_r[p(r) \leqslant \varepsilon] = \varepsilon$
- if p-value is below some threshold, say, 10^{-5}, we say that r looks non-random with p-value 10^{-5}

What is a statistical test?

- input: bit sequence r (typical length: 10^7)
- output: not just "pass" or "fail", but some real $p(r) \in (0, 1)$
- "p-value": $\Pr_r[p(r) \leqslant \varepsilon] = \varepsilon$
- if p-value is below some threshold, say, 10^{-5}, we say that r looks non-random with p-value 10^{-5}
- "r is random with probability 10^{-5}": misleading

What is a statistical test?

- input: bit sequence r (typical length: 10^7)
- output: not just "pass" or "fail", but some real $p(r) \in (0,1)$
- "p-value": $\Pr_r[p(r) \leqslant \varepsilon] = \varepsilon$
- if p-value is below some threshold, say, 10^{-5}, we say that r looks non-random with p-value 10^{-5}
- "r is random with probability 10^{-5}": misleading
- compressors as random tests: compression by k bits corresponds to p-value below 2^{-k}

Randomness and cryptography

Randomness and cryptography

- password/secret key from the random table book: a bad idea

Randomness and cryptography

- password/secret key from the random table book: a bad idea
- password generated by a pseudorandom number generator?

Randomness and cryptography

- password/secret key from the random table book: a bad idea
- password generated by a pseudorandom number generator? seeds needed

Randomness and cryptography

- password/secret key from the random table book: a bad idea
- password generated by a pseudorandom number generator? seeds needed
- complexity-based cryptography: Yao – Blum – Micali

Randomness and cryptography

- password/secret key from the random table book: a bad idea
- password generated by a pseudorandom number generator? seeds needed
- complexity-based cryptography: Yao – Blum – Micali
- PRNG: short seed $\rightarrow$ long bit string

Randomness and cryptography

- password/secret key from the random table book: a bad idea
- password generated by a pseudorandom number generator? seeds needed
- complexity-based cryptography: Yao – Blum – Micali
- PRNG: short seed $\to$ long bit string
- indistinguishable by tests of bounded circuit complexity

Randomness and cryptography

- password/secret key from the random table book: a bad idea
- password generated by a pseudorandom number generator? seeds needed
- complexity-based cryptography: Yao – Blum – Micali
- PRNG: short seed $\to$ long bit string
- indistinguishable by tests of bounded circuit complexity
- conditional existence (factoring is hard, the existence of one-way functions)

Physical randomness

Physical randomness

- "fast electronic coin"

Physical randomness

- “fast electronic coin”
- examples: https://en.wikipedia.org/wiki/Comparison_of_hardware_random_number_generators

Physical randomness

- “fast electronic coin”
- examples: https://en.wikipedia.org/wiki/Comparison_of_hardware_random_number_generators
- more exotic solutions: lava lamps at CloudFlare https://www.youtube.com/watch?v=1cUUfMeOijg

Physical randomness

- “fast electronic coin”
- examples: https://en.wikipedia.org/wiki/Comparison_of_hardware_random_number_generators
- more exotic solutions: lava lamps at CloudFlare https://www.youtube.com/watch?v=1cUUfMeOijg
- basic problem: distribution not under control

Physical randomness

- “fast electronic coin”
- examples: https://en.wikipedia.org/wiki/Comparison_of_hardware_random_number_generators
- more exotic solutions: lava lamps at CloudFlare https://www.youtube.com/watch?v=1cUUfMeOijg
- basic problem: distribution not under control
- solution attempt: “extracting randomness from weak randomness”

The project goals

The project goals

- understanding relations between different approaches

The project goals

- understanding relations between different approaches
- better understanding of existing practices and their weaknesses

The project goals

- understanding relations between different approaches
- better understanding of existing practices and their weaknesses
- making tests robust

The project goals

- understanding relations between different approaches
- better understanding of existing practices and their weaknesses
- making tests robust
- trying new type of tests

The project goals

- understanding relations between different approaches
- better understanding of existing practices and their weaknesses
- making tests robust
- trying new type of tests
- trying new ways of randomness extraction

The project goals

- understanding relations between different approaches
- better understanding of existing practices and their weaknesses
- making tests robust
- trying new type of tests
- trying new ways of randomness extraction
- this report: mostly practical aspects

The project goals

- understanding relations between different approaches
- better understanding of existing practices and their weaknesses
- making tests robust
- trying new type of tests
- trying new ways of randomness extraction
- this report: mostly practical aspects
- see below (and also pdf report) for more theoretical work

Randomness tests: problems

Randomness tests: problems

- p-value function requires exact answer for probabilities;

Randomness tests: problems

- p-value function requires exact answer for probabilities;
- in many cases only a bound available or even an empirical estimate

Randomness tests: problems

- p-value function requires exact answer for probabilities;
- in many cases only a bound available or even an empirical estimate
- unsuitable for "secondary test" when several p-values on independent inputs are tested against a uniform distribution on $(0, 1)$ with Kolmogorov–Smirnov test

Randomness tests: problems

- p-value function requires exact answer for probabilities;
- in many cases only a bound available or even an empirical estimate
- unsuitable for "secondary test" when several p-values on independent inputs are tested against a uniform distribution on $(0, 1)$ with Kolmogorov–Smirnov test
- `diehard` uses dependent inputs when independence is required

dieharder documentation

speaks about "test failures"

> *Many dieharder tests, despite our best efforts, are numerically unstable or have only approximately known target statistics or are straight up asymptotic results, and will eventually return a failing result even for a gold-standard generator (such as AES), or for the hypercautious the XOR generator with AES, threefish, kiss, all loaded at once and xor'd together.* $\langle \ldots \rangle$ *Failure with numbers of psamples within an order of magnitude of the AES thresholds should probably be considered possible test failures, not generator failures. Failures at levels significantly less than the known gold standard generator failure thresholds are, of course, probably failures of the generator.*

More reliable approach needed

More reliable approach needed

- *any* function p (no assumptions about having p-value properties)

More reliable approach needed

- *any* function p (no assumptions about having p-value properties)
- $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(R_1), \ldots, p(R_n)$ where R_i are true random strings

More reliable approach needed

- *any* function p (no assumptions about having p-value properties)
- $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(R_1), \ldots, p(R_n)$ where R_i are true random strings
- we may use Kolmogorov–Smirnov criterion for *two* distributions

More reliable approach needed

- *any* function p (no assumptions about having p-value properties)
- $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(R_1), \ldots, p(R_n)$ where R_i are true random strings
- we may use Kolmogorov–Smirnov criterion for *two* distributions
- guaranteed to be reliable (assuming true randomness)

More reliable approach needed

- *any* function p (no assumptions about having p-value properties)
- $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(R_1), \ldots, p(R_n)$ where R_i are true random strings
- we may use Kolmogorov–Smirnov criterion for *two* distributions
- guaranteed to be reliable (assuming true randomness)
- almost as sensitive as the original test

Testing without etalon randomness

Testing without etalon randomness

- correctness depends on the etalon randomness

Testing without etalon randomness

- correctness depends on the etalon randomness
- improvement: $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(r_{n+1} \oplus R_1), \ldots, p(r_{2n} \oplus R_n)$ where R_i are presumably true random strings

Testing without etalon randomness

- correctness depends on the etalon randomness
- improvement: $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(r_{n+1} \oplus R_1), \ldots, p(r_{2n} \oplus R_n)$ where R_i are presumably true random strings
- guaranteed to be reliable (no assumptions about R_i)

Testing without etalon randomness

- correctness depends on the etalon randomness
- improvement: $p(r_1), \ldots, p(r_n)$ (where r_i are independent parts of a test stream) compared with $p(r_{n+1} \oplus R_1), \ldots, p(r_{2n} \oplus R_n)$ where R_i are presumably true random strings
- guaranteed to be reliable (no assumptions about R_i)
- equally sensitive if R_i are truly random

Probabilistic arguments as random tests

Probabilistic arguments as random tests

- combinatorial results obtained by a probabilistic method

Probabilistic arguments as random tests

- combinatorial results obtained by a probabilistic method
- object with some combinatorial properties (e.g., expander graphs) generated with high probability using random bits

Probabilistic arguments as random tests

- combinatorial results obtained by a probabilistic method
- object with some combinatorial properties (e.g., expander graphs) generated with high probability using random bits
- testing bit string r: use it as a random string in the algorithm and measure the properties of the object generated by it

Probabilistic arguments as random tests

- combinatorial results obtained by a probabilistic method
- object with some combinatorial properties (e.g., expander graphs) generated with high probability using random bits
- testing bit string r: use it as a random string in the algorithm and measure the properties of the object generated by it
- suitable for our scheme even if nothing is formally proved about the algorithm

Probabilistic arguments as random tests

- combinatorial results obtained by a probabilistic method
- object with some combinatorial properties (e.g., expander graphs) generated with high probability using random bits
- testing bit string r: use it as a random string in the algorithm and measure the properties of the object generated by it
- suitable for our scheme even if nothing is formally proved about the algorithm
- some preliminary results (M.Popov, master thesis under supervision of A.Romashchenko)

Physical random number generators

Physical random number generators

- goal: add some “unpredictability” or “entropy” from physical sources

Physical random number generators

- goal: add some "unpredictability" or "entropy" from physical sources
- "`/dev/random` typically blocks if there is less entropy available than requested", "the generator keeps an estimate of the number of bits of noise in the entropy pool", etc. (wikipedia)

Physical random number generators

- goal: add some “unpredictability” or “entropy” from physical sources
- “`/dev/random` typically blocks if there is less entropy available than requested”, “the generator keeps an estimate of the number of bits of noise in the entropy pool”, etc. (wikipedia)
- (naive) idea of “entropy” as some kind of liquid that can be measured, kept in a pool, etc. similar to caloric theory

RFC 4086

on extracting randomness from weak random source:

> *For an example of using a strong mixing function, reconsider the case of a string of 308 bits, each of which is biased 99% toward zero. The parity technique* $\langle \ldots \rangle$ *reduces this to one bit, with only a* 1/1000 *deviance from being equally likely a zero or one. But, applying the equation for information* $\langle \ldots \rangle$ *[Shannon entropy], this* 308*-bit skewed sequence contains over* 5 *bits of information. Thus, hashing it with SHA-1 and taking the bottom* 5 *bits of the result would yield* 5 *unbiased random bits and not the single bit given by calculating the parity of the string.*

RFC 4086

on extracting randomness from weak random source:

> *For an example of using a strong mixing function, reconsider the case of a string of 308 bits, each of which is biased 99% toward zero. The parity technique* $\langle \ldots \rangle$ *reduces this to one bit, with only a* $1/1000$ *deviance from being equally likely a zero or one. But, applying the equation for information* $\langle \ldots \rangle$ *[Shannon entropy], this* 308*-bit skewed sequence contains over* 5 *bits of information. Thus, hashing it with SHA-1 and taking the bottom* 5 *bits of the result would yield* 5 *unbiased random bits and not the single bit given by calculating the parity of the string.*

[Not justified: parity argument uses independence, and SHA-1 trick is not justified even in the independence case]

Alternative ways to extract randomness

Alternative ways to extract randomness

- theoretical work: randomness extractors

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random
- or two long independent weak random sources

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random
- or two long independent weak random sources
- not directly practical

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random
- or two long independent weak random sources
- not directly practical
- some practical approaches inspired by them

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random
- or two long independent weak random sources
- not directly practical
- some practical approaches inspired by them
- using expander walk over weakly random edges

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random
- or two long independent weak random sources
- not directly practical
- some practical approaches inspired by them
- using expander walk over weakly random edges
- using $B[x][y]$ where B is a balanced matrix and x and y are independent weak random sources

Alternative ways to extract randomness

- theoretical work: randomness extractors
- two inputs: long weak random and independent short truly random
- or two long independent weak random sources
- not directly practical
- some practical approaches inspired by them
- using expander walk over weakly random edges
- using $B[x][y]$ where B is a balanced matrix and x and y are independent weak random sources
- some preliminary experiments done

Planned work

Planned work

- convert some tests from standard suites in a robust form (may be incorporating them in the existing open source software)

Planned work

- convert some tests from standard suites in a robust form (may be incorporating them in the existing open source software)
- adding some new tests based on probabilistic constructions

Planned work

- convert some tests from standard suites in a robust form (may be incorporating them in the existing open source software)
- adding some new tests based on probabilistic constructions
- making experiments with existing physical randomness inputs (sound cards, physical devices) and analyzing their properties and ways to extract good random bits out of them

Planned work

- convert some tests from standard suites in a robust form (may be incorporating them in the existing open source software)
- adding some new tests based on probabilistic constructions
- making experiments with existing physical randomness inputs (sound cards, physical devices) and analyzing their properties and ways to extract good random bits out of them
- last, but not least: theoretical work to understand properties of randomness (algorithmic information theory, computability theory approach to randomness, models of computation, randomness in game-theoretic approach to probability theory, etc.)

Theoretical work

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)
- properties of randomness tests in algorithmic information theory (G.Novikov)

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)
- properties of randomness tests in algorithmic information theory (G.Novikov)
- detection of regularities and algorithmic statistics (A.Milovanov, N.Vereshchagin), presented at CCC 2017)

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)
- properties of randomness tests in algorithmic information theory (G.Novikov)
- detection of regularities and algorithmic statistics (A.Milovanov, N.Vereshchagin), presented at CCC 2017)
- conditional and image randomness (L.Bienvenu, A.Shen)

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)
- properties of randomness tests in algorithmic information theory (G.Novikov)
- detection of regularities and algorithmic statistics (A.Milovanov, N.Vereshchagin), presented at CCC 2017)
- conditional and image randomness (L.Bienvenu, A.Shen)
- randomness, normality, automatic complexity (A.Shen)

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)
- properties of randomness tests in algorithmic information theory (G.Novikov)
- detection of regularities and algorithmic statistics (A.Milovanov, N.Vereshchagin), presented at CCC 2017)
- conditional and image randomness (L.Bienvenu, A.Shen)
- randomness, normality, automatic complexity (A.Shen)
- randomness and expanders (A.Romashchenko)

Theoretical work

- physical models of computation (work of O.Bournez and his group, accepted by JACM); other non-standard models of computation (B.Durand, G.Lafitte and others)
- properties of randomness tests in algorithmic information theory (G.Novikov)
- detection of regularities and algorithmic statistics (A.Milovanov, N.Vereshchagin), presented at CCC 2017)
- conditional and image randomness (L.Bienvenu, A.Shen)
- randomness, normality, automatic complexity (A.Shen)
- randomness and expanders (A.Romashchenko)
- mutual information and its operational characterization (A.Romashchenko, with M.Zimand, Towson University)

Chronological report about RaCAF progress, including references and texts of RaCAF-related papers, http:www.lirmm.fr/~ashen/racaf.html

M. Andreev, G. Posobin, A. Shen, Plain stopping time and conditional complexities revisited, preprint, https://arxiv.org/abs/1708.08100

O. Bournez, D.S. Gracça, A. Pouly, Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length, *Journal of the ACM*, Volume 64, Issue 6, November 2017, Article No. 38

O. Bournez, A. Pouly, A Universal Ordinary Differential Equation, *International Colloquium on Automata, Language and Programming*, ICALP'2017, 116:1–116:14

B. Bauwens, A. Shen, H. Takahashi, Conditional Probabilities and van Lambalgens Theorem Revisited, *Theory of Computing Systems*, 2017, doi:10.1007/s00224-017-9789-2

M. Carl, B. Durand, G. Lafitte, S. Ouazzani, Admissible in Gaps, *CiE 2017: Unveiling Dynamics and Complexity, Proceedings*, Lecture

Notes in Computer Science, 10307, Springer, 2017, 175–186.
https://doi.org/10.1007/978-3-319-58741-7_18

J. Cervelle, G. Lafitte, On shift-invariant maximal filters and hormonal cellular automata, *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, 1–10,
https://doi.org/10.1109/LICS.2017.8005145

O. Defrain, B. Durand, G. Lafitte, Infinite Time Busy Beavers, *CiE 2017: Unveiling Dynamics and Complexity, Proceedings*, Lecture Notes in Computer Science, 10307, Springer, 2017, 221–233.
https://doi.org/10.1007/978-3-319-58741-7_22

B. Durand, A. Romashchenko, On the Expressive Power of Quasi-Periodic SFT, *Mathematical Foundations of Computer Science*, 2017, https://doi.org/10.4230/LIPIcs.MFCS.2017.5

L. Bienvenu, M. Hoyrup, A. Shen, Layerwise Computability and Image Randomness, *Theory of Computing Systems*, 2017, doi:10.1007/s00224-017-9791-8

Guilhem Marion, *Le hasard et sa production*, report de stage, LIRMM, see RaCAF diary above.

A. Milovanov, Algorithmic Statistics: Normal Objects and Universal Models, *Computer Science in Russia 2016*, Lecture Notes in Computer Science, v. 9691 (2016), 280–293.

A. Milovanov, Some Properties of Antistochastic Strings, *Theory of Computing Systems*, published online 21 June 2016, DOI 10.1007/s00224-016-9695-z.

A. Milovanov, On Algorithmic Statistics for space-bounded algorithms. In *Proceedings of 12th International Computer Science Symposium in Russia* (CSR 2017) LNCS, vol. **10304**, pp. 232–234, 2017.

A. Milovanov, N. Vereshchagin, Stochasticity in Algorithmic Statistics for Polynomial Time, *32nd Computational Complexity Conference*(CCC 2017) proceedings (Leibniz International Proceedings in Informatics, LIPIcs), doi:10.4230/LIPIcs.CCC.2017.17, 17:1–17:18

G. Novikov, Randomness Deficiences, *CiE 2017: Unveiling Dynamics and Complexity, Proceedings*, Lecture Notes in Computer Science, 10307, Springer, 2017, 338–350. https://link.springer.com/chapter/10.1007/978-3-319-58741-7_32

A. Romashchenko, *Coding in the fork network in the framework of Kolmogorov complexity*, preprint, arXiv:1602.02648.

A. Shen, Algorithmic Information Theory, book section in *The Routledge handbook of philosophy of information*, Routletge, 2016, 37–43.

A. Shen, Automatic Kolmogorov complexity and normality revisited, *FCT 2017 Conference, Bordeaux, France, Proceedings*, full version: https://arxiv.org/pdf/1701.09060.pdf

A. Shen, V. Uspensky, N. Vereshchagin, Kolmogorov Complexity and Algorithmic Randomness. A book accepted for publication (in 2017) by the American Mathematical Society. Draft: http://www.lirmm.fr/~ashen/kolmbook-eng.pdf

N. Vereshchagin, A. Shen, *Algorithmic statistics: forty years later*. Book chapter in *Computability and Complexity. Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*. Lecture Notes in Computer Science, v. 10010, Springer, 2017, p. 669–737.