Supplementary material for article: Kmer-db: instant evolutionary distance estimation

Sebastian Deorowicz Adam Gudyś Maciej Długosz Marek Kokot
Agnieszka Danek

June 13, 2018

Contents

1 Illustration of the data structure

Figures 1–5 show the contents of the Kmer-db data structure after adding successive samples (one per figure). The only parts stored in practice are yellow-headed tables, the remaining are given for clearer presentation. Let us focus at Figure 2 which shows data structure after processing two genomes ("Samples" table). Table "k-mers" presents all unique k-mers in these genomes, each assigned with a list of sample ids containing it. E.g., `CTGGA` is present in samples 0 and 1. The "Templates" table contains all unique lists of sample ids. After processing two samples there are 3 such unique lists, called templates: (`0`), (`1`), (`0,1`).

The "k-mers to templates" ($K2T$) table is essentially the same as the "k-kmers" table but lists of sample ids are replaced by template ids. The templates are stored in a compressed way ("compacted templates" table) in which shared parts of the lists of ids are merged. E.g., template `3` contains sample id `1` as well as all sample ids present in its parental template `1`.

As was said above, in practice only $K2T$ and CT tables (yellow-headed) are stored. The internal representation is as follows:

- "k-mer to templates" ($K2T$) is a hash table with linear probing (to support fast queries); the entries are:
  - k-mer — 64-bit integer, which allows to store k-mers up to $k = 32$, alternatively, the 64-bit minhash value can be stored here,
  - t_{id} — 64-bit integer pointing to CT table;
- "compacted templates" (CT) is an array of entries:
  - s_{id} — 32-bit integer for sample id,
  - p_{id} — 64-bit integer for id of parent template id,
  - list of Elias-gamma-coded sample ids (s_{id} list) stored in array of size rounded up to 16 bytes.

Samples

s_{id}	Sequence
0	ACTGGATGCAG

k-mers

k-mer	s_{id} list
ACTGG	0
ATGCA	0
CTGGA	0
GATGC	0
GGATG	0
TGCAG	0
TGGAT	0

Templates

t_{id}	s_{id} list
1	0

$K2T$ (k-mers to templates)

k-mer	t_{id}
ACTGG	1
ATGCA	1
CTGGA	1
GATGC	1
GGATG	1
TGCAG	1
TGGAT	1

CT (compacted templates)

t_{id}	s_{id} list	p_{id}
1	0	—

Figure 1: Database state ($k = 5$) after processing one sample. Symbols: s_{id}—sample id, t_{id}—template id, p_{id}—parent template id.

Samples

s_{id}	Sequence
0	ACTGGATGCAG
1	GCTGGATGGAG

k-mers

k-mer	s_{id} list
ACTGG	0
ATGCA	0
ATGGA	1
CTGGA	0, 1
GATGC	0
GATGG	1
GCTGG	1
GGATG	0, 1
TGCAG	0
TGGAG	1
TGGAT	0, 1

Templates

t_{id}	s_{id} list
1	0
2	1
3	0, 1

$K2T$ (k-mers to templates)

k-mer	t_{id}
ACTGG	1
ATGCA	1
ATGGA	2
CTGGA	3
GATGC	1
GATGG	2
GCTGG	2
GGATG	3
TGCAG	1
TGGAG	2
TGGAT	3

CT (compacted templates)

t_{id}	s_{id} list	p_{id}
1	0	—
2	1	—
3	1	1

Figure 2: Database state ($k = 5$) after processing 2 samples. Symbols: s_{id}—sample id, t_{id}—template id, p_{id}—parent template id.

Samples

s_{id}	Sequence
0	ACTGGATGCAG
1	GCTGGATGGAG
2	ACTGGATGGAG

k-mers

k-mer	s_{id} list
ACTGG	0, 2
ATGCA	0
ATGGA	1, 2
CTGGA	0, 1, 2
GATGC	0
GATGG	1, 2
GCTGG	1
GGATG	0, 1, 2
TGCAG	0
TGGAG	1, 2
TGGAT	0, 1, 2

Templates

t_{id}	s_{id} list
1	0
2	1
3	0, 1, 2
4	0, 2
5	1, 2

$K2T$ (k-mers to templates)

k-mer	t_{id}
ACTGG	4
ATGCA	1
ATGGA	5
CTGGA	3
GATGC	1
GATGG	5
GCTGG	2
GGATG	3
TGCAG	1
TGGAG	5
TGGAT	3

CT (compacted templates)

t_{id}	s_{id} list	p_{id}
1	0	—
2	1	—
3	1, 2	1
4	2	1
5	2	2

Figure 3: Database state ($k = 5$) after processing 3 samples. Symbols: s_{id}—sample id, t_{id}—template id, p_{id}—parent template id.

Samples

s_{id}	Sequence
0	ACTGGATGCAG
1	GCTGGATGGAG
2	ACTGGATGGAG
3	ATGCAGTTGGT

k-mers

k-mer	s_{id} list
ACTGG	0, 2
AGTTG	3
ATGCA	0, 3
ATGGA	1, 2
CAGTT	3
CTGGA	0, 1, 2
GATGC	0
GATGG	1, 2
GCAGT	3
GCTGG	1
GGATG	0, 1, 2
GTTGG	3
TGCAG	0, 3
TGGAG	1, 2
TGGAT	0, 1, 2
TTGGT	3

Templates

t_{id}	s_{id} list
1	0
2	1
3	0, 1, 2
4	0, 2
5	1, 2
6	0, 3
7	3

$K2T$ (k-mers to templates)

k-mer	t_{id}
ACTGG	4
AGTTG	7
ATGCA	6
ATGGA	5
CAGTT	7
CTGGA	3
GATGC	1
GATGG	5
GCAGT	7
GCTGG	2
GGATG	3
GTTGG	7
TGCAG	6
TGGAG	5
TGGAT	3
TTGGT	7

CT (compacted templates)

t_{id}	s_{id} list	p_{id}
1	0	—
2	1	—
3	1, 2	1
4	2	1
5	2	2
6	3	1
7	3	—

Figure 4: Database state ($k = 5$) after processing 4 samples. Symbols: s_{id}—sample id, t_{id}—template id, p_{id}—parent template id.

Samples

s_{id}	Sequence
0	ACTGGATGCAG
1	GCTGGATGGAG
2	ACTGGATGGAG
3	ATGCAGTTGGT
4	CGCAGTTGGT

k-mers

k-mer	s_{id} list
ACTGG	0, 2
AGTTG	3, 4
ATGCA	0, 3
ATGGA	1, 2
CAGTT	3, 4
CGCAG	4
CTGGA	0, 1, 2
GATGC	0
GATGG	1, 2
GCAGT	3, 4
GCTGG	1
GGATG	0, 1, 2
GTTGG	3, 4
TGCAG	0, 3
TGGAG	1, 2
TGGAT	0, 1, 2
TTGGT	3, 4

Templates

t_{id}	s_{id} list
1	0
2	1
3	0, 1, 2
4	0, 2
5	1, 2
6	0, 3
7	3, 4
8	4

$K2T$ (k-mers to templates)

k-mer	t_{id}
ACTGG	4
AGTTG	7
ATGCA	6
ATGGA	5
CAGTT	7
CGCAG	8
CTGGA	3
GATGC	1
GATGG	5
GCAGT	7
GCTGG	2
GGATG	3
GTTGG	7
TGCAG	6
TGGAG	5
TGGAT	3
TTGGT	7

CT (compacted templates)

t_{id}	s_{id} list	p_{id}
1	0	—
2	1	—
3	1, 2	1
4	2	1
5	2	2
6	3	1
7	3, 4	—
8	4	—

Figure 5: Database state ($k = 5$) after processing 5 samples. Symbols: s_{id}—sample id, t_{id}—template id, p_{id}—parent template id.

2 Examined programs

The following programs were used in the experimental part. The running parameters are also given.

- Mash v. 2.0:
  Mash was downloaded from `https://github.com/marbl/Mash/releases`.
  - To perform all-to-all comparison we run Mash with the following commands:
    ```
    mash sketch -p <threads> -k <k> -s <sketch_size> -o reference -l <list_of_fasta>
    mash dist -p <threads> -t reference.msh reference.msh > table
    ```
  - To perform one-to-all comparison we run Mash with the following commands:
    ```
    mash sketch -p <threads> -k <k> -s <sketch_size> -o reference_fastq -l <list_of_fastq>
    mash dist -p <threads> -t reference.msh reference_fastq.msh > table
    ```
- Kmer-db v. 1.1 with KMC v. 3.0.1:
  Kmer-db was downloaded from `https://github.com/refresh-bio/kmer-db`
  KMC was downloaded from `https://github.com/refresh-bio/KMC`.
  - To perform all-to-all comparison we run Kmer-db with the following commands:
    ```
    kmer-db-1.1 build -k <k> <list_of_fasta> kmers_build
    kmer-db-1.1 all2all kmers_build matrix
    ```
  - To perform one-to-all comparison we run Kmer-db with the following commands:
    ```
    kmc -k<k> -r -ci5 -fq <fastq_file_name> fastq_db tmp
    kmer-db-1.1 build -k <k> <list_of_fasta> kmers_build
    kmer-db-1.1 one2all kmers_build fastq_db vector
    ```
  - To perform all-to-all comparison with k-mer filtration we run Kmer-db with the following commands:
    ```
    kmer-db-1.1 build -f <filter_value> -k <k> <list_of_fasta> kmers_build
    kmer-db-1.1 all2all kmers_build matrix
    ```
  - To perform one-to-all comparison with k-mer filtration we run Kmer-db with the following commands:
    ```
    kmc -k<k> -r -ci5 -fq <fastq_file_name> fastq_db tmp
    kmer-db-1.1 build -f <filter_value> -k <k> <list_of_fasta> kmers_build
    kmer-db-1.1 one2all kmers_build fastq_db vector
    ```

  The algorithms parameters were:

- `threads` – number of threads (48),
- `k` – k-mer length (20),
- `list_of_fasta` – file containing names of genome files listed in a specified order,
- `fastq_file_name` – name of a file with reads (Kmer-db),
- `filter_value` – threshold for k-mers filtering (Kmer-db),
- `list_of_fastq` – file containing name of a file with reads (Mash),
- `sketch_size` – sketch size (Mash).

3 Datasets

3.1 Complete genomes

As a set of genomes we used 40,715 FASTA files downloaded from `https://www.ncbi.nlm.nih.gov/pathogens/` (all assembled genomes available to download on Nov 14, 2017). The complete list of file names can be found at `https://github.com/refresh-bio/kmer-db`. This set was used to prepare the database and run the *all2all* mode.

3.2 Sequenced reads

The datasets for *one2all* tests were downloaded also from `https://www.ncbi.nlm.nih.gov/pathogens/` with fastq-dump. The accession numbers are given in Table 1. We used a single sample for each species.

Table 1: *one2all* mode datasets accession numbers

Pathogen name	Reads accession numbers
Salmonella enterica	SRR3219070
E.coli and Shigella	SRR5385225
Listeria monocytogenes	SRR974852
Campylobacter jejuni	SRR4106490
Mycobacterium tuberculosis	ERR133980
Klebsiella pneumoniae	SRR5666403
Acinetobacter baumannii	SRR2558787
Neisseria	SRR969389
Pseudomonas aeruginosa	SRR001657 SRR001656
Enterobacter	SRR005471
Clostridioides difficile	ERR1015522
Vibrio parahaemolyticus	SRR873695
Legionella pneumophila	ERR376714
Serratia marcescens	SRR3927534
Klebsiella oxytoca	SRR2965664
Cronobacter	SRR1771714
Citrobacter freundii	SRR1631758
Elizabethkingia anophelis	SRR3105503
Staphylococcus pseudintermedius	SRR5875113
Providencia alcalifaciens	SRR036284
Morganella morganii	SRR1919351
Kluyvera intermedia	SRR2965721

4 Environment

The computer used in test was of the following configuration:

- 2 Intel Xeon E5-2670 v3 CPUs, 12 double-threaded cores per CPU, each clocked at 2.3 GHz,
- 256 GiB RAM,
- 6 HDDs of size 1 TiB each in RAID-5 configuration, `hdparm -t` reported buffered read speed 528.45 MB/s.

For compilation we used G++ v. 7.2.0. The machine was running Debian 9.3.1 x86-64 OS.

5 Additional results

Table 3 shows detailed results of construction of Kmer-db database and calculation of distance matrix for k-mer size $k = 20$. The meaning of the columns is as follows:

- No. samp. — number of samples in the input dataset.
- *build* processing time — time of construction of Kmer-db database from assembled genomes. This stage is necessary if we want to include new genomes to the analysis.
- *all2all* processing time — time of calculation of the distance matrix from Kmer-db database. This stage is also necessary when we include new genomes to the analysis.
- total processing time — total time of determination of the distance matrix from the genomes (*build* + *all2all*).
- RAM [GB] — maximal amount of memory allocated in the analysis stages (build, all2all). The amount of memory in KMC stage is much smaller.
- No. k-mers [M] — total number of unique k-mers (expressed in millions) in all input genomes.
- No. temp. [M] — total number of templates (expressed in millions) in Kmer-db database.
- Sum of cells [G] — sum of all cells in the output (triangular) distance matrix (expressed in billions). This says exactly how many single additions must be made by any naive algorithm that compares samples in pairs k-mer by k-mer.
- No. of add. [G] — total number of additions made by Kmer-db. This value is smaller than "sum of cells" since Kmer-db increments the counter by number of k-mers pointing to a single template.

The cause of a stepped increase of RAM usage for growing number of samples in the input set is the internal representation. For datasets examined in the article the *K2T* table dominates the memory usage. It is implemented as a hash table with linear probing of size being a power of 2. Thus, when the filling factor achieves the limit (by default set to 0.8) the size of the hash table is doubled. This can be observed for 2,000 samples and 10,000 samples).

Table 4 shows similar information, but the input sets of k-mers were filtered using minhash to retain desired fraction of elements (similarly as made by Mash). There is a single additional column:

- minhash processing time — time necessary to filter the KMC databases. This stage is made only once for each assembled genome, so we do not include the time in the total time of analyses.

Table 2: Mean absolute error (MAE) of Mash distance estimation w.r.t. actual Mash distance (calculated on the basis of all k-mers). Evaluation was done on the set of 500 *E. coli* genomes from Ondov et al (2016) study.

Mash		Kmer-db	
sketch	error	fraction	error
1,000	0.001053	0.02%	0.000930
10,000	0.000442	0.2%	0.000269
100,000	0.000122	2%	0.000124

Table 3: Detailed results of running the *build* and *all2all* modes for various orderings of input data and various numbers of samples in the input set

No. samp.	Processing time [h:mm:ss]			RAM [GB]	No. k-mers [M]	No. temp. [M]	Sum of cells [G]	No. of. add. [G]
	build	*all2all*	total					
Species tax id ordering (default)								
1,000	0:04:21	0:00:15	0:04:36	15.2	361	7.8	307	42
2,000	0:07:22	0:00:24	0:07:46	27.9	459	12.8	1,395	246
5,000	0:15:00	0:01:33	0:16:33	28.8	678	26.7	8,540	2,201
10,000	0:30:09	0:03:40	0:33:49	55.2	860	40.3	28,400	6,537
20,000	0:52:25	0:13:10	1:05:35	57.4	1,039	58.8	107,256	28,280
40,715	1:32:05	1:27:19	2:59:24	60.7	1,406	97.3	412,913	164,102
Lexicographical ordering								
1,000	0:04:25	0:00:16	0:04:41	14.8	361	8.1	307	56
2,000	0:07:02	0:00:30	0:07:32	27.8	459	13.4	1,395	323
5,000	0:14:21	0:01:56	0:16:17	27.8	678	27.7	8,540	2,631
10,000	0:27:11	0:04:48	0:31:59	56.2	860	41.8	28,400	7,573
20,000	0:46:43	0:18:06	1:04:49	56.2	1,039	60.7	107,256	31,457
40,715	1:26:17	1:53:01	3:19:18	57.4	1,406	99.9	412,913	175,133
Random ordering								
1,000	0:04:13	0:00:15	0:04:28	14.8	361	8.4	307	53
2,000	0:07:08	0:00:30	0:07:38	28.0	459	13.8	1,395	296
5,000	0:14:53	0:02:09	0:17:02	28.3	678	28.4	8,540	2,577
10,000	0:29:55	0:06:12	0:36:07	59.1	860	42.7	28,400	7,608
20,000	0:49:55	0:26:24	1:16:19	60.6	1,039	62.2	107,256	31,272
40,715	1:29:57	2:58:50	4:28:47	79.6	1,406	102.3	412,913	180,729
Tax id. ordering								
1,000	0:04:13	0:00:15	0:04:28	14.8	361	8.1	307	56
2,000	0:07:02	0:00:30	0:07:32	27.8	459	13.3	1,395	317
5,000	0:14:32	0:01:53	0:16:25	27.9	678	27.4	8,540	2,626
10,000	0:28:48	0:04:36	0:33:24	56.1	860	41.5	28,400	7,664
20,000	0:51:10	0:13:14	1:04:24	56.3	1,039	60.4	107,256	28,828
40,715	1:33:04	1:23:06	2:56:10	59.3	1,406	99.6	412,913	169,859

Table 4: Detailed results of running the *build* and *all2all* modes for various sizes of minhash filter (preserving from 0.0002 to 0.1 of input k-mers). Species tax id ordering was used here.

No. samp.	Processing time [mm:ss] *build*	*all2all*	total	RAM [GB]	No. k-mers [K]	No. temp. [K]	Sum of cells [M]	No. of. add. [M]
				fraction 0.0002				
1,000	00:08	00:00	00:08	1.3	73	20	63	51
2,000	00:17	00:01	00:18	1.4	92	28	284	249
5,000	00:40	00:01	00:41	1.4	136	51	1,740	1,632
10,000	01:14	00:05	01:19	1.4	173	73	5,762	4,825
20,000	02:23	00:15	02:38	1.4	208	99	21,661	19,345
40,715	05:06	01:03	06:09	3.2	282	151	83,515	80,104
				fraction 0.001				
1,000	00:16	00:00	00:16	1.4	361	77	304	231
2,000	00:19	00:01	00:20	1.4	459	113	1,382	1,165
5,000	00:49	00:02	00:51	1.4	679	204	8,480	7,775
10,000	01:18	00:05	01:23	1.5	861	291	28,382	22,769
20,000	02:19	00:18	02:37	1.5	1,039	402	107,033	92,924
40,715	04:33	01:13	05:46	3.3	1,408	617	410,979	389,500
				fraction 0.002				
1,000	00:07	00:00	00:07	1.3	723	138	610	444
2,000	00:14	00:01	00:15	1.4	918	206	2,774	2,267
5,000	00:37	00:02	00:39	1.4	1,359	373	17,004	15,361
10,000	01:13	00:06	01:19	1.5	1,722	533	56,695	44,473
20,000	02:27	00:20	02:47	1.5	2,080	737	214,128	182,617
40,715	04:58	01:25	06:23	3.4	2,816	1,136	823,034	774,167
				fraction 0.005				
1,000	00:09	00:01	00:10	1.5	1,807	298	1,528	1,062
2,000	00:19	00:01	00:20	1.4	2,294	453	6,944	5,470
5,000	00:54	00:03	00:57	1.6	3,393	826	42,536	37,545
10,000	01:47	00:07	01:54	1.7	4,300	1,179	141,717	107,412
20,000	03:21	00:27	03:48	1.7	5,195	1,641	535,362	443,478
40,715	07:38	01:57	09:35	3.7	7,031	2,540	2,060,379	1,908,860
				fraction 0.01				
1,000	00:14	00:01	00:15	1.5	3,612	530	3,069	2,035
2,000	00:30	00:01	00:31	1.6	4,587	818	13,949	10,563
5,000	01:19	00:04	01:23	1.8	6,785	1,500	85,402	73,612
10,000	02:42	00:11	02:53	1.9	8,602	2,138	284,076	208,714
20,000	05:03	00:38	05:41	2.0	10,392	2,988	1,073,244	862,904
40,715	10:56	02:53	13:49	4.3	14,063	4,648	4,132,467	3,766,741
				fraction 0.002				
1,000	00:18	00:01	00:19	1.7	7,228	926	6,148	3,810
2,000	00:35	00:02	00:37	1.8	9,178	1,450	27,967	20,018
5,000	01:33	00:07	01:40	2.2	13,572	2,687	171,272	142,067
10,000	03:04	00:17	03:21	2.4	17,206	3,828	569,038	400,368
20,000	06:05	00:58	07:03	2.5	20,785	5,372	2,147,568	1,655,775
40,715	12:40	04:36	17:16	5.3	28,131	8,403	8,273,572	7,351,439
				fraction 0.05				
1,000	00:24	00:02	00:26	2.0	18,071	1,844	15,378	8,284
2,000	00:45	00:04	00:49	2.2	22,942	2,943	69,936	44,333
5,000	01:49	00:14	02:03	3.0	33,918	5,551	428,147	325,122
10,000	03:27	00:33	04:00	3.3	43,002	7,931	1,423,932	913,123
20,000	06:49	01:54	08:43	3.8	51,947	11,200	5,374,216	3,786,495
40,715	14:04	09:20	23:24	7.5	70,308	17,638	20,692,365	17,327,899
				fraction 0.1				
1,000	00:34	00:03	00:37	2.8	36,138	2,920	30,692	13,965
2,000	01:00	00:06	01:06	3.0	45,875	4,711	139,579	76,110
5,000	02:22	00:22	02:44	4.4	67,825	9,046	854,372	577,709
10,000	04:31	00:55	05:26	4.9	85,988	13,012	2,841,591	1,625,067
20,000	08:52	03:11	12:03	5.8	103,886	18,480	10,724,805	6,756,069
40,715	18:03	16:29	34:32	11.2	140,606	29,264	41,289,544	31,989,192

Table 5: Detailed results of running the *sketch* and *dist* of Mash

No. samples	Processing time [h:mm:ss]		RAM
	sketch	*dist*	[GB]
Sketch size 1,000; lexicographical ordering			
1,000	0:00:11	0:00:01	0.6
2,000	0:00:32	0:00:04	0.7
5,000	0:01:17	0:00:28	0.8
10,000	0:02:39	0:01:53	0.9
20,000	0:04:40	0:07:24	1.1
40,715	0:09:41	0:30:39	1.6
Sketch size 1,000; Species tax id ordering			
1,000	0:00:15	0:00:01	0.6
2,000	0:00:22	0:00:04	0.7
5,000	0:00:59	0:00:28	0.8
10,000	0:02:01	0:01:50	0.9
20,000	0:03:55	0:07:17	1.1
40,715	0:08:08	0:30:05	1.5
Sketch size 10,000; lexicographical ordering			
1,000	0:00:17	0:00:06	0.8
2,000	0:00:26	0:00:24	1.1
5,000	0:01:06	0:02:25	1.7
10,000	0:02:22	0:09:34	2.7
20,000	0:04:21	0:37:49	4.8
40,715	0:08:58	2:37:00	9.8
Sketch size 10,000; Species tax id ordering			
1,000	0:00:12	0:00:06	0.9
2,000	0:00:24	0:00:24	1.1
5,000	0:01:03	0:02:25	1.7
10,000	0:02:07	0:09:33	2.7
20,000	0:04:10	0:37:45	4.8
40,715	0:08:35	2:36:54	9.8

Table 5 shows the running times of Mash analyses. We used two orderings of the input datasets (lexicographical and species tax id) without significant impact on the processing times. Two sketch sizes were examined: 1,000 (default Mash sketch size) and 10,000 (value used in the NCBI Pathogen Detection). For the examined species 10,000 sketch size is similar to 0.002 fraction in Kmer-db.

The meaning of the columns in the table is as follows:

- No. samples — number of samples in the input set.
- Sketch processing time — time of calculation of sketches. This stage is made only one time, when new genome is added, so we do not include it in the analysis time.
- Dist processing time — time of calculation of matrix of distances for known sketches.
- RAM [GB] — maximum amount of memory allocated in the distance calculation stage.