Supplementary Material

A survey of BWT variants for string collections

Davide Cenzato*1 and Zsuzsanna Lipták*2

1Department of Environmental Sciences, Informatics and Statistics, Ca’ Foscari University, Venice, Italy and
2Department of Computer Science, University of Verona, Verona, Italy

*Corresponding author. zsuzsanna.liptak@univr.it

A. Missing proofs

Proof of Lemma 1 For 1., note that the order of the strings $T_1$ is the lexicographical order, since $T_1 < ω T_2$ if and only if $T_1 < _{lex} T_2$ if and only if $T_1 < lex T_2$. For 2., consider that no two rotations can be one prefix of the other, resulting in the lexicographic order between rotations, with the dollar-signs breaking ties. The only difference between the two transforms is now that for $T_i$’s, and creating a run for every character of $\mathcal{L}$.

Proof of Lemma 2 Follows from the fact that no two distinct substrings ending in $\$\$ can be one prefix of the other.

Proof of Proposition 1 1. Let $L_1[i] = x$ and $L_2[i] = y$. Since all separator-based BWT variants use the lexicographical order of the rotations, this means that there exists a substring $U$ which is preceded by $x$ in one string $T_j$ and by $y$ in another $T_j$, the first occurrence has rank $i$ in one BWT and the other has rank $j$ in the other BWT variant. This implies that the two occurrences are followed by two dollars, and either the two dollars are different, or they are the same dollar, and the subsequent substrings are different. Therefore, $U$ defines an interesting interval. Parts 2. and 3. follow from 1.

Proof of Proposition 2 Let $I = [b_1, c_1]$ be an interesting interval containing $d$ distinct characters, and let $U$ be the shared suffix defining $I$. Since the strings are listed according to the colex order, all strings in which $U$ is preceded by the same character will appear in one block, and therefore, $L$ has exactly $d$ runs in the interval $I$. Let $L_{b_1 - 1} = x$ and $L_{c_1 + 1} = y$. If $x$ occurs in $I$ and it is not the first run of $I$ (i.e., $L_{b_1} \neq x$), then listing first the strings where $U$ is preceded by $x$ would reduce the number of runs by 1; similarly, listing those where $y$ precedes $U$ as last of the group would reduce the number of runs by 1. By Prop. 1, this is the only possibility for varying the number of runs.

B. Experimental setup

All datasets are stored in FASTA format.

We used three tools for computing the five BWT variants; pfpebwt, ropebut2 and Big-BWT. In order to make the BWTs comparable we did some adaptations to both tools and inputs. We modified ropebut2 to make it work with the same character order as the other tools, i.e. $\$ < A < C < G < N < T$. Then we used ropebut2 for computing both the mdoEBWT and the colexBWT using the -R and -r -s flags respectively. We used pfpebwt for constructing both the eBWT and the dolEBWT variants. In order to compute the dolEBWT, we modified the input files, appending an end-of-string character at the end of each sequence. Finally, for computing the concBWT, we removed the headers from the FASTA files, arranging the sequences in newline separated files, and ran Big-BWT without additional flags on these newline separated files.

C. Further information on the tools

We tested all 18 tools extensively, and determined which data structure they compute, using both our tests and the algorithm descriptions in the respective papers. In this section, we include further information about some of these tools.

- pfpebwt is a tool computing the eBWT of string collections (https://github.com/davidecenzato/PFP-eBWT.git). It takes in input a fasta file and gives in output the eBWT in either
Table 1. Summary of the most important parameters of the eight datasets. From left to right we report the dataset name, the number of sequences, the total length, the average, minimum and maximum sequence length, and the average runlength $n/r$ of the optimum BWT according to Bentley et al. [2].

| dataset                      | no. seq | total length    | avg  | min  | max  | $n/r$ (opt) |
|------------------------------|---------|-----------------|------|------|------|-------------|
| SARS-CoV-2 short             | 500,000 | 25,000,000      | 50   | 50   | 50   | 35.125      |
| Simons Diversity reads       | 500,000 | 50,000,000      | 100  | 100  | 100  | 8.133       |
| 16S rRNA short               | 500,000 | 75,929,833      | 152  | 69   | 301  | 44.573      |
| Influenza A reads            | 500,000 | 115,692,842     | 231  | 60   | 251  | 50.275      |
| SARS-CoV-2 long              | 50,000  | 53,726,351      | 1,075| 265  | 3,355| 74.498      |
| 16S rRNA long                | 16,741  | 25,142,323      | 1,502| 1,430| 1,549| 47.140      |
| Candida auris reads          | 50,000  | 124,150,880     | 2,483| 214  | 8,791| 1,732       |
| SARS-CoV-2 genomes           | 2,000   | 59,610,692      | 29,805| 22,871| 29,920| 523.240     |

plain ASCII text or RLE (run-length-encoded) format. We used (a) no flags for long sequences, and (b) the flags -w 10 -p 10 -n 3 --reads for short sequences. We included it in two different rows of Table 1 because by default `pfpebwt` computes the eBWT, but it can compute the dolEBWT if the sequences have explicit end-of-string characters (not in multi-thread mode).

- **cais** is a tool implementing the SAIS_for_eBWT algorithm [3], which computes both the eBWT and the dolEBWT (https://github.com/davidecenzato/cais.git) depending on the input flag. It takes in input a fasta file, a fastq file, or a plain text file and gives output in the final transform in plain ASCII text. The -c and -a flags enable to output the conjugate array along with the resulting BWT.

- **G2BWT** is a tool computing the dolEBWT of short sequence collections (https://bitbucket.org/DiegoDiazDominiguez/lms_grammar/src/bwt_imp2). It takes in input newline separated files. Even though it is not stated explicitly, this tool computes the dolEBWT because, when it constructs the grammar, it uses dollars for separating adjacent strings. Thus, also the string rotations will contain dollars. We tested it using the default settings.

- **mnbut** is a tool implementing the Holt and McMillan [4] merge-based BWT construction algorithm (https://github.com/holtjma/mnbut.git). It takes in input a list of one or several fastq files. Even if this tool uses the BCR approach [1] for computing the BWTs to merge, it actually computes the dolEBWT. This is because it features a preprocessing where it sorts the input strings lexicographically. Thus, the resulting dolBWT corresponds to the dolEBWT.

- **BEETL** is a suite containing several tools, including a tool computing the dolBWT of string collections using an implementation of the BCR and BCR-ext algorithms [1] (https://github.com/BEETL/BEETL.git). This tool requires that all input sequences have to have the same length. We tested this tool using --output-format ASCII and --concatenate-output flags. This tool also computes the BWT variant similar to the colexBWT by using the --sap-ordering flag (BCR-ext mode only).

- **BCR_LCP_GSA** is a tool computing the mdolBWT of string collections in semi-external memory (https://github.com/giovannarosone/BCR_LCP_GSA). It implements an algorithm similar to BCR contained in the BEETL tool, but it can process a string collection containing sequences of different lengths. It takes in input a fasta file, a fastq file, or a gz-compressed fastq file. It computes the mdolBWT following the method of Bauer et al., described in [1]. We set the `datatypeLengthSequences` variable in Parameters.h to 1.

- **ropebwt2** is a tool computing the FM-index and the mdolBWT of string collections (https://github.com/lh3/ropebwt2.git), using an approach similar to BCR. It takes in input a fasta file, a fastq file, or a gz compressed fastq file. We listed it in two different rows of Table 1 because it computes the mdolBWT or the colexBWT, depending on the flags. We used the -R and the -aR -a flags, respectively, to obtain the two transforms. In addition, we modified main.c in order to change the order of the characters to $< A < C < G < N < T$.

- **merge-BWT** computes the mdolBWT of a string collection by merging the BWTs of subcollections of the input (https://github.com/jltsiren/bwt-merge.git). It takes in input a list of one or several mdolBWTs. The order of the dollars will depend on the order in which the input BWTs are listed. We tested it using -i plain_sorted and -o plain_sorted flags. We computed the BWTs of the subcollections using ropebwt2.

- **nvSetBWT** is a tool included in nvbio suite (https://github.com/NVlabs/nvbio.git). It takes in input either a fastq or a newline separated file. We tested it using the -R flag for skipping the reverse strand. However, even if the algorithmic descriptions in [7, 5] seem to describe the mdolBWT, the output of the current version (version 1.1) does not correspond to a possible BWT because the Parikh vector is different from that of the input.

- **eGSA** computes the generalized enhanced suffix array and the mdolBWT of a string collection (https://github.com/felipelouza/egsa.git). It takes in input a newline text file, a fasta file, or a fastq file. It uses the gSACA-K algorithm for computing the suffix array of subcollections of the input and then merges all suffix arrays. Thus it computes the mdolBWT. We tested it with the -b flag.

- **eGAP** computes the mdolBWT, and optionally the LCP-array (longest common prefix array) and DA (document array) of a string collection (https://github.com/felipelouza/egap.git). It takes in input a newline separated file, a fasta file, or a fastq file. It uses the gSACA-K algorithm for computing the suffix array of subcollections of the input and then merges all suffix arrays. Thus it computes the mdolBWT. We tested it with the -b flag.

- **bwt-lcp-parallel** computes the mdolBWT and the LCP-array of a collection of short sequences (https://github.com/AlgoLab/bwt-lcp-parallel.git). It takes in input fasta files and does not support the N character. We tested it using standard settings.

- **gsufsort** computes the SA, LCP and mdolBWT of a string collection (https://github.com/felipelouza/gsufsort.git), using the gSACA-K algorithm of [6]. It takes in input a newline
Supplementary Material

separated file, a fasta file, or a fastq file. We tested it using --fasta and --bwt flags.

• `grlBWT` is a tool computing the mдолBWT of string collections using an induced suffix sorting based algorithm that keeps the intermediate data structures in compressed form (https://github.com/ddiazdom/grlBWT). It takes in input a concatenated string collection and gives in output the mдолBWT in run-length compressed form. We tested it with the default parameters and used new-line separated files as input.

• `BigBWT` computes the concBWT, and optionally the suffix array, of a highly repetitive text or string collection (https://github.com/alshai/Big-BWT.git) using the Prefix-free parsing (PFP) algorithm. It takes in input a newline separated file or a fasta file. This tool with the -f flag is used internally in the r-index implementation (https://github.com/alshai/r-index), producing the BWT of the strings concatenated without dollars, thus, the end-of-string symbols have to be added explicitly. On the other hand, the tool without the -f flag will compute the BWT of the fasta files without skipping the fasta headers. We used standard parameters and as input newline separated files, the output then is the concBWT.

• `r-pfbwt` is a tool which computes the run-length encoded concBWT by using a similar algorithm than `BigBWT` (https://github.com/marco-oliva/r-pfbwt). However, unlike `BigBWT`, `r-pfbwt` employs an improved version of the PFP algorithm, which allows the process of even larger datasets through a recursive pre-processing of the input. We tested it using the --bwt-only flag and computed the PFP data structures using the pfp++ software (https://github.com/marco-oliva/pfp.git).

• `CMS-BWT` is a tool computing the concBWT by using the matching statistics to speed up the BWT computation and reduce the memory footprint for large and repetitive datasets (https://github.com/fmasillo/CMS-BWT.git). Unlike the other software it requires two input files, one containing a string collection and another containing a reference sequence. It takes in input fasta files and outputs the resulting BWT in plain format or run-length encoding. We tested it using the default parameters.

• `optimalBWT` is a tool computing the optimal BWT of Bentley et al., it features two different construction algorithms, a variant of SAIS of Nong et al. which works in internal memory and a variant of BCR working in semi-external memory (https://github.com/davidecenzato/optimalBWT.git). It takes in input either a fasta or fastq file and outputs the resulting BWT in plain ascii text. We tested it using both -a sais and -a bcr flags.

D. Full results on individual datasets

References

1. M. J. Bauer, A. J. Cox, and G. Rosone. Lightweight algorithms for constructing and inverting the BWT of string collections. *Theor. Comput. Sci.*, 483:134–148, 2013.

2. J. W. Bentley, D. Gibney, and S. V. Thankachan. On the complexity of BWT-runs minimization via alphabet reordering. In *Proc. of 28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *LIPIcs*, pages 15:1–15:13, 2020.

3. C. Boucher, D. Cenzato, Zs. Lipták, M. Rossi, and M. Sciortino. Computing the original eBWT faster, simpler, and with less memory. In *Proc. of 28th International Symposium on String Processing and Information Retrieval (SPIRE 2021)*, volume 12944 of *LNCS*, pages 129–142, 2021.

4. J. Holt and L. McMillan. Merging of multi-string BWTs with applications. *Bioinform.*, 30(24):3524–3531, 2014.

5. C. Liu, R. Luo, and T. W. Lam. GPU-accelerated BWT construction for large collection of short reads. *CoRR*, abs/1401.7457, 2014.

6. F. A. Louza, S. Gog, and G. P. Telles. Inducing enhanced suffix arrays for string collections. *Theor. Comput. Sci.*, 678:22–39, 2017.

7. J. Pantaleoni. BWT of large string sets. *CoRR*, abs/1410.0562, 2014.
SARS-CoV-2 short (500,000 short sequences)

| Hamming d. | Hamming distance on the big dataset | dataset properties |
|------------|-------------------------------------|-------------------|
|            | dolEBWT  | m dolBWT | concBWT | colexBWT | no. sequences | 500,000 |
| dolEBWT    | 0       | 3,014,183 | 2,926,602 | 2,912,860 |                      |         |
| m dolBWT   | 0.11820 | 0        | 3,013,908 | 3,102,887 |                      |         |
| concBWT    | 0.11477 | 0.11819 | 0        | 3,013,634 |                      |         |
| colexBWT   | 0.11423 | 0.12168 | 0.11818 | 0        |                      |         |

| no. runs big dataset |
|----------------------|
| r       | n/r       |
| eBWT    | 1,902,148 | 13.143 |
| dolEBWT | 1,868,581 | 13.647 |
| m dolBWT | 3,113,818 | 8.189 |
| concBWT | 3,402,513 | 7.494 |
| colexBWT | 808,906  | 31.524 |
| optBWT   | 725,979   | 35.125 |

| small dataset properties |
|--------------------------|
| no. of sequences | 5,000 |
| total length     | 250,000 |
| average length   | 100 |
| no. of interesting intervals | 2,476 |
| total length intr.int.s | 180,038 |
| fraction pos.s in intr.int.s | 0.706 |
| variability      | 0.173 |

| no. runs small dataset |
|------------------------|
| r       | n/r       |
| eBWT    | 52,979    | 4.719 |
| dolEBWT | 50,803    | 5.019 |
| m dolBWT | 54,766    | 4.656 |
| concBWT | 54,698    | 4.662 |
| colexBWT | 37,320   | 6.833 |
| optBWT   | 35,904    | 7.102 |

Table 2. Results for the SARS-CoV-2 short dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength (n/r) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength (n/r) of all BWT variants on a subset of the input collection.
Table 3. Results for the Simons Diversity reads dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength \((n/r)\) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength \((n/r)\) of all BWT variants on a subset of the input collection.
### 16S rRNA short (500,000 short sequences)

| Hamming d. | Hamming distance on the big dataset | dataset properties |
|------------|------------------------------------|--------------------|
|            | dolEBWT | mdolBWT | concBWT | colexBWT | no. of sequences | total length |
| dolEBWT    | 0       | 2,202,008 | 2,540,310 | 1,748,072 | 500,000         | 75,929,833  |
| mdolBWT    | 0.02881 | 0        | 2,201,003 | 2,202,717 |                | 152        |
| concBWT    | 0.03324 | 0.02880  | 0        | 2,784,600 |                | 54,366     |
| colexBWT   | 0.02287 | 0.02882  | 0.03643  | 0        |                | 56,708,529 |

| Hamming distance on a subset of 5,000 sequences | no. runs big dataset |
|-------------------------------------------------|----------------------|
| dolEBWT | mdolBWT | concBWT | colexBWT | n/r  |
| dolEBWT | 1,992,130 | 38.115  |
| mdolBWT | 1,992,211 | 38.364  |
| concBWT | 4,057,541 | 18.836  |
| colexBWT | 2,767,797 | 27.614  |

| Hamming d. | Hamming distance on a subset of 5,000 sequences | small dataset properties |
|------------|-----------------------------------------------|--------------------------|
|            | dolEBWT | mdolBWT | concBWT | colexBWT | no. of sequences | total length |
| dolEBWT    | 0       | 20,159  | 23,229  | 15,835  | 5,000          | 765,037     |
| mdolBWT    | 0.02635 | 0       | 20,024  | 20,092  |                | 152        |
| concBWT    | 0.03036 | 0.02617 | 0       | 25,464  |                | 1,376       |
| colexBWT   | 0.02070 | 0.02626 | 0.03329 | 0       |                | 139,041     |

| edit d. | edit distance on a subset of 5,000 sequences | no. runs small dataset |
|---------|---------------------------------------------|------------------------|
|         | eBWT | dolEBWT | mdolBWT | concBWT | colexBWT | r | n/r |
| eBWT    | 0       | 51,683  | 62,799  | 63,303  | 61,732  | 35,262 | 21.554 |
| dolEBWT | 0.06756 | 0       | 16,968  | 20,180  | 14,166  | 35,293 | 21.677 |
| mdolBWT | 0.08209 | 0.02218 | 0       | 16,965  | 19,371  | 50,581 | 15.125 |
| concBWT | 0.08274 | 0.02638 | 0.02182 | 0       | 21,683  | 38,900 | 19.667 |
| colexBWT | 0.08069 | 0.01852 | 0.02532 | 0.02834 | 0       | 30,568 | 25.027 |

Table 4. Results for the 16S rRNA short dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength (n/r) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength (n/r) of all BWT variants on a subset of the input collection.
Table 5. Results for the Influenza A reads dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength \((n/r)\) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength \((n/r)\) of all BWT variants on a subset of the input collection.
SARS-CoV-2 long (50,000 long sequences)

| norm. Hamming d. | Hamming distance on the big dataset | dataset properties |
|------------------|------------------------------------|--------------------|
| dolEBWT          | 0                                  | no. sequences      |
| mdolBWT          | 0.00462                            | 50,000             |
| concBWT          | 0.00462                            | total length       |
| colexBWT         | 0.00475                            | 53,776,351         |
| dolEBWT          | 248,189                            | average length     |
| mdolBWT          | 0                                  | 1,075              |
| concBWT          | 248,572                            | no. of interesting |
| colexBWT         | 248,205                            | intervals          |
| dolEBWT          | 255,357                            | total length intr.|
| mdolBWT          | 248,631                            | intr.s             |
| concBWT          | 248,765                            | fraction pos.s     |
| colexBWT         | 0                                  | in intr.int.s      |
|                  |                                    | variability        |
|                  |                                    | 0.17548            |
|                  |                                    | 0.03716            |

| no. runs big dataset |
|----------------------|
| eBWT                 |
| dolEBWT              |
| mdolBWT              |
| concBWT              |
| colexBWT             |

| small dataset properties |
|--------------------------|
| no. sequences            |
| total length             |
| average length           |
| no. of interesting       |
| total length intr. intr.|
| fraction pos.s intr. int.|
| variability              |
| 1,500                    |
| 1,612,956                |
| 1,075                    |
| 1,046                    |
| 152,035                  |
| 0.094                    |
| 0.047                    |

| Table 6. Results for the SARS-CoV-2 long dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength \((n/r)\) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength \((n/r)\) of all BWT variants on a subset of the input collection. |
Table 7. Results for the 16S rRNA long dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength (n/r) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength (n/r) of all BWT variants on a subset of the input collection.
Table 8. Results for the Candida auris reads dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength \((n/r)\) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength \((n/r)\) of all BWT variants on a subset of the input collection.
SARS-CoV-2 genomes (2,000 long sequences)

| norm. Hamming d. | Hamming distance on the big dataset |  |
|------------------|-----------------------------------|---|
| dolEBWT          | 0                                 | 7,958 | 7,900 | 7,263 |
| mdolBWT          | 0.00013                           | 0     | 7,958 | 7,957 |
| concBWT          | 0.00013                           | 0.00013 | 0     | 7,990 |
| colexBWT         | 0.00012                           | 0.00013 | 0.00013 | 0     |

| dataset properties |
|--------------------|
| no. sequences      | 2,000               |
| total length       | 59,612,692          |
| average length     | 29,085              |
| no. interesting intervals | 1,863       |
| total length intr. intr.s | 80,486    |
| fraction pos.s in intr. intr.s | 0.001   |
| variability        | 0.148               |

| no. runs big dataset |
|----------------------|
| r        | n/r       |
| eBWT     | 117,628 | 506.773 |
| dolEBWT  | 117,410 | 507.731 |
| mdolBWT  | 118,876 | 501.495 |
| concBWT  | 119,334 | 499.549 |
| colexBWT | 114,287 | 521.605 |
| optBWT   | 113,930 | 523.240 |

| small dataset properties |
|--------------------------|
| no. sequences | 50 |
| total length  | 1,490,184 |
| average length | 29,802 |
| no. interesting intervals | 43 |
| total length intr. intr.s | 271 |
| fraction pos.s in intr. intr.s | 1.8 · 10⁻⁴ |
| variability | 0.690 |

| no. runs small dataset |
|------------------------|
| r        | n/r       |
| eBWT     | 25,258 | 58.997 |
| dolEBWT  | 25,255 | 59.006 |
| mdolBWT  | 25,274 | 58.961 |
| concBWT  | 25,285 | 58.936 |
| colexBWT | 25,221 | 59.085 |
| optBWT   | 25,210 | 59.113 |

Table 9. Results for the SARS-CoV-2 genomes dataset. First row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants. First row right: summary of the dataset properties. Second row: number of runs and average runlength \((n/r)\) of all BWT variants. Third row left: absolute and normalized pairwise Hamming distance between separator-based BWT variants on a subset of the input collection. Third row right: summary of the dataset properties of a subset of the input collection. Fourth row left: absolute and normalized pairwise edit distance between all BWT variants on a subset of the input collection. Fourth row right: number of runs and average runlength \((n/r)\) of all BWT variants on a subset of the input collection.