Supplementary Materials for
"The String Decomposition Problem and its Applications to Centromere Analysis and Assembly"

Appendices

Appendix: Uniqueness of the solution of the String Decomposition Problem

Backtracking from one of the sinks to the source in the String Decomposition Graph reveals the sequence of block-switching edges in the backtracking path and thus provides a solution of the String Decomposition Problem (each block-switching edge is labeled by the corresponding block). To check if this solution is unique we modify the standard backtracking procedure in sequence alignment (that arbitrarily selects a single backtracking path (Compeau and Pevzner, 2013)) by constructing the *backtracking graph* that contains all edges traversed by all backtracking paths from one of the sinks to the source. Two vertices in the backtracking graph are called *close* if there is a path between them that does not contain block-switching edges. We form the *block-switching graph* by gluing all close vertices in the backtracking graph and collapsing parallel identically labeled block-switching edges into a single edge. The String Decomposition Problem has a unique solution if the block-switching graph represents a single path.

Appendix: SD implementation details

run_decomposer.py script. StringDecomposer is publicly available at https://github.com/ablab/stringdecomposer. The "run_decomposer.py" script accepts the following two files in fasta format as the Input: (i) a file containing the read-set or a genomic

sequence, and (ii) a file containing monomer sequences. It runs the SD algorithm (implemented in C++) and converts the alignment scores of individual monomers into percent identities using the Edlib library (Šošic and Šikic, 2017). The default indel and mismatch penalties for indel are equal to 1 but the “run,decomposer.py” script allows one to assign arbitrary user-defined penalties. StringDecomposer saves the monomer alignments to a given read-set (or a genomic sequence) in the tsv-format. To generate the string decomposition described in the Results section commit 6fc0b4a64 was used.

SD parallelization. The String Decomposition Graph becomes rather large in the case when the string *R* is long (e.g., when *R* is an ultralong read or an assembly of an entire centromere) or when the block-set is large, leading to a large memory footprint. To reduce the memory footprint, we represent the string *R* as a set of overlapping segments of length *SegmentLength* with default value *SegmentLength* = 5500 bp (the last segment can be shorter) so that the consecutive segments overlap by *Overlap* nucleotide (Figure S1). The default value *Overlap* (500 bp) is chosen to be larger than the monomer size and to ensure that each monomer is positioned fully inside at least one segment. Afterward, each segment of the read is processed separately and all segments are “glued” together.

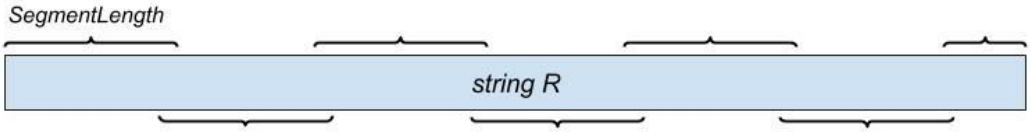

Fig. S1. Partitioning a read into overlapping segments.

Appendix: Identification of reliable monomers

In order to distinguish false monomer alignments (denoted by the gap-symbol “?”) from the true monomer alignments, we trained a logistic regression model with two features defined for each monomer-to-sequence alignment: 1) the identity score *TopIdentity* of the highest-scoring monomer, and 2) the difference *IdentityDiff* between *TopIdentity* and the identity of the second-highest scoring monomer. The simplest *baseline* logistic regression model makes a decision solely on the *TopIdentity* value, i.e, classifies a monomer as reliable if *TopIdentity* exceeds a threshold *MinTopIdentity*. Below we demonstrate the logistic regression model with two predictors (*TopIdentity* and *IdentityDiff*) improves on the baseline one-predictor logistic regression model.

In order to train both models, we selected 1,000 ONT reads that were mapped to cenX by tandemQUAST (further referred as the CENX read-set) and 1,000 ONT reads from the same rel3 CHM13 dataset mapped to non-centromeric regions of chromosome X T2T assembly v0.7 with minimap2 (further referred as the NOTCENX read-set). Centromeric reads containing the LINE element were excluded from the CENX set. Only reads with alignment identity exceeding 80% and the fraction of aligned sequence exceeding 90% were included in the NOTCENX set. We attempted to decompose each read from CENX and NOTCENX into the monomer alphabet. The training dataset *TrainSet* is constructed from 80% of CENX and NOTCENX reads. The test dataset *TestSet* is constructed from the remaining 20% of CENX and NOTCENX reads. Both baseline and two-predictor logistic regression models were trained on *TrainSet*.

The optimal *MinTopIdentity* threshold identified with the baseline model on the *TrainSet* is 67.7%. At this threshold, the baseline model produces 0 false-positive and 241 false-negative alignments out of 65,842 monomer alignments in *TestSet* (false-positive rate is 0%, false-negative rate is 0.37%). The two-predictor model showed superior results: 0 false positive and 198 false negative alignments out of the same 65,842 monomer alignments in *TestSet* (false-positive rate is 0%, false-negative rate is 0.30%). Figure S2 illustrates the results of the classification produced by the baseline and the two-predictor regression models.

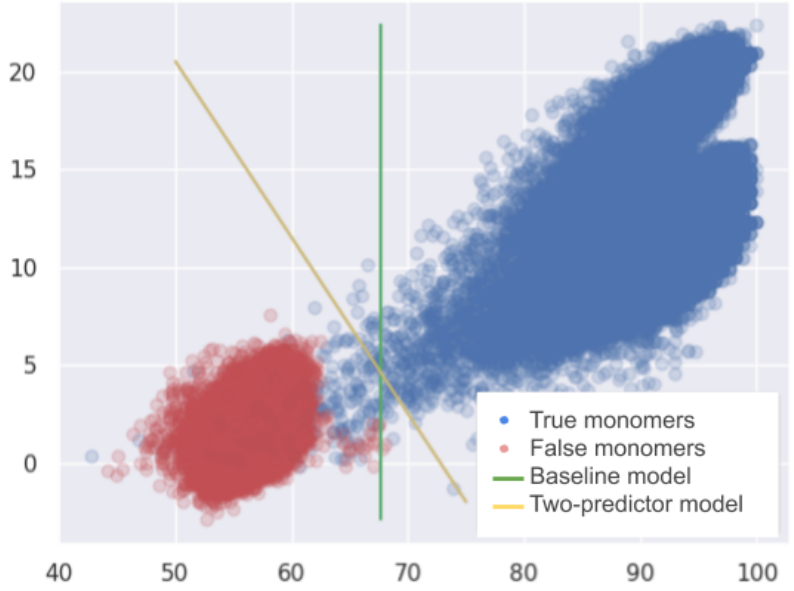

Fig. S2. The distribution of *TopIdentity* (x-axis) and *IdentityDiff* (y-axis) for monomer alignments in centromeric and non-centromeric regions. Each dot in the graph represents either a true monomer alignment that belongs to the centromeric region (blue dots) or false monomer alignment (red dots). The green line separates true and false monomers according to baseline classifier based on *TopIdentity* value of each alignment, while the yellow line separates the true and false monomers alignments according to the logistic regression trained on both *TopIdentity* and *IdentityDiff* values.

Appendix: Processing gaps in the monomer alignments

Some reads are translated into monoreads with gaps represented by the "?" symbols that reveal regions of low identity to all monomers. The SD algorithm generates a read decomposition where each position is covered by a monomer and replaces all unreliable monomers by the "?" symbol in the monoread. However, the number of the predicted "?" symbols in a monoread is not necessarily an accurate approximation of the total length of non-monomeric positions in a read. Additionally, the AC algorithm produces a decomposition that does not cover all positions of a read, resulting in *non-covered* positions in the AC monomer decomposition, with no monomer alignment covering these positions.

We thus modified a transformation of a read *R* into a monoread *mono(R)* by replacing a run of non-covered positions of length *L* by a run of the gap symbol "?" with length *L/MonomerLength*, where *MonomerLength* is the average length of monomers.

Appendix: Benchmarking string decomposition tools

Alpha-CENTAURI. Alpha-CENTAURI v.0.2 was run with default parameters. While HMMer search from the first stage of the Alpha-CENTAURI algorithm (partitioning a read into consequent monomers locations) was successful, the second stage (monomer sequences clustering and monomer identification) did not generate a precise read decomposition into monomers, reporting many abnormal HORs alignments, and was removed from further analysis.

TandemRepeatsFinder. TRF 4.09 version was run with recommended parameters for human genome (https://tandem.bu.edu/trf/trf.whatnew.html). Though TRF has identified the correct monomer length (~170 bp) in all centromeric reads, its output is difficult to use for further analysis. In particular, it is not clear how to identify monomers from the putative positions identified by TRFs as these positions are often shifted. For example a read *bcc5e5d2-f12f-4b59-b952-bd10f81ac89f* in rel2 T2T dataset is fully covered by DXZ1* monomers both according to SD and TRF, but TRF alignments positions have a 40 bp shift with respect to SD positions, making it difficult to identify monomers with high identity scores. In contrast, TRF identified a rather small shift (~10-15 bp) in a read *c500a3b1-f00c-40c1-94af-e33bae40ca71,* resulting in a successful prediction of monomers from the TRF alignments.

NCRF. We launched NCRF v1.01.02 to search for repeats of DXZ1* with parameters "--scoring=nanopore --minlength=5000". Appendix "Benchmarking NCRF" analyzes NCRF results using monomer-free metrics and compares it with other string decomposition approaches.

Appendix: Benchmarking NCRF

We compared NCRF with the AC and SD approaches using the dataset *Reads* defined in the Results section and analyzing two monomer-free metrics:

- *read coverage,* the fraction of reads' length partitioned into monomers for (AC and SD approaches) or covered by the DXZ1* HOR (NCRF approach).
- *percentage of unaligned segments.* Two consecutive aligned monomers in a read are separated by an unaligned segment if the distance from the end of the first monomer alignment to the start of the second monomer alignment exceeds *MinUnalignedLength* (default value *MinUnalignedLength*=10 bp). Since NCRF reports HORs without spaces in the alignment, it has 0 unaligned segments.

Table S1 illustrates that NCRF has lower *read coverage* than the SD and AC approaches but improves on these approaches with respect to the number of unaligned segments.

Approach	**read coverage (%)**	**% unaligned segments (#unaligned segments / #monomers)**

AC	98	2.95
NCRF	92	0
SD	99	0.14

Table S1. Monomer-free metrics for the AC, NCRF and SD tools.

Appendix: cenX monomers

In order to extract twelve monomer sequences we launched “chop,to,monomers.py” script from Alpha-CENTAURI v.0.2G on the consensus monomer HMM (https://github.com/volkansevim/alpha-CENTAURI/blob/master/example/alpha.hmm) and a concatenation of two DXZ1* sequences derived in Bzikadze and Pevzner, 2019. The twelve cenX monomers are derived from the Alpha-CENTAURI output. The twelve monomers forming cenX HOR monomers are usually reported as CDEABCDEABCD since this sequence of monomers reflects the ancestral pentamer structure (CDEAB) of the HOR from which cenX HOR (DXZ1) originated. Since this representation is inconvenient for analyzing string decomposition of cenX, we instead represent DXZ1 as ABCDEFGHIKL.

Appendix: Generating accurate read alignments to cenX

In order to generate a set of accurate alignments, positions of alignments generated by the TandemMapper tool (Mikheenko et al., 2020) were compared to the positions of read alignments in the cenX assembly generated by centroFlye. It turned out that some TandemMapper alignments differ from centroFlye alignments. This is likely caused either by an incorrect mapping of some reads to centromere (generated by TandemMapper) or by an erroneous recruitment of non-cenX reads to cenX (provided by centroFlye). We thus filtered out reads with differing starting positions (by more than 2 kbp) of TandemMapper and centroFlye alignments, resulting in 1442 read alignments.

Appendix: Errors in monoread-to-monocentromere alignments

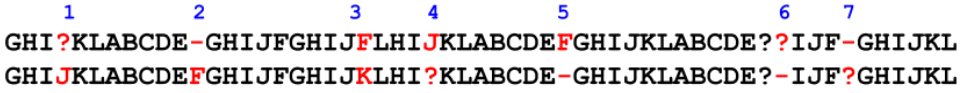

Fig. S3. Monoread-to-monocentromere alignments. The first row represents *mono(origin(Read))*, the second row represents *mono(Read)*. The matching positions are shown in black and positions with errors are shown in red. The following types of errors are shown: gap-monomer mismatch (1), monomer-insertion (2), monomer-monomer mismatch (3), monomer-gap mismatch (4), monomer-deletion (5), gap-deletion (6), gap-insertion (7).

Appendix: Detailed analysis of errors in string decomposition

Most alignment errors between monoreads and monocentromere for both SD and AC approaches occur due to inconsistencies between (inaccurate) reads and (accurate) centromere assembly.

Since 91% of mismatches for the AC approach are *monomer-gap mismatches* (Table 1), we analyzed monomers predicted by SD but missed by AC. All SD monomer alignments that have overlap longer than 100 bp with some gap symbol ("?") output by AC were considered. All monomer predicted by SD were divided into three groups: (i) the highest-scoring monomer is a true monomer, (ii) the second highest-scoring monomer is a true monomer, and (iii) none of the two highest-scoring monomers is a true monomer. Figure S4 presents the scatter-plot of the scores of the highest-scoring and the second highest-scoring monomers for each group (left) and the distribution of their differences (right). All alignments have relatively low identity (below 85%) as compared to the average identity of all monomers (93%). However, the highest-scoring monomer is correct in ~99% of cases and the difference in identity between the highest-scoring and the second-highest scoring monomers is rather substantial (more than 4% in most cases). Both the highest-scoring and the second-highest scoring monomers are incorrect in approximately 0.3% of cases.

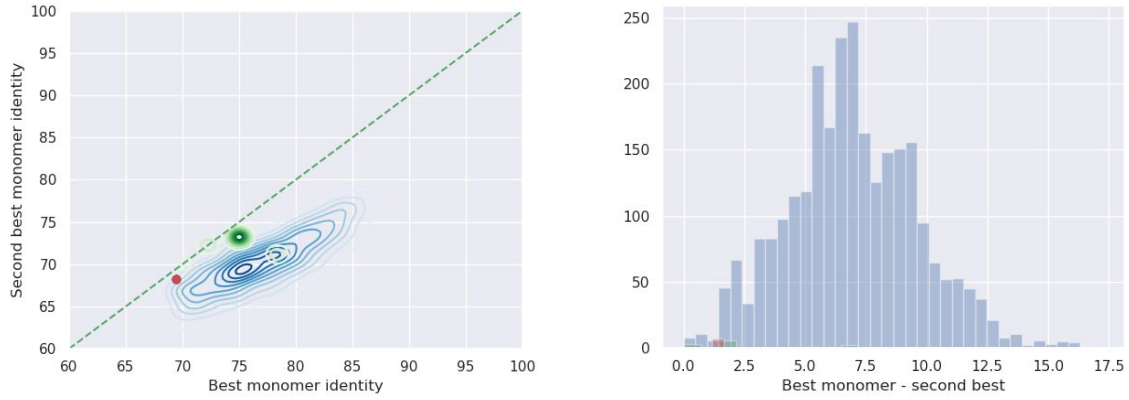

Fig. S4. Statistics of scores for monomers that the AC approach failed to predict. (Left) Analysis of underpredicted monomers that were classified as monomer-gap mismatches: the scatter-plot of the highest monomer score and the second-highest monomer score for three cases: the highest-scoring monomer is correct (blue), the second highest-scoring monomer is correct (green), neither the highest-scoring nor the second highest-scoring monomer is correct (red). The intensity of the color reflects the number of points with such identity values. (Right) Distribution of differences between the identity of the highest-scoring and the second highest-scoring monomers.

SD and AC made 18 (20) gap-insertions, 0(6) monomer-insertions, and 117 (154) monomer-deletions. Most such errors arise in corrupted regions of reads with low alignment quality — the identities of flanking monomers located next to such regions usually falls below 80%. AC has more insertions (deletions) than SD, as the run of "?" identified by AC are sometimes longer (shorter) than the correct number of monomers in the run.

Appendix: Most frequent human monomers

It turned out that 21 out of 965 monomers in the set *AllMonomers* do not appear in any monoreads. Figure S5 presents the histogram of frequencies of the remaining 965-21=944 monomers. Table S2 presents frequencies of 100 most frequent monomers in the set *MonoReads*.

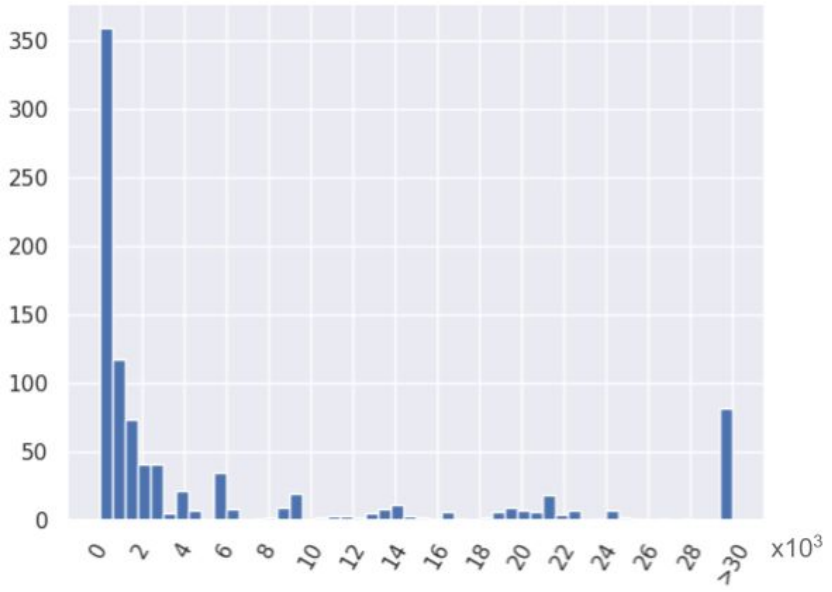

Fig. S5. Histogram of frequencies 944 monomers with non-zero frequencies in the set *MonoReads*. The x-axis shows the monomer frequency (in thousands) and the height of each bar represents the number of monomers with this frequency (The bin size is 50).

Monomer	Frequency	Monomer	Frequency	Monomer	Frequency	Monomer	Frequency	Monomer	Frequency
1	314720	21	49294	41	36522	61	35598	81	29801
2	229898	22	46100	42	36519	62	35555	82	29718
3	93658	23	45628	43	36509	63	35344	83	27612
4	93124	24	43689	44	36480	64	34449	84	26599
5	92475	25	42939	45	36474	65	34059	85	25583
6	92044	26	42093	46	36435	66	34054	86	25148
7	90964	27	38588	47	36414	67	34005	87	24961
8	90839	28	38582	48	36098	68	33513	88	24563
9	81976	29	38575	49	35970	69	33485	89	24521
10	81947	30	38545	50	35967	70	33436	90	24486
11	81169	31	38463	51	35932	71	33393	91	24485
12	81155	32	38461	52	35903	72	33264	92	24286
13	81126	33	37400	53	35862	73	33148	93	24177
14	80986	34	37321	54	35763	74	32778	94	24167
15	80885	35	36794	55	35722	75	31031	95	23505
16	80768	36	36754	56	35717	76	30763	96	22792
17	64762	37	36746	57	35708	77	30452	97	22786
18	59226	38	36731	58	35674	78	29971	98	22727
19	58511	39	36567	59	35672	79	29954	99	22724
20	51012	40	36529	60	35612	80	29807	100	22642

Table S2. Frequencies of 100 most frequent monomers in the set *MonoReads*. The total number of identified monomers is 7,577,262. 40 monomers have frequency below 10, and 9 of them have frequency 1. 33 monomers out of 40 have very low average alignment identity (below 75%).

Appendix: Most frequent putative human HORs

Table S3 presents information about 100 most frequent HORs in the set *MonoReads*.

ID	Size	HOR	*TandemCount*	*Count*
1	2	_1_2_	179339	214076
2	6	_3_4_6_8_7_5_	67874	80205
3	8	_10_9_12_11_16_15_14_13_	59607	71244
4	4	_1_24_25_17_	23364	39748
5	12	_37_41_42_44_39_47_45_46_38_35_40_43_	22503	29704
6	5	_89_88_91_93_90_	20409	22634
7	7	_18_30_27_28_29_32_31_	14466	29166
8	16	_71_63_55_59_56_52_53_60_26_50_61_57_62_49_36_48_	14265	22005
9	12	_21_81_80_82_21_66_64_67_65_51_54_58_	10094	14085
10	4	_185_193_191_19_	8548	11074
11	10	_72_73_20_1_20_78_79_74_77_76_	7259	11546
12	11	_129_128_126_18_30_27_28_29_32_31_18_	6764	12111
13	16	_157_141_142_19_145_139_140_136_143_19_146_149_148_147_96_137_	6325	10308
14	17	_175_174_177_179_178_183_184_182_181_171_170_169_172_173_180_186_176_	5531	8855
15	8	_187_190_188_200_194_192_195_197_	5447	8225
16	8	_152_154_153_151_150_69_70_68_	5092	10522
17	18	_114_116_127_166_164_165_123_111_121_119_125_124_117_115_120_108_109_110_	5003	8525
18	13	_210_207_216_214_209_212_203_202_213_196_222_206_204_	4996	6952
19	10	_73_2_1_20_78_79_74_77_76_72_	4939	7454
20	7	_68_135_132_131_167_69_70_	4937	8377
21	10	_228_229_231_230_225_223_226_227_224_234_	4569	5818
22	19	_107_22_92_95_85_22_130_113_101_112_122_134_102_98_97_106_105_87_86_	4154	7493
23	8	_260_269_268_267_276_238_244_242_	3993	4745
24	12	_51_54_58_21_81_80_82_163_66_64_67_65_	3749	6270
25	12	_1_2_1_2_103_17_34_33_1_2_1_2_	3508	8300
26	15	_123_111_121_119_125_124_117_115_120_108_109_110_114_116_127_	3278	7498
27	11	_252_249_232_248_261_263_266_264_262_239_251_	3047	4194
28	14	_257_258_241_254_265_270_274_237_275_236_272_259_26_253_	3015	4213
29	8	_163_66_64_67_65_51_54_58_	2959	4300
30	11	_286_281_283_287_273_285_243_246_245_250_247_	2578	3282
31	4	_23_75_144_155_	2285	9293

32	11	_152_154_153_151_150_69_70_68_135_132_131_	2239	3626
33	6	_199_94_99_138_133_118_	2199	6213
34	10	_1_2_1_2_103_17_34_33_1_2_	2097	6054
35	15	_23_168_84_83_23_75_84_83_162_158_156_23_75_144_155_	2013	3651
36	14	_94_99_138_133_118_199_94_99_138_133_118_198_201_189_	1966	3696
37	18	_18_30_27_28_29_32_31_18_129_128_126_18_30_27_28_29_32_31_	1903	4804
38	8	_94_99_138_133_118_198_201_189_	1748	4567
39	2	_104_104_	1662	5784
40	4	_5_3_4_6_	1617	7557
41	4	_8_7_5_3_	1369	10281
42	15	_152_154_153_151_150_69_70_68_135_132_131_167_69_70_68_	1357	2589
43	14	_1_2_1_2_103_17_34_33_1_2_1_2_1_2_	1280	3656
44	11	_84_83_162_158_156_23_75_144_155_23_168_	1210	2163
45	10	_315_317_316_312_310_308_418_311_313_314_	1173	1456
46	15	_215_217_211_220_218_221_219_205_159_161_160_278_159_161_160_	1170	2062
47	11	_215_217_211_220_218_221_219_205_159_161_160_	1168	3171
48	16	_1_24_25_17_1_24_25_17_34_33_1_100_1_24_25_17_	1160	2645
49	8	_1_2_103_17_34_33_1_2_	1141	4138
50	12	_1_24_25_17_34_33_1_100_1_24_25_17_	1130	3227
51	15	_159_161_160_19_159_161_160_215_217_211_220_218_221_219_205_	1107	1722
52	10	_8_7_5_3_4_6_8_7_5_3_	1030	6550
53	16	_343_369_370_347_346_328_348_342_341_352_349_350_351_345_340_344_	1007	1579
54	15	_240_366_362_356_309_358_361_364_355_354_357_360_353_365_240_	972	1527
55	8	_24_25_17_34_33_1_100_1_	937	2774
56	2	_96_137_	928	3669
57	10	_308_459_311_313_314_315_317_316_312_310_	885	1106
58	16	_141_142_19_145_139_140_136_143_19_146_149_148_147_96_137_96_	860	1416
59	4	_21_81_80_82_	848	4110
60	2	_10_9_	825	4800
61	16	_1_2_1_2_1_2_1_2_103_17_34_33_1_2_1_2_	805	2058
62	10	_5_3_4_6_8_7_5_3_4_6_	797	4054
63	10	_225_223_226_227_224_454_228_229_231_230_	783	1106
64	4	_51_54_58_21_	728	3055
65	16	_8_7_5_3_4_6_8_7_5_3_4_6_8_7_5_3_	720	3755
66	16	_408_414_405_403_410_413_407_412_417_406_377_404_409_455_401_402_	690	1050
67	4	_18_129_128_126_	682	2509
68	15	_63_55_59_56_52_53_60_26_50_61_57_62_49_36_48_	674	1726
69	6	_103_17_34_33_1_2_	642	2717
70	10	_10_9_12_11_16_15_14_13_10_9_	618	3051

71	2	_64_67_	585	2167
72	11	_273_391_243_246_245_250_247_463_386_411_458_	578	891
73	11	_23_75_84_83_162_158_156_23_75_144_155_	577	1793
74	2	_191_19_	568	4877
75	8	_20_78_79_74_77_76_72_73_	563	2623
76	8	_23_75_144_155_23_168_84_83_	558	1614
77	14	_99_475_99_489_497_118_502_499_501_453_491_488_189_94_	494	716
78	3	_104_104_104_	489	1861
79	14	_543_551_36_546_541_237_531_236_534_527_26_500_532_518_	481	669
80	9	_433_559_572_469_542_579_452_480_449_	474	575
81	2	_100_1_	461	1887
82	15	_55_59_56_52_53_60_26_50_61_57_62_49_36_48_71_	454	1306
83	16	_508_520_517_512_507_495_510_506_516_514_509_513_240_523_309_515_	450	673
84	15	_106_105_87_86_107_22_482_474_101_481_457_134_102_98_97_	432	624
85	8	_485_564_568_560_367_503_363_367_	427	572
86	4	_92_95_85_22_	426	1226
87	8	_376_383_382_394_395_439_189_392_	411	1080
88	16	_21_81_80_82_21_66_64_67_65_51_54_58_21_81_80_82_	401	1300
89	18	_10_9_12_11_16_15_14_13_10_9_12_11_16_15_14_13_10_9_	369	1471
90	16	_51_54_58_21_81_80_82_163_66_64_67_65_51_54_58_21_	364	1146
91	6	_14_13_10_9_12_11_	363	2417
92	18	_96_137_157_141_142_19_145_139_140_136_143_19_146_149_148_147_96_137_	361	978
93	4	_205_159_161_160_	346	1279
94	16	_5_3_4_6_8_7_5_3_4_6_8_7_5_3_4_6_	332	1822
95	4	_161_160_278_159_	316	1081
96	15	_113_101_112_122_134_102_98_97_106_105_87_86_107_22_92_	310	687
97	6	_67_65_51_54_58_21_	304	1410
98	4	_6_8_7_5_	294	2460
99	2	_3_4_	293	3148
100	10	_65_51_54_58_21_81_80_82_163_66_	276	540

Table S3. Top 100 *k*-mers (putative HORs) with the highest tandem counts. Each putative HORs is represented by a sequence of monomers from the set *AllMonomers*.

Bibliography

Šošić, M., Šikić, M. (2017). Edlib: a C/C ++ library for fast, exact sequence alignment using edit distance. Bioinformatics *33*, 1394–1395.

Compeau, P., Pevzner, P.A Bioinformatics Algorithms: An Active Learning Approach, 3rd edition. La Jolla, California: Active Learning Publishers, 2018.