Supplementary Material of CRISPRitz: rapid, high-throughput, and variant-aware *in silico* off-target site identification for CRISPR genome editing

1 Enumeration and completeness of sgRNA on- and off- target sites by general short aligners

Although well-known, general purpose short read sequence aligners, such as BWA (Li and Durbin, 2009) are commonly used by guide design tools to enumerate potential off-targets, this is not an optimal choice. In fact, using short aligner is difficult and also fine-grained searches are not possible, for example is not possible to use ambiguous nucleotide (IUPAC notation) or "N" nucleotide. Those details add difficulties in the use of short read aligners as CRISPR off-targets search tools.

To test this we use a well-known guide (GAGTCCGAGCAGAAGAAGAA) targeting the EMX1 gene. We count all the sites recovered by Cas-Offinder (a brute force approach) and BWA, when using a up to 4 mismatches and no DNA/RNA bulges and the well-known Cas9 PAM (NGG) (see Fig.S1A). We show how searching with BWA using a guide with PAM (e.g. GAGTCCGAGCAGAAGAAGAANGG) may lead to unexpected results.

Before starting the search with BWA, we created an index of human reference genome 19 (hg19), using the command:

```
bwa index hg19.fa
```

The alignment phase was performed with this command:

```
bwa aln -o 0 -m 20000000 -n 4 -k 4 -N -l 20 hg19.fa emx1.fa
```

The mapping with this command:

```
bwa samse -n 600000000000000000 hg19.fa 4mm.bwa.sai emx1.fa
```

Finally, we created a BED file with the regions recovered using samtool and bedtools, with those commands:

```
samtools view -Sb 4mm.bwa.sam
bedtools bamtobed -i 4mm.bwa.bam
```

For Cas-Offinder we performed the same analysis with the command (Cas-OFFinder uses settings written into the guide file to perform the search):

```
cas-offinder emx1.guide G emx1.out
```

As shown in Fig. S1A, BWA finds 20 off-targets in common with Cas-OFFinder and 7 unique off-targets. This behaviour can be explained by analyzing the file we input in BWA. It contains the sequence GAGTCCGAGCAGAAGAAGAANGG created adding the PAM "NGG" to emx1 guide GAGTCCGAGCAGAAGAAGAA. BWA treats each "N" nucleotide as a mismatch, so each off-target returned by BWA has one mismatch more than the Cas-OFFinder counterpart. Also, BWA tries to match the PAM sequence of the guide to the genome. Due to this behaviour, BWA reports some off-targets containing mismatches in the PAM sequence (in our example the 7 unique off-targets) that are not reported by ad-hoc CRISPR off-targets search tools (such as Cas-OFFinder in the example). To overcome these difficulties in the set-up to perform an off-targets search with BWA, some tools (ie CRISPOR (Haeussler *et al.*, 2016)) first perform a search of all the occurrences of the guides (without the PAM) and then, in each found site, checks the PAM compatibility. Taking as example our input guide, CRISPOR first performs a search with BWA using only the guide (GAGTCCGAGCAGAAGAAGAA) and than in the collection of the targets, runs a filtering by extracting the sequences from the genome used in the search and collecting only those containing the correct PAM (NGG, in the example).

Importantly, we have confirmed that CRISPRitz recover all the sites recovered by Cas-OFFINDER when executed with the command:

```
CRISPRitz search hg19.index pamNNN emx1.guide -mm 4 -r
```

2 CRISPRitz recovers complex DNA/RNA bulges missed by CAS-Offinder

A well-known problem of CRISPR off-targets search tools is to efficiently enumerate all the possible off-targets that may occur during CRISPR wet-lab experiments. One difficulty added to this problem is to efficiently enumerate off-target targets containing bulges. In this section we show that CRISPRitz enumerates more off-targets with bulges with respect to the number of off-targets found by Cas-OFFinder. The two searches were run with the following thresholds: up to 4 mismatches, up to 2 DNA and 2 RNA bulges.

For Cas-OFFinder we run the command (CAS-OFFinder uses settings written into the guide file to perform the search):

```
cas-offinder emx1.guide G emx1.out
```

For CRISPRitz the used command was:

```
CRISPRitz search hg19.index pamNNN emx1.guide -index -mm 4 -bDNA
2 -bRNA 2 -r
```

As shwon in Fig. S1B, Cas-OFFinder returns a smaller number of total off-targets. In fact, Cas-OFFinder performs a filter to avoid collecting all the off-targets containing "jumps" between bulges, for example, this off-target $GAGTCaGAGCA - gA - cAcAAGTG$ is collected by CRISPRitz and not by Cas-OFFinder. Cas-OFFinder does not take into account sequences with matches/mismatches between two bulges. This behaviour can lead to the loss of some possible off-targets, compromising the analysis.

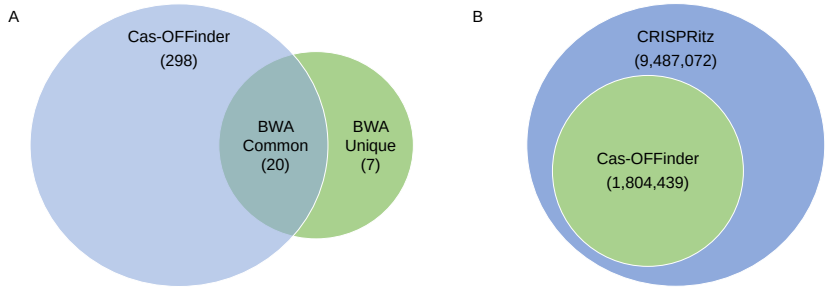

Figure S1: Enumeration and comparison of targets and off-targets by Cas-OFFinder, BWA and CRISPRitz `CRISPRitz`. (A) Comparison between Cas-OFFinder and BWA, showing the differences in terms of off-targets found with a short aligner (BWA) and an ad-hoc software (CAS-OFFinder), enlightening the necessity of dedicated software to perform in-silico CRISPR/Cas analysis. (B) Comparison between Cas-OFFinder and CRISPRitz, showing the differences in off-targets count when considering 2 bulges.

3 PAM search with Aho-Corasick algorithm

To efficiently find all the genome regions compatible with a given PAM, the algorithm creates a small deterministic automata machine used to scan and enumerate all the potential PAM matches in linear time (based on the length of the PAM). The automata machine reduces the searching time complexity to the minimum number of comparisons $O(N)$, where N is the genome length (i.e., the algorithm scans each nucleotide in the genome only once). Automata machines are usually represented as *extended* graphs. In this work, the automata machine is a rooted directed tree (also called *trie*) enriched with additional connections among nodes (see Paper Fig. 2). Each path from the root to a leaf represents a PAM. Paper Fig. 2 shows the patterns corresponding to the NGG PAM i.e.: GGG, AGG, TGG, CGG, CCC, CCT, CCA, CCG. With this data structure, the PAM search in the genome is performed by reading one nucleotide at a time and by matching it with a node of the graph. If the nucleotide matches with one

of those nodes, then the algorithm follows the corresponding path by moving one node forward in the tree and one nucleotide in the genome. This node traversal iteratively continues as long as the current genome nucleotide matches the current node. If it is not possible to match the next available node, the algorithm jumps back to the parent node of the graph by following special failure edges (dashed lines in Paper Fig. 2) and the search continues from the parent node. The paths following the failure edges reduce the number of comparisons by allowing the search to restart from the longest common substring that could be matched at that stage. Every time the graph visit, reaches a leaf node, the algorithm saves the current index (i.e., the nucleotide position in the reference genome) as it may represent the starting position of a candidate off-target site. The full list of indices of candidate off-target sites is the output of the PAM search. Of note, the indices are saved in two arrays that differentiate between positive and negative strands.

4 Ternary Search Tree for efficient genome indexing

CRISPRitz uses a *ternary search tree* (TST) data structure (Bentley and Sedgewick, 1998) to index genome for rapid bulge searches. In a TST, each node represents a nucleotide and can have a maximum of three children: left, center, and right. The TST is built by inserting one candidate off-target sequence at a time, where the insertion is implemented through a recursive search. Starting from the first nucleotide of the candidate target sequence and from the TST root, the algorithm compares if the two nucleotide match, if they match, the comparison continues to the next nucleotide on the candidate target sequence and descend in the TST through the *center* child node and no new node is added to the TST. If the nucleotide pair between the *center* child node and the nucleotide from the candidate off-target sequence do not match, then the algorithm verifies the lexicographical order of the nucleotide on the candidate target. If it is smaller than the current nucleotide in the TST node, the search recursively continues to the left child, otherwise if the candidate target nucleotide is lexicographical greater, the search recursively continues to the right child. Finally, if the node in the chosen direction does not exist, a new node is inserted in the TST with the current nucleotide of the candidate target sequence. Every genomic sequence inserted in the TST is 2 characters longer than the PAM sequence to allow for searches with up to two DNA bulges. For example, for the PAM in Paper Fig. 2, the tool inserts a string of length 25 nucleotides (23 nucleotides for the guide plus PAM sequence as well as an additional two nucleotides for DNA bulges). The additional nucleotides represent the appropriate PAM flanking (upstream or downstream) sequence in the reference genome. Paper Fig. 4, shows an example of TST construction, by considering three strings (candidate off-target sites) named CT1, CT2, and CT3. The algorithm first inserts CT1, which is represented (by construction) by the central vertical path of the TST. The algo-

rithm then inserts CT2 by visiting the first seven central nodes (corresponding to the common prefix with CT1). Since the eighth base (C) does not match with G, a new left child is created (since C < G). The procedure recursively completes the insertion by forming the left vertical path of the TST. CT3 is similarly inserted, by sharing a common prefix of length three after which a new branch is created on the right starting from T (T > A).

5 Ternary Search Tree for efficient genome search

The candidate off-target site search is a function that recursively visits the TST, in a similar manner as during the TST construction, described in the previous section. With the constructed TST, the search is performed in two different ways, the first only considers mismatches (hereafter referred to as 'mismatch-search'), and the second considers mismatches in which bulges are permitted (hereafter referred to as 'mismatch-bulge-search'). In the mismatch-search type, it is not possible to reach Any leaf of the TST by construction (see Section 2.4) and every candidate off-target site is written only once (a candidate off-target site cannot be saved, for example, with 2 mismatches and with 4 mismatches considering the same guide and the same genomic position). The mismatch-search type stops either when the pattern (the candidate off-target site) has been found or when the number of maximum mismatches has been exceeded (the pattern, or candidate off-target site, is not reported). In the mismatch-bulge-search type, mismatches and bulges are permitted for candidate off-target sites (see Fig. 4 for an example). To avoid computationally expensive search iterations on the same branch, the algorithm implements a recursive approach that tries all possible combinations of results a candidate off-target site can generate. Those combinations are tested recursively on the TST branch at runtime, avoiding an iterative search restarting every time from the TST root. For example, if the target guide is *CCCAACCC*, two possible combinations with bulges that share trace-backs in the recursion are *CCCA-ACCC* and *CCCAA-CCC*, since they share the common prefix *CCCA*. Every time an off-target site has been saved, the algorithm computes a recursive trace-back and tests further *combination* of mismatches and bulges on the same branch.

6 Guides comparison and results analysis between reference genome and enriched geneome with variants

Fig. S2 and S3 show the different behaviour of two guides extracted from the therapeutic dataset for targeting CCR5. The images are divide in three plots. The first plot, the radar chart, is created to help the user to assess the guide behaviour, compared to the Gecko Library v2. By looking at the area formed by the point on the y-axis, the user can quickly evaluate the guide behaviour in

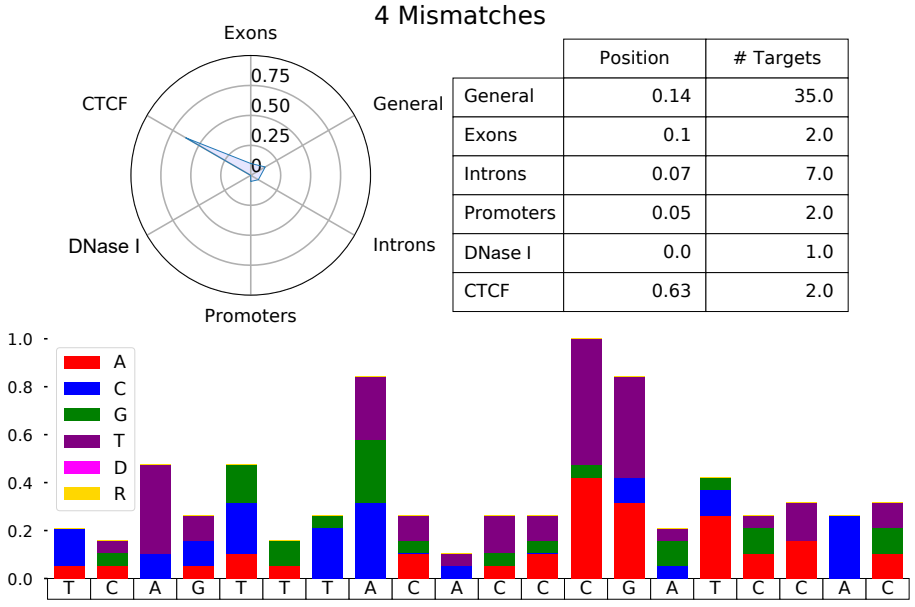

Figure S2: A complete graphical visualization of a CCR5 guide with 4 mismatches allowed. The guide considered is *TCAGTTTACACCCGATCCAC*. This is a good guide, with respect to the radar chart and the table, compared to all others guides of the Gecko Library. This is visible from the small area in the radar chart and from the position values close to 0 in the table and, for example, in the exons region, the guide shows a y value of 0.1 and an off-target count of 2.

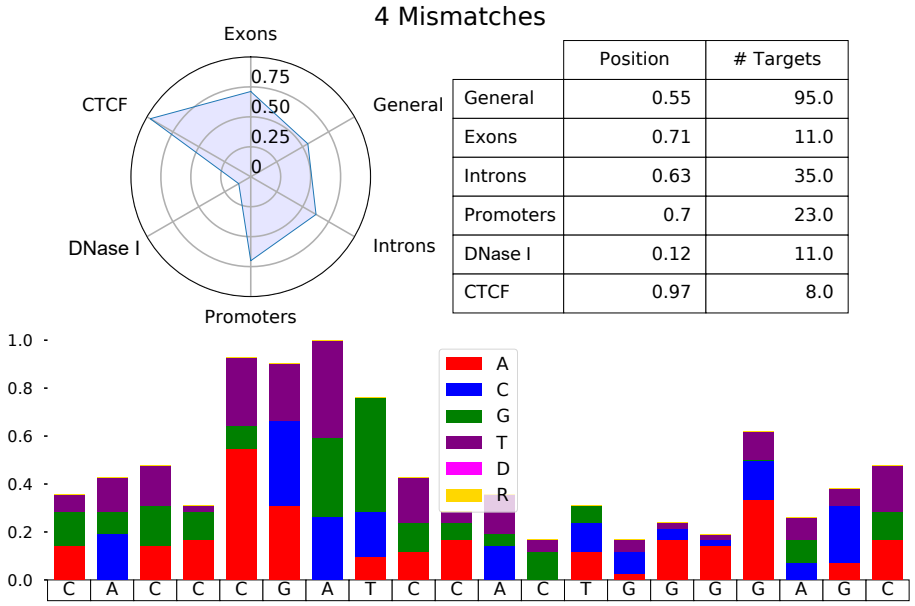

Figure S3: A complete graphical visualization of a CCR5 guide with 4 mismatches allowed. The guide considered is *CACCCGATCCACTGGGGAGC*. This is a medium guide, with respect to the radar chart and the table, compared to all others guides of the Gecko Library. This is visible from the medium area in the radar chart and from the position values range from to 0.12 to 0.97 in the table and, for example, in the exons region, the guide shows a y value of 0.71 and an off-target count of 11.

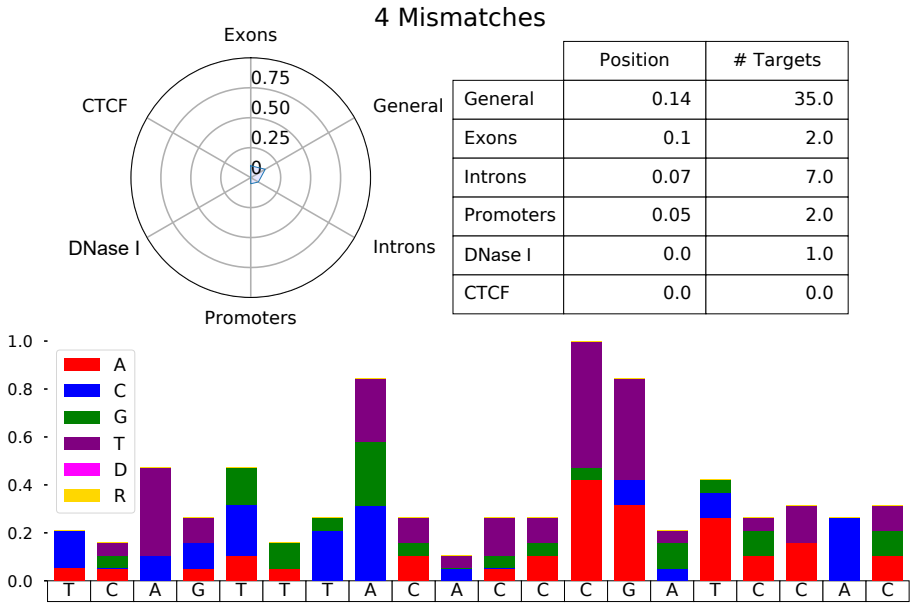

Figure S4: A complete graphical visualization of a CCR5 as reported in Fig. S2 but using for the CTCF annotation only putative binding sites within peaks predicted by a motif model from the Jaspar database (Khan *et al.*, 2017). Comparing this analysis with the one in Fig. S2we observe a decrease from 2 to 0 due to the use of a more precise CTCF annotation.

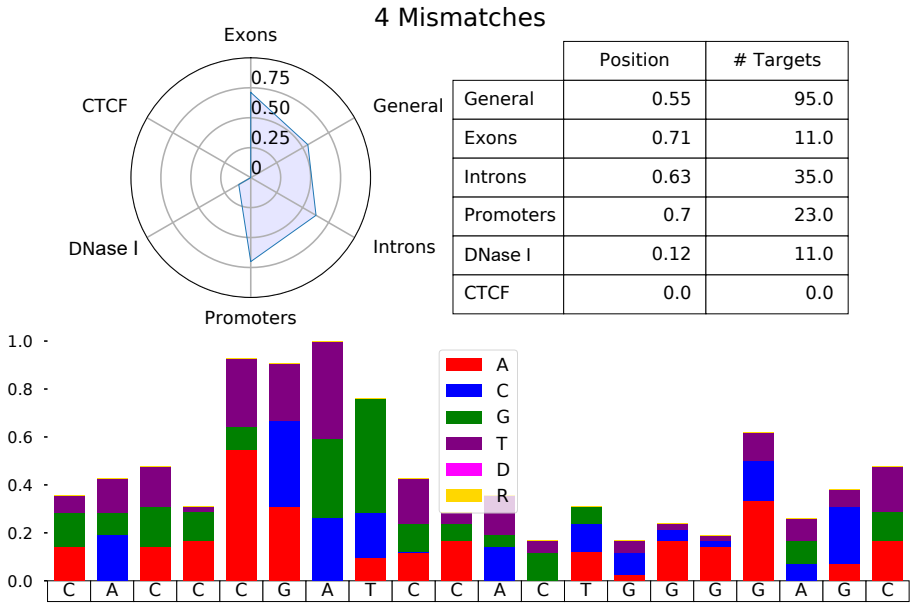

Figure S5: A complete graphical visualization of a CCR5 guide as reported in Fig. S3 but using for the CTCF annotation only putative binding sites within peaks predicted by a motif model from the Jaspar database (Khan *et al.*, 2017). Comparing this analysis with the one in Fig. S3 we observe a net decrease from 8 to 0 due to the use of a more precise CTCF annotation.

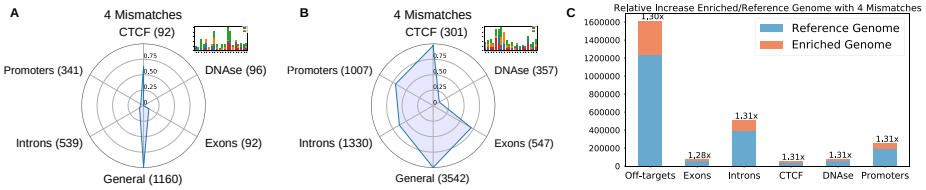

Figure S6: Visual representation of CRISPRitz results. (A) and (B) show behaviours of 2 guides from the CCR5 set (hg19 enriched genome, with up to 4 mismatches and 1 DNA and 1 RNA bulge). (A) and (B) were created by comparing results from the CCR5 dataset with the previous computed results based on the Gecko Library v2, as explained in Paper Section 2.6. (C) Barplot to show the relative increase in the count of off-targets when accounting for genetic variants from the 1000 Genomes project.

different genomic regions. The second plot, the table, is created to summarized the guide behaviour in a more schematic way, showing the actual position of each point in the y-axis of the radar chart and showing the actual count. The third plot, the motif barplot, represents the motif matrix of the guide, showing the count of every nucleotide in every base-pair to quickly assess mutational preferences. To improve the readability, every column is normalized against the maximal value of the entire plot. Fig. S4 and S5 show the same analysis reported in Fig. S2 and S3 changing the annotation for the CTCF binding sites, previously based on ChIP-seq peaks from ENCODE to predicted binding sites by a motif model. To this end, we used FIMO (Grant *et al.*, 2011), with default settings and a Position Weight Matrix (PWM) model from the Jaspar database (Khan *et al.*, 2017) to find all the putative binding sites within the CTCF peaks. As expected, we found that fewer putative off-targets overlapping with CTCF binding sites predicted by the PWM. More precisely, testing the guides targeting CCR5 (4 mismatches, no bulges and the enriched hg19 genome), we found that the number of targets in CTCF peaks with no motif restriction to binding sites are 1045 in the reference genome and 1642 in the enriched variant genome; instead when considering the predicted binding sites based on the PWM model we obtain only 73 in hg19 reference and 104 in hg19 variant genome. We believe this analysis should better reflect the potential disruption of CTCF binding sites, and that the peak based annotation may overestimate this effect. Therefore we provided the PWM based annotation by default in `CRISPRitz`.

We also include an analysis, presented in Fig. S6 to show the general behaviour of CCR5 guide dataset when including bulges in the search. The comparison was computed allowing up to 4 mismatches and 1 DNA plus 1 RNA bulge. The figure shows how dramatic may be the differences between two potential guides targeting the same gene. This figure also illustrates the increase in terms of putative off-target sites when using an enriched genome with respect to a reference genome.

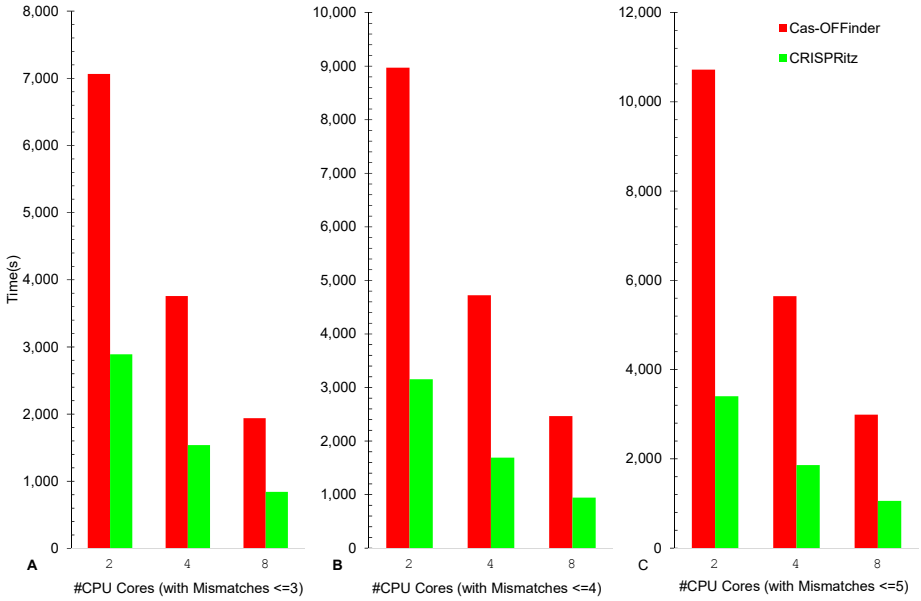

Figure S7: Scalability comparison between `CRISPRitz`and Cas-OFFinder.

7 Software Scalability and performance evaluation as function of the number of discovered off-targets

Fig. S7 shows the scalability of CRISPRitz compared to Cas-OFFinder by varying the number of CPU cores (from 2 to 8) used for the guide search (mismatch threshold of 3, 4, and 5). The figure shows that both the tools scale very well over the number of CPU cores used for the computation. By doubling the number of cores, the obtained speedup is almost linear ($\simeq 1.8$). This underlines the portability of CRISPRitz to parallel architectures. The tests were performed with the random dataset containing 1000 guides, on a machine equipped with an Intel(R) Xeon(R) CPU E5-2650 v4 with 8 cores, clocked at 2200 Mhz and 64 GBs RAM, and the Ubuntu operating system (version 16.04). Table S1 reports execution time of CRISPRitz and Cas-OFFinder as a function of the number of discovered off-targets per guide. We have sampled 3 groups of 10 guides (based on the hg19 reference genome) with similar but increasing number of off-targets (50, 10000 and 1000000). On this dataset, we run CRISPRitz and Cas-OFFinder with 2 cores, 4 mismatches and no bulges. This analysis shows that the number of the off-targets per guide has almost no impact on the execution time. Notably, the speed-up of CRISPRitz over Cas-OFFinder is always close to 2.5x.

Software	50 Off-Targets	10000 Off-Targets	1000000 Off-Targets
CRISPRitz	71.82	81.86	82.93
Cas-OFFinder	200.80	201.38	211.53

Table S1: Execution time of CRISPRitz and Cas-OFFinder as a function of the number of discovered off-targets per guide. We have sampled 3 groups of 10 guides (based on the hg19 reference genome) with similar but increasing number of off-targets (50, 10000 and 1000000). On this dataset, we run CRISPRitz and Cas-OFFinder with 2 cores, 4 mismatches and no bulges. This analysis shows that the number of the off-targets per guide has almost no impact on the execution time. Notably, the speed-up of CRISPRitz over Cas-OFFinder is always close to 2.5x.

8 Time Comparison between CRISPRitz and other CRISPR/Cas analysis software

We compared CRISPRitz with other 7 software (Cas-OFFinder (Bae *et al.*, 2014), Flashfry (McKenna and Shendure, 2018), Off-spotter (Pliatsika and Rigoutsos, 2015), CRISPOR (Haeussler *et al.*, 2016), CHOPCHOP (Montague *et al.*, 2014), CRISPRseek (Zhu *et al.*, 2014), and CRISPRtool (Lessard *et al.*, 2017)) which perform off-target analysis and guide design. The tests were executed on a machine equipped with an Intel(R) Xeon(R) CPU E5-2650 v4, clocked at 2200 Mhz and 64 GBs RAM, and the Ubuntu operating system (version 16.04). Experiments were run with 2 core on CRISPRitz and Cas-OFFinder because they support the parallel execution, and 1 core on all the others, to better reflect a typical mid-size personal computer. We used the random generated dataset of 1000 guides as described in Section 2.7. In Table S2 the guide search is performed with 3, 4, 5 mismatches and no bulges on the hg19 reference. As expected, the software that run a pre-processing step to build a database containing candidate targets outperform all the other software if more than one guide is searched. Among the tools that support only searches with mismatches and are based on a precomputed index, Flashfry is the fastest followed by Off-Spotter and CHOPCHOP. However, among the software that can perform an online search without a pre-processing step, CRISPRitz is the fastest. Table S3 reports the search running time allowing mismatches on the hg19 enriched genome with variants from the 1000 Genome project encoded using the IUPAC notation (see Section 2.3 in Paper). CRISPRitz and CRISPRtool are the only 2 software capable of accounting for genetic variants during the search. CRISPRitz outperforms CRISPRtool in every test reaching a speed-up of 35x when using 5 mismatches. In Table S4 we show the results searching with 3, 4, 5 mismatches, 1 DNA and 1 RNA bulges using the hg19 reference. CRISPRitz and Cas-OFFinder are the only 2 software that can perform this search since the other tools dont support the incorporation of bulges. In this case, CRISPRitz indexes the reference genome to perform analysis with bulges (see Section 2.4). This step is independent of the guide to search and performed

only once. CRISPRitz outperforms Cas-OFFinder in every test reaching 74x of speed-up in the 3 mismatches case. Notably, the time required by CRISPRitz to build the database and run 1 search is about 40x less of the time than Cas-OFFinder takes to perform 1 search. In Table S5, the guide search is run with 3, 4, 5 mismatches, 1 DNA and 1 RNA bulges and using the hg19 enriched genome as defined before. The only software capable of performing this analysis is CRISPRitz.
Finally, Table S6 summarizes the features, limitations, and capabilities of the existing software. Several software share the same capabilities and limitations. Only CRISPRitz and Cas-OFFinder perform bulge analysis. Only CRISPRitz and CRISPRtool can search using reference genomes and accounting for genetic variants. We believe this extensive comparison highlights the utility and flexibility of CRISPRitz in performing exhaustive and complete searches using enriched genomes, any number of mismatches and bulges, support for arbitrary PAM, activity and off-target scores and providing visual reports based on user defined functional annotations.

Software	**3 MM**	**4 MM**	**5 MM**	**DB Time**	**RAM**
CRISPRitz	2890,51	3154,05	3402,73	-	4,26
CAS-OFFinder	7063,56	8969,93	10721,75	-	0,77
FlashFry	40,49	58,44	106,95	3268,29	1,43
OFF-Spotter	114,34	182,28	499,64	743,41	38,09
CRISPOR	5133,99	6160,78	7392,93	3742,59	64,71
CHOPCHOP	1791,01	ND	ND	6744,57	57,41
CRISPRseek	97283,42	100135,78	104152,26	-	65,28
CRISPRtool	20599,78	49637,65	109127,31	-	10

Table S2: Running time for searches with only mismatches (3,4 or 5) on the reference genome (hg19). To simulate a common laptop as a execution platform, the searches were run on a virtual machine using 2 cores for CRISPRitz and Cas-OFFinder (to account for their parallel support), and 1 core for the others. Time are expressed in seconds and RAM consumption in GBytes.

Software	3 MM	4 MM	5 MM	DB Time	RAM
CRISPRitz	2835,65	3078,14	3300,39	-	5,06
CAS-OFFinder	-	-	-	-	-
FlashFry	-	-	-	-	-
OFF-Spotter	-	-	-	-	-
CRISPOR	-	-	-	-	-
CHOPCHOP	-	-	-	-	-
CRISPRseek	-	-	-	-	-
CRISPRtool	24534,67	56345,76	116437,56	-	11,23

Table S3: Running time for searches with mismatches (3,4 or 5 mismatches) on enriched genome (hg19) with genetic variants. Variants are from the 1000 Genome Project. To simulate a common laptop as a execution platform, the searches were run on a virtual machine using 2 cores for CRISPRitz and Cas-OFFinder (to account for their parallel support), and 1 core for the others. Time are expressed in seconds and RAM consumption in GBytes.

Software	3 MM	4 MM	5 MM	DB Time	RAM
CRISPRitz	2132,54	14258,46	52687,21	1832,24	7,12
CAS-OFFinder	158853,74	201561,34	245363,59	-	2,17
FlashFry	-	-	-	-	-
OFF-Spotter	-	-	-	-	-
CRISPOR	-	-	-	-	-
CHOPCHOP	-	-	-	-	-
CRISPRseek	-	-	-	-	-
CRISPRtool	-	-	-	-	-

Table S4: Running time for searches with mismatches (3,4 or 5) and bulges (1 DNA and 1 RNA bulges) on the reference genome (hg19). To simulate a common laptop as a execution platform, the searches were run on a virtual machine using 2 cores for CRISPRitz and Cas-OFFinder (to account for their parallel support), and 1 core for the others. Time are expressed in seconds and RAM consumption in GBytes.

Software	**3 MM**	**4 MM**	**5 MM**	**DB Time**	**RAM**
CRISPRitz	8627,13	57475,97	210748,84	1832,24	8,17
CAS-OFFinder	-	-	-	-	-
FlashFry	-	-	-	-	-
OFF-Spotter	-	-	-	-	-
CRISPOR	-	-	-	-	-
CHOPCHOP	-	-	-	-	-
CRISPRseek	-	-	-	-	-
CRISPRtool	-	-	-	-	-

Table S5: Running time for searches with mismatches (3,4 or 5) and bulges (1 DNA and 1 RNA bulges) on enriched genome (hg19) with genetic variants. Variants are from the 1000 Genome Project. To simulate a common laptop as a execution platform, the searches were run on a virtual machine using 2 cores for CRISPRitz and Cas-OFFinder (to account for their parallel support), and 1 core for the others. Time are expressed in seconds and RAM consumption in GBytes.

Software	MM	Bulges	Enzymes	Genomes	IUPAC	Score	Annotation
CRISPRitz	0-Any	0-Any	User specified	Every fasta genome in UCSC format	Yes	Doench2016azimuth, CFD	Any
CAS-OFFinder	0-Any	0-Any	User specified	Every fasta genome in UCSC format	No	None	None
FlashFry	0-Any	None	SpCas9, Cpf1	Every fasta genome in UCSC format	No	Hsu2013, Doench2014, Doench2016azimuth, Moreno-Mateos	Any
OFF-Spotter	0-5	None	SpCas9NGG, SpCas9NAG, saCas9, CjCas9	GRCh38, GRCh37, GRCm38, w303	No	User customizable	Any
CRISPOR	0-5	None	SpCas9, saCas9, CjCas9, Cpf1	Every fasta genome in UCSC format	No	MIT, CFD, Doench2016azimuth, Doench2016old, Chari, Xu, Wu-Crispr, Doench2014, Wang, Moreno-Mateos, Azimuth in-Vitro, CCTop, deepCpf1, Najm et al 2018	Any
CHOPCHOP	0-3	None	User specified	very fasta genome in UCSC format	No	Doench2016azimuth, Chari, Xu, Doench2014, Moreno-Mateos	Any
CRISPRseek	0-Any	None	User specified	All BSgenome available	No	Hsu 2013	mRNA exons
CRISPRtool	0-Any	None	User specified	All BSgenome available	Yes	Sanjan2014, Hsu2013, CFD	None

Table S6: Software features comparison.

References

Bae, S. *et al.* (2014). Cas-offinder: a fast and versatile algorithm that searches for potential off-target sites of cas9 rna-guided endonucleases. *Bioinformatics*, **30**(10), 1473–1475.

Bentley, J. and Sedgewick, B. (1998). Ternary search trees. *Dr. Dobb's Journal*, **23**(4).

Grant, C. E. *et al.* (2011). Fimo: scanning for occurrences of a given motif. *Bioinformatics*, **27**(7), 1017–1018.

Haeussler, M. *et al.* (2016). Evaluation of off-target and on-target scoring algorithms and integration into the guide rna selection tool crispor. *Genome biology*, **17**(1), 148.

Khan, A. *et al.* (2017). Jaspar 2018: update of the open-access database of transcription factor binding profiles and its web framework. *Nucleic acids research*, **46**(D1), D260–D266.

Lessard, S. *et al.* (2017). Human genetic variation alters crispr-cas9 on-and off-targeting specificity at therapeutically implicated loci. *Proceedings of the National Academy of Sciences*, page 201714640.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *bioinformatics*, **25**(14), 1754–1760.

McKenna, A. and Shendure, J. (2018). Flashfry: a fast and flexible tool for large-scale crispr target design. *BMC biology*, **16**(1), 74.

Montague, T. G. *et al.* (2014). Chopchop: a crispr/cas9 and talen web tool for genome editing. *Nucleic acids research*, **42**(W1), W401–W407.

Pliatsika, V. and Rigoutsos, I. (2015). off-spotter: very fast and exhaustive enumeration of genomic lookalikes for designing crispr/cas guide rnas. *Biology direct*, **10**(1), 4.

Zhu, L. J. *et al.* (2014). Crisprseek: a bioconductor package to identify target-specific guide rnas for crispr-cas9 genome-editing systems. *PloS one*, **9**(9), e108424.