Faster and better CRISPR guide RNA design with the Crackling method

Jacob Bradford [1], Timothy Chappell [1] and Dimitri Perrin [1]*

[1] School of Computer Science, Queensland University of Technology, Brisbane, 4000, Australia

*To whom correspondence should be addressed.

Abstract

Motivation: CRISPR-Cas9 systems have become a leading tool for gene editing. However, the design of the guide RNAs used to target specific regions is not trivial. Design tools need to identify target sequences that will maximise the likelihood of obtaining the desired cut, and minimise the risk of off-target modifications. Achieving this across entire genomes is also computationally challenging. There is a clear need for a tool that can meet both objectives while remaining practical to use on large genomes.
Results: Here, we present *Crackling*, a new method for whole-genome identification of suitable CRISPR targets. We test its performance on 12 genomes, of length 375 to 9965 megabases, and on data from validation studies. The method maximises the efficiency of the guides by combining the results of multiple scoring approaches. On experimental data, the set of guides it selects are better than those produced by existing tools. The method also incorporates a new approach for faster off-target scoring, based on Inverted Signature Slice Lists (ISSL). This approach provides a gain of an order of magnitude in speed, while preserving the same level of accuracy. Overall, this makes Crackling a faster and better method to design guide RNAs at scale.
Availability: Crackling is available at https://github.com/bmds-lab/Crackling under the Berkeley Software Distribution (BSD) 3-Clause license.
Contact: dimitri.perrin@qut.edu.au

1 Introduction

CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) is an adaptable immune system found in archaea and bacteria [13]. Through extensive research, many wild-type and engineered CRISPR nucleases exist and present great opportunity in the field of gene editing. Wild-type CRISPR provides immunity in three steps [12]:

1. when infected by phage, a DNA snippet is obtained and stored within the CRISPR array, constructing a memory of past viral infection;
2. the array is transcribed to produce duplicates of previously obtained DNA snippets (or *guides*);
3. a guide and an RNA-guided endonuclease (e.g. Cas9, in the case of *S. pyogenes*) binds to enable site-specific cleavage.

This last step is the basis for the use of CRISPR systems for gene editing: the endonuclease is delivered to the cells to be edited, along with a synthetic guide RNA that contains a short sequence ($\tilde{2}$0bp) corresponding to the genomic region being targeted. The design of this guide RNA is a crucial step for any CRISPR experiment. However, guide design is not trivial, as the *efficiency* and *specificity* of guides are crucial factors. Efficiency broadly refers to the guide correctly binding to the targeted region and to the endonuclease. This is influenced by a number of factors that depend both on the guide itself (e.g. secondary structure) and on the targeted region (e.g. chromatin accessibility). Specificity refers to the need for the guide RNA not to induce off-target modifications. This means not only that the targeted sequence must be unique, but also that closely related sequences (typically less than four mismatches) elsewhere in the genome must be carefully considered.

We recently benchmarked existing guide design tools [2]. Our two main findings were that:

1. When considering efficiency, for any given genomic sequence there is a limited overlap between the set of guides that each tool is producing.
2. Several tools have inadequate filtering on specificity, and when that is considered carefully, it tends to be a computationally expensive task.

The limited overlap between the tools can be exploited for guide selection: when a guide is recommended by multiple tools, there is a higher chance that it will actually be efficient. We explored consensus approaches in detail [3], and they provide better guides. However, it is not necessarily practical to run several tools. Our *Crackling* method directly integrates multiple scoring approaches.

A number of methods have been implemented to tackle specificity evaluation. One of the earliest methods, published in 2014, is *Cas-OFFinder* [1]. The algorithm, as seen in Figure 1, is as follows: (i) read the genome sequence from a FASTA formatted file, distribute it amongst the processing units available on the machine and identify off-target sites; (ii) compare the off-target sites with the candidate guides which have been provided, count the number of mismatches between each of these sequence-pairs, and; (iii) if the number of mismatches is less than a threshold (e.g. four), then write the pair to file. The resulting file can then be used to get a sense of the off-target risk for each candidate guide. Cas-OFFinder utilises multi-threaded programming and is capable of running on a modern workstation.

```
Require: $O$, $C$, $N$, where
    $O$ = list of off-target sites,
    $C$ = list of candidate guides to be evaluated,
    $N$ = maximum number of mismatches
1: for all $c \in C$ do
2:     for all $o \in O$ do
3:         $n \leftarrow$ count number of mismatches between $c$ and $o$
4:         if $n < N$ then
5:             write $c : o : n$ to file
6:         end if
7:     end for
8: end for
```

Fig. 1. Algorithm for Cas-OFFinder. The authors describe the algorithm in three wrappers, where wrappers 1 and 2 are parallelised using OpenCL kernels: (i) a FASTA formatted genome is read file and distributed amongst processing units to identify off-target sites; (ii) the number of mismatches between each off-target site and each candidate guide is calculated, and; (iii) if the number of mismatches for a pair is less than a threshold (e.g. four), then the pair is written to file.

While identifying off-target sites that are within a fixed number of mismatches is useful, it is not necessarily enough. It has been experimentally shown that the position of the mismatches also matters; a risk score can be derived from the number and position of the mismatches [10]. This score is sometimes referred to as the *Zhang* score, from the corresponding author of that paper. To calculate the score, a candidate guide is compared against all potential CRISPR sites in the genome of interest. When a site is at most four mismatches away, a *local* score is calculated using this formula:

$$\prod_{e \in M} (1 - W[e]) \times \frac{1}{\frac{19-d}{19} \times 4 + 1} \times \frac{1}{n^2} \qquad (1)$$

where W is an array of position-specific weights, d is the pairwise distance between mismatches, n is the number of mismatches between target and sequence, and M is the list of mismatch positions. The *global* score for the candidate guide is then obtained by combining all the local scores:

$$S_{global} = \frac{100}{100 + \sum_{i=1}^{q} S_i} \qquad (2)$$

where S_i represents the local score between the candidate guide and off-target site i that partially matches the target (calculated using Eq. 1), and q is the number of such sites that partially matched.

Published in 2014 like Cas-OFFinder, the *mm10db* method uses a custom implementation of that score [18], in a function called *findMismatches*. The objective of that function is to reject all guides for which the global score is below a fixed threshold $\tau = 0.75$. Based on Eq. 2, $S_{global} \leq \tau$ is equivalent to Eq. 3.

$$\sum_{i=1}^{q} S_i \geq \frac{100}{\tau} - 100 \qquad (3)$$

Instead of calculating an exact global score, it is therefore possible to keep track of the running sum of local scores, and terminate early as soon as a guide is guaranteed to end up being rejected. The overall algorithm is summarised in Figure 2. The method supports multi-threading, to score multiple candidate guides in parallel. The initial implementation of findMismatches relies on string comparisons. For this paper, we have reimplemented it using a binary encoding of the sequences, so that mismatches can be identified more efficiently.

The recently published *Crisflash* also relies on the Zhang score to evaluate guides, but without the early termination. It is reported that Crisflash can out-perform Cas-OFFinder by an order of magnitude on the human genome when more than a couple of hundred guides are supplied [11]. Crisflash also provides support for variants, which are not used here. It is also noted that Crisflash relies on a volume of memory which would not be available on a modern workstation (90 Gb), but rather a server cluster. Figure 3 describes how the tool utilises a tree structure for evaluating the specificity of a guide.

FlashFry presents a discovery approach that utilises a table of bit-encoded off-target prefixes [16]. The algorithm is summarised in Figure 4.

```
Require: $C$, $O$, $S$ where
    $C$ = list of candidate guides to be scored,
    $O$ = list of off-target sites,
    $S$ = sum of local Zhang scores
1: for all $c \in C$ do
2:     $S \leftarrow 0$
3:     for all $o \in O$ do
4:         if number of mismatches between $c$ and $o \leq 4$ then
5:             S ← S + local Zhang score of $c$ against $o$
6:         end if
7:         if $S > threshold$ then
8:             Break inner loop
9:         end if
10:    end for
11:    $S \leftarrow$ global Zhang score of $S$
12:    Print $c : S$ to file
13: end for
```

Fig. 2. Algorithm for findMismatches and findMismatchesBit, for which the comparison of characters in findMismatches is replaced with bit operations in findMismatchesBit.

```
Require: $O$, $T$, $C$, $M$, $S$ where
    $O$ = list of off-target sites,
    $T$ = N-ary tree containing identified off-target sites,
    $C$ = list of candidate guides to evaluate,
    $M$ = maximum number of mismatches,
    $S$ = sum of local Zhang scores

    (i) Construct off-target sites tree
1: for all $o \in O$ do
2:     for all $i \in$ length of $o$ do
3:         $T \leftarrow Add\_node(o[i])$
4:     end for
5: end for

    (ii) Evaluate candidate guides against off-target sites tree
6: for all $c \in C$ do
7:     $m \leftarrow$ number of mismatches
8:     $i \leftarrow$ first node of tree
9:     while $m < M$ and $i \leq$ depth of tree do
10:        if node $i$ of tree is different to base $i$ of $s$ then
11:            $m \leftarrow m + 1$
12:        end if
13:        $i \leftarrow$ move to next node
14:    end while
15:    if $m \leq M$ and $i =$ depth of tree then
16:        Calculate Zhang score for $c$
17:    end if
18: end for
```

Fig. 3. Algorithm for Crisflash.

```
Require: $O, T, C, M, S$ where
    $O =$ list of off-target sites,
    $T =$ Off-targets table,
    $C =$ list of candidate guides to evaluate,
    $M =$ maximum number of mismatches,
    $S =$ candidate guide score

    (i) Database construction
1: for all $o \in O$ do
2:    $T$[prefix of $o$] $\leftarrow o$
3: end for

    (ii) Off-target discovery and scoring
4: for all $c \in C$ do
5:    for all $o \in T$[prefix of $c$] do
6:       if number of mismatches between $c$ and $o \leq M$ then
7:          $S \leftarrow S+$ score of $c$ against $o$
8:       end if
9:    end for
10: end for
```

Fig. 4. Algorithm for FlashFry.

Similar to other methods, a candidate guide is only scored on a given off-target if the number of mismatches is fewer than the specified threshold. FlashFry does not report the Zhang score, but rather the cutting-frequency determination (CFD) score [8]. CFD is more descriptive than Zhang, such that indels and nucleic identity changes are also considered. The authors claim that FlashFry is able to run two to three orders of magnitude faster than Cas-OFFinder.

Other tools such as CRISPRseek [21], CasOT [20] and CRISPRitz [4] exist. However, we do not consider them here as already published results show that they perform poorly compared to others which we have included [11, 4].

2 Approach

In this article, we present Crackling, a method aimed at addressing the two challenges of efficiency and specificity when designing CRISPR-Cas9 guide RNAs. To increase the efficiency of the set of guides we produce, we combined three scoring strategies. To speed up the off-target scoring, we introduce a solution for constant-time lookup of sequence neighbourhoods.

The tool is implemented in Python (version 3) for most steps, with the high-performance off-target scoring in C++. It can run on any platform. Minimal dependencies are required: Bowtie2 [14] for guide realignment onto the input genome and RNAfold [15] for secondary structure prediction. Where possible, components of the tool are parallelised for improved performance. Pre-processing is minimal, only requiring the input genome to be indexed by Bowtie2 and off-target sites to be indexed for ISSL. The ISSL index is simple to build and can be reused at any time for the genome it is constructed for.

To ensure that any user is able to utilise the full capability of the tool, a well-documented, Python-based configuration method is implemented. The configuration of the pipeline is duplicated when ran and once configured, the pipeline is called via any command line terminal using the Python3 interpreter. Post-processing, which provides the ability to annotate guides with any genomic features which they target, is an optional final step, for which we also provide code.

Multi-approach efficiency evaluation

We previously showed that, when tools actively filter or score guides based on their predicted efficiency, they only rarely agree with each other [2]. We also showed that this can be leveraged for consensus-based approaches that combine the output of multiple tools [3]. However, having to install and run multiple tools limits how practical such approaches can be.

Here, our Crackling tool directly incorporates three scoring approaches and only recommends candidate guides that have been accepted by at least two of the approaches. As summarised in Table 1, we are combining elements from three methods to assess candidate guides:

1. As in CHOPCHOP [17], we are looking for the presence of a guanine at position 20 (also known as the *G20* rule).
2. We score guides based on the model from sgRNAScorer 2.0 [7], and consider candidate guides to be accepted if their score is positive.
3. We used the filtering steps from mm10db [18] (GC content, secondary structure, etc.), and only accept candidate guides that passed all steps.

The sgRNAScorer 2.0 model is included in the Crackling code repository (see Data Availability), in addition to the raw data and a script to retrain the model for version compatibility reasons.

Off-target scoring using Inverted Signature Slice Lists

Inverted Signature Slice Lists (ISSL) can be used to rapidly perform approximate nearest neighbourhood searches in collections of locality-sensitive signatures [5]. By using fixed-length signatures as search-keys, items in a neighbourhood can be found in constant time. This approach was initially proposed as a high-performance method for searching web-scale collections of data. Given that genomic data is at a comparable scale, it provided a feasible method for evaluating CRISPR guide specificity.

The approach utilises bit-encoding to reduce memory and storage requirements, and to use CPU instructions most effectively. The four letters of the genomic alphabet (*A*, *T*, *C* and *G*) are two-bit encoded, hence a 20-bp genomic sequence requires 40-bits, but is stored in a 64-bit word. The critical 40 bit segment is portioned into $n+1$ slices, where n is the maximum number of mismatches (here, $n = 4$ to match other methods). The position of the slice is retained when the index is generated, thus yielding each slice as a locality-sensitive signature. Each off-target signature is inserted into a dual-keyed, tabular data structure, where $n : signature$ is the key. Finally, the data structure is written to file for later reuse.

Separately, this process is repeated at run time for all candidate guides. Given that n mismatches are allowed and there exist $n+1$ slices, there will always be one slice shared between a candidate guide and a potential off-target site. The time complexity of this discovery step is constant given the structure of the index, and the number of potential off-target sites is less than that in other approaches (such as using a sequential search where all off-target sites are considered). Similar to the findMismatches approach, the candidate guide is scored against off-target sites using the Zhang score until a specified threshold is reached (Eq. 3). At the point when the score drops below the threshold, subsequent off-target sites in the current slice and subsequent slices, are not evaluated. This approach is visualised in Figure 5 and the algorithm is summarised in Figure 6.

Performance evaluation

In this paper, we are interested in the computational performance of our off-target scoring relative to methods such as Cas-OFFinder and findMismatches, as well as in the performance of the overall Crackling tool against other end-to-end tools.

For the former, we time each of the off-target tools described above on genomes of increasing size, which were obtained from the National Center for Biotechnology Information (NCBI), see Table 2. For each genome,

Table 1. The consensus approach in Crackling reimplements the approaches of three methods. By default, a candidate guide is predicted to be effective if two of three methods agree.

Tool	Design Principles	Details
mm10db	Filtering based on features such as: GC content, presence of TTTT, gRNA secondary structure	[18]
sgRNA Scorer 2.0	Scoring using machine learning model, trained on [6]	[7]
CHOPCHOP	Flag indicating position 20 containing guanine	[17]

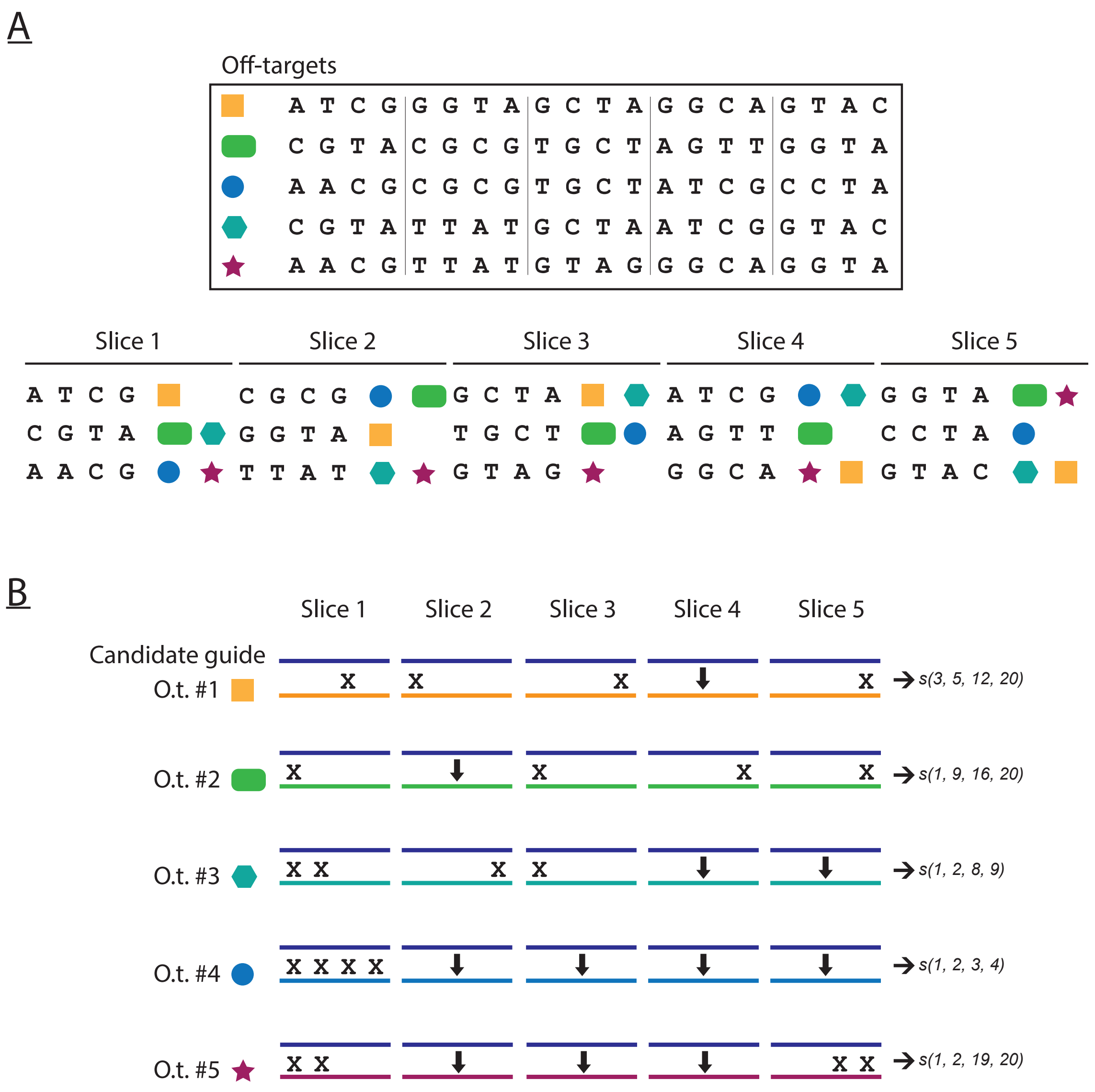

Fig. 5. A: The ISSL off-targets index is a dual-key tabular data structure, constructed using locality-sensitive hashes. The key of the first dimension is the slice number. The key of the second dimension is the bit-encoded genomic sequence of the slice. The contents is the bit-encoded genomic sequence of the off-target site (depicted by the coloured shapes). B: Scoring a candidate guide against off-target sites. An x denotes a mismatch between the candidate guide and the off-target site. An arrow indicates that the given slice is common for both sequences. Each candidate guide is portioned into $n + 1$ slices ($n = 4$ in this figure). Each matching slice is used a locality-sensitive look-up key into the index. For each off-target site, the positions of mismatches are used in calculating the specificity score. If more than four mismatches exist, then the sequence is not deemed as an off-target site. An off-target may be identified for multiple slices, however ISSL strictly considers all distinct off-targets per candidate guide (e.g. off-target #3 has been identified twice but only considered for scoring once).

candidate guides were extracted from the forward and reverse strands using a regular expression: $[ACG][ATCG]^{20}GG$ (and the complement for the reverse strand). Two editions of these target sites were created: one including the PAM sequence and one without, as this was required by the tools. This process was repeated for off-target sites, however using the following expression: $[ACG][ATCG]^{20}[AG]G$ (again, with a complementing expression for the reverse strand).

Using the candidate guides extracted for each genome, we further extract five distinct sets of 10,000 candidate guides. For *O. sativa*, we repeated this but for sets of size 5,000 to 25,000, incrementing by 5,000.

```
Require: $B, N, L, C, S$ where
    $B$ = list of bit-encoded off-target sites,
    $N$ = number of mismatches,
    $L$ = list of off-target site slices,
    $C$ = list of target sites,
    $S$ = sum of local Zhang score

    (i) Slice bit-encoded off-target sites into $(N+1)$ portions
1: for all $n \in (N+1)$ do
2:   for all $b \in B$ do
3:     $L[n][n$'th portion of $b] \leftarrow b$
4:   end for
5: end for

    (ii) Evaluate specificity of each target site.
    Perform step (i) for $C$ in place of $O$
6: for all $c \in C$ do
7:   $S \leftarrow 0$
8:   $slices \leftarrow$ slice $x$ into $(N+1)$ slices
9:   for all $i \in (N+1)$ do
10:    if $slices[i] \in L[i]$ then
11:      $n \leftarrow$ number of mismatches between $c$ and off-target sites of $L$
12:      if $n \leq N$ then
13:        $S \leftarrow$ local Zhang score of $t$
14:      end if
15:    end if
16:  end for
17:  $S \leftarrow$ global Zhang score for $S$
18:  Print $c : S$ to file
19: end for
```

Fig. 6. Algorithm for ISSL

Table 2. The genomes used in this study. Mb is megabases. An asterisk indicates these genomes were only used on the high-performance machine due to memory limitations.

Genome	Size	Number of off-target sites
$O.sativa$	374.4 Mb	61.7×10^6
$P.hallii$	507.4 Mb	91.9×10^6
$X.couchianus$	685.5 Mb	100.6×10^6
$P.major$	998.3 Mb	173.0×10^6
$C.viridis$	1,297.0 Mb	193.8×10^6
$D.rerio$	1,679.4 Mb	214.8×10^6
$O.mykiss$	1,950.0 Mb	295.9×10^6
$M.musculus$	2,730.7 Mb	499.7×10^6
$M.domestica$	3,502.4 Mb	570.9×10^6
$T.urartu$	4,661.6 Mb	829.4×10^6
$R.bivittatum*$	5,178.6 Mb	984.7×10^6
$T.turgidum*$	9,964.3 Mb	$1{,}763.2 \times 10^6$

For each of these, we tailored copies for tools which required custom formats: (i) Crisflash: a multi-FASTA formatted file, (ii) Cas-OFFinder: a space-separated values file where the first item is the candidate guide and the second item is the maximum number of mismatches for the guide (we chose four for all tools), and where the first line is the directory which contains a file for each chromosome, and the second line is the IUPAC-formatted pattern for identifying off-target sites. ISSL required an index of the off-target sites. Crisflash required a single genome file, as opposed to a file for each chromosome. For this, we interspaced each chromosome with sufficient N's to prevent the tool from detecting target sites which may overlap chromosomes. FlashFry required a custom index for each genome but could not produce one larger than that for *O. mykiss*.

We captured the output of the tools by redirecting the standard streams to file. However, for Crisflash, they were redirected to */dev/null* as the tool produced hundreds of gigabytes of data.

For findMismatches, findMismatchesBit, ISSL and Crisflash, we modified the source code to time the guide specificity evaluation method in each tool. Additionally, for Crisflash, we made modifications to time the construction of its tree data structure. For Cas-OFFinder, we timed it with the *time* application with microsecond precision. All preliminary tests were performed on a high-performance Linux workstation, with two 18-core Intel Xeon E5-2699 v3 (2.3 GHz) CPU's, 512 GB RAM, 4.2 GB allocated swap space and Hewlett-Packard Enterprise MB4000GEFNA 4TB HDD. Further testing was performed on a different Linux workstation with one 8-core Intel Core i7-5960X (3.0 GHz), 32 GB RAM, 32 GB allocated swap space, and Samsung PM87 SSD.

For all tests, we used a strict 30-hour walltime, as tools slower than this would not scale to the analysis of entire genomes.

3 Results and Discussion

We ran experiments on two machines, as described earlier. All preliminary tests were executed on a high-performance machine, and followed up on a machine where the available memory is more limited. Cas-OFFinder could not be tested on the high-performance machine due to compatibility issues. For FlashFry, it was not possible to generate the required indexes for genomes larger than *O. mykiss*, and the tool was therefore not tested past this point. Generating the index for the larger genomes was attempted on both machines, with no success.

The ISSL approach powering Crackling is the fastest off-target scoring method

Three tools (findMismatches, findMismatchesBit and ISSL) were capable of completing tests on all twelve genomes on the high-performance machine. They also completed tests on the ten genomes considered on the workstation (with the two largest being excluded due to the memory limitation of this machine). On the high-performance machine, Crisflash failed on genomes larger than *O. mykiss* due to segmentation faults. On the workstation, it saturated physical memory, causing swapping to occur.

For both machines, the run time for findMismatches and findMismatchesBit were proportional, confirming that bit-encoding alone provides a performance benefit. For example, on the high-performance machine with *C. viridis*, using bit-encoding provides a 6.6x speed-up. The results for the *T. urartu* dataset on the workstation are an outlier due to memory being swapped to disk. Overall, these results provide further motivation for bit-encoding, which is also used in ISSL.

On the high-performance machine, ISSL performs an order of magnitude faster than the next best performing tool (findMismatchesBit), and up to two orders of magnitude faster than the worst performing tool (findMismatches). It was able to evaluate 10,000 candidate guides for *O. sativa* in three seconds on average; whereas, findMismatches completed analysis of the same datasets in an average of 7.5 minutes. This is a 153x speed-up. ISSL also significantly outperforms Cas-OFFinder. For instance, it completed the *T. urartu* test in 5.5 minutes on the workstation, compared to 10 hours for Cas-OFFinder. This is a 112x speed-up.

The mean run times for each test are given in Table 3, and visualised in Figures 7 and 8. ISSL is the fastest method for off-target scoring. For large genomes on low-memory machines, findMismatchesBit can also be a useful alternative, due to its lower memory footprint.

For ISSL and Flashfry, the supporting indexes containing off-target sites are generated in a pre-processing step that is required only once. For Crisflash, the tree data structure is constructed at run-time so we modified the source code to include a timer for this. We timed the construction of the indexes for each of these tools and report the results in Table 4, and Figures 9 and 10. Once again, ISSL is the best performing tool; followed by Crisflash and finally Flashfry.

Impact of the number of targets

We ran each tool on the *O. sativa* genome and varied the number of candidate guides to be evaluated. It would be expected that runtime is a linear function of input size, seeing that each candidate guide is evaluated individually. For all tools, this is true. The results are shown in Table 5 and Figure 11 for mean run times over five tests.

Overall, the performance of each tool is similar to that discussed in the previous section: Cas-OFFinder is the poorest performing tool; findMismatches and findMismatchesBit are proportional; ISSL is the best performing. ISSL completed all tests in under 25 seconds, whereas Cas-OFFinder completed the test on the smallest sized dataset in 23 minutes and largest sized dataset in 2 hours. Here, ISSL performed up to 300x faster than Cas-OFFinder, and at least an order of magnitude faster than the next tool.

Crackling is a fast method to produce efficient guides

Previously, we benchmarked leading CRISPR guide design methods using custom datasets derived from the mouse genome [2]. These datasets were of size 500,000 bp (*500k*), 1,000,000 bp (*1m*), 5,000,000 bp (*5m*) and the full 61m bp chromosome 19 (*full*). Some tools require an annotation file, and only consider guides in coding regions. To account for this we calculated the *effective base-pairs per second* (EFPS), which uses the length of the regions used for candidate extraction. Crackling considered the entire input genome, thus the genome length was used when calculating EFPS.

We constructed the ISSL index for these datasets and ran identical experiments on Crackling from the original paper. Crackling was able to

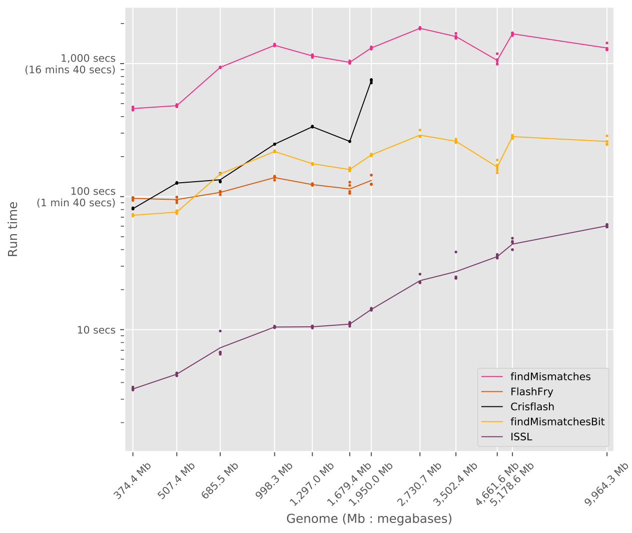

Fig. 7. Run time on the high-performance machine for an increase in genome size. Crisflash produced a segmentation fault when tested on genomes larger than O. mykiss. FlashFry could not produce an index larger than that for O. mykiss.

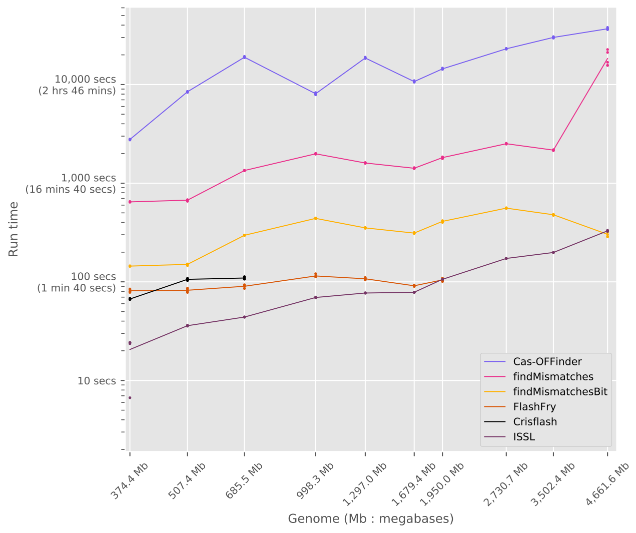

Fig. 8. Run time on the workstation for an increase in genome size. The memory requirements of Crisflash exceeded that available and caused swapping to occur; these tests were stopped.

Table 3. Mean run time, of 5 tests, when genome size is increased (hh:mm:ss). - indicates the test failed due to memory limitations. fMb is findMismatchesBit. fM is findMismatches.

	High-performance machine					**Workstation**					
Genome	ISSL	fMb	fM	FlashFry	Crisflash	ISSL	fMb	fM	FlashFry	Crisflash	Cas-OFFinder
$O.sativa$	00:00:03	00:01:12	00:07:39	00:01:37	00:01:21	00:00:20	00:02:24	00:10:46	00:01:21	00:01:07	00:46:17
$P.hallii$	00:00:04	00:01:16	00:08:02	00:01:35	00:02:06	00:00:35	00:02:30	00:11:12	00:01:22	00:01:46	02:20:49
$X.couchianus$	00:00:07	00:02:28	00:15:37	00:01:47	00:02:13	00:00:44	00:04:56	00:22:24	00:01:30	00:01:49	05:15:20
$P.major$	00:00:10	00:03:38	00:22:52	00:02:19	00:04:07	00:01:09	00:07:19	00:33:05	00:01:54	-	02:14:29
$C.viridis$	00:00:10	00:02:56	00:19:02	00:02:03	00:05:35	00:01:17	00:05:52	00:26:42	00:01:47	-	05:09:51
$D.rerio$	00:00:11	00:02:39	00:17:01	00:01:54	00:04:20	00:01:18	00:05:12	00:23:37	00:01:31	-	02:58:37
$O.mykiss$	00:00:14	00:03:25	00:21:50	00:02:12	00:12:24	00:01:46	00:06:50	00:30:17	00:01:44	-	04:00:19
$M.musculus$	00:00:23	00:04:50	00:30:41	-	-	00:02:52	00:09:18	00:41:52	-	-	06:23:51
$M.domestica$	00:00:27	00:04:20	00:26:37	-	-	00:03:18	00:07:58	00:36:04	-	-	08:20:43
$T.urartu$	00:00:35	00:02:46	00:17:37	-	-	00:05:28	00:05:03	05:06:15	-	-	10:12:32
$R.bivittatum$	00:00:43	00:04:42	00:27:54	-	-					-	
$T.turgidum$	00:01:00	00:04:19	00:21:49	-	-					-	

complete five repeat tests on each genome without failure. The average EFPS for the datasets were:

- *500k*: 9,584 EFPS
- *1m*: 9,992 EFPS
- *5m*: 12,167 EFPS
- *full*: 25,542 EFPS

This is also visualised in Figure 12. In the benchmark, we used this Figure to define low-, medium- and high-performance groups. Crackling enters the high-performance group, and is the only method in that group to correctly filter guides (CasFinder and CRISPR-ERA provide a score, but it did not prove to be informative). Compared to other tools that filter guides, Crackling is at least an order of magnitude faster.

The quality of the guides produced by Crackling is also the highest amongst the tools reviewed. Table 6 describes the precision of each tool on two experimentally validated datasets, *Wang* [19] and *Doench* [9]. As seen in Figure 13 for the *Doench* dataset, Crackling correctly selected many guides that are *efficient* and very few that are *inefficient*, and again for the *Doench* dataset in Figure 14. Notably, Crackling, by default, requires a candidate guide to be accepted by two of the three scoring approaches in order to be selected. When increasing this to require all three approaches to agree, the precision of Crackling will increase (to 91.2% and 48.5% on the *Wang* and *Doench* datasets, respectively), but the recall drops to values that are only acceptable when a low number of candidates is needed (7.1% and 8.9%, respectively).

Taken together, all these results show that Crackling is able to produce high quality results in a relatively short period of time. It is able to perform better than any tool on both critical properties (specificity and efficiency) that are considered when evaluating CRISPR guides.

4 Conclusion

One of the main challenges of experiments using CRISPR-Cas9 systems is the design of suitable guide RNAs. It is crucial, but not trivial, to ensure that these guides will be efficient, but also specific enough not to lead to off-target modifications. There was no tool that could meet both objectives while remaining practical to use on large genomes.

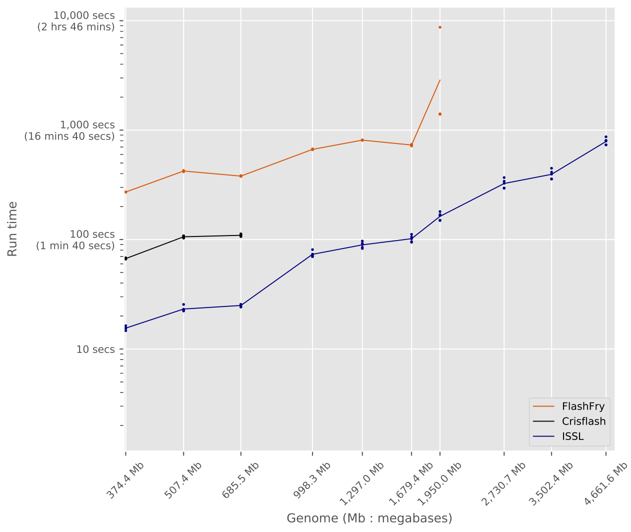

Fig. 9. Run time for generating off-targets indexes on the workstation.

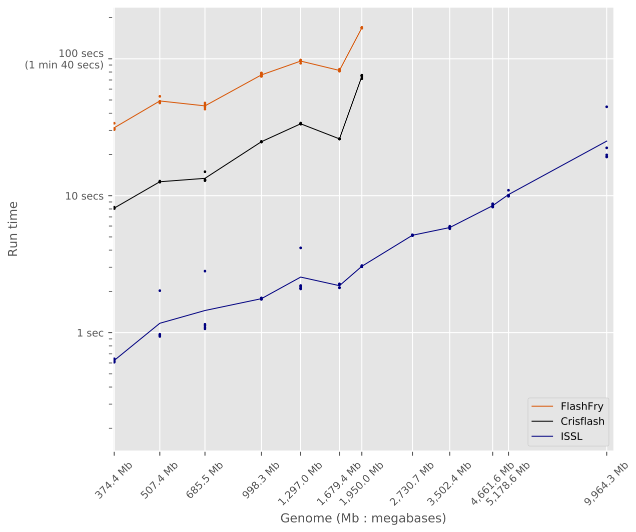

Fig. 10. Run time for generating off-targets indexes on the high-performance machine.

Table 4. The mean run time for generating the off-targets index, of 5 tests, when genome size is increased (mm:ss). - indicates the test failed due to memory limitations.

	High-performance machine			Workstation		
Genome	FlashFry	Crisflash	ISSL	FlashFry	Crisflash	ISSL
$O.sativa$	5:13	1:21	0:06	4:31	1:06	0:15
$P.hallii$	8:11	2:06	0:11	7:02	1:45	0:23
$X.couchianus$	7:32	2:13	0:14	6:20	1:49	0:24
$P.major$	12:42	4:07	0:17	11:06	-	1:13
$C.viridis$	16:01	5:35	0:25	13:30	-	1:29
$D.rerio$	13:41	4:20	0:22	12:10	-	1:41
$O.mykiss$	28:03	12:24	0:30	47:42	-	2:42
$M.musculus$	-	-	0:51	-	-	5:25
$M.domestica$	-	-	0:58	-	-	6:34
$T.urartu$	-	-	1:24	-	-	13:07
$R.bivittatum$	-	-	1:41	-	-	20:00
$T.turgidum$	-	-	4:10	-	-	-

Table 5. Run time for when increasing the number of candidate guides to evaluate (mm:ss). Tested on O. sativa.

Dataset	ISSL	fMb	fM	Cas-OFFinder	Crisflash
5k	00:00:06	00:01:12	00:05:22	00:23:11	00:06:33
10k	00:00:10	00:02:24	00:10:46	00:46:17	00:13:04
15k	00:00:15	00:03:34	00:16:02	01:08:51	00:19:35
20k	00:00:19	00:04:47	00:21:23	01:32:10	00:26:10
25k	00:00:23	00:05:58	00:26:44	01:55:11	00:32:48

In this paper, we presented Crackling, a new tool for whole-genome identification of suitable CRISPR targets. We showed that it provides the fastest way to calculate the off-target risk, by using binary encoding and the ISSL method to identify closely related throughout the genome. This was confirmed on twelve genomes of increasing length. We also showed that by combining the results of multiple scoring approaches, we can maximise the efficiency of the guides being designed. On experimental data, the set of guides it selects are better than those produced by existing tools.

Overall, this makes Crackling a faster and better method to design guide RNAs at scale.

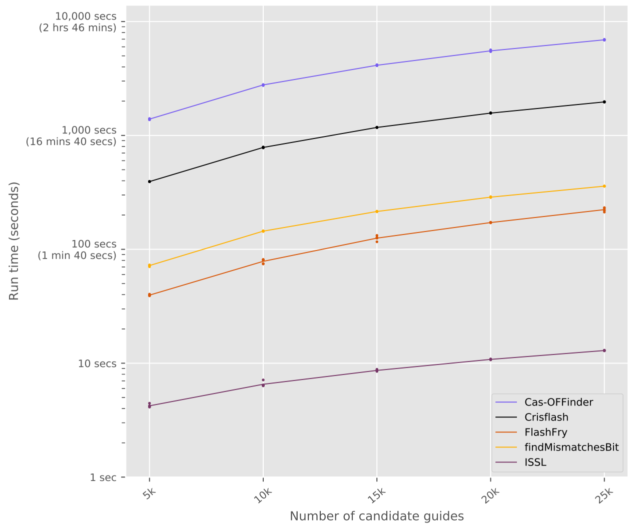

Fig. 11. Run time for an increase in the number of candidate targets

Table 6. Precision of tools on experimentally validated datasets.- indicates the tool was not tested on this dataset as it was used for training. Adapted from [3].

	Wang		**Doench**	
Tool name	Precision	Recall	Precision	Recall
Crackling	**85.7%**	**36.0%**	**29.3%**	**45.3%**
CRISPR-DO	-	-	30.4%	57.7%
CHOPCHOP	84.3%	31.5%	29.4%	42.6%
SSC	-	-	27.7%	78.7%
sgRNAScorer2	83.3%	55.1%	27.0%	65.0%
PhytoCRISP-Ex	76.4%	36.4%	23.5%	33.2%
mm10db	65.2%	29.4%	22.7%	23.5%
Cas-Designer	61.2%	17.2%	21.0%	37.7%
FlashFry	84.4%	16.3%	-	-
WU-CRISPR	81.8%	32.0%	-	-
TUSCAN	71.5%	72.1%	-	-

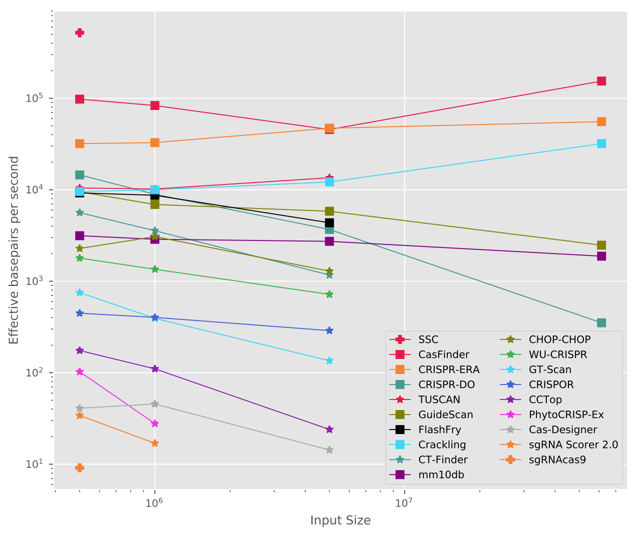

Fig. 12. Run time on mouse chromosome 19 datasets. This plot is extended from [2] to include Crackling.

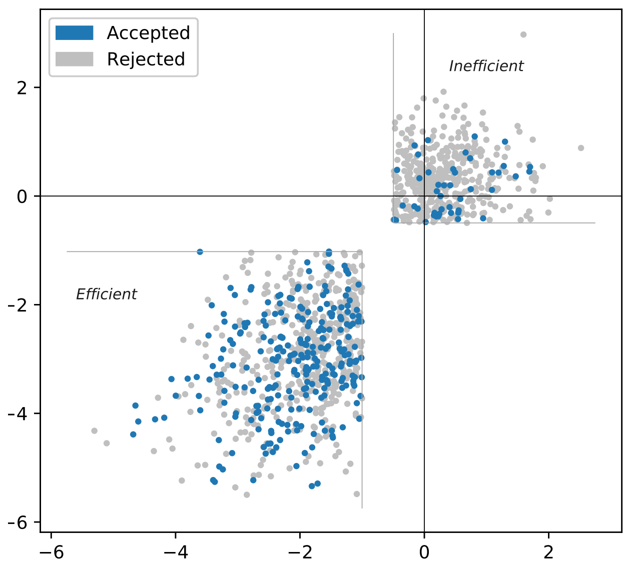

Fig. 13. Results on the Wang dataset. Each marker represents a candidate guide that was tested experimentally to be efficient or inefficient. Guides appearing in the efficient region of the plot are those that caused most knock-out effects, while those in the inefficient region had no effect. Each blue marker is a guide accepted by Crackling. It is desirable that markers appear blue only if they are in the efficient region.

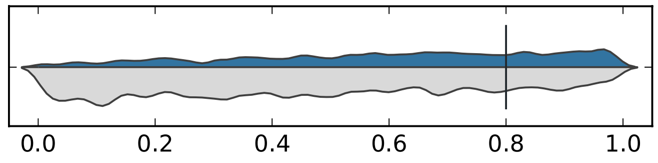

Fig. 14. Results on the Doench dataset. As per [3], "The blue distribution shows the number of guides accepted, and the grey distribution shows the number of guides rejected. The vertical marker at 0.8 shows the threshold used to determine efficiency; guides with a gene rank score greater than this were deemed experimentally efficient".

Acknowledgements

We would like to thank the Big Data lab at the Queensland University of Technology for providing access to their high-performance machine, and the authors of the other tools for making their source code available.

References

[1] Sangsu Bae, Jeongbin Park, and Jin Soo Kim. Cas-OFFinder: A fast and versatile algorithm that searches for potential off-target sites of Cas9 RNA-guided endonucleases. *Bioinformatics*, 30(10):1473–1475, May 2014.

[2] Jacob Bradford and Dimitri Perrin. A benchmark of computational CRISPR-Cas9 guide design methods. *PLoS Computational Biology*, 15(8):e1007274, 2019.

[3] Jacob Bradford and Dimitri Perrin. Improving CRISPR guide design with consensus approaches. *BMC Genomics*, 20(Suppl 9):1–11, 2019.

[4] Samuele Cancellieri, Matthew C Canver, Nicola Bombieri, Rosalba Giugno, and Luca Pinello. CRISPRitz: rapid, high-throughput and variant-aware in silico off-target site identification for CRISPR genome editing. *Bioinformatics*, 11 2019.

[5] Timothy Chappell, Shlomo Geva, and Guido Zuccon. Approximate nearest-neighbour search with inverted signature slice lists. In *Advances in Information Retrieval*, pp. 147–158, Cham, 2015. Springer International Publishing.

[6] Raj Chari, Prashant Mali, Mark Moosburner, and George M. Church. Unraveling CRISPR-Cas9 genome engineering parameters via a library-on-library approach. *Nature Methods*, 12(9):823–826, 2015.

[7] Raj Chari, Nan Cher Yeo, Alejandro Chavez, and George M. Church. SgRNA Scorer 2.0: A Species-Independent Model to Predict CRISPR/Cas9 Activity. *ACS Synthetic Biology*, 6(5):902–904, 2017.

[8] John G Doench, Nicolo Fusi, Meagan Sullender, Mudra Hegde, Emma W Vaimberg, Katherine F Donovan, Ian Smith, Zuzana Tothova, Craig Wilen, Robert Orchard, et al. Optimized sgrna design to maximize activity and minimize off-target effects of crispr-cas9. *Nature biotechnology*, 34(2):184, 2016.

[9] John G. Doench, Ella Hartenian, Daniel B. Graham, Zuzana Tothova, Mudra Hegde, Ian Smith, Meagan Sullender, Benjamin L. Ebert, Ramnik J. Xavier, and David E. Root. Rational design of highly active sgRNAs for CRISPR-Cas9-mediated gene inactivation. *Nature Biotechnology*, 32(12):1262–1267, 2014.

[10] Patrick D. Hsu, David A. Scott, Joshua A. Weinstein, F. Ann Ran, Silvana Konermann, Vineeta Agarwala, Yinqing Li, Eli J. Fine, Xuebing Wu, Ophir Shalem, Thomas J. Cradick, Luciano A. Marraffini, Gang Bao, and Feng Zhang. DNA targeting specificity of RNA-guided Cas9 nucleases. *Nature Biotechnology*, 31(9):827–832, 2013.

[11] Adrien L S Jacquin, Duncan T Odom, and Margus Lukk. Crisflash: open-source software to generate CRISPR guide RNAs against genomes annotated with individual variation. *Bioinformatics*, 35(17):3146–3147, 01 2019.

[12] Fuguo Jiang and Jennifer A Doudna. CRISPR-Cas9 structures and mechanisms. *Annual Review of Biophysics*, 46:505–529, May 2017.

[13] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A Doudna, and Emmanuelle Charpentier. A Programmable Dual-RNA-Guided DNA Endonuclease in Adaptive Bacterial Immunity. *Science*, 337(6096):816–822, 2012.

[14] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357, 2012.

[15] Ronny Lorenz, Stephan H Bernhart, Christian Höner Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

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

[17] Tessa G. Montague, José M. Cruz, James A. Gagnon, George M. Church, and Eivind Valen. CHOPCHOP: A CRISPR/Cas9 and TALEN web tool for genome editing. *Nucleic Acids Research*, 42(W1):401–407, 2014.

[18] Genshiro A. Sunagawa, Kenta Sumiyama, Maki Ukai-Tadenuma, Dimitri Perrin, Hiroshi Fujishima, Hideki Ukai, Osamu Nishimura, Shoi Shi, Rei-ichiro Ohno, Ryohei Narumi, Yoshihiro Shimizu, Daisuke Tone, Koji L. Ode, Shigehiro Kuraku, and Hiroki R. Ueda. Mammalian Reverse Genetics without Crossing Reveals Nr3a as a Short-Sleeper Gene. *Cell Reports*, 14(3):662–677, January 2016.

[19] Tim Wang, Jenny J Wei, David M Sabatini, and Eric S Lander. Genetic screens in human cells using the CRISPR-Cas9 system. *Science*, 343(6166):80–84, 2014.

[20] An Xiao, Zhenchao Cheng, Lei Kong, Zuoyan Zhu, Shuo Lin, Ge Gao, and Bo Zhang. CasOT: A genome-wide Cas9/gRNA off-target searching tool. *Bioinformatics*, 30(8):1180–1182, 2014.

[21] Lihua J. Zhu, Benjamin R. Holmes, Neil Aronin, and Michael H. Brodsky. CRISPRseek: A Bioconductor package to identify target-specific guide RNAs for CRISPR-Cas9 genome-editing systems. *PLoS ONE*, 9(9), 2014.