The manual of CIDP

目录

Catalogue

1、介绍

CIDP 是用于设计 CRISPR 系统中 single guide RNA (sgRNA)的程序。选择背景物种以计算脱靶率是 sgRNA 设计软件的一个共同特点。以往的设计软件往往会尽可能收集不同的物种以构成数据库，从而为用户在设计 sgRNA 时可以提供亲缘关系尽可能近的物种以便更为准确地计算脱靶率。然而，随着生命科学研究的不断进步，越来越多的物种被纳入到了研究体系中而以往设计软件中所包含的数据库却无法及时更新并且随着研究物种的增多，将所有物种纳入程序数据库也是一个很难实现的设想。这就使人们不得不以亲缘关系较远的物种为背景来设计 sgRNA，而这往往会导致误差以及一些难以预料的问题。因此，我们开发了一款名为 CIDP 的软件。与以往同类型软件相比，该软件所需要的背景数据库是由用户提供的基因组序列文件构建而来并且 CIDP 对输入文件没有任何特殊的格式要求，这使 CIDP 能够更为广泛地适应不同物种的研究需求。除查找 sgRNA 外，CIDP 同时还具有序列提取、引物及特殊结构设计、数据筛选，可视化等扩展功能，使得 CIDP 成为了设计 sgRNA 的综合性平台。

2、安装系统

CIDP 能够在 Windows，MacOS 以及 Linux 系统中安装使用。

3、安装

对 Windows 系统：

安装程序有 32 位和 64 位的区别，其中用 x32 是用于 32 位系统的而 x64 则是可用于 64 系统安装的程序。在安装时需要用户根据自身电脑的配置进行灵活选择。双击 exe，即可进行程序安装，安装时要避免路径含有空格。

对 MacOS 系统：

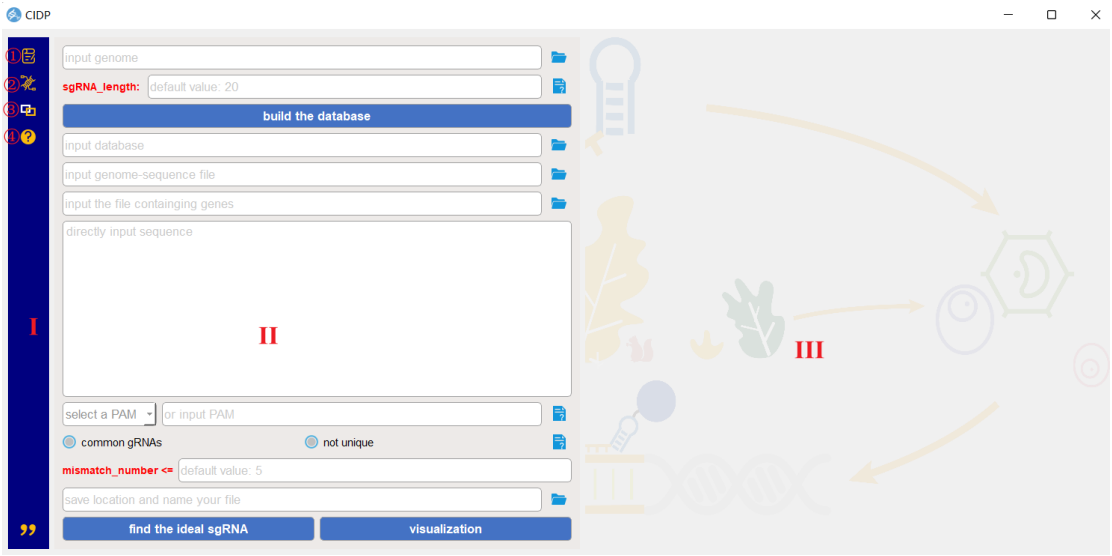
在制作软件及进行软件测试时，我们所使用的系统是 64 位系统，M1 芯片。安装时双击 pkg 文件即可，同样要注意避免安装路径中含有空格。

对 Linux 系统

这里的 Linux 系统指的是 Linux 系统服务器，需要用户安装有 python 3.8 及以上版本同时需要安装的包是 primer3-py 及 psutil 且需要将 python 3 放入环境变量中。

4、关于 CIDP 软件使用的具体信息

4.1 软件界面（Windows and MacOS 版本）



I, 模块选择区域，包括数据库构建以 sgRNA 查询模块（①），功能扩展模块（②），其他功能模块（③）以及帮助模块（④）；

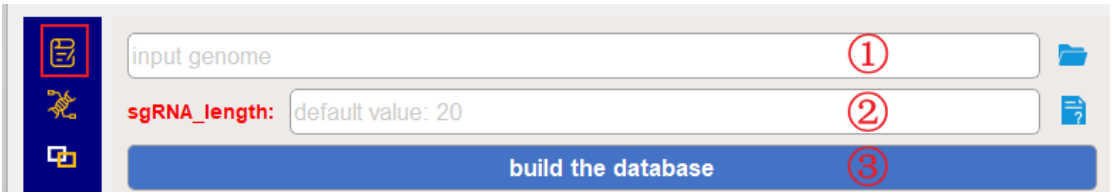
II, 功能设置区域，包含各功能按键分布及参数设置；

III, 结果展示区域。

4.2 功能分布及介绍

4.2.1 建库及 sgRNA 查询

建库



在①放入基因组序列文件，fasta 格式，非压缩；在②设置希望的 sgRNA 大小，一般为 20 bp

且 20 为默认值，如果 20 能够满足需求则不需要设置；点击③执行建库。建库所需时间受基因组大小及个人电脑配置影响，该步骤的核心目的是以设定的 sgRNA 大小筛选全基因组范围内具有唯一性的 sgRNA。因此，该库建立后，如果是同一份基因组序列，则不需要反复建库。但同时，由于是针对整个基因组进行操作，从 Windows 和 MacOS 电脑的一般配置来看，所需时间一般较长。一般规律是 64 位版本比 32 位版本快，200 Mb 基因组文件在内存 8 Gb 电脑上操作大约需要 2 小时。如果想要快速建库和操作可以考虑使用 Linux 版本。

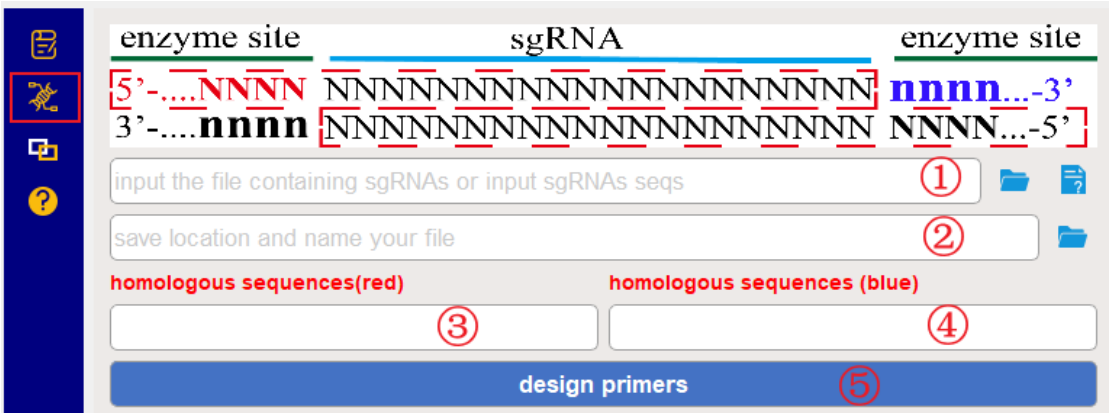
特异性 sgRNA 的查询

在①放入构建的 sgRNA 数据库；在②放入基因组序列文件；对基因序列而言，如果只想查找一个基因上的 sgRNA，则可以直接将序列放入④，③接收的是一个含有多条基因序列的 fasta 文件，如果对③进行了设置则可以进行 sgRNA 的批量设计；⑤主要负责提供 PAM 选项，可选择的 PAM 主要有 20 种，如果这些 PAM 没有办法满足需求，可以在后面输入框中直接输入 PAM 的序列来进行查询，其中默认的 PAM 是 NGG，需要注意，如果是直接输入 PAM，请不要输入 PAM 模型，比如 AGG 是 PAM 序列而 NGG 则是 PAM 模型，直接输入这里仅支持输入序列；之后，在⑧设置允许的最大错配数，默认值为 5；在⑨设置保存位置并对结果文件命名；点击按钮 10 执行功能；按钮 11 则可对结果进行可视化展示。

查询的基本原理解释：CIDP 首先根据 PAM，在用户输入的序列上查找可能的 sgRNA，之后检验这些 sgRNA 是否在 sgRNA 库中并因此过滤 sgRNA；根据允许的最大错配数，将得到的 sgRNA 进行全基因组范围的搜索，以明确其可能发生的脱靶位点并计算脱靶率。因此，最终设计的、具有极低脱靶率的 sgRNA 其对目的基因而言应该具有极高序列特异性从而也最大限度的降低了脱靶发生的可能。

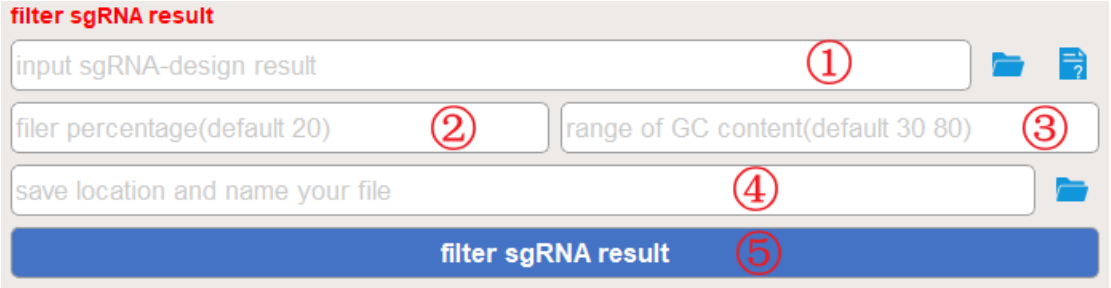
两种特殊情况：1、最近的研究表明如果所使用的 sgRNA 是一组基因共有的 sgRNA 则可以利用 CRISPR 技术快速产生多突变体，基于此，我们设计了 common sgRNAs 功能⑥，该功能会根据用户输入的一组序列查找，该组序列共有的 sgRNA。2、如果有些用户不需要特异性的 sgRNA 而只是想要统计所输入序列中 sgRNA 的数量及分布情况，可以选择⑦。

4.2.2 引物设计



在获得 sgRNA 后，需要将其插入 CRISPR 载体中，一般选择的插入方法是选择载体上合理的酶切位点，将酶切位点两侧的序列添加到 sgRNA，直接引物合成，单链退火后，将其与切割后的载体进行融合。基于该过程，我们在 CIDP 中安排了引物设计功能。首先将得到的 sgRNA 放入①，设置保存位置并对结果文件命名②，根据载体图的指示，将酶切位点左侧的序列放入③，右侧序列放入④，之后点击⑤即可执行引物设计功能。

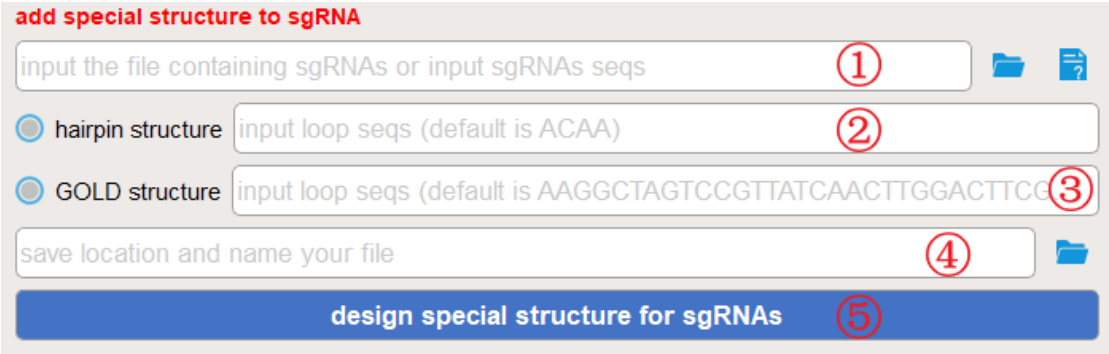
4.2.3 对 sgRNA 设计的结果进行过滤



有些基因可能会得到大量的 sgRNA，这样就涉及到选择 sgRNA 的问题。过滤 sgRNA 功能

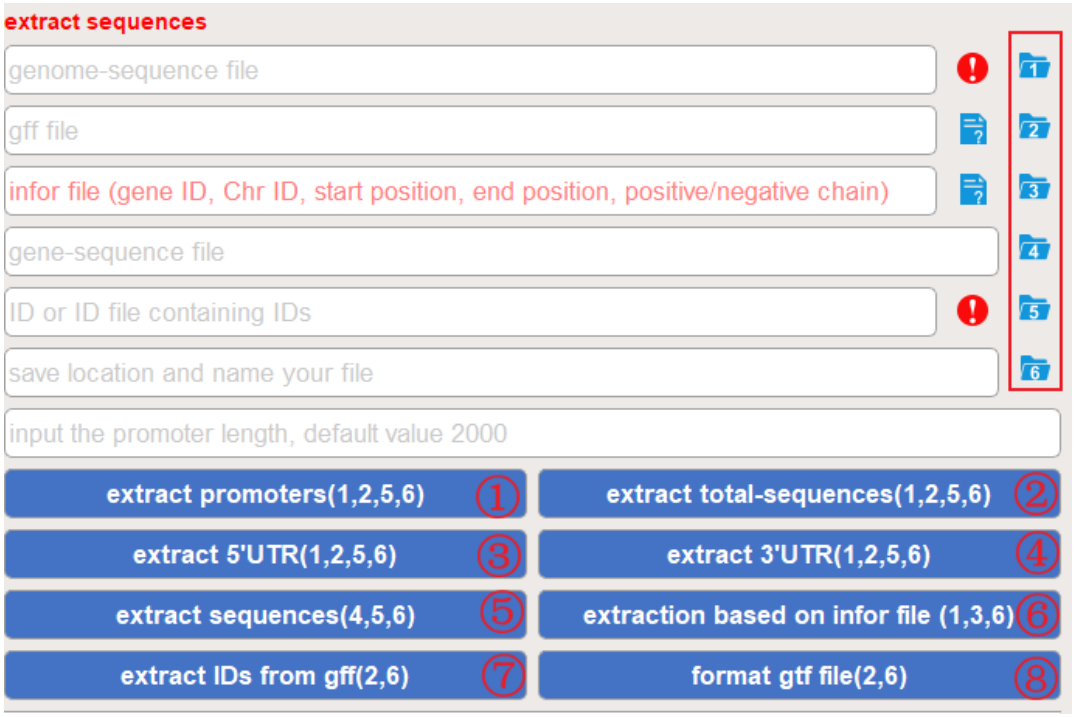
主要从脱靶率（②）和 GC 含量（③）两方面对结果过滤。将 sgRNA 放入①；在②设置脱靶率阈值（整数即可，默认值 20）及 GC 含量阈值（GC 含量是一个范围，设置的格式已经用灰色字体注明，简言之，30 80 中间用空格连接，即代表保留 GC 含量在 30%到 80%的 sgRNA）；设置保存位置并对结果文件命名④；点击⑤执行功能。

4.2.4 为 sgRNA 添加特殊结构



最近的一些研究表明通过向 sgRNA 添加特殊结构可以减弱脱靶效应。可通过序列添加的方式加以利用的，主要有添加发卡结构（doi: 10.1038/s41587-019-0095-1）和 GOLD 结构（doi: 10.1038/s41467-022-28137-7）两种。为此，我们设置了自动添加这些结构的功能。将 sgRNA 放入①；如果要添加发卡结构，则点击②处按钮，默认的添加序列是 ACAA，如果需要其他序列形式则直接把序列加到后面的输入框即可；同理，GOLD 结构则需要选择③；在④设置保存位置并命名结果文件，点击⑤执行功能。

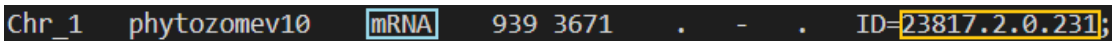
4.2.5 序列、信息提取以及格式转换



CIDP 是专门为实验人员设置 sgRNA 设计工具，从这个角度讲，我们的基本假设是用户不熟悉编程及命令行操作。尽管我们知道有不少工具可用于从基因组中顺利地提取序列，但这仍然涉及到了不同软件的切换。因此，我们在 CIDP 中设计了用于自动提取各类序列的功能。如上图所示，能够提取的类型包括从基因组序列中提取启动子①，基因全长（指一个基因整个的 mRNA 序列）②，UTR 区（Untranslated Regions）③④，从 fasta 文件中提取序列⑤，根据 bed 文件提取序列⑥，从 gff 提取 ID⑦以及将 gtf 文件转为 gff 文件等功能⑧。为了方便用户操作，我们在 CIDP 界面保留了足够的信息用户仅需要按照界面提示放入相应序列即可进行操作，因此，这里不再进行各个功能的详细描述了，仅举一例以及叙述相应的注意事项。例如用户要提取启动子，涉及到的功能按钮是①，其提示的数字是 1，2，5，6，这些数字对应的是右侧红色方框显示的地方。1，2，5，6 即代表需要在 1，2，5，6 这些区域放入相应文件，放入后点击按钮执行功能即可。

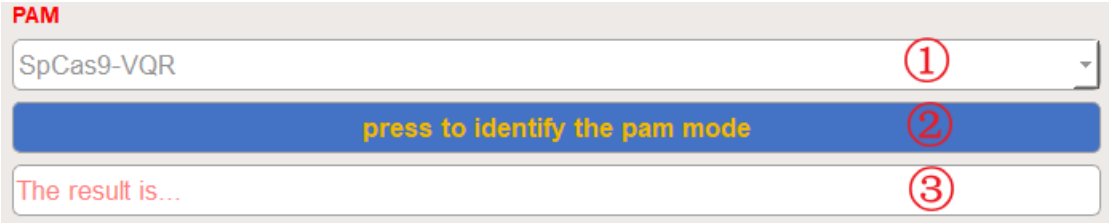
注意事项：1、除了基因组序列文件外，我们还需要文件以提供基因在基因组上的位置信息。默认的位置信息文件是通用的 gff 文件，这里仍然需要用户去确认两件事：I，手里的 gff 文件是否是标准格式的 gff 文件。gff 文件的标准格式可以参考示例文件中的 gff 文件；II，如果手里确实没有 gff 文件而有 gtf 文件可以通过按钮⑧进行转换（注意，这里的转换结果，仅仅只适用于 CIDP）。如果用户既没有 gff 也没有 gtf 格式的文件，CIDP 支持用户自己构建

位置信息文件来获取序列，位置信息文件的基本格式是基因 ID，染色体 ID，起始位置，终止位置以及这段序列是在基因组的正链还是负链上，这些元素中间的分隔符由空格键进行分隔。2、提取功能需要用户输入基因 ID（下图黄色框），这里的 ID 指的是 gff 文件中 mRNA 关键词（下图青色框）所对应的 ID，如下图所示：



3、按键⑤所指功能是从基因 fasta 文件中提取序列而不是从基因组序列文件中提取。

4.2.6 关于如何选择 PAM 模型

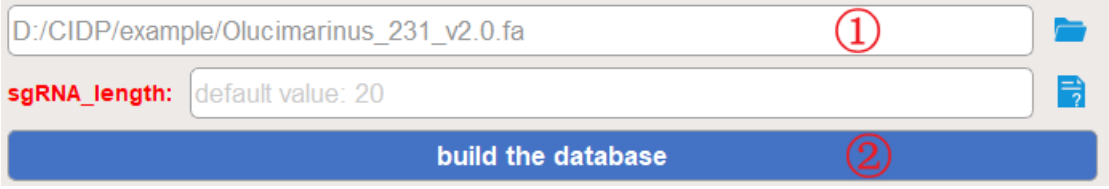


部分用户可能会遇到不知如何选择 PAM 的问题。原因是用户更容易知道的是自己所用载体以及载体上对应的 cas 酶。为了帮助用户进行选择，我们设置了 PAM 选择功能。根据载体中 cas 酶在①中进行选择；然后点击②即可在③中显示较为理想的 PAM。

5、快速开始

以上是我对 CIDP 工作原理及参数的具体介绍。事实上，我们根据一般的设置需求已经给用户设置了大量的默认值，这些默认值可以帮助用户在几乎不需要任何设置的情况下就可以正常运行 CIDP。下面，我将使用默认值来介绍如何快速使用 CIDP。

5.1 建库

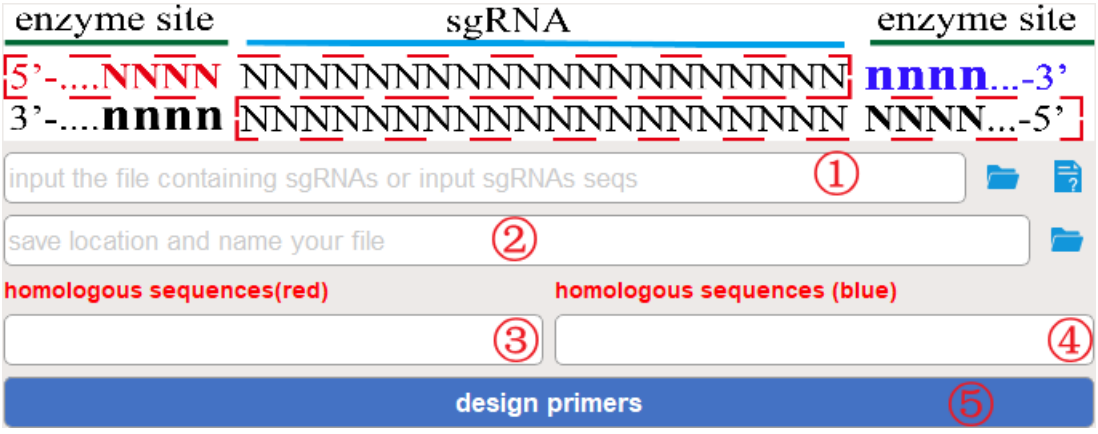


可直接拖拽基因组序列文件到①，点②建库

5.2 查询 sgRNA

将库文件放入①，②放入基因组序列文件，③放入基因 fasta 文件或者在④直接放入序列，⑤设置保存位置，⑥查询 sgRNA，如有需要，点⑦展示 sgRNA 位置

5.3 设计引物

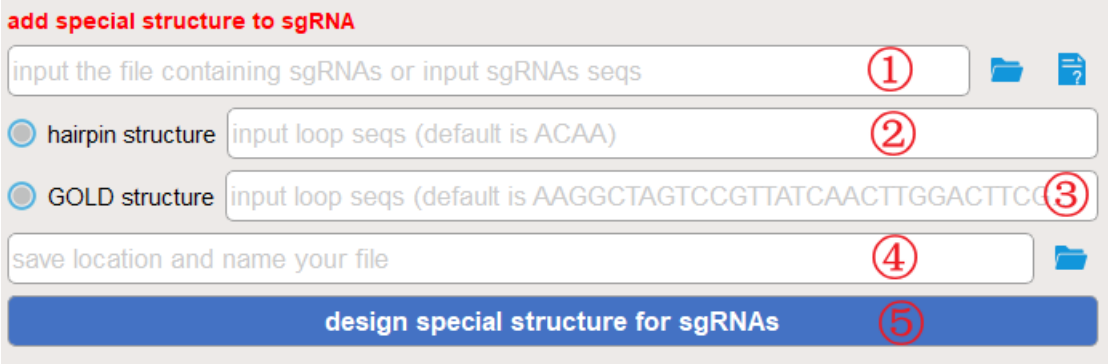


将 sgRNA 放入①，②设置保存位置，③切割位点左侧序列，④切割位点右侧序列，点⑤执行功能。

5.4 过滤 sgRNA

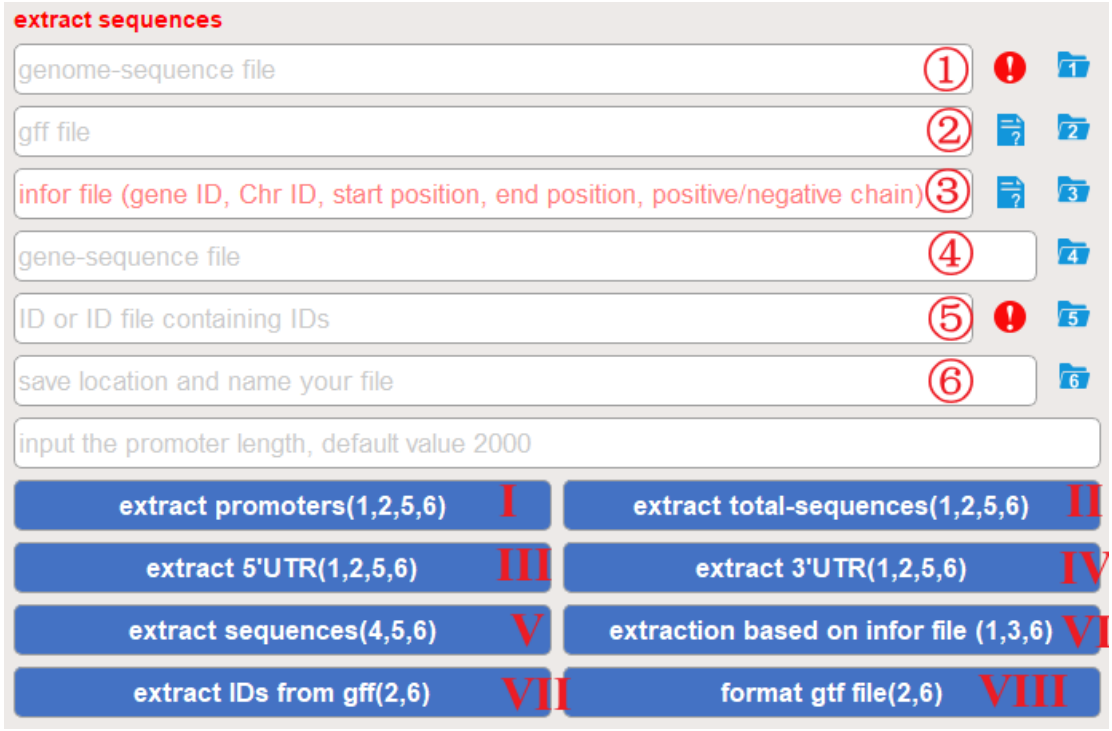
将 sgRNA 放入①，②设置保存位置并命名，③执行功能

5.5 为 sgRNA 添加特殊结构



将 sgRNA 放入①，根据需要选择②或者③，④设置保存位置并命名，⑤执行功能

5.6 序列提取



提取启动子： ①→②→⑤→⑥ 点击 I 执行;

提取全长： ①→②→⑤→⑥ 点击 II 执行;

提取 5'UTR: ①→②→⑤→⑥ 点击 III 执行;

提取 3'UTR: ①→②→⑤→⑥ 点击 IV 执行;

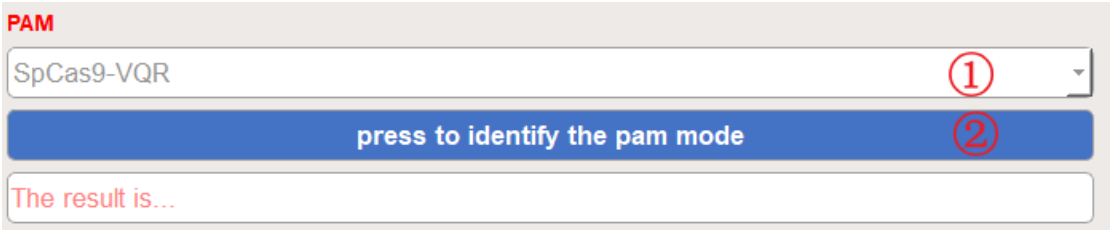
提取序列: ④→⑤→⑥ 点击 V 执行;

根据 bed 文件提取序列: ④→⑤→⑥ 点击 VI 执行;

从 gff 文件中提取 ID: ②→⑥ 点击 VII 执行;

将 gtf 转为 gff 文件: ②→⑥ 点击 VIII 执行.

5.7 选择 PAM



在①选择 cas 类型，点②执行。

6、关于 Linux 系统版本的操作

该版本主要为解决 Windows 和 MacOS 版本由于内存不足而导致设计效率低的问题，所以这里指的是 Linux 是指 Linux 服务器。

6.1 建库

命令行

python CIDP-linux.py -md -gf 基因组序列文件 -o 库名称

例如：

python CIDP-linux.py -md -gf ./Osativa_323_v7.0.fa -o pt_kmer.txt

所需时间：水稻基因组 365Mb，建库时间<30 min。

6.2 确定 sgRNA

命令行

python CIDP-linux.py -ss -ff 基因序列文件 -misnumber 允许的最大错配数 -gf 基因组序列文件 -df 第一步生成的库文件 -o 保存结果

例如：

python CIDP-linux.py -ss -ff try.fa -misnumber 4 -gf ./Olucimarinus_231_v2.0.fa -df ol.txt -o ./designs.txt

6.3 提取启动子

命令行

python CIDP-linux.py -exp -gff_file gff 文件 -gf 基因组文件 -ID_file 要提取的基因 ID (注意这个 ID 应与 gff 文件中以 mRNA 为关键词的行中标注的 ID 保持一致) -pl (想要提取的启动子长度，如果不设置，则该值默认为 2000) -o 保存位置

例如：

python CIDP-linux.py -exp -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -pl 1000 -o promoters2.txt

6.4 提取基因总长

命令行

python CIDP-linux.py -ext -gff_file gff 文件 -gf 基因组 -ID_file (与提取启动子注意事项相同) -o (保存并命名)

例如：

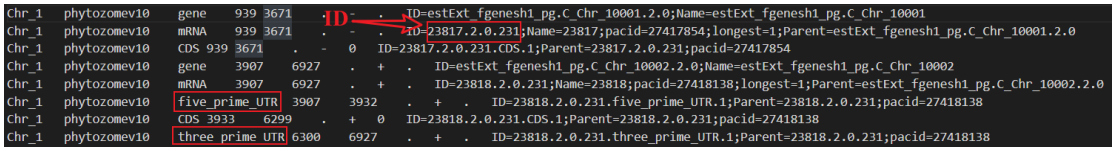
python CIDP-linux.py -ext -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -o total_length.txt

6.5 提取 5'UTR 区

命令行

python CIDP-linux.py -ex5 -gff_file gff 文件 -gf 基因组 -ID_file (基因 ID 文件) -o (保存并命名)

注意：UTR 的提取功能主要依据 gff 文件中所标注的 UTR 区域进行提取。因此，如果某些基因没有相应标注则无法提取。例如，示例文件中这两个基因的区别：



例如：

python CIDP-linux.py -ex5 -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -o 5utr.txt

6.6 提取 3'UTR 区

命令行

python CIDP-linux.py -ex5 -gff_file gff 文件 -gf 基因组 -ID_file (基因 ID 文件) -o (保存并命名)

例如：

python CIDP-linux.py -ex3 -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -o 3utr.txt

6.7 根据基因 ID，从 fasta 文件中提取相应序列

命令行

python CIDP-linux.py -excds -ff fasta 文件 -ID_file (基因 ID 文件) -o (保存并命名)

例如：

python CIDP-linux.py -excds -ff fasta_example.txt -ID_file fasta_ID.txt -o extract_seqs.txt

6.8 根据 bed 文件从基因组中提取序列

命令行

python CIDP-linux.py - exbed -bed_file (bed 文件) - gf 基因组序列文件 -o (保存并命名)

例如：

python CIDP-linux.py -exbed -bed_file bed.txt -gf Olucimarinus_231_v2.0.fa -o bed_seqs.txt

6.9 根据 gff 文件中提取 ID

命令行

python CIDP-linux.py -exids - gff_file (gff 文件) -o (保存并命名)

例如：

python CIDP-linux.py -exids -gff_file Olucimarinus_231_v2.0.gene.gff3 -o ids.txt

6.10 将 gtf 文件转 gff 文件

命令行

python CIDP-linux.py - formatgtf -gtf_file (gtf 文件) -o (保存并命名)

例如：

python CIDP-linux.py -formatgtf -gtf_file GCF_000214015.3_version_140606_genomic.gtf -o ids.gff

7、使用的注意事项

1、不同的系统，程序可能有差别；

2、目前脱靶率的计算只能针对 NGG 进行；

3、利用 ID 从基因组提取序列，一定注意 ID 必须符合要求；

4、UTR 区序列的提取是以 gff 文件中含有“UTR”标识符进行的（如下）。因此，如果某基因在 gff 文件中缺乏“UTR”标志则无法提取。

```
Chr_1   phytozomev10    five_prime_UTR  9830    9839    .   +   .   ID=23820.2.0.231
```

1. Introduction

CIDP is a program used to design the single guide RNA (sgRNA) in the CRISPR system. Selecting background species to calculate the off-target rate is a common feature of sgRNA design software. In the past, design software often collected as many different species as possible to form a database, so as to provide users with the most closely related species to calculate the off-target rate more accurately when designing sgRNA. However, with the continuous progress of life science research, more and more species have been included in the research system, but the database contained in the previous design software cannot be updated in time. Also, as the number of species under study increases, including all species in the program database is a difficult idea to realize. This would force the design of sgRNA in the context of distantly related species, which often leads to errors and unpredictable problems. So, here we developed a program called CIDP. Compared with the previous similar software, the background database required by this software is constructed from the genome sequences provided by the user, and CIDP does not have any special format requirements for the input file, which enables CIDP to more widely adapt to the research needs for different species. In addition to sgRNA search, CIDP also has extended functions such as sequence extraction, primer and special structure design, data screening and visualization, making CIDP a comprehensive platform for sgRNA design.

2. Requirements of operation system

CIDP can be installed and used in Windows, MacOS and Linux systems.

3. Installation

For windows:

There are two installer programs (32-bit and 64-bit installers), with x32 for 32-bit systems and x64 for 64-bit systems. Users can choose a suitable version according to the configuration of their computers during the installation, Double-click the exe file to install the program. Please make sure

that the installation path contains no spaces.

For MacOS:

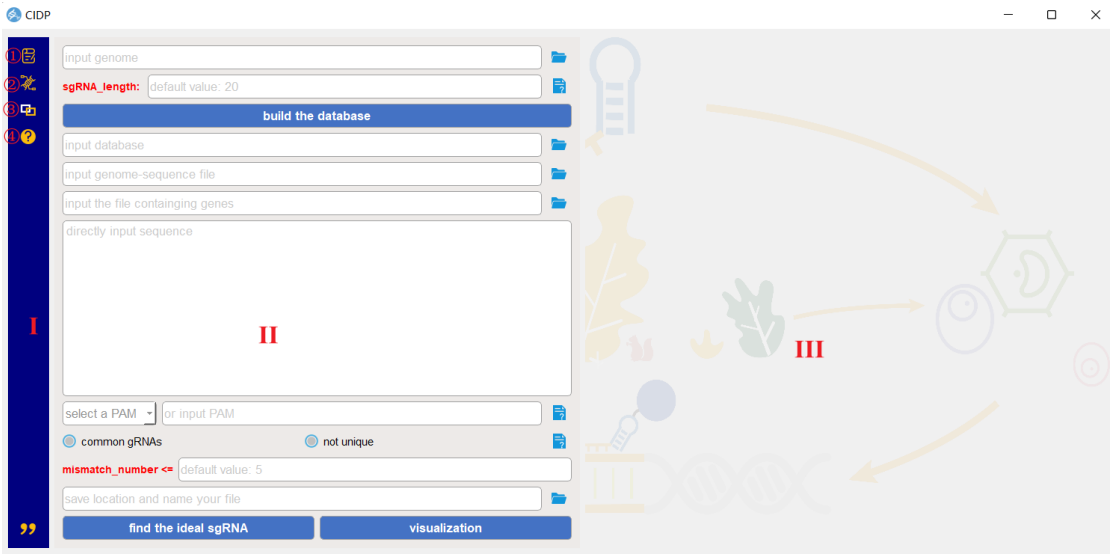
When developing and testing this software, the system we use is a 64-bit system, M1 chip. Double-click the pkg file to install the program and avoid spaces in the installation path.

For Linux:

The Linux system here refers to a server version of Linux system, which requires a preliminary installation of python 3.8 or later version, with packages of primer3-py and psutil.

4. Usage of CIDP

4.1 Software interface (Windows and MacOS versions)



I. Module selection area, including database construction and sgRNA query (①), function expansion (②), other functions (③) and help information (④).

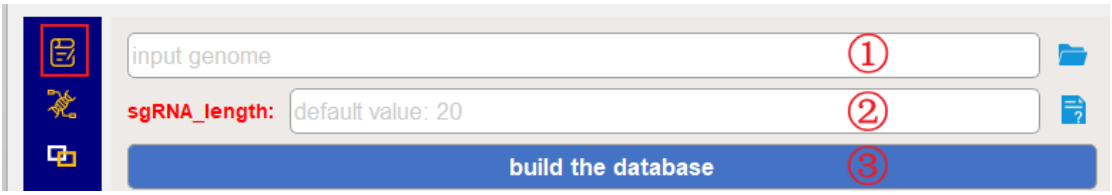
II. Function setting area, including the layout of function keys and the parameter settings.

III. Display area for results.

4.2 Introduction of functions

4.2.1 Database construction and sgRNA search

Database construction



① input the genome sequence file in fasta format, uncompressed;② set the desired size of sgRNA, which is generally 20 bp (default value). Users can leave this blank if 20 can meet their requirements here. ③ click to build the database. The time consuming for building a database is affected by genome size and the computer configuration. The key purpose of this step is to screen unique sgRNAs across the whole genome with the set sgRNA size. Therefore, after the establishment of the database, there is no need to build database repeatedly for a same genome sequence. Because the operation is on the whole genome, it generally takes a long time in a common Windows or MacOS computer, e.g., ~2h in a computer with 8G RAM for 200Mb of genome sequence. The 64-bit version program is usually faster than the 32-bit version. You can consider to use the Linux version of this program if you want to build and operate libraries more efficiently.

Search for specific sgRNA

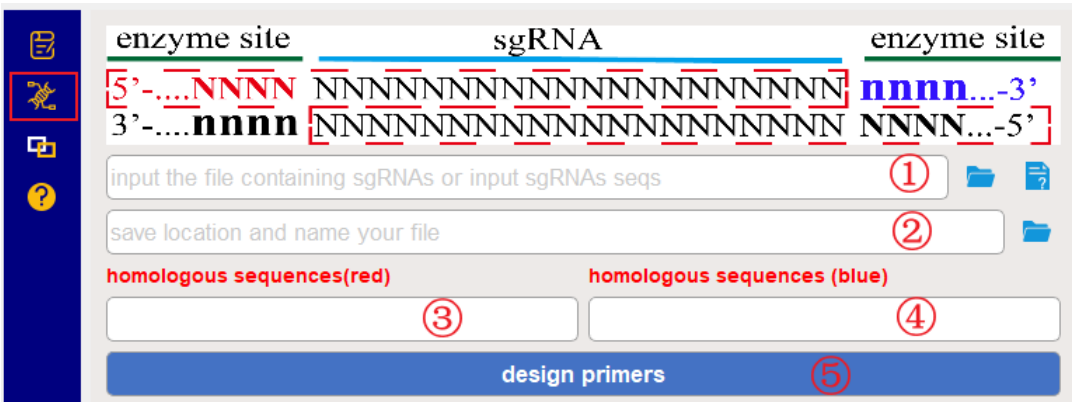
① input the constructed sgRNA database. ② input the genome sequence file. ③ input a fasta

file containing multiple gene sequences. If this option is set, the program will carry out a batch design of sgRNA; ④ This option accepted only one gene sequence. If you want to find the sgRNA from only one gene, you can directly put the DNA sequence here. ⑤ PAM options can be found here. There are 20 kinds of PAM models to choose, in which the default PAM is NGG. If these PAM cannot meet the requirements, you can directly input the sequence of PAM in the text box for query. Notably, DO NOT enter a PAM model (e.g., NGG) in the text box, instead you should only use a PAM sequence (e.g., AGG) in the text box. ⑧ input the maximum number of mismatches allowed (default: 5). ⑨ set a output file for saving the results. Click the button 10 to execute the function. Click the button 11 to visualize the results.

The basic principles of query are as follows: CIDP first searches for possible sgRNA from the input sequences based on the PAM, then checks whether the obtained sgRNA are in the sgRNA library and filters them accordingly. The resulted sgRNA are further searched though the whole genome with the option of allowed maximum number of mismatches to identify potential off-target loci and calculate the off-target rate. Therefore, the final designed sgRNA with a low off-target rate should have extremely high sequence specificity for the target gene which minimizing the possibility of off-target.

Two special cases: (1) Recent studies have shown that if the sgRNA used is shared by a group of genes, multiple mutants can be rapidly generated by using CRISPR technology. Based on this, we designed a common sgRNAs function (⑥), which can search for shared sgRNA from a group of sequences input by the user. (2) If some users do not want to identify specific sgRNA but just count the number and distribution of sgRNA in the input sequence, they can choose ⑦.

4.2.2 Primer design

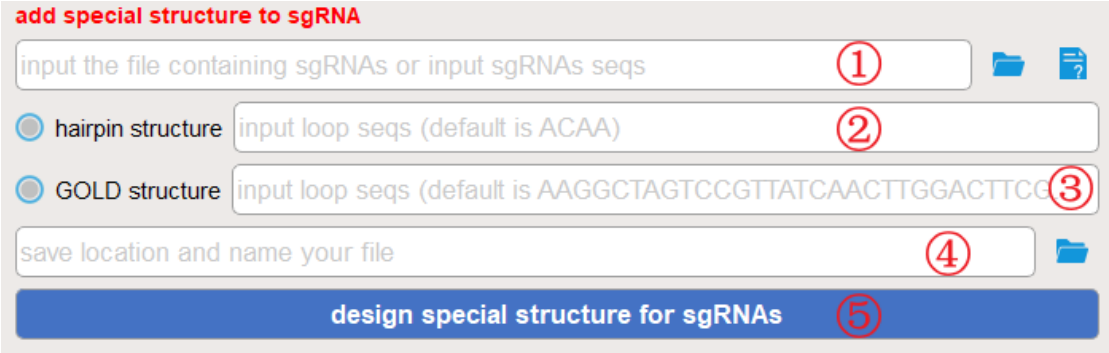


After obtaining the sgRNA, it needs to be inserted into the CRISPR vector. The general approach is to select the target enzyme restriction site on the vector, add the sequences on both sides of the enzyme restriction site to the sgRNA to construct the primer sequence; The pair of primer can be synthesized and annealed and then can be used to fuse with the cut vector. Based on this process, we added primer design functions in CIDP. ① input the obtained sgRNA. ② set the storage location and the result file. ③ According to the vector diagram, input the sequence on the left of the enzyme restriction site; and ④ the sequence on the right. ⑤ click to design the primers.

4.2.3 Filter the results of designed sgRNA

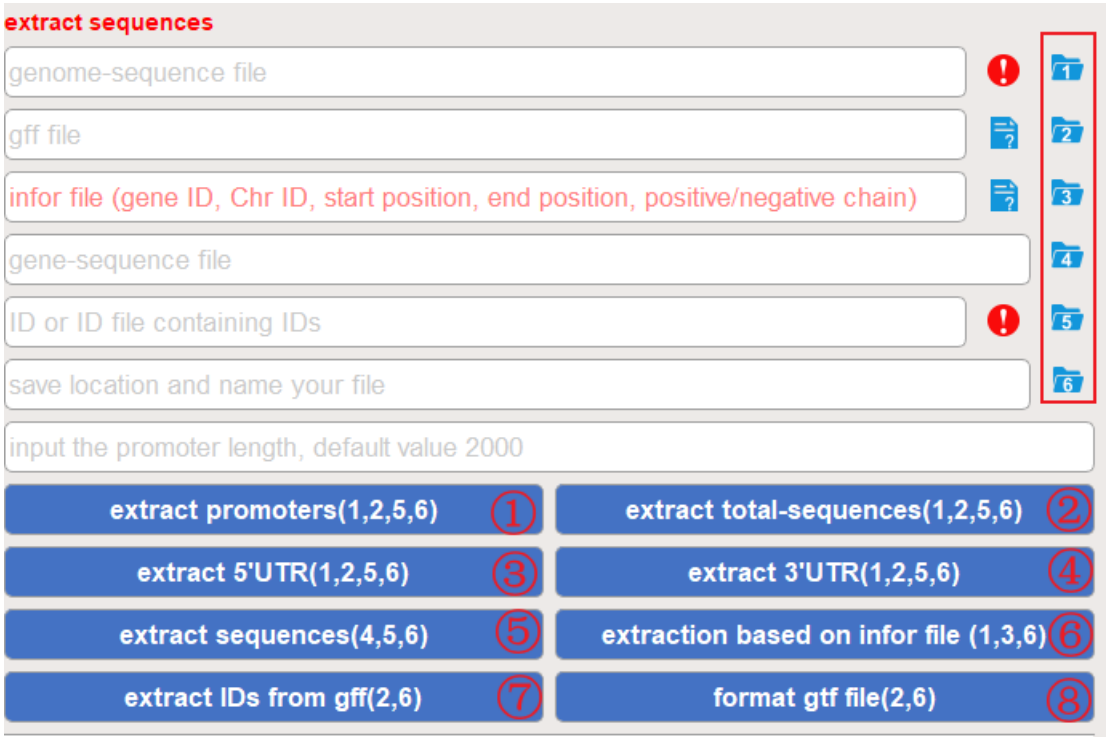
Some genes may yield a lot of sgRNA, so there is a need to filter the resulted sgRNA. Two params, the off-target rate (②) and GC content (③), were considered in filtering the sgRNA. ① input the sgRNA. ② set the off-target rate threshold (integer, default value 20). ③ set the GC content range (input minimum and maximum values, separated by a space, e.g., 30 80, means the sgRNA with GC content between 30% and 80% will be retained). ④ set the save location and the result file. ⑤ Click to execute the function.

4.2.4 Add special structures for sgRNA



Some recent studies have shown that off-target effects can be attenuated by adding special structures to sgRNA, e.g., adding a pin structure (doi: 10.1038/s41587-019-0095-1) or a GOLD structure (doi: 10.1038/s41467-022-28137-7). To do this, we provided a module to automatically add these structures. ① input the sgRNA. ; ② select this option if you want to add the hairpin structure. The default sequence is ACAA. If you need other sequence forms, just add the sequence to the following text box. ③ select this option if need the GOLD structure.④ set the save location and the result file. ⑤ Click to execute the function.

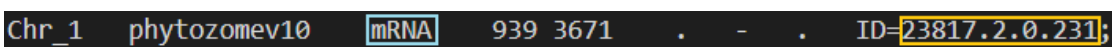
4.2.5 Sequence, information extraction, and format conversion



CIDP is a sgRNA design tool for experimentalists. Therefore, our basic assumption is that the user

is not familiar with programming and command line operations. Although we know that a number of tools are available to extract sequences smoothly from genomes, it still involves switching between different software. Thus, we designed a function in CIDP for automatic extraction of various sequences. As shown in the figure above, the types that can be extracted include the promoter (①) extracted from the genome sequence, the full length of the gene (the whole mRNA sequence of a gene, ②), the UTR Regions (Untranslated Region, ③④), extracting sequences from a fasta file (⑤), extracting sequences from a bed file (⑥), extracting ID from a gff file (⑦) and converting gtf files into gff files (⑧). For user convenience, we presented enough information in the CIDP interface. Users only need to follow the interface prompts to input the corresponding sequence to operate. Therefore, a detailed description of each function will not be provided here. An example can be illustrated here. For example, if a user wants to extract the initiator, the function button involved is ①, and the numbers presented are 1,2,5,6. These numbers denote the places displayed in the red box on the right. 1,2,5,6 means that the corresponding files need to be placed in the areas of 1, 2, 5, and 6, and then click the target button to perform the function.

Notes: 1. In addition to the genome sequence file, we also need the gene location information file to provide the location information of genes on the genome. The default gene location information file is a generic gff file. The user needs to confirm two things here: I, whether the provided gff file is in the standard gff format, which can be found as shown by the gff file in the sample files; II, if you have a gtf file, you can convert it to gff format with the button ⑧ (note that the converted gff files may only work with CIDP). If the user has neither gff nor gtf format files, CIDP allows the users to construct their own location information files. The basic format of location information file includes 5 columns: gene ID, chromosome ID, starting position, ending position and gene direction on the reference genome, separated by spaces. 2. The extraction function requires the user to input the gene ID (as shown in the yellow box below). The ID here refers to the ID corresponding to the mRNA keywords in the gff file (as shown in the cyan box below), as shown in the figure below.



3. Button ⑤ refers to the function of extracting sequences from the gene fasta file rather than from the genome sequence file.

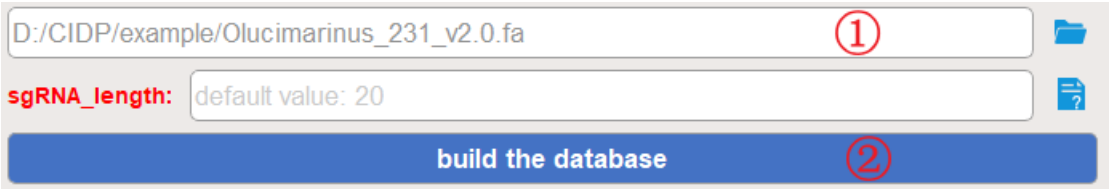
4.2.6 How to select the PAM model

Some users may don't know how to select a suitable PAM model. And they usually can remember the vector they are using and the corresponding Cas enzyme on the vector. To help users choose, we design a PAM selection function. Select the Cas enzyme in the vector (①), and then click ② to display the ideal PAM (③).

5. Quick start

The above is a detailed introduction to the working principle and parameters of CIDP. In fact, we have set a number of default values for users based on common requirements, which can help the user to run CIDP without changing almost any params in most cases except for the source genome sequences. Below, I'll show a quick start with CIDP by using the default values.

5.1 Library construction

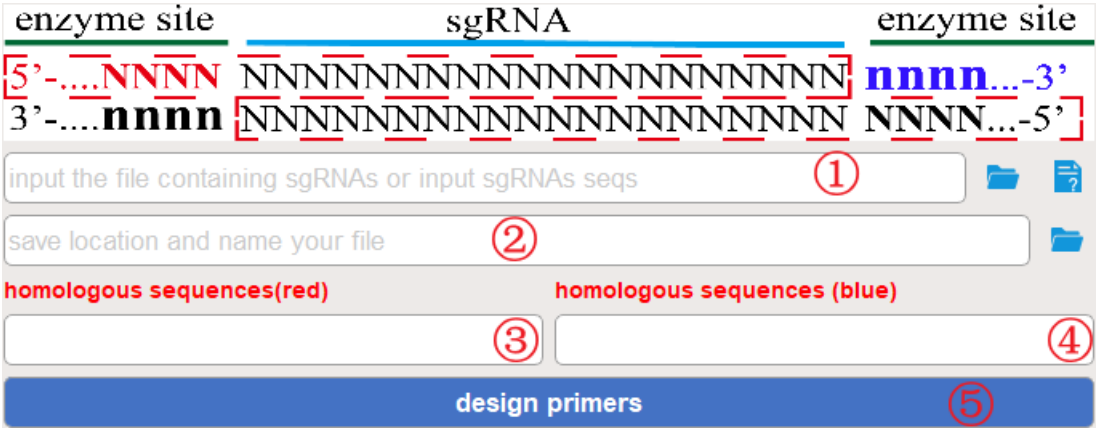


Drag the genome sequence file to ① and click ② to create the library.

5.2 sgRNA Search

Put the library file into ①, the genome sequence file into ②, the gene sequence file (fasta format) into ③ or just the DNA sequence into ④, and set the save location (⑤). Click ⑥ to query the sgRNA. Click ⑦ to show the sgRNA location if necessary,

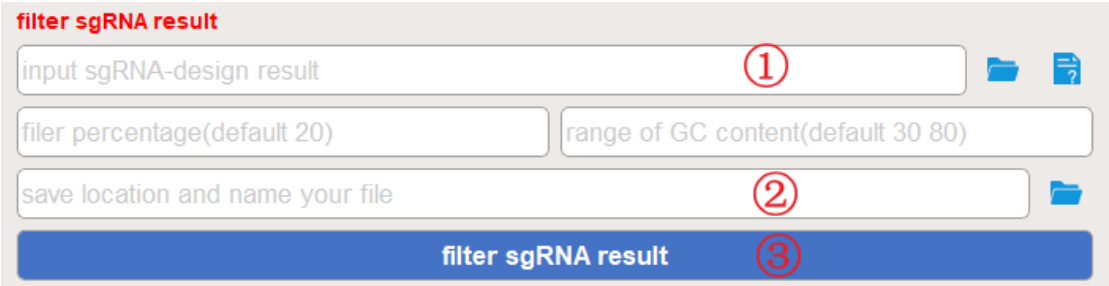
5.3 Primer design



Put sgRNA into ①, set the save location (②), input the sequence on the left side of the cleavage site (③) and the sequence on the right side of the cleavage site (④), and click ⑤ for executive the

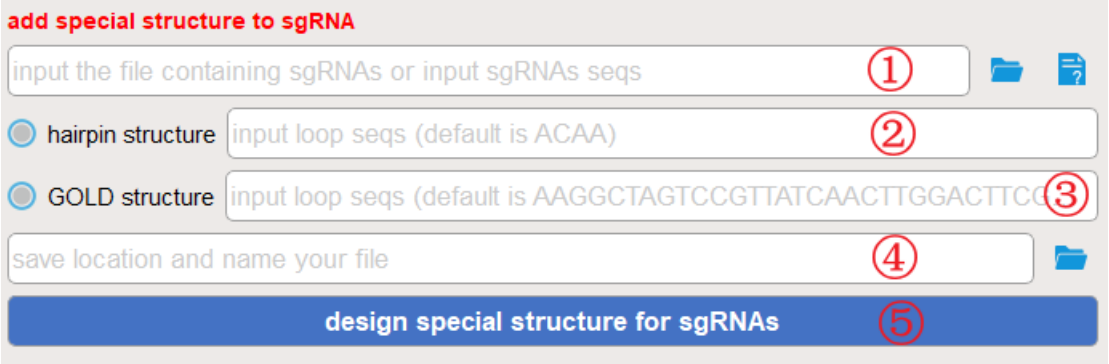
function.

5.4 Filter the sgRNA



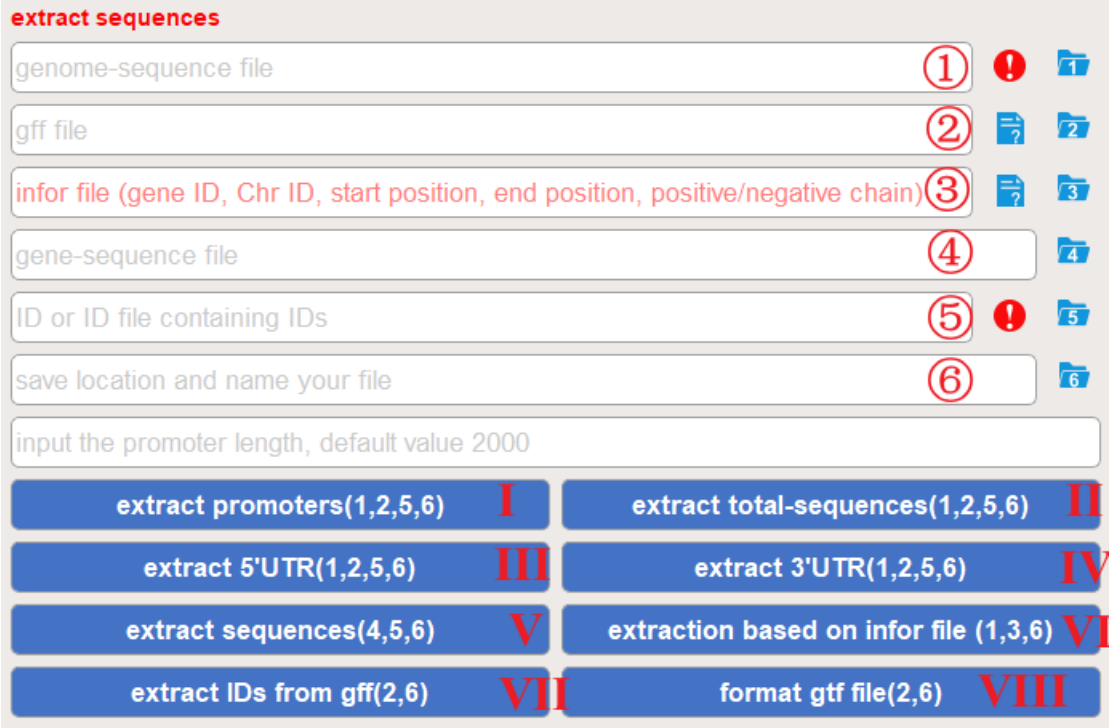
Put the sgRNA into ①, set the save location (②) and click ③ to executive the function.

5.5 Add special structures to sgRNA



Put the sgRNA into ①, select ② or ③ according to their needs, set the save location and the output file at ④, and click ⑤ to execute the function.

5.6 Extract the sequences



Extract the promoter: ① → ② → ⑤ → ⑥ and click I to execute the function.

Extract total sequence of genes: ① → ② → ⑤ → ⑥ and click II to execute the function.

Extract 5'UTR: ① → ② → ⑤ → ⑥ and click III to execute the function.

Extract 3'UTR: ① → ② → ⑤ → ⑥ and click IV to execute the function.

Extract sequences from fasta file: ④ → ⑤ → ⑥ and click V to execute the function.

Extract sequences from bed file: ④ → ⑤ → ⑥ and click VI to execute the function.

Extract ID from gff file: ② → ⑥ and click VII to execute the function.

Convert gtf file into gff file: ② → ⑥ and click VIII to execute the function.

5.7 Select PAM

Select the Cas type in ① and click ② to execute the function.

6. Operations for the Linux version of CIPD

This version of the CIPD is designed to address the inefficiency of the Windows and MacOS versions due to lack of memory. By Linux, I mean a Linux server here.

6.1 Library construction

Usage:

python CIDP-linux.py -md -gf <genome sequence file> -o <library file name>

example:

python CIDP-linux.py -md -gf ./Osativa_323_v7.0.fa -o pt_kmer.txt

Time consuming: <30 min for a rice genome with total length of 300Mb.

6.2 sgRNA search

Usage:

python CIDP-linux.py -ss -ff <gene sequence file> -misnumber <allowed max number of mismatches> -gf <genome sequence file> -df < library file generated by the above step > -o <output file name>

example:

python CIDP-linux.py -ss -ff try.fa -misnumber 4 -gf ./Olucimarinus_231_v2.0.fa -df ol.txt -o ./designs.txt

6.3 Extract the promoter

Usage:

python CIDP-linux.py -exp -gff_file <gff file> -gf <genome sequence file> -ID_file <ID file containing the IDs of target gene (must be identical with the IDs related to the mRNA information

in the gff file) > -pl <length of the target promoter (default 2000)> -o <output file>

Example:

python CIDP-linux.py -exp -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -pl 1000 -o promoters2.txt

6.4 Extract the total sequence of genes

Usage:

python CIDP-linux.py -ext -gff_file <gff file> -gf <genome sequence file> -ID_file <ID file containing the IDs of target gene (must be identical with the IDs related to the mRNA information in the gff file) > -o <output file>

Example:

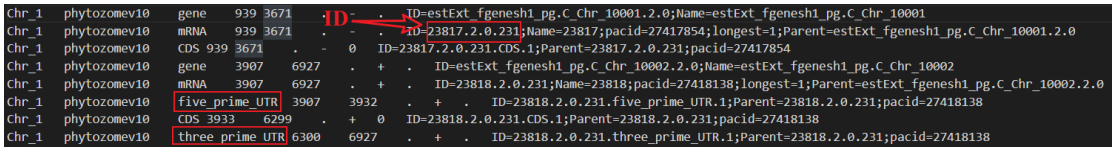
python CIDP-linux.py -ext -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -o total_length.txt

6.5 Extract the 5'UTR

Usage:

python CIDP-linux.py -ex5 -gff_file <gff file> -gf <genome sequence file> -ID_file <ID file containing the IDs of target gene (must be identical with the IDs related to the mRNA information in the gff file) > -o <output file>

Notes: CIPD extracts the UTR regions by using the UTR information in the gff files. Therefore, if some genes have no UTR information, then their UTR regions cannot be extracted. For example, the difference between these two genes in the sample file:



Example:

python CIDP-linux.py -ex5 -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -o 5utr.txt

6.6 Extract the 3'UTR

Usage:

python CIDP-linux.py -ex5 -gff_file <gff file> -gf <genome sequence file> -ID_file <ID file containing the IDs of target gene (must be identical with the IDs related to the mRNA information in the gff file) > -o <output file>

Example:

python CIDP-linux.py -ex3 -gff_file Olucimarinus_231_v2.0.gene.gff3 -gf Olucimarinus_231_v2.0.fa -ID_file 1.txt -o 3utr.txt

6.7 Extract sequences from fasta file

Usage:

python CIDP-linux.py -excds -ff <fasta file> -ID_file <ID file containing the IDs of target sequence> -o <output file>

Example:

python CIDP-linux.py -excds -ff fasta_example.txt -ID_file fasta_ID.txt -o extract_seqs.txt

6.8 Extract sequences from a genome according to information in a bed file

Usage:

python CIDP-linux.py - exbed -bed_file <bed file> -gf <genome sequence file> -o <output file>

Example:

python CIDP-linux.py -exbed -bed_file bed.txt -gf Olucimarinus_231_v2.0.fa -o bed_seqs.txt

6.9Extract gene IDs from a gff file

Usage:

python CIDP-linux.py -exids -gff_file <gff file> -o <output file>

Example:

python CIDP-linux.py -exids -gff_file Olucimarinus_231_v2.0.gene.gff3 -o ids.txt

6.10 Convert gtf file into gff file

Usage:

python CIDP-linux.py - formatgtf -gtf_file <gtf file> -o <output file>

Example:

python CIDP-linux.py -formatgtf -gtf_file GCF_000214015.3_version_140606_genomic.gtf -o ids.gff

7. Cautions

(1) Functions of different versions of CIPD may be different;

(2) Currently, the calculation of off-target rate can only be carried out for NGG;

(3) Make sure the ID meet the requirements when extracting sequence from a genome;

(4) The extraction of UTR sequences is carried out by identifying the "UTR" identifier in gff file (as follows). Therefore, if a gene lacks the "UTR" identifier in the gff file, its UTR cannot be extracted.