Easy$prime Documentation$

Yichao Li, Yong Cheng

May 28, 2021

CONTENTS

CHAPTER

ONE

CONTENTS:

1.1 Easy-Prime Installation steps

1.1.1 Summary

Installation of easy-prime is really easy, however, you might experience errors due to lower conda version problem. Please make sure that you have conda installed and conda version >= 4.9.

1.1.2 Steps

The installation may take 20 min.

Stage 1. Type the installation command

```
conda create -n easy_prime -c cheng_lab easy_prime
```

Please note that `-n ENV_NAME`, the `ENV_NAME` can be anything strings without space. `-c cheng_lab easy_prime` means installation the compiled conda package (namely easy_prime) from `cheng_lab` channel.

```
[yli11@noderome176 ~]$ conda create -n easy_prime -c cheng_lab easy_prime
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.9.2
  latest version: 4.10.1

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /home/yli11/.conda/envs/easy_prime

  added / updated specs:
    - easy_prime
```

Stage 2. Type y to start installation

Once you have typed in the `conda create` command, the conda program will start to gather information, for example, informing you about new conda version. Then it tells you a "Package Plan", for new packages to be downloaded and installed.

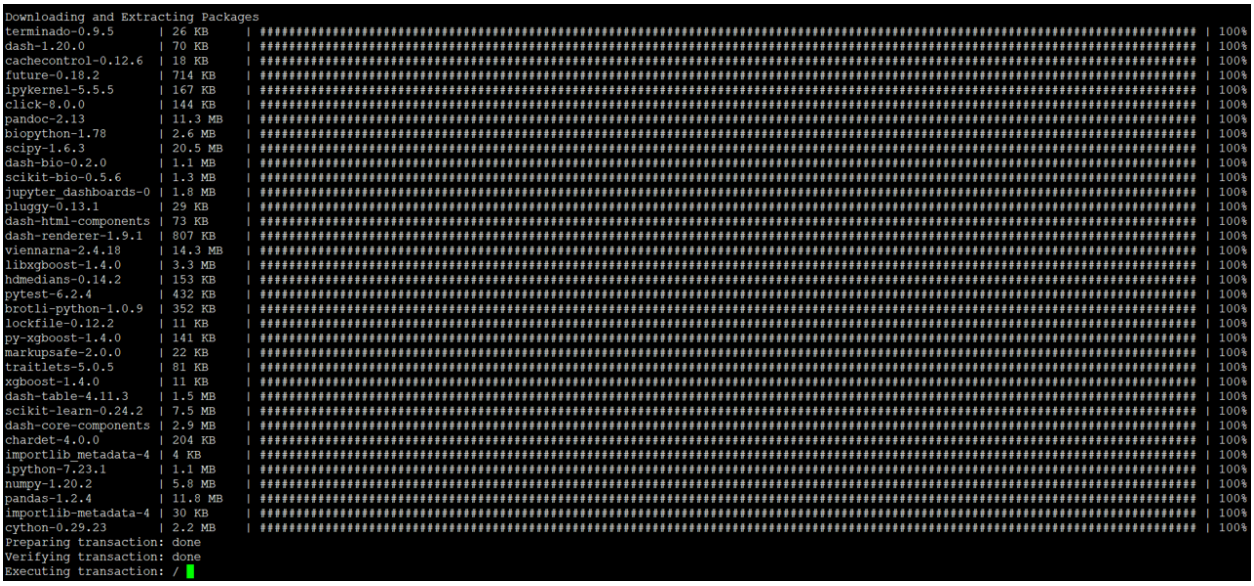
```
The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    biopython-1.78             |   py37h5e8e339_2         2.6 MB  conda-forge
    brotli-python-1.0.9        |   py37hcd2ae1e_4         352 KB  conda-forge
    cachecontrol-0.12.6        |              py_0          18 KB  conda-forge
    chardet-4.0.0              |   py37h89c1867_1         204 KB  conda-forge
    click-8.0.0                |   py37h89c1867_0         144 KB  conda-forge
    cython-0.29.23             |   py37hcd2ae1e_0         2.2 MB  conda-forge
    dash-1.20.0                |     pyhd8ed1ab_0          70 KB  conda-forge
    dash-bio-0.2.0             |           py37_0         1.1 MB
    dash-core-components-1.16.0|     pyhd8ed1ab_0         2.9 MB  conda-forge
    dash-html-components-1.1.3 |     pyhd8ed1ab_0          73 KB  conda-forge
    dash-renderer-1.9.1        |     pyhd8ed1ab_0         807 KB  conda-forge
    dash-table-4.11.3          |     pyhd8ed1ab_0         1.5 MB  conda-forge
    future-0.18.2              |   py37h89c1867_3         714 KB  conda-forge
    hdmedians-0.14.2           |   py37h902c9e0_0         153 KB  conda-forge
    importlib-metadata-4.0.1   |   py37h89c1867_0          30 KB  conda-forge
    importlib_metadata-4.0.1   |       hd8ed1ab_0           4 KB  conda-forge
    ipykernel-5.5.5            |   py37h085eea5_0         167 KB  conda-forge
    ipython-7.23.1             |   py37h085eea5_0         1.1 MB  conda-forge
    jupyter_dashboards-0.7.0   |py37hc8dfbb8_1002         1.8 MB  conda-forge
    libxgboost-1.4.0           |       h9c3ff4c_0         3.3 MB  conda-forge
    lockfile-0.12.2            |              py_1          11 KB  conda-forge
    markupsafe-2.0.0           |   py37h5e8e339_0          22 KB  conda-forge
    numpy-1.20.2               |   py37h038b26d_0         5.8 MB  conda-forge
    pandas-1.2.4               |   py37h219a48f_0        11.8 MB  conda-forge
    pandoc-2.13                |       h7f98852_0        11.3 MB  conda-forge
    pluggy-0.13.1              |   py37h89c1867_4          29 KB  conda-forge
    py-xgboost-1.4.0           |   py37h89c1867_0         141 KB  conda-forge
    pytest-6.2.4               |   py37h89c1867_0         432 KB  conda-forge
    scikit-bio-0.5.6           |   py37ha21ca33_4         1.3 MB  conda-forge
    scikit-learn-0.24.2        |   py37h18a542f_0         7.5 MB  conda-forge
```

```
The following NEW packages will be INSTALLED:

  _libgcc_mutex       conda-forge/linux-64::_libgcc_mutex-0.1-conda_forge
  _openmp_mutex       conda-forge/linux-64::_openmp_mutex-4.5-1_gnu
  _py-xgboost-mutex   conda-forge/linux-64::_py-xgboost-mutex-2.0-cpu_0
  argon2-cffi         conda-forge/linux-64::argon2-cffi-20.1.0-py37h5e8e339_2
  async_generator     conda-forge/noarch::async_generator-1.10-py_0
  attrs               conda-forge/noarch::attrs-21.2.0-pyhd8ed1ab_0
  backcall            conda-forge/noarch::backcall-0.2.0-pyh9f0ad1d_0
  backports           conda-forge/noarch::backports-1.0-py_2
  backports.functoo~ conda-forge/noarch::backports.functools_lru_cache-1.6.4-pyhd8ed1ab_0
  bedtools            bioconda/linux-64::bedtools-2.30.0-h7d7f7ad_1
  biopython           conda-forge/linux-64::biopython-1.78-py37h5e8e339_2
  bleach              conda-forge/noarch::bleach-3.3.0-pyh44b312d_0
  brotli-python       conda-forge/linux-64::brotli-python-1.0.9-py37hcd2ae1e_4
  brotlipy            conda-forge/linux-64::brotlipy-0.7.0-py37h5e8e339_1001
  bzip2               conda-forge/linux-64::bzip2-1.0.8-h7f98852_4
  ca-certificates     conda-forge/linux-64::ca-certificates-2020.12.5-ha878542_0
  cachecontrol        conda-forge/noarch::cachecontrol-0.12.6-py_0
  certifi             conda-forge/linux-64::certifi-2020.12.5-py37h89c1867_1
  cffi                conda-forge/linux-64::cffi-1.14.5-py37hc58025e_0
  chardet             conda-forge/linux-64::chardet-4.0.0-py37h89c1867_1
  click               conda-forge/linux-64::click-8.0.0-py37h89c1867_0
  cryptography        conda-forge/linux-64::cryptography-3.4.7-py37h5d9358c_0
  cycler              conda-forge/noarch::cycler-0.10.0-py_2
  cython              conda-forge/linux-64::cython-0.29.23-py37hcd2ae1e_0
  dash                conda-forge/noarch::dash-1.20.0-pyhd8ed1ab_0
  dash-bio            pkgs/main/linux-64::dash-bio-0.2.0-py37_0
  dash-core-compone~ conda-forge/noarch::dash-core-components-1.16.0-pyhd8ed1ab_0
  dash-html-compone~ conda-forge/noarch::dash-html-components-1.1.3-pyhd8ed1ab_0
  dash-renderer       conda-forge/noarch::dash-renderer-1.9.1-pyhd8ed1ab_0
  dash-table          conda-forge/noarch::dash-table-4.11.3-pyhd8ed1ab_0
  dataclasses         conda-forge/noarch::dataclasses-0.8-pyhc8e2a94_1
  decorator           conda-forge/noarch::decorator-5.0.7-pyhd8ed1ab_0
  defusedxml          conda-forge/noarch::defusedxml-0.7.1-pyhd8ed1ab_0
```

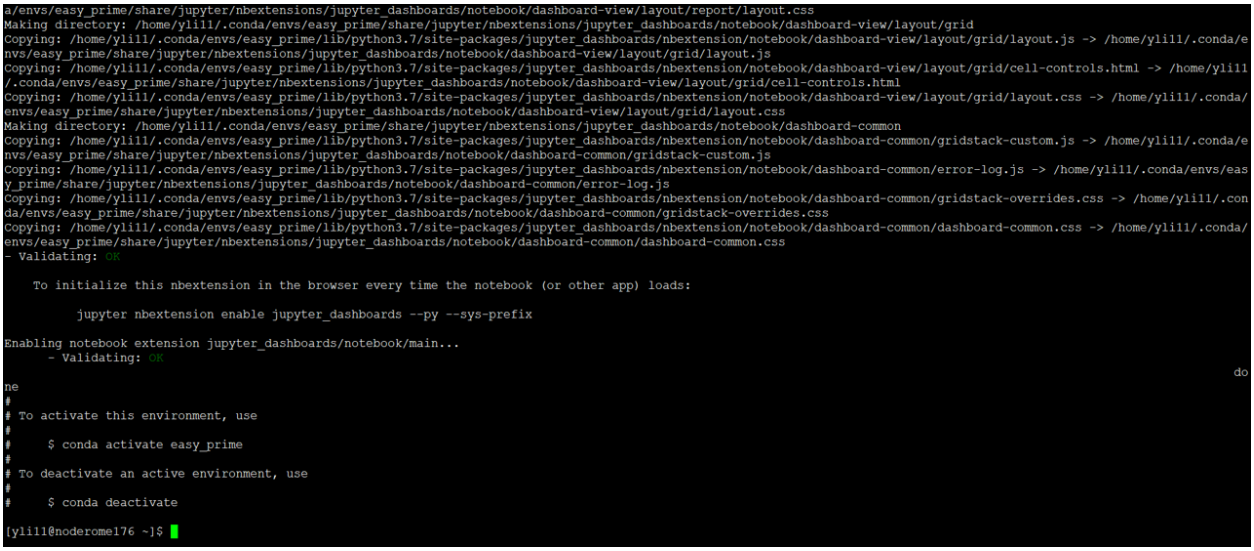
```
  pyparsing           conda-forge/noarch::pyparsing-2.4.7-pyh9f0ad1d_0
  pyrsistent          conda-forge/linux-64::pyrsistent-0.17.3-py37h5e8e339_2
  pysocks             conda-forge/linux-64::pysocks-1.7.1-py37h89c1867_3
  pytest              conda-forge/linux-64::pytest-6.2.4-py37h89c1867_0
  python              conda-forge/linux-64::python-3.7.10-hffdb5ce_100_cpython
  python-dateutil     conda-forge/noarch::python-dateutil-2.8.1-py_0
  python_abi          conda-forge/linux-64::python_abi-3.7-1_cp37m
  pytz                conda-forge/noarch::pytz-2021.1-pyhd8ed1ab_0
  pyyaml              conda-forge/linux-64::pyyaml-5.4.1-py37h5e8e339_0
  pyzmq               conda-forge/linux-64::pyzmq-22.0.3-py37h336d617_1
  readline            conda-forge/linux-64::readline-8.1-h46c0cb4_0
  requests            conda-forge/noarch::requests-2.25.1-pyhd3deb0d_0
  retrying            conda-forge/noarch::retrying-1.3.3-py_2
  scikit-bio          conda-forge/linux-64::scikit-bio-0.5.6-py37ha21ca33_4
  scikit-learn        conda-forge/linux-64::scikit-learn-0.24.2-py37h18a542f_0
  scipy               conda-forge/linux-64::scipy-1.6.3-py37h29e03ee_0
  send2trash          conda-forge/noarch::send2trash-1.5.0-py_0
  setuptools          conda-forge/linux-64::setuptools-49.6.0-py37h89c1867_3
  six                 conda-forge/noarch::six-1.16.0-pyh6c4a22f_0
  sqlite              conda-forge/linux-64::sqlite-3.35.5-h74cdb3f_0
  terminado           conda-forge/linux-64::terminado-0.9.5-py37h89c1867_0
  testpath            conda-forge/noarch::testpath-0.4.4-py_0
  threadpoolctl       conda-forge/noarch::threadpoolctl-2.1.0-pyh5ca1d4c_0
  tk                  conda-forge/linux-64::tk-8.6.10-h21135ba_1
  toml                conda-forge/noarch::toml-0.10.2-pyhd8ed1ab_0
  tornado             conda-forge/linux-64::tornado-6.1-py37h5e8e339_1
  traitlets           conda-forge/noarch::traitlets-5.0.5-py_0
  typing_extensions   conda-forge/noarch::typing_extensions-3.7.4.3-py_0
  urllib3             conda-forge/noarch::urllib3-1.26.4-pyhd8ed1ab_0
  viennarna           bioconda/linux-64::viennarna-2.4.18-py37hfecc14a_0
  wcwidth             conda-forge/noarch::wcwidth-0.2.5-pyh9f0ad1d_2
  webencodings        conda-forge/noarch::webencodings-0.5.1-py_1
  werkzeug            conda-forge/noarch::werkzeug-2.0.0-pyhd8ed1ab_0
  wheel               conda-forge/noarch::wheel-0.36.2-pyhd3deb0d_0
  xgboost             conda-forge/linux-64::xgboost-1.4.0-py37h89c1867_0
  xz                  conda-forge/linux-64::xz-5.2.5-h516909a_1
  yaml                conda-forge/linux-64::yaml-0.2.5-h516909a_0
  zeromq              conda-forge/linux-64::zeromq-4.3.4-h9c3ff4c_0
  zipp                conda-forge/noarch::zipp-3.4.1-pyhd8ed1ab_0
  zlib                conda-forge/linux-64::zlib-1.2.11-h516909a_1010


Proceed ([y]/n)?
```

Now, type y and enter.

Stage 3. Waiting for installation, may take 20 min



Stage 4. Installation is completed



The terminal says, “To activate, use conda activate easy_prime”.

To use `conda activate` or `source activate` depends on the operating system. In Mac and Linux, please use `source activate easy_prime`.

Stage 5. Print Easy_prime help message

```
# To activate this environment, use
#
#     $ conda activate easy_prime
#
# To deactivate an active environment, use
#
#     $ conda deactivate

[yli11@noderome176 ~]$ source activate easy_prime
(easy_prime) [yli11@noderome176 ~]$ easy_prime -h
usage: easy_prime [-h] -f VCF_FILE [-c CONFIG] [-v VERSION] [-o OUTPUT]

easy_prime for pegRNA design

optional arguments:
  -h, --help            show this help message and exit
  -f VCF_FILE, --vcf_file VCF_FILE
                        input target mutations to look for pegRNAs (default:
                        None)
  -c CONFIG, --config CONFIG
                        A YAML file specifying parameters (default: None)
  -v VERSION, --version VERSION
                        print version (default: 1.1.3)
  -o OUTPUT, --output OUTPUT
                        output dir (default:
                        easy_prime_yli11_2021-05-14_result_dir)
(easy_prime) [yli11@noderome176 ~]$
```

Type, `easy_prime -h`

1.2 Easy-Prime Web server tutorial

1.2.1 Welcome to Easy-Prime

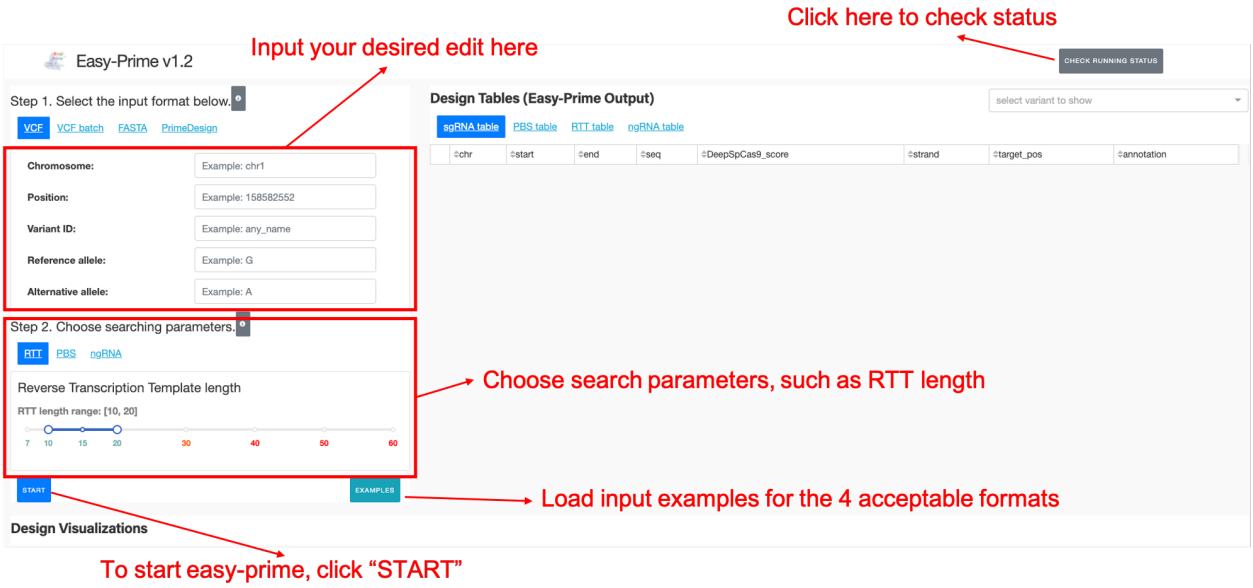
Easy-Prime is a machine learning based tool for prime editing gRNA (pegRNA) design. Please input your desire edits in VCF format or FASTA format and click start. Additionally, you can play with the pegRNA/ngRNA searching parameters. Outputs include a bed-like table and genome-browser visualization.

This web server is based on Dash. URL is: http://easy-prime-test-dev.us-west-2.elasticbeanstalk.com/

Currently it only supports hg19.

1.2.2 Get Started

Go to the easy-prime web portal, the webpage looks like below:



Here, you can find areas to input target mutations, to choose different searching parameters, and output visualizations, including a bed-like table and a genome-browser visualization.

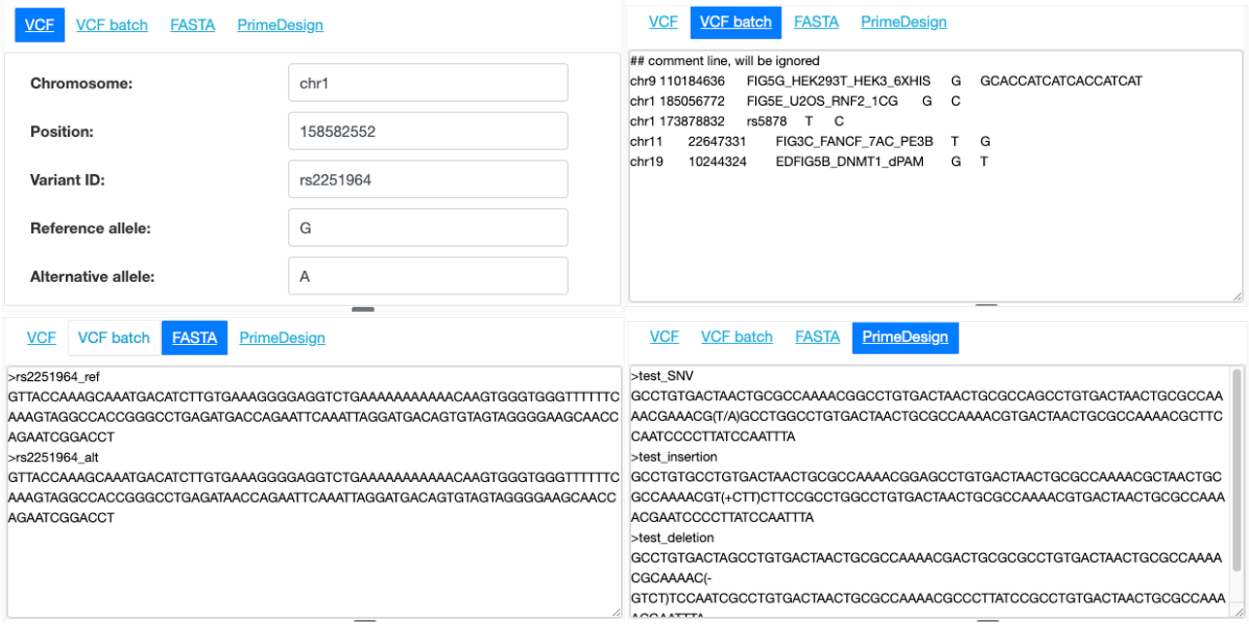
For starter, you can first click `Examples` to automatically load input examples for the 4 acceptable formats.

Sometimes you might experience error (very likely due to incorrect input format), you can click the `check running status` button for error messages. Note that it may not be able to capture all kinds of errors.

Note: If you do experience error and everything seems not working, please refresh the browser and start over. If the issue is still there, please email us.

1.2.3 Input formats

The program accepts 4 types of formats. The first two are VCF-like formats. Basically we need 5 types of information, which are: chr, pos, ID, ref, alt, specified in the first 5 columns in a vcf file.



The last two are fasta-like formats. Basically users can input DNA sequences and the program will automatically determine the target mutation and optimize pegRNA/ngRNA design.

VCF format

```
## comment line, will be ignored
chr9    110184636       FIG5G_HEK293T_HEK3_6XHIS        G       GCACCATCATCACCATCAT
chr1    185056772       FIG5E_U2OS_RNF2_1CG     G       C
chr1    173878832       rs5878  T       C
chr11   22647331        FIG3C_FANCF_7AC_PE3B    T       G
chr19   10244324        EDFIG5B_DNMT1_dPAM      G       T
```

The `VCF` tab is used for single target mutation and the `VCF batch` tab is used for any number of target mutations (prefer less than 10 mutations). The server prohibits output file size > 50M. If you want to design pegRNAs for large number of mutations, please download the command line program.

Note that this format is a tsv format, please do not confuse the program with space or comma. You can first create the input in excel and then copy and paste it to the text box.

FASTA format

```
>rs2251964_ref
GTTACCAAAGCAAATGACATCTTGTGAAAGGGGAGGTCTGAAAAAAAAAAAACAAGTGGGTGGGTTTTTTCAAAGTAGGCCACCGGGCCTGAGATGACCAGAAT
>rs2251964_alt
GTTACCAAAGCAAATGACATCTTGTGAAAGGGGAGGTCTGAAAAAAAAAAAACAAGTGGGTGGGTTTTTTCAAAGTAGGCCACCGGGCCTGAGATAACCAGAAT
```

We use a keyword to recognize the reference and mutated sequences and they are `_ref` and `_alt`. In this example, variant name is `rs2251964`, but it can be string without spaces.

We suggest the input sequence length is at least 100bp.

PrimeDesign format

```
>test_SNV
GCCTGTGACTAACTGCGCCAAAACGGCCTGTGACTAACTGCGCCAGCCTGTGACTAACTGCGCCAAAACGAAACG(T/
↪A)GCCTGGCCTGTGACTAACTGCGCCAAAACGTGACTAACTGCGCCAAAACGCTTCCAATCCCCTTATCCAATTTA
>test_insertion
GCCTGTGCCTGTGACTAACTGCGCCAAAACGGAGCCTGTGACTAACTGCGCCAAAACGCTAACTGCGCCAAAACGT(+CTT)CTTCCGCCTGGCCTGTGACTAA
>test_deletion
GCCTGTGACTAGCCTGTGACTAACTGCGCCAAAACGACTGCGCGCCTGTGACTAACTGCGCCAAAACGCAAAAC(-
↪GTCT)TCCAATCGCCTGTGACTAACTGCGCCAAAACGCCCTTATCCGCCTGTGACTAACTGCGCCAAAACGAATTTA
```

Please see https://github.com/pinellolab/PrimeDesign#primedesign-input-sequence-format for more information.

We use PrimeDesign format as a FASTA format, the fasta header is used as the variant name.

Please note that the `Combinatorial edits` format is not accepted, e.g., `GC(G/T)CCA(+ATCG)AAA`

1.2.4 Searching Parameters

Here users can change RTT length, PBS length, and nick-gRNA distance. We suggest users just use the default settings.

1.2.5 Output pegRNA/ngRNA design tables

Once easy-prime is finished, default sgRNA, PBS, RTT, ngRNA selection is set to be the one with the highest predicted editing efficiency.

Users can click on each tab (e.g., PBS table tab) to choose other sequences. Selection of sgRNA triggers updates of PBS, RTT, and ngRNA table, since there 3 components are unique for each sgRNA. Each selection triggers the genome browser visualization in the bottom.

To download all results for current Easy-Prime prediction, click the `Download all prediction` button. This will download all prediction in a bed-like format as a zip file. Remember that Easy-Prime exhaustively searches all combinations, this is a big file.

To download your current selection, click "Download current selection". This is a bed-like format containing the 4 components of a pegRNA/ngRNA, which are sgRNA, PBS, RTT, and ngRNA.

1.2.6 Output pegRNA/ngRNA genome browser visualization

Genome browser view is powered by Protein Paint (https://pecan.stjude.cloud/proteinpaint). You can zoom in to actually see the DNA bases.

Howevery, we only support hg19 in the tracks. So then the second visualization, will be better if your input is in FASTA format (e.g., if you have hg38 variant, you can first extract +/- 100bp sequence and input here).

1.3 Ask questions here

https://github.com/YichaoOU/easy_prime

1.4 Summary

PE design involves carefully choosing a standard sgRNA, a RT template that contains the desired edits, a PBS that primes the RT reaction, and a ngRNA that nicks the non-edit strand. Usually thousands of combinations are available for one single disired edit. Therefore, it is overwhelming to select the most likely high-efficient candidate from the huge number of combinations.

Easy-Prime applies a machine learning model (i.e., XGboost) that learned important PE design features from public PE amplicon sequencing data to help researchers selecting the best candidate.

1.5 Installation

```
conda create -n genome_editing -c cheng_lab easy_prime

source activate genome_editing

easy_prime -h
```

For detailed installation with screenshots, see: Installation

1.6 Input

1. vcf input example

VCF headers will be ignored. Only the first 5 columns from the vcf file will be used; they are: chr, pos, name/id, ref, alt.

```
## comment line, will be ignored
chr9	110184636	FIG5G_HEK293T_HEK3_6XHIS	G	GCACCATCATCACCATCAT
chr1	185056772	FIG5E_U2OS_RNF2_1CG	G	C
chr1	173878832	rs5878	T	C
chr11	22647331	FIG3C_FANCF_7AC_PE3B	T	G
chr19	10244324	EDFIG5B_DNMT1_dPAM	G	T
```

2. fasta input example

To specify reference and alternative allele, you need two fasta sequences; *_ref* is a keyword that will be recognized as the reference allele and *_alt* is a keyword for target mutations.

```
>test_ref
AAAAAAAAAAAAAAAAAAAAAAAAAGGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
>test_alt
AAAAAAAAAAAAAAAAAAAAAAAAAGGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

1.7 Config file

Default values are shown in the following yaml files.

```
genome_fasta: /path/to/genome.fa
scaffold: GTTTTAGAGCTAGAAATAGCAAGTTAAAATAAGGCTAGTCCGTTATCAACTTGAAAAAGTGGCACCGAGTCGGTGC
debug: 0
n_jobs: 4
min_PBS_length: 8
max_PBS_length: 17
min_RTT_length: 10
max_RTT_length: 25
min_distance_RTT5: 3
max_ngRNA_distance: 100
max_target_to_sgRNA: 10
sgRNA_length: 20
offset: -3
PAM: NGG
```

1.8 Output

The output folder contains:

- topX_pegRNAs.csv
- rawX_pegRNAs.csv.gz
- X_p_pegRNAs.csv.gz
- summary.csv

The top candidates are provided in *topX_pegRNAs.csv*. This is a rawX format file.

CHAPTER

TWO

RAWX FORMAT

X means the input to machine learning models. Here, rawX basically means the file before machine learning featurization. Specifically, rawX contains 11 + 1 columns. The first 5 columns are from the input vcf file: sample_ID, chr, pos, ref, alt, where sample_ID ends with *_candidate_xxx*, this indicates the N-th combination. The next 6 columns are genomic coordinates: type, seq, chr, start, end, strand, where the *type* could be sgRNA, PBS, RTT, or ngRNA. Since for one PE design, it has to have these 4 components, which means that for one unique *sample_ID*, it has 4 rows specifying the sequences for each of them. The 12-th column, which is optional, is the predicted efficiency; in other words, the Y for machine learning.

Both *topX_pegRNAs.csv* and *rawX_pegRNAs.csv.gz* use this format.

CHAPTER

THREE

X FORMAT

X format is the numeric representation of rawX. *X_p* format appends the predicted efficiency to the last column of X.

CHAPTER

FOUR

MAIN RESULTS

The main results, which is the top condidates, is provided in *topX_pegRNAs.csv*.

CHAPTER

FIVE

PE DESIGN VISUALIZATION

Users can visualize the predicted combinations using:

```
easy_prime_vis -f topX_pegRNAs.csv -s /path/to/genome_fasta.fa
```

This will output pdf files to a result dir.

5.1 Usage

```
git clone https://github.com/YichaoOU/easy_prime

cd easy_prime/test

easy_prime -h

easy_prime --version

## Please update the genome_fasta in config.yaml

easy_prime -c config.yaml -f test.vcf

## Will output results to a folder
```

CHAPTER

SIX

DASH APPLICATION

Easy-Prime also provides a dash application.

Please have dash installed before running the dash application.

```
git clone https://github.com/YichaoOU/easy_prime

cd easy_prime/dash_app

python main.py
```