**Introduction:**The following file is provided as a detailed ReadMe file, for someone who would like to edit and use this program. Here are descriptions of each script, files necessary, and files produced. Everything was coded in Python 3.6 using Pycharm (2018.1) with a Conda (Anaconda) environment, “Base Editing”, the details for which are provided in a separate .yml file (Base\_Editing\_Env.yml). The final directory contains this file, a less detailed text version of this file, the zipped PyCharm files, the input and output files that were used and produced, and the Conda environment file.

**General Notes:**

* Suffix “T” refers to Top strand
* Suffix “B” refers to Bottom strand
* Suffix “df” refers to the type of the variable (pandas DataFrame)
* Suffix “dict” refers to the type of the variable (dictionary)
* Suffix “list” refers to the type of the variable (list)
* “PAM”, “AA”, “CDS” are always kept all capitalized

**User Required Input:**   
The main file has places where the user input is required, these are labeled “USER\_INPUT\_REQUIRED”. The instances are summarized below.

1. (STEP 1) Provide the path to the top strand sequence file (txt)
   1. # USER\_INPUT\_REQUIRED - change the path in TopSequenceFile
2. (STEP 1) Provide a path out for the complement sequence file (filecomp)
   1. # USER\_INPUT\_REQUIRED - change the path in filecomp
3. (STEP 2) To change the PAM sequence, go into the position\_pam.py function, findPAM and change (both instances of) “([atc]gg)” to the letter combination of interest.
4. (STEP 2) To change the length and position of the editable area (default is -19 to -11 – an example of -19 to -16 is commented out) in the protospacer change (both instances of) the number of base pairs (bp#) and the number in the “[start-#]”.
5. (STEP 20) Provide the path out for the final excel sheet with all the calculations.
   1. # USER\_INPUT\_REQUIRED - change the path in writePathout

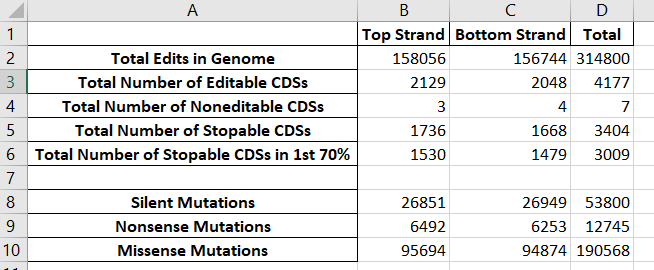
**External Files Required:**

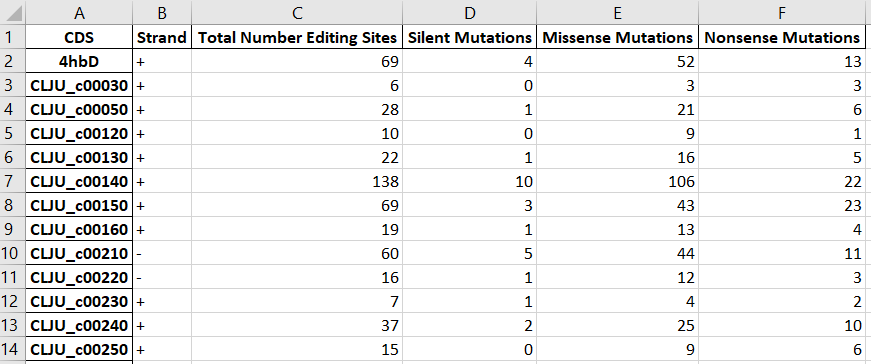
1. DNA top strand sequence file:
   * A text file with the nucleotide sequence
   * The reader (**readtxt**) will skip a line that starts with “>”
2. File with the CDS (gene) information
   * A tab delimited text/csv file
   * Should (bold are optional) to contain the following columns/headings “CDS, CDS length, direction, start, stop, **annotation**”
     1. E.g. “locus length bp direction start stop NCBI Name NCBI Locus NCBI Annotation”
   * Direction must be given as “=>” as forward and “<=” for backward.

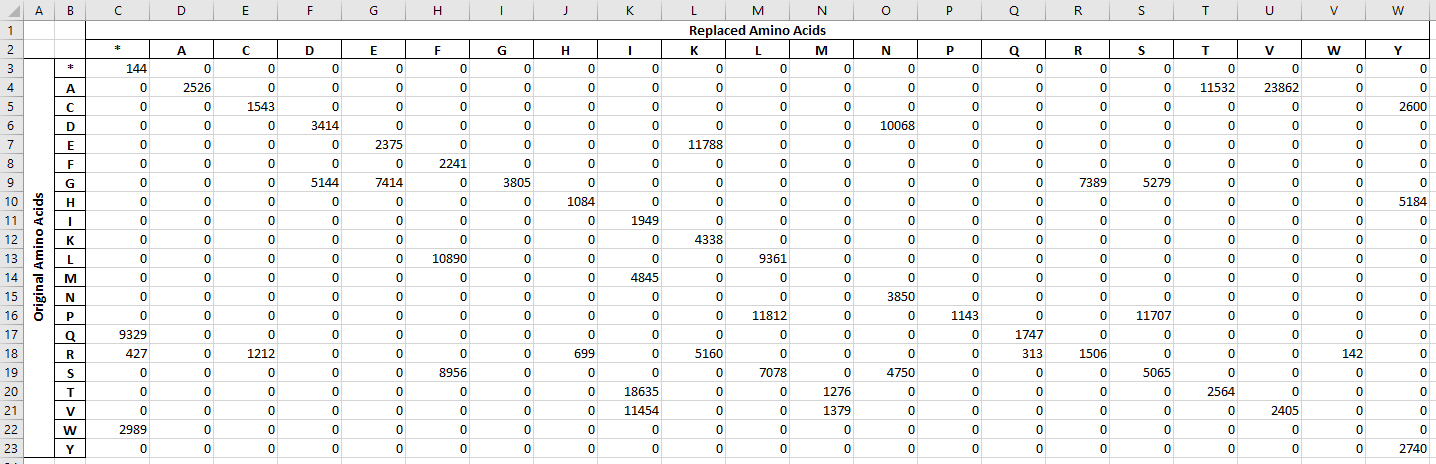
**Table 1**. Describes the included script files, required input files, and output files. The descriptions for the functions can be accessed through the Python “Help()” function.

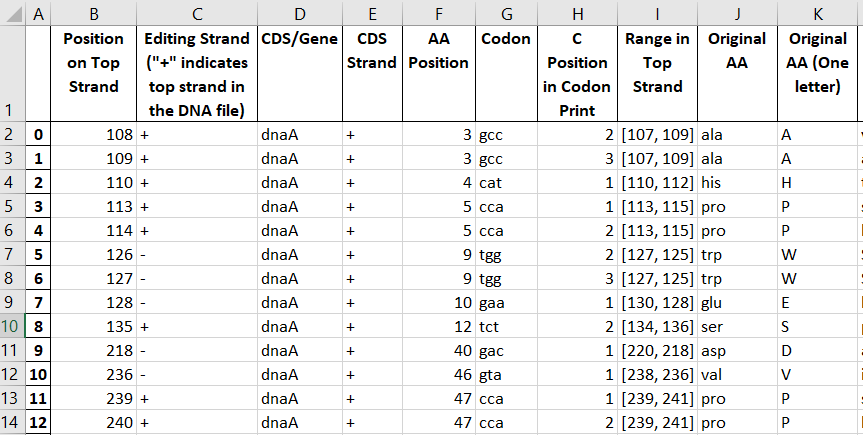
|  |  |
| --- | --- |
| **File Name** | **Description** |
| main.py | **STEP 1**: Read in Top strand sequence file and make the complement in the 5'->3' direction (the Bottom strand). Using **readtxt** function in read\_txt.py. Then the complement strand is made with the function **complement** in complement.py.  **STEP 2**: Find all the PAM Sequence information (considering "ngg" 5'->3'). Using the **findPAM** function in position\_PAM.py. For both the top and bottom strand, a lists of lists with [PAM ID #, PAM seq, start, stop, 20 bp upstream, bp -19 to -11] is returned  **STEP 3**: Find all the Cs that are in the 2-10 (-19 to -11) bp range of the PAM sequence, done on each strand. Using the **countCedit** function in all\_editable.py.  **STEP 4**: This step reads in a txt file with the CDS information from Snapgene and NCBI (CDS, CDS length, direction, start, stop, annotation) as a pandas dataframe. Using **readtxt2df** in read\_txt.py.  **STEP 5**: Makes a series of dictionaries and lists of the CDS and the range (start,stop), ["CDS", [start,stop]] or {"CDS":[start,stop]} using **makeCDSdict** from check\_CDS.py.  **STEP 6**: This step makes a list for each strand that shows the [C position, C strand, Locus, Locus strand] using **checkinCDS** from check\_CDS.py.  **STEP 7**: Remove duplicates from the listCCDS\_T/B (and creates new lists), there are duplicates because the same "C" can be edited sometimes by multiple PAM sequences. Using **remove\_dup**(listCCDS).  **STEP 8**: This will make a dataframe (df) for each strand with the following 8 columns: 'Position on Top Strand', '"C" Strand', 'Locus', 'Locus Strand', 'AA Number', 'Codon', 'C Position in Codon', 'C Position in Codon Print', 'Range in Top Strand' using the function **make\_codon**(listCCDS, dictranges, SequenceTop, SequenceBottom) from AA\_dict.py.  **STEP 9**: Determine the Amino Acid (AA) (and the single letter code) that was originally in the coding strand and then edited one, using **AA\_determinator\_df**(df) from AA\_dict.py.  **STEP 10**: Export the PAM data for the given C Positions (want the repeats etc), using **addPAM**(listPAM, df) from AA\_dict.py.  **STEP 11**: Add the new AA data to the dataframes (DF) from the top and bottom strand, this includes the columns: “Old AA”, "Old AA Code", "New AA", "New AA Code", "New Codon", "Term", "PAM Sequence", "PAM Range", "20bp up PAM"  **STEP 12**: Clean-up of the DF, remove the 'C Position in Codon' column (the user doesn't care about it), combine the DF for the top and bottom strands and sort them by the C position.  **STEP 13**: Puts together the summary information. 1st it extracts all the locus’ that can be edited from the DF (this is done separately from top and bottom strand). Because each locus can be edited in multiple spots, duplicates are removed (using **remove\_dup** from check\_CDS.py) to produce a unique list but each instance is counted  **STEP 14**: Turn information into a DF  **STEP 15**: combine the CDS/number of sites between top and bottom strand into a DF  **STEP 16**: Uses **countsCDSedits** to get all CDSs with edit info (their strand, the total number of editing sites, the unique nonsense, missense and stop edits), strand of the CDS, the num/count of prematurely stopable CDSs in top strand, the num/count of prematurely stopable CDSs in bottom strand, the num/count of non-editable CDSs on the top strand, the num/count of non-editable CDSs on the bottom strand  **STEP 17**: Creates information for the summary sheet of the workbook. Sums the following for each strand and then the total: "Total Edits in Genome", "Total Number of Editable CDSs", "Total Number of Noneditable CDSs", "Total Number of Stopable CDSs", "Nonsense", "Nonsense (stop)", "Missense". The information converted into a dataframe.  **STEP 18**: Find the number of instances of each type of change (old amino acid to new amino acid) using **numAAchanges** from heatmap.py. This information is converted into a dataframe and a dataframe with a structure for a heatmap using **make\_matrix**(df) from heatmap.py.  **STEP 19**: Edit the column names of the dataframes for the final excel sheet to make more user-friendly. These names are the ones that are found in the output Excel sheet (“DataSet”)  **STEP 20**: Write out all the information to a workbook in 5 different sheets (Summary, Base Editing in CDS, Amino Acid Replacement, List of Editable Sites, Heatmap). The path out needs to be defined (marked with “USER\_INPUT\_REQUIRED”). This step also adds labels to the amino acid replacement sheet that is in the matrix form.  **STEP 21**: Checks that the program exited correctly and prints how long it took |
| all\_editable.py | Has function **countCedit**(listPAMinfo, strand) that finds all instances of the base C in the 2nd --> 10th base pairs before the PAM sequence (counting backward from the PAM they are identified as base pairs -19 to -11). |
| AA\_dict.py | Has function **make\_codon**(listCCDS, dictranges, SequenceTop, SequenceBottom) that returns a dataframe with the following columns: 'Position on Top Strand', '"C" Strand', 'locus\_list', 'locus\_list Strand', 'AA Number', 'codon\_list', 'C Position in codon\_list', 'C Position in codon\_list Print', 'Range in Top Strand'. Note: these are renamed in Step 19 of the main.py to longer more user-friendly names.  Has function **AA\_determinator\_df**(df) that takes the dataframe from **make\_codon** and finds the corresponding amino acids (AA), both the original and the edited ones (3 letter code, 1 letter code, and they type of edit).  Has function **addPAM**(listPAM, df) that first takes in the list with the C position and the PAM information, and combines it to a dictionary. This is done while preserving/listing the multiple PAM sequences for each C position if there are any – throughout the whole genome. It returns the following dictionary and three lists:  CPAMdict, (key is the C position (top strand), the value has the list of the PAM number, the PAM range, the PAM sequence and the 20 bp before the PAM sequence)  listseq (a list of the all the PAM sequences, a list (of lists) if there are multiple that can edit the same C)  listrange (a list of the range of the PAM sequences, a list (of lists) if there are more than one for a given C)  list20bp (the 20 bp before the PAM sequence, a list (of lists) if there are more than one for a given C).  Has function **inversebp**(bp) that provides the complement base pair. This is used in the **make\_codon** function. |
| check\_CDS.py | Has function **makeCDSdict**(dataCDS) that takes the dataframe of the CDS information and puts that information on the ranges of the locus in lists and dictionaries to be used later.  Has the function **checkinCDS**(listlistrange, locC, strandC) that checks to see which of the Cs in editable regions are also in a CDS (on either strand). It returns a list (listCCDS) with the following information: location of the C, strand of the C, locus, strand of the locus.  Has the function **remove\_dup**(listCCDS) that removes any duplicate values *e.g.,* any C’s that can be edited by the same PAM sequence (thus show up more than once). |
| complement.py | Has function **dna\_complement**(sequence) that reads in a string of the top strand DNA sequence and returns the string of the complement (bottom strand) also in the 5’->3’ direction. |
| edit\_stats.py | Has function **find\_duplicates**(df\_all) that finds the count of the “unique” number of nonsense, nonsense (stop), and missense edits for each CDS. “Not unique” is defined as the same type of edit at the same amino acid number within the CDS. This function returns a dataframe (df\_CDS) with the CDSs as the row indexes and the unique edit types as the columns.  Has function **getnoneditable**(df\_all, dictCDSinfo) that makes a dictionary of all the CDS that are not editable and the count based on top and bottom strand.  Has function **countsCDSedits**(countCDSdf\_all,df\_all,dictCDSinfo) that returns a dataframe (with all CDSs, their strand, the total number of editing sites, the unique nonsense, missense and nonsense (stop) edits), strand of the CDS, the number/count of prematurely stoppable CDSs in top strand, the number/count of prematurely stoppable CDSs in bottom strand, the number/count of non-editable CDSs on the top strand, the number/count of non-editable CDSs on the bottom strand. The function also prints the CDSs that are not stoppable and the total count. This function uses both **find\_duplicates** and **getnoneditable**. |
| heatmap.py | Has function **numAAchanges**(df\_all) that counts the number of types a certain edit occurs from x AA to y AA.  Has the function **make\_matrix**(countAAchange\_df) that creates a matrix with the old amino acids as rows and the new/edited amino acids as columns and fills in the number of edits of each type there are. It returns a dataframe with a structure that can be used as a heatmap. |
| position\_PAM.py | Has function **findPAM**(sequence, strand) that finds all the PAM sequences with the sequence “ngg” in the 5’->3’ direction. The function returns a list of lists with the following information: “Sequence #”, "PAM Sequence 5'->3'", "Start Position 5'->3'", "Stop Position 5'->3'", "20bp before PAM 5'->3'", "bp 2-10 5'->3'". The function determines if the file is the top or bottom strand in order to adjust all numbering to be in terms of the top strand (as is convention). |
| read\_txt.py | Has function **readtxt**(filepathname) that skips a line that starts with “>” and then reads in the rest of the file while removing any new lines (“\n”).  Has function **readtxt2df**(filepathname) that makes a pandas Dataframe from a tab delimited csv file, skipping the 1st row and using the 2nd as a header. |
| add\_functions.py | This file contains two functions that are used to calculate if a stop codon occurs in the first 70% of the gene.   Has function **cds\_aa**(dictCDSinfo) which edits the input dictionary (key = locus/CDS, value = [[start position, stop position], “strand (either “+” or “-“)]. The CDS is kept as the key and the number of amino acids in the locus is appended the list of the value. The output dictionary is called, dictCDSinfo\_new.  Has function **first\_70**(dictCDSinfo\_new,df\_all) which will go through all the edits (found in df\_all) and if the edit creates a premature stop codon then it will check if the codon is in the first 70% of the gene. It will count how many unique instances there are sorted by strand - based on the CDS. |
| Full\_genome\_clostridium\_ljungdahlii\_DSM13528.txt | This file is the DNA top sequence file. The format used for the input file should follow this one. |
| COMP\_SEQUENCE.txt | This file is produced during the running of the program. It is the complement sequence of the top strand. |
| DataSet.xlsx | This file is the file output of the program (the name can be changed when assigning the path out), it contains 5 sheets “Summary”, “Base Editing in CDS”, “Amino Acid Replacement” “List of Editable Sites”, “Heatmap”. |

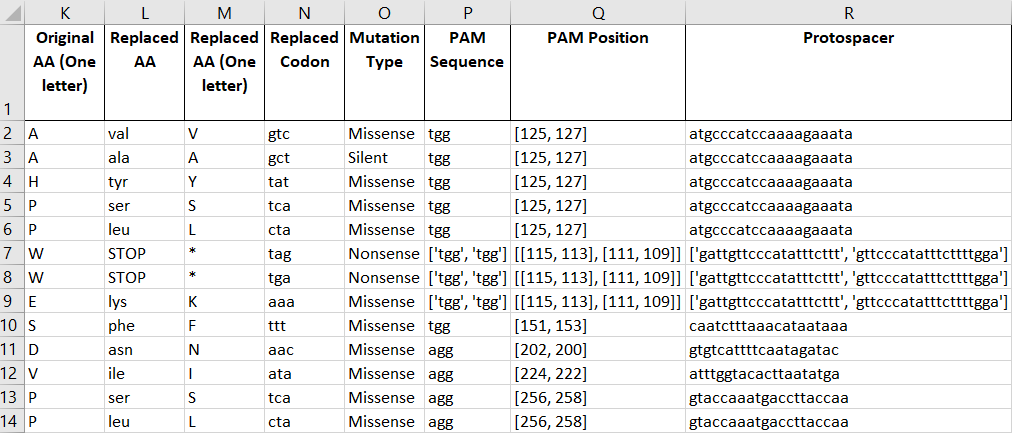
**Preview of Sheets in the Output File**:

Summary:   


Base Editing in CDS:  


Amino Acid Replacement:  


List of Editable Sites:



Heatmap:  
