BLASTer.py Tutorial

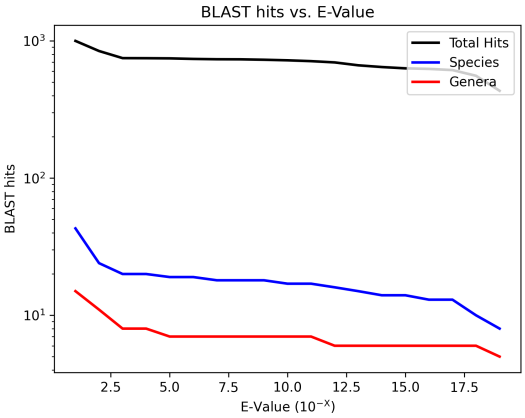
Michael J. Gray, M.S., Ph.D.

April 23, 2020

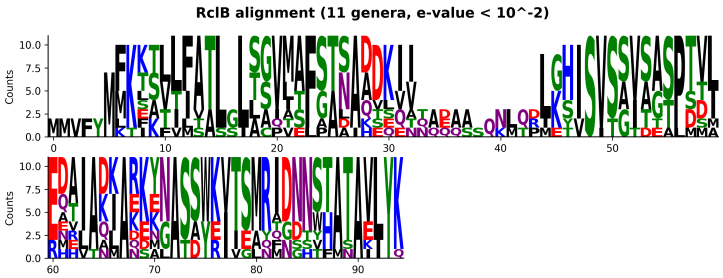
Introduction

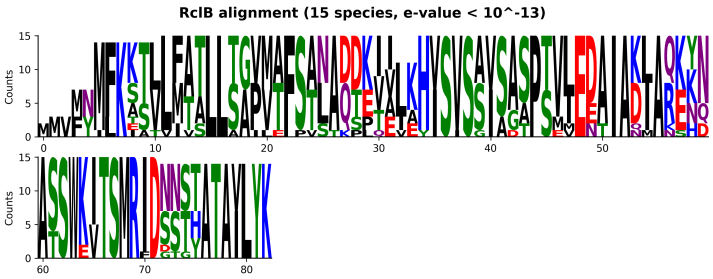
BLASTer.py is a Python script that automates the identification and visualization of protein homology and conservation across biological sequence space. It was developed to simplify and automate a work flow that I use to understand the features and distribution of any protein I work with. BLASTer has two basic elements: the “Align-O-Tron” and a set of tools for generating high-quality protein alignments.

The Align-O-Tron is a tool for determining what statistical E-value cutoff might be meaningful for identifying homologs of your protein of interest. What it does is plot the number of BLAST search hits (from NCBI’s non-redundant RefSeq database) versus the E-value cutoff, generating a graph that shows the number of total hits, as well as how many genera and how many species contain homologs of your protein at a range of E-value cutoffs. This graph for the small *E. coli* protein RclB is shown below.



The Align-O-Tron will also automatically generate alignments and sequence logos at E-value cutoffs with fewer than 100 genera or species, as appropriate. Two examples of RclB alignments from the run that generated the graph above are shown below.





Using the information you've gathered from the Align-O-Tron, BLASTer then allows you to define conditions for a high quality protein alignment at a specific E-value cutoff. This alignment can be processed in a variety of flexible ways, and comes with some tables of useful data:

1) It can be "pruned" to contain only 1 (or any number) of hits per species or per genus.
2) It can be "trimmed" to remove unusually short sequences, unusually long ones, both, or to save *only* those sequences that are longer or shorter than average.
3) A table can be generated that BLASTs each hit in the alignment pairwise with the original sequence, reporting percent identity and percent similarity, as well as reporting the taxonomic lineage of the species from which each homolog originates.
4) A pretty good sequence logo is generated, which might be publication quality. You'll probably want to tweak it, though, and are provided with the raw data files as well.

This document will walk you through installation and use of BLASTer for analysis of the *E. coli* transcription factor RclR (Parker *et al.* 2013 J Biol Chem 288[45]:32574-84).

Installation and Setup

BLASTer was written on a Macintosh computer with Python 3.8.2. Python scripts should run equally well on any type of computer, but I can only 100% vouch for the installation instructions below on a Mac. There is at least one part of the script that will need to change if you are using a PC, and that is indicated below.

The first thing you need to do is download and install Python 3 on your computer. Go to the official Python distribution site and download the most recent version for your operating system.

With the Macintosh distribution of Python, you will next need to run the "Install Certificates.command" script (found in the Python 3.8 folder in your Applications directory) to certify your SSL certificates and allow Python to communicate with the internet. Just double click and let it do its thing.

BLASTer uses three additional Python modules:

Biopython (version 1.76 or greater)
matplotlib (version 3.2.1 or greater)
Logomaker (version 0.8 or greater)

Installation of these is easy. Open up the Terminal and type each of the following:

```
pip3 install biopython
pip3 install matplotlib
pip3 install logomaker
```

After each of those commands has run, your Python 3 installation will be able to access the functionality of those modules, as well as the numpy and pandas modules, which should be automatically installed along with Biopython and Logomaker, respectively.

Finally, you will need standalone applications for protein alignment and local BLAST.

The alignment software used by BLASTer is MUSCLE 3.8.31. Download the muscle3.8.31_i86darwin64 executable from that site and store it in a directory you can easily access.

IMPORTANT: The “i86darwin64” MUSCLE executable is Mac-only. If you are running a PC or Linux computer you will need to download the appropriate MUSCLE executable AND change the name of the program that BLASTer expects to find on lines 770 and 1718 of BLASTer.py.

The local BLAST application is part of the BLAST+ package available for download from NCBI. Download the version appropriate for your operating system and move the “blastp” executable from that package into the same directory where you have stored the MUSCLE executable. This is the “standalone” directory, and BLASTer will need to know where this is. I don’t *think* you will need to make any changes to the script to get it to communicate with blastp on PC or Linux, but I am not 100% certain of that.

The first time you run BLASTer on a Mac, when it attempts to open the executables, you will get pop-up windows that tell you that MUSCLE or blastp “cannot be opened because the developer cannot be verified.” What you need to do is open the General tab of the Security & Privacy panel in your System Preferences. After you click “Cancel” in the pop-up window with the error message, you will get an option in your Preferences window to “Allow Anyway”. Click that, and then BLASTer should run fine afterwards.

To run BLASTer.py, open it with IDLE (or another Python IDE) and select “Run Module”. The first time you run the program, it will ask for some information:

```
BLASTer version 4
April 20, 2020
A tool for examining sequence conservation in proteins.
by Michael J. Gray, M.S., Ph.D.

No settings file has been created. You will need to enter
1. a valid email address (required by NCBI):
```

In order to make automated requests from NCBI’s Entrez servers, you need to provide them with your email address. Enter a valid email address here. If you start using

unreasonable amounts of server time, this allows NCBI to contact you before blocking your IP address.

```
BLASTer version 4
April 20, 2020
A tool for examining sequence conservation in proteins.
by Michael J. Gray, M.S., Ph.D.

No settings file has been created. You will need to enter
1. a valid email address (required by NCBI): mjgray@uab.edu
2. the path to the folder containing the standalone MUSCLE and blastp
applications:
```

This is asking for the directory where you have stored the MUSCLE and blastp executables. On my computer, that's "/Users/michaelgray/Desktop/BLASTer/Standalones", which is what I enter into the script.

```
BLASTer version 4
April 20, 2020
A tool for examining sequence conservation in proteins.
by Michael J. Gray, M.S., Ph.D.

No settings file has been created. You will need to enter
1. a valid email address (required by NCBI): mjgray@uab.edu
2. the path to the folder containing the standalone MUSCLE and blastp
applications: /Users/michaelgray/Desktop/BLASTer/Standalones
3. the path to the folder you want to save your data in:
```

Finally, you will need to enter the path to a folder in which you would like to have your data files saved, On my computer, again, that's "/Users/michaelgray/Desktop/BLASTer".

After you enter the file save directory, you will have something like the following on your screen, and you are ready to start analyzing proteins.

```
BLASTer version 4
April 20, 2020
A tool for examining sequence conservation in proteins.
by Michael J. Gray, M.S., Ph.D.

No settings file has been created. You will need to enter
1. a valid email address (required by NCBI): mjgray@uab.edu
2. the path to the folder containing the standalone MUSCLE and blastp
applications: /Users/michaelgray/Desktop/BLASTer/Standalones
3. the path to the folder you want to save your data in: /Users/michaelgray/
Desktop/BLASTer

File created: BLASTer_settings.txt

Entrez email = mjgray@uab.edu
Standalone application directory = /Users/michaelgray/Desktop/BLASTer/
Standalones
Output file directory = /Users/michaelgray/Desktop/BLASTer
```

```
At any prompt, enter 'Q' to quit or 'H' for contextual help.
------------------------------------------------------------
Enter the protein name:
```

A “BLASTer_settings.txt” file containing the information you just entered will have been created in the directory where BLASTer.py itself is found, and you can edit that if you need to. The script will only ask for this information if no valid settings file is present.

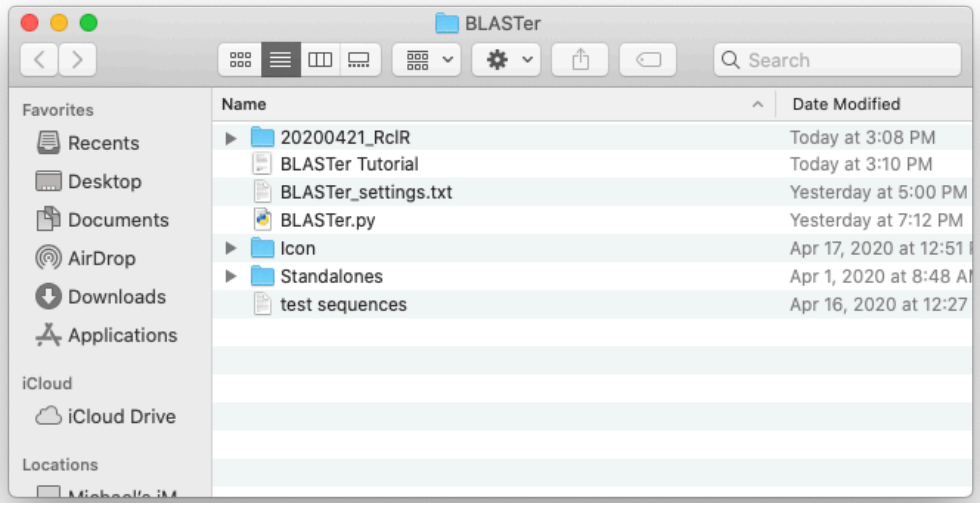
Note that you can type “Q” at any prompt to quit the program, or “H” to access the built-in help function (which is simple, but hopefully somewhat helpful).

Part 1: The Align-O-Tron

The first thing BLASTer will ask for is the name of your protein, which can be any text, and will be used, along with the date, as the basis of the names of any files and folders created in the course of your analysis. As I mentioned above, for this tutorial I will be examining the transcription factor RclR.

```
BLASTer version 4
April 20, 2020
A tool for examining sequence conservation in proteins.
by Michael J. Gray, M.S., Ph.D.

Entrez email = mjgray@uab.edu
Standalone application directory = /Users/michaelgray/Desktop/BLASTer/
Standalones
Output file directory = /Users/michaelgray/Desktop/BLASTer

At any prompt, enter 'Q' to quit or 'H' for contextual help.
------------------------------------------------------------
Enter the protein name: RclR
```

That will cause BLASTer to create a folder in your output file directory with the day’s date and the name of your protein of interest:



Next, BLASTer will ask for your protein sequence. This can be either a full-length or partial protein sequence. RclR is an AraC-family transcription factor, and we are most interested in identifying homologs of its N-terminal sensing domain, not its C-terminal DNA-binding domain, so we enter just the first 162 amino acids (of 284 total):

```
Enter the protein name: RclR
Enter the full or partial protein sequence (with no linebreaks):
MDALSRLLMLNAPQGTIDKNCVLGSDWQLPHGAGELSVIRWHALTQGAAKLEMPTGEIFTLRPGNVVLLPQNSAHRL
SHVDNESTCIVCGTLRLQHSARYFLTSLPETLFLAPVNHSVEYNWLREAIPFLQQESRSAMPGVDALCSQICATFFT
LAVREWIA
```

BLASTer will now ask several questions:

- Do you want to generate a hits vs. value graph (*i.e.* run the Align-O-Tron)?
- If so, do you want to automatically generate alignments and logos?
- What E-value cutoff do you want the Align-O-Tron to begin at? I would recommend either 10^{0} or 10^{-1} as reasonable starting points.
- Finally, it will ask for the maximum number of hits you would like returned. The default is 20,000 (and you can just hit return to get this many), but really what you want is a number larger than the total number of hits at the initial e-value cutoff.

Once you've entered these parameters, BLASTer will perform a BLAST search through the NCBI servers, which sometimes runs very quickly, and sometimes is quite slow. Be patient.

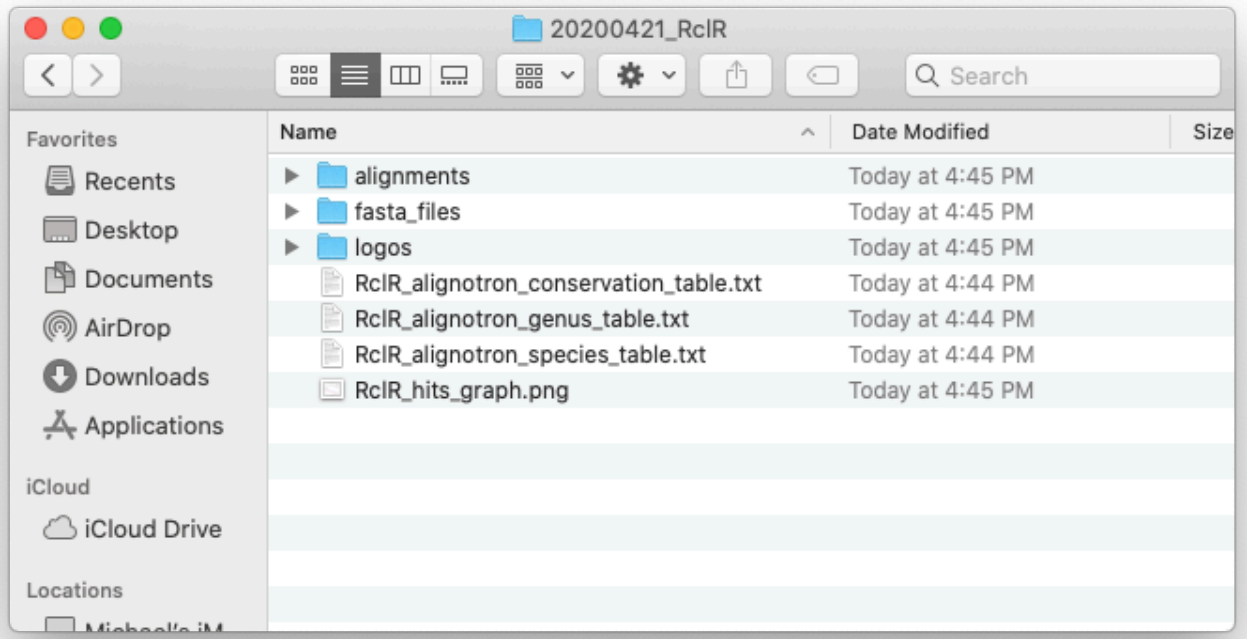
```
Generate a hits vs. e-value graph? Y or N: y
Automatically generate alignments and logos? Y or N: y
Initial e-value cutoff (10^-X): 1
Maximum number of hits you would like returned (recommended = 20000):

Running BLAST search (this may take a while)...
```

Eventually, the BLAST search will complete, and the Align-O-Tron will begin to output a table reporting the number of hits at each successive E-value, and will begin retrieving full-length protein sequences, aligning them and producing sequence logos once the number of genera drop below 100. It will stop when the number of species drop below 10, in this case at an E-value of 10^{-101}.

```
Beginning homology detection loop...

E-value (10^-X)    Total Hits        Species     Genera
1      5029  728   271
2      4330  598   227
3      3550  493   176
4      3096  386   134
5      2648  296   100
6      2360  219   81
RclR_e^-6_81genera.fasta saved, containing 81 genera.
MUSCLE alignment result file saved as RclR_e^-6_81genera.aln
Sequence logo saved as RclR_e^-6_81genera_logo.png
```

```
7     2176  161   65
RclR_e^-7_65genera.fasta saved, containing 65 genera.
MUSCLE alignment result file saved as RclR_e^-7_65genera.aln
Sequence logo saved as RclR_e^-7_65genera_logo.png
8     2072  124   52
RclR_e^-8_52genera.fasta saved, containing 52 genera.
MUSCLE alignment result file saved as RclR_e^-8_52genera.aln
Sequence logo saved as RclR_e^-8_52genera_logo.png

****

100   539   10    4
RclR_e^-100_10species.fasta saved, containing 10 species.
MUSCLE alignment result file saved as RclR_e^-100_10species.aln
Sequence logo saved as RclR_e^-100_10species_logo.png
101   524   9     4
RclR_e^-101_9species.fasta saved, containing 9 species.
MUSCLE alignment result file saved as RclR_e^-101_9species.aln
Sequence logo saved as RclR_e^-101_9species_logo.png

Hits vs. E-Value graph saved as RclR_hits_graph.png
```

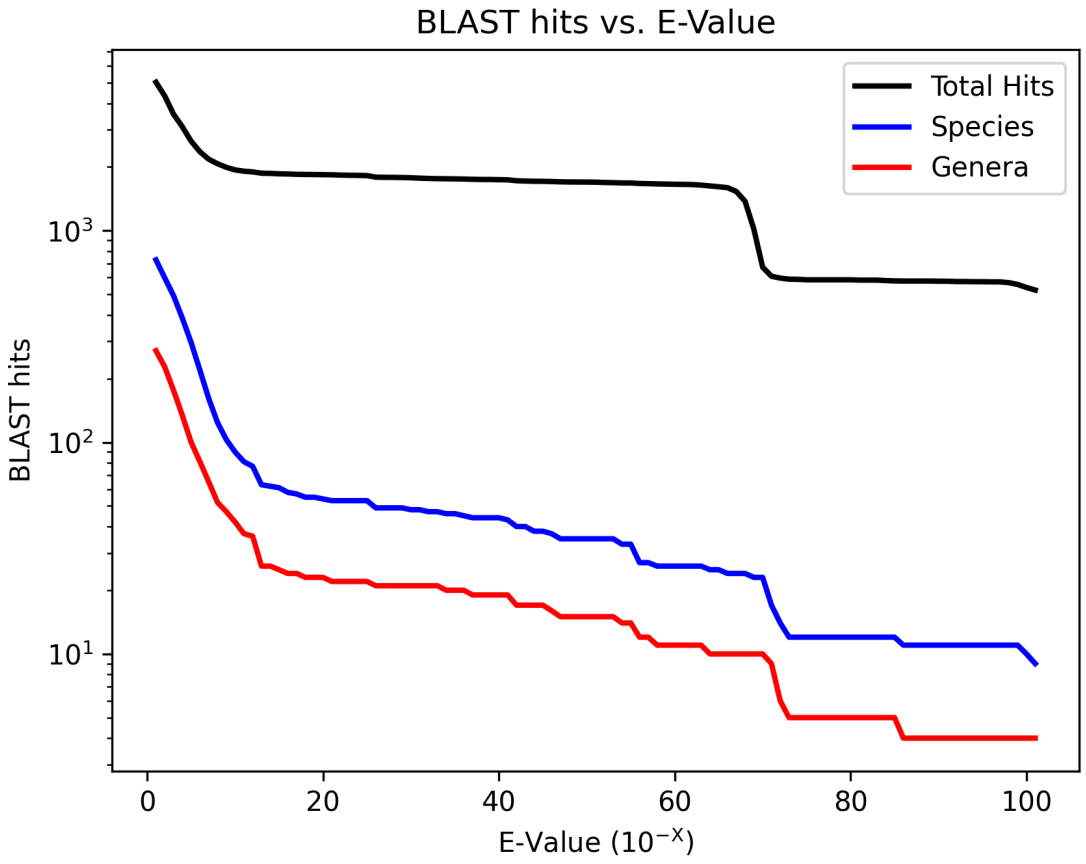
At this point, several different files and folders will have been generated:



These contain the following information:

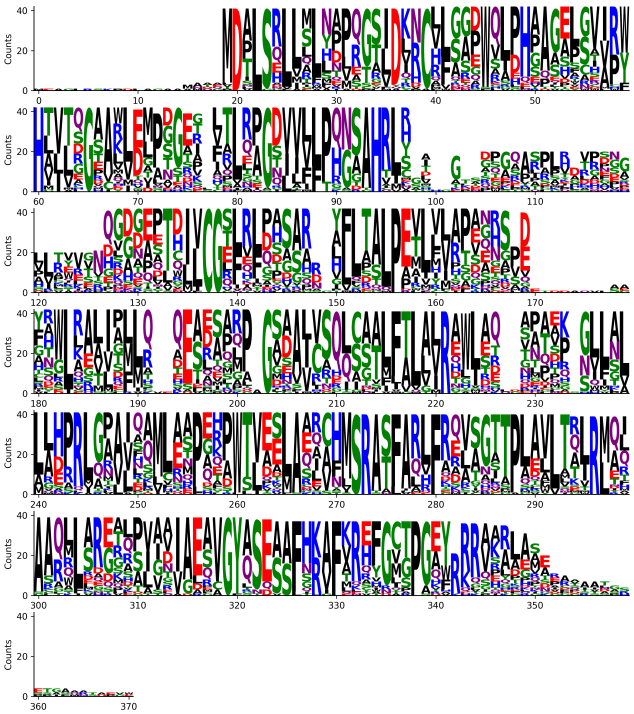
- “fasta_files” directory = unaligned FASTA files containing the protein sequences retrieved for E-values of 10^{-6} through 10^{-101}
- “alignments” directory = MUSCLE-aligned FASTA files containing the protein sequences retrieved for E-values of 10^{-6} through 10^{-101}

- “logos” directory = PNG images derived from alignments of protein sequences retrieved for E-values of 10^{-6} through 10^{-101}
- “RclR_alignotron_conservation_table.txt” = plain text, tab-delimited table of the number of total hits and hits per genus / species at each E-value cutoff
- “RclR_alignotron_genus_table.txt” = plain text, tab-delimited table of the number of hits for each genus at each E-value cutoff
- “RclR_alignotron_species_table.txt” = plain text, tab-delimited table of the number of hits for each species at each E-value cutoff
- “RclR_hits_graph.png” = graph of BLAST hits *vs.* E-value cutoffs (shown below)

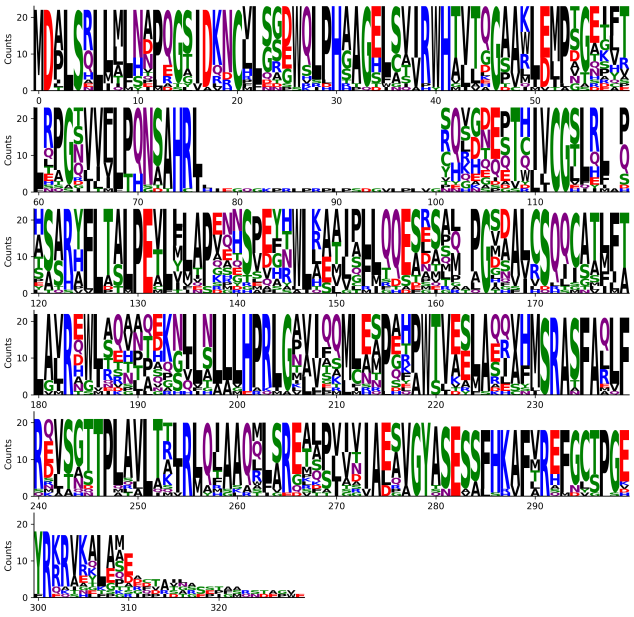


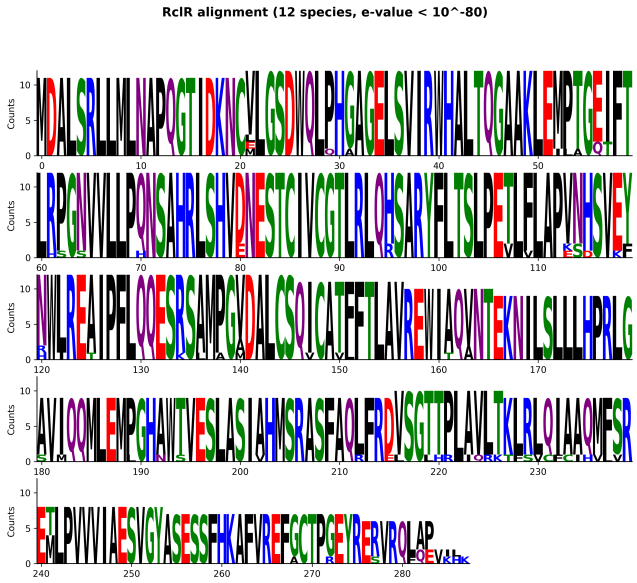
Examination of the individual logos can give a rough sense of the conservation of a protein at different levels of statistical significance. Here are a few examples:

RclR alignment (42 genera, e-value < 10^-10)



RclR alignment (23 genera, e-value < 10^-20)





(One fun thing to do with the sequence logos generated by the Align-O-Tron is to input them into ezgif.com to make an animated GIF.)

Based on these alignments and on the species and genus information contained in the generated tables and the goals of your particular analysis, you can then choose an appropriate E-value cutoff for high quality protein alignment.

Part 2: High Quality Alignment

For the next step, BLASTer will ask if you want to generate a high-quality alignment, and if so, it will request more information:

```
Generate a high quality alignment? Y or N: y
Enter an e-value cutoff (10^-X): 20
How many hits would you like you returned? 10000
```

The E-value cutoff will be determined by how closely related you want the homologs in the final alignment to be (in this case, I chose a cutoff of 10^{-20}), and the number of hits you want returned should be a number larger than the number of total hits at that E-value cutoff (as determined from the hits *vs.* E-value graph). In this case, I chose 10000, since there were about 2000 hits at that E-value.

Once again, at this point BLASTer will contact the NCBI servers to perform the BLAST search, which may take some time, but the output will be somewhat simpler:

```
Running BLAST search (this may take a while)...

Results: 1839 hits.
```

```
Species occurrence table for RclR at e-value < 1e-20 (54 species) saved as
RclR_species_table_20200421.txt
Genus occurrence table for RclR at e-value < 1e-20 (23 genera) saved as
RclR_genus_table_20200421.txt
```

The species and genus occurrence tables are simple tab-delimited text tables containing the number of hits for each species or genus at the chosen E-value.

Next, you will be given the option to repeat the search with a different E-value cutoff or continue to the “pruning” stage, at which point you can choose to keep only 1 (or any number) hit from each genus or species. In this case, I chose to keep 1 hit per species (the most common application in my hands):

```
(C)ontinue to data extraction step or (R)epeat search with changed
parameters? c

Would you like to generate a pruned result file? (Y/N): y
Prune by species or by genus? (S/G): s
How many hits per species? 1
Extracting relevant data...
Requesting GenBank files...
Reformatting results to FASTA file, containing 54 records:
RclR_20200421.fasta
```

What BLASTer does during this step is request the full-length sequences for each hit from GenBank and save them into a FASTA format file, as indicated. You will then be given the option to prune differently (in which case the “RclR_20200421.fasta” file will be overwritten by the new result) or move on to the “trimming” stage:

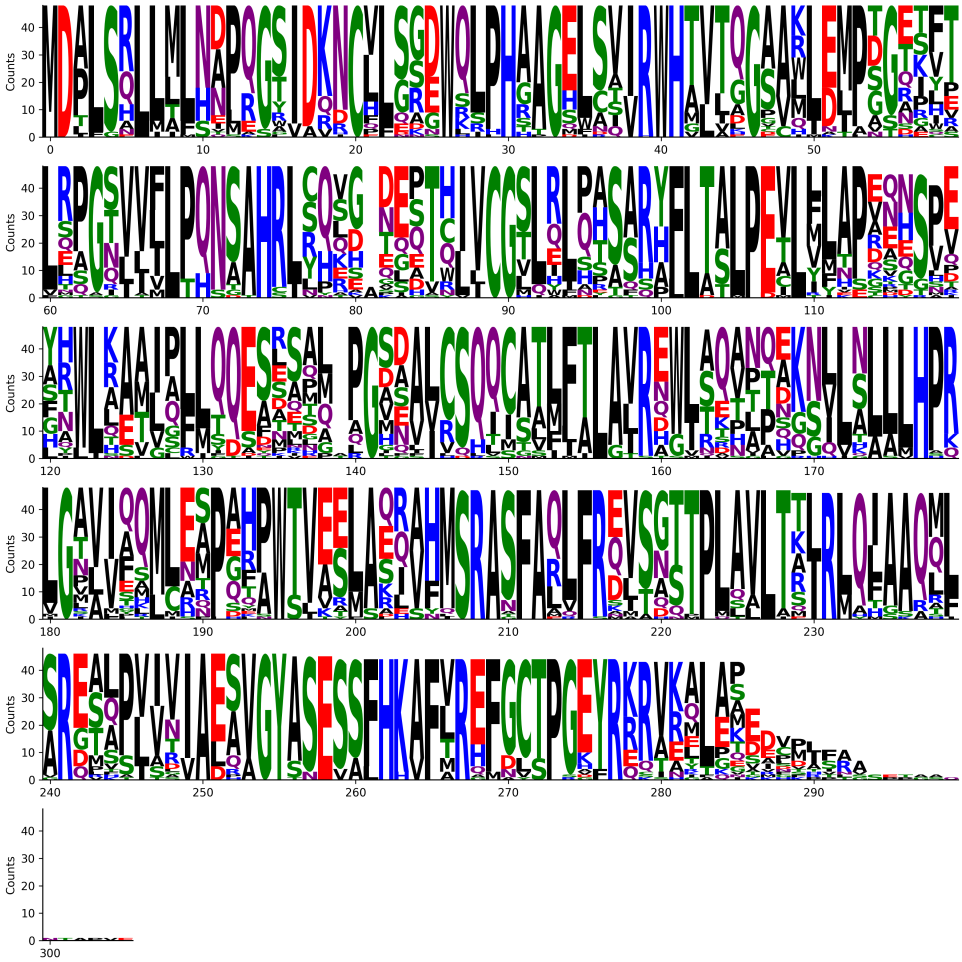
```
(C)ontinue to trimming step or (R)epeat pruning with changed parameters? c

RclR is a 254 amino acid protein. RclR_20200421.fasta contains 54 sequences,
with an average length of 269.1296296296296 and standard deviation of
48.14947647740328.
5 are sequences less than 190.5 amino acids (75% of RclR)
1 are sequences more than 317.5 amino acids (125% of RclR)

Choose one:
Trim (S)hort sequences
Trim (L)ong sequences
Trim (B)oth long and short sequences
Keep only long sequences (OL)
Keep only short sequences (OS)
Do (N)o trimming
b
Trim by (P)ercent length (75-125%), (S)tandard deviation, or (D)efined
length? p
RclR_20200421.fasta trimmed to 48 sequences between 190.5 and 317.5 amino
acids.
```

BLASTer reports the length distribution of your hits, and tells you how many of your hits are substantially shorter or longer than the original protein of interest. These may represent sequencing errors, like partial sequences drawn from incomplete genome sequences or incorrectly annotated stop codons, or interesting biological variants and protein fusions, so be cautious when trimming.

In this case, I chose to eliminate both the long and short sequences to focus on sequences most likely to be similar in structure and function to *E. coli* RclR, and chose the simple “percent length” method for determining the cutoffs for sequences to remove. This brings the number of RclR homologs in my analysis down to 48.

(As you can see, there are lots of trimming options and methods for telling BLASTer which sequences to keep and which to remove.)

The next step is simply asking whether you want to generate an alignment and sequence logo:




```
(C)ontinue to alignment step or (S)kip alignment? c
```

```
MUSCLE alignment result file saved as RclR_20200421.aln
Sequence logo saved as RclR_20200421_logo.png
```
The “RclR_20200421.aln” file is a FASTA-formatted MUSCLE alignment of the 48 RclR homologs, and “RclR_20200421_logo.png” is a sequence logo generated from that alignment, as shown on the previous page.

By comparing this to the logo generated by the Align-O-Tron at E < 10^{-20} (page 9), which contained hits from 23 genera as opposed to the 48 species represented here, you can see that by trimming out long sequences, we eliminated an RclR homolog with a fairly large insertion in the N-terminal half of the protein. Was that a good idea? Maybe, maybe not, but it would probably be a good idea to take a closer look at that protein.

If you want a differently formatted logo and are comfortable with Python, Logomaker is extremely flexible. I have also gotten good results with WebLogo, which is a web-based tool for sequence logo generation that works very well. Either one will accept the “.aln” FASTA-formatted alignment file generated in this step.

The last thing that BLASTer can do is generate a homology and taxonomy table, for which each individual hit will be BLASTed pairwise against the input protein, and the taxonomic lineage of the species from which each hit originated will be retrieved and reported.

```
(G)enerate homology and taxonomy table or (S)kip step? g
Running local BLAST comparisons...
Generating homology table (requesting taxonomic data from NCBI)...
Table row: 1
Table row: 2

***

Table row: 47
Table row: 48
Homology table of data for 48 aligned regions saved as
RclR_homology_table_20200421.txt

(R)epeat analysis, analyze a (N)ew protein, or (Q)uit? q
```

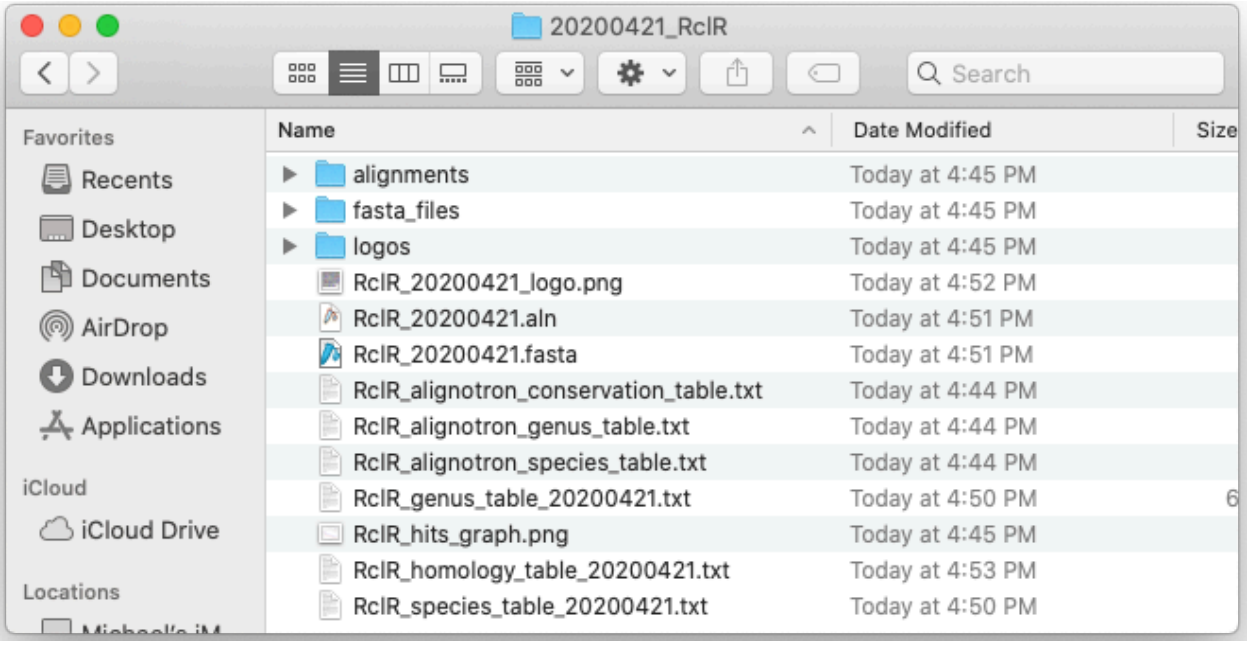
The first few rows of this table are shown on the next page, divided into two chunks to make it fit into this portrait-formatted document.

Species	Accession	Length of Homology	Percent Identical	Percent Similar
Escherichia coli	WP_032251713.1	254 / 254	100.0	100.0
Shigella dysenteriae	MJD66596.1	284 / 254	89.08450704225	89.4366197183099
Escherichia sp. HH26CH	WP_160524165.1	284 / 254	88.73239436619	89.4366197183099
Shigella sp. FC3196	WP_146760262.1	284 / 254	88.38028169014	88.7323943661972
Shigella boydii	EAA0844780.1	276 / 254	90.94202898550	91.3043478260870
Trichuris trichiura	CDW60614.1	284 / 254	88.02816901408	88.7323943661972
Escherichia marmotae	WP_038355734.1	287 / 254	80.48780487804	83.9721254355401

E Value	Taxonomy
0.0	cellular organisms; Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enteroba
0.0	cellular organisms; Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enteroba
0.0	cellular organisms; Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enteroba
0.0	cellular organisms; Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enteroba
0.0	cellular organisms; Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enteroba
0.0	cellular organisms; Eukaryota; Opisthokonta; Metazoa; Eumetazoa; Bilateria; Protostomia; Ecdy
1.81049E-175	cellular organisms; Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enteroba

What you can see from this is that several species of *Escherichia* and *Shigella* (unsurprisingly) have nearly identical RclR proteins to that of *E. coli*. However, you can also see that a protein from *Trichuris trichura* (a eukaryotic nematode) also appears on this list, which is very surprising! This could represent an exciting example of horizontal gene transfer between a gammaproteobacterium and an animal or (more likely) contamination of the *T. trichura* genome sequence with bacterial DNA.

All of the files generated in the previous steps are saved in the same output folder as the Align-O-Tron outputs:



Summary

BLASTer provides a convenient workflow for several types of protein sequence analysis that are inconvenient to do “by hand”, and I hope others will find it useful as well.

I also hope that, by releasing the BLASTer.py script into the world, it can serve as a working example of how to integrate multiple scientific Python packages to achieve a fairly complex result. I have tried to make my code as well-commented and transparent as possible. I have no formal training in programming (which may be obvious!), so there are certainly aspects of BLASTer.py that are not “best practice”, but it does illustrate one way to produce a flexible and fairly robust computational tool. I hope it inspires other biologists to dip their toes in programming and make interesting new tools of their own!

The following books and resources were especially helpful in the process of writing and updating the most recent form of BLASTer, and I recommend them highly:

Python for Everyone: Exploring Data With Python 3 by Dr. Charles Severance

the Biopython Tutorial by Jeff Chang *et al.*

Automate the Boring Stuff with Python by Al Sweigart