User's Guide
Version 1.0

Khalid K Alam[1], Jonathan L Chang[2] & Donald H Burke[1,2]

[1]Department of Biochemistry
[2]Department of Molecular Microbiology and Immunology
University of Missouri, Columbia, Missouri, USA

FASTAptamer

If you use FASTAptamer, please cite the paper:

Khalid K. Alam, Jonathan L. Chang, Donald H. Burke. "FASTAptamer: A Bioinformatic Toolkit for High-Throughput Sequence Analysis of Combinatorial Selections." Molecular Therapy – Nucleic Acids. 2015; 4:e1 DOI: 10.1038/mtna.2015.4

For feedback, suggestions, technical support, etc., please email us at burkelab@missouri.edu or tweet us @BurkeLabRNA.

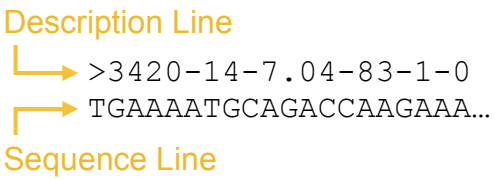
Table of Contents

1. Introduction

FASTAptamer is an open source toolkit designed to address the primary sequence analysis needs from high-throughput sequencing of combinatorial selection populations. FASTAptamer performs the simple tasks of counting, normalizing, ranking and sorting the abundance of each unique sequence in a population, comparing sequence distributions for two populations, clustering sequences into sequence families based on Levenshtein edit distance, calculating fold-enrichment for all of the sequences present in 2 or 3 populations, and searching degenerately for nucleotide sequence motifs. While FASTAptamer was originally developed for analysis of high-throughput sequencing data from aptamer selections, it offers broad utility for those working on ribozyme or DNAzyme selections, surface display (phage display, mRNA display, etc.) selections, *in vivo* SELEX, protein mutagenesis selection, or any biocombinatorial selection that results in a DNA-encoded library for sequencing.

What FASTAptamer ***cannot*** do is merge paired-end reads, trim constant regions from FASTQ files, calculate secondary structure, perform sequence alignments, or any other function for which software already exists (see **section 4c** for more on these resources). Rather than re-inventing the wheel, we decided to simply address the needs of the selections field and ensure that the output from FASTAptamer remains compatible for downstream analysis. Doing so allows users the flexibility of plugging into the FASTAptamer toolkit for rapid identification of candidate biomolecules and performing additional analysis on a smaller subset of their high-throughput sequencing data, all while preserving the sequence metrics important in combinatorial selections.

FASTAptamer makes extensive use of the FASTA file format, the widely used and *de facto* format for representing nucleotide or amino acid sequence information. This format contains two main features - a description line and a sequence line. FASTAptamer exploits the format by utilizing the description line to preserve sequence metrics, such as the abundance of the sequence or it's degree of relatedness to other sequences. By assigning each sequence in a population a unique description line, FASTAptamer is able to use this information throughout the toolkit to perform a variety of primary sequence analysis tasks.



1a. Overview of Features

FASTAptamer-Count is the gateway to the FASTAptamer toolkit and will rapidly parse through a FASTQ file to perform the following tasks:

- Count the occurrence of each unique sequence (often referred to as abundance, read counts, copy number, multiplicity or frequency).
- Normalize the sequence abundance to reads per million (RPM).
- Rank the abundance of each sequence in the population.
- Sort the population by decreasing abundance.

Output from FASTAptamer-Count is provided in FASTA format and is required for all subsequent FASTAptamer scripts to function properly.

FASTAptamer-Compare is a tool to compare the sequence distribution of two populations. Using two input files from FASTAptamer-Count, the script will:

- List the RPM for each sequence present in both input populations, along with the sequence information itself, to allow rapid generation of XY-scatter plots.
- Calculate the binary logarithm (Log_2) of the ratio of RPM in each sequence in both populations.
- Generate “bin buckets” of the binary log values for effortless generation of a sequence distribution histogram.

Output from FASTAptamer-Compare is a tab-separated (or “tab-delimited”) plain text file.

FASTAptamer-Cluster is a tool that can generate families, or “clusters”, of closely-related sequences based on a user-defined Levenshtein edit distance. Using an input file processed with FASTAptamer-Count, the script will:

- Identify “seed” sequences for cluster generation based on abundance.
- Calculate the Levenshtein edit distance (the number of insertions, deletions or substitutions necessary to transform a sequence into the seed sequence) for each unclustered sequence.
- Cluster sequences together if the edit distance from the seed sequence is less than or equal to the edit distance specified.
- Assign each sequence within a cluster a rank based on abundance.

Output from FASTAptamer-Cluster remains in the FASTA format, allowing for downstream analysis within the toolkit or with other software.

1a. Overview of Features

FASTAptamer-Enrich will accept up to 3 input files from FASTAptamer-Count or FASTAptamer-Cluster and rapidly:

- Calculate the fold-enrichment ratio for each sequence present in more than one population.
- List each sequence along with its length, rank, reads, RPM and cluster information (if provided) for each population, in a sortable format for facile candidate identification.
- Filter output to include only those sequences present across all input populations greater than a user-defined RPM.

Output from FASTAptamer-Enrich is provided as a tab-separated plain text file.

FASTAptamer-Search accepts multiple input files from FASTAptamer-Count or FASTAptamer-Cluster and will:

- Search for multiple sequence motifs at a time, using degenerate IUPAC-IUBMB single letter nomenclature for nucleotides.
- Highlight sequence motif matches by enclosing each match in parentheses.
- Generate a new file containing only matched sequences for downstream analysis.

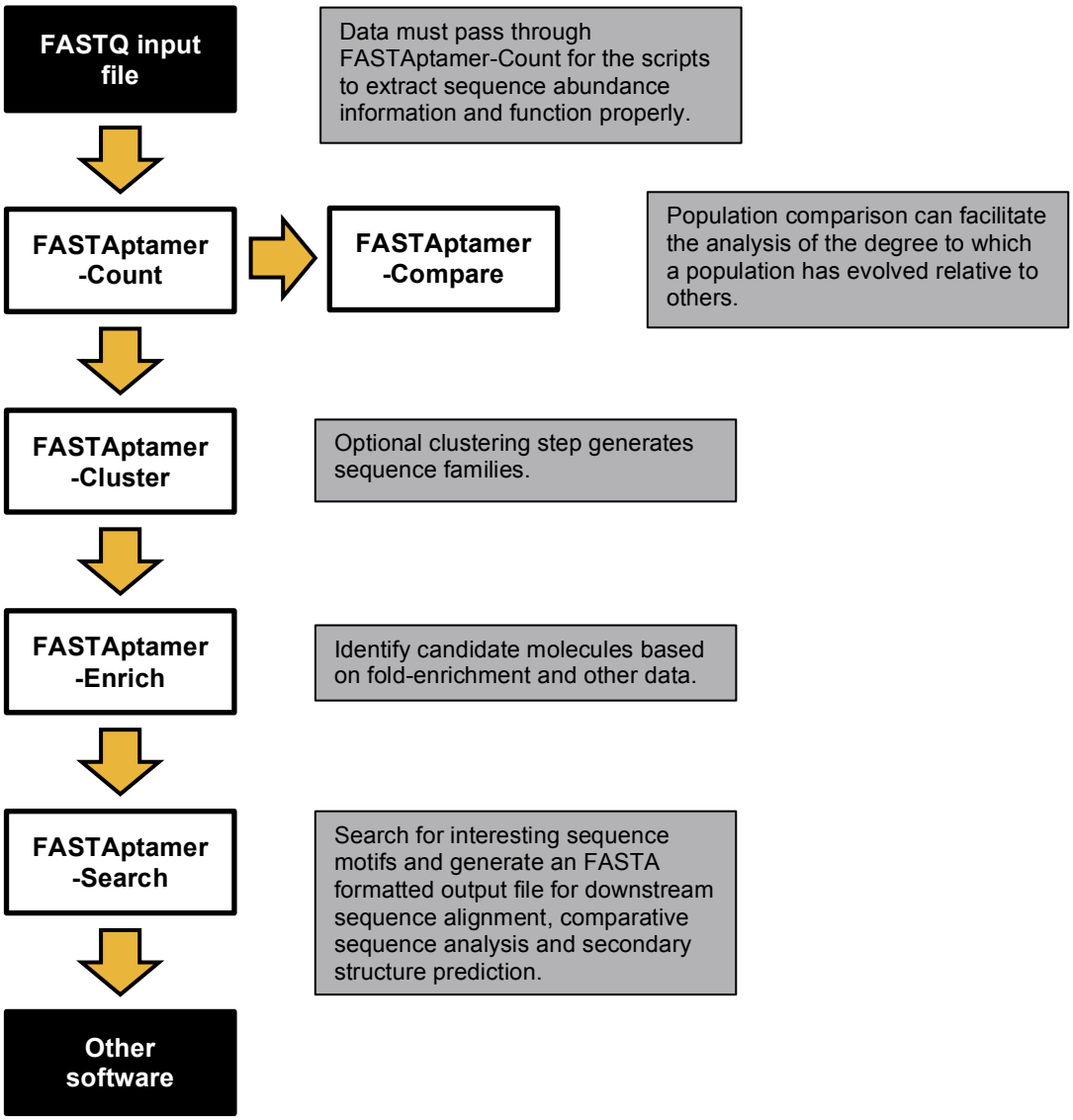
Output from FASTAptamer-Search preserves the FASTA format of each input file.

For more detailed information on FASTAptamer, please see our publication:

Khalid K. Alam, Jonathan L. Chang, Donald H. Burke. “FASTAptamer: A Bioinformatic Toolkit for High-Throughput Sequence Analysis of Combinatorial Selections.” Molecular Therapy – Nucleic Acids. 2015; 4:e22X DOI: 10.1038/mtna.2015.4

1b. Sample Pipeline

FASTAptamer is provided as a modular collection of scripts that can be configured in several ways to extract the information from the dataset that you deem important. Below is a sample pipeline that we use in our research, but be aware that several more configurations exist. For a complete list of input and output compatibilities for each script refer to **section 4a**.



2. Installation & Use

FASTAptamer is provided as a compressed folder containing:

- The 5 FASTAptamer scripts.
- LICENSE.txt – a plain text file containing the GNU GPL v3 software licensing information.
- README.txt – a plain text file with the essential information.
- This PDF user's guide.

After "unzipping" the folder, the FASTAptamer scripts can be installed as executable programs or simply used without installation. Refer to **sections 2c** and **2d** for more information.

2a. User Requirements

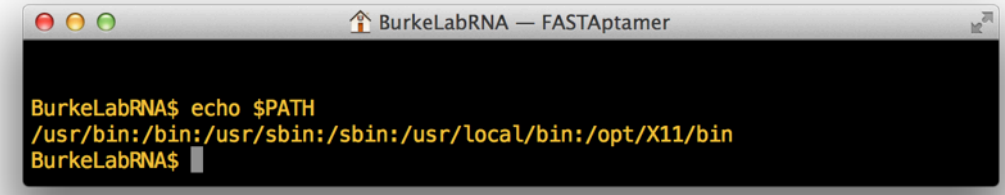
FASTAptamer is designed to be EASY to use. Installation and use of the FASTAptamer toolkit assumes a *basic* working knowledge of command line operation. If you can navigate around your computer's directories ("`cd`"), copy ("`cp`") and move ("`mv`") files, and can tolerate the inability of using a mouse, then you should be able to start using FASTAptamer immediately. If you can't – don't fear. Several resources are provided in **section 4c** that should get you up to speed quickly. The installation instructions and tutorials are designed for the inexperienced user. If you continue to experience problems don't hesitate to tweet us (@BurkeLabRNA) or email us (burkelab@missouri.edu) for support.

2b. System Requirements

FASTAptamer is written in the Perl programming language with no external dependencies. What this means for you is that virtually every modern computer can run it. Linux and Mac users rejoice, as nearly every instance of Linux and Mac OS X can run Perl out of the box. If you're running Windows you'll need to download a Perl interpreter such as Strawberry Perl (open source - always free) or ActiveState's ActivePerl (they provide a free "community distribution"). We've personally tested the toolkit, without issue, on CentOS Linux 5.4, Mac OS X 10.6+, Debian GNU/Linux 7.0 and Strawberry Perl 5.20.1.

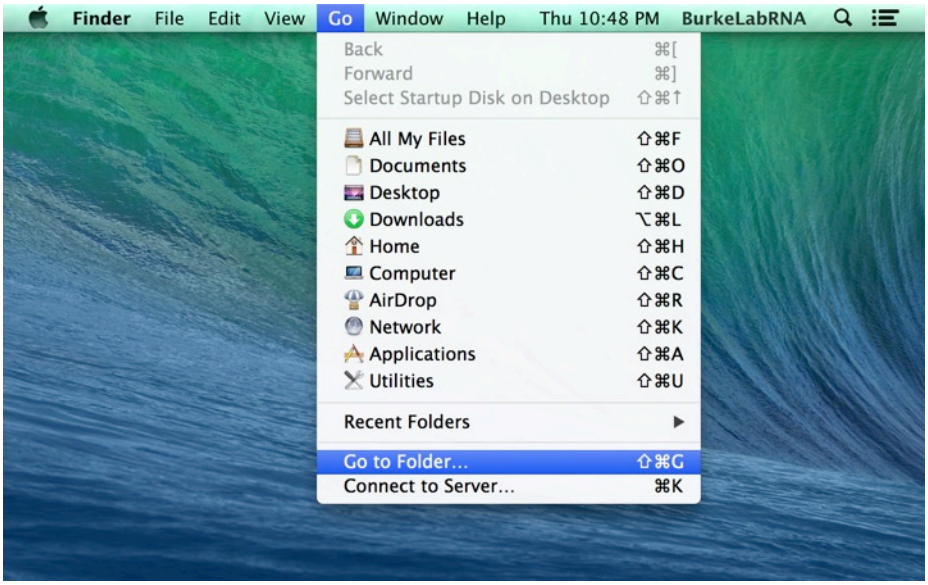
2c. Installing as Executable

The PATH variable in a UNIX-like system is where “executable” programs are called upon by the operating system. Having the FASTAptamer scripts saved in one of these directories will ensure that no matter where you are in the file system, you’ll be able to call upon the scripts to execute from the prompt. To find these directories, open up a terminal emulator (the “Terminal” app on all Macs) and enter `echo $PATH`



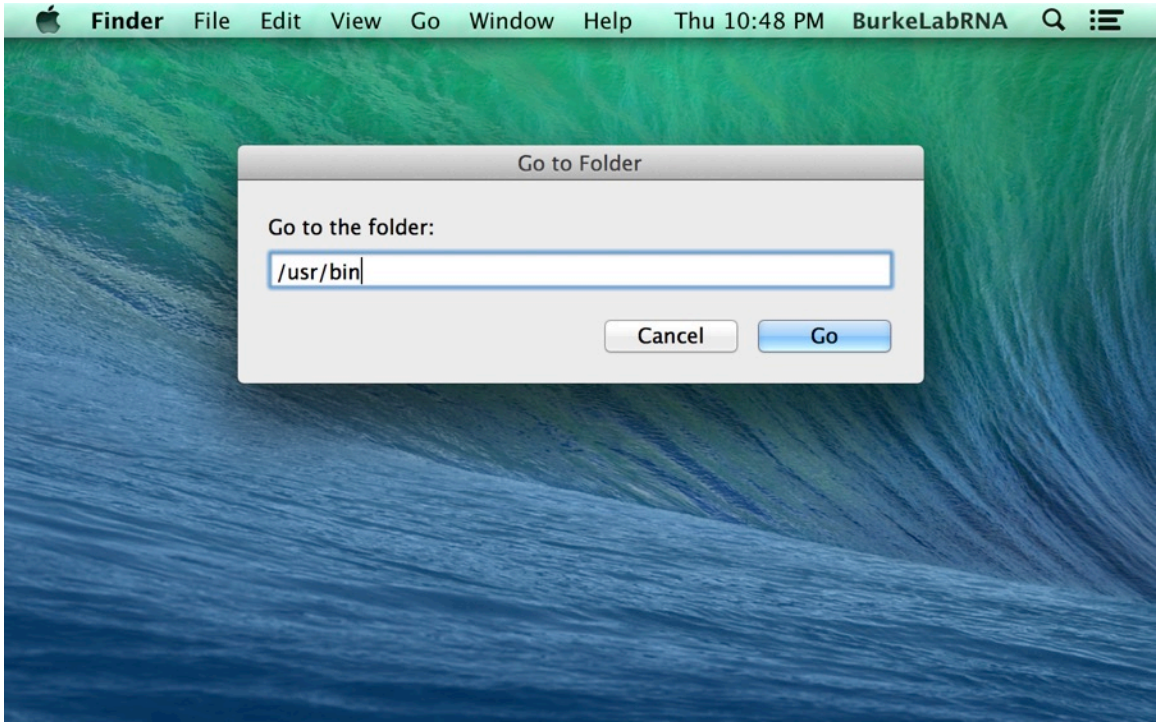
Each directory in the PATH variable is then displayed and separated by a colon. When you enter a command in the command line, the system searches from the left-most directories first for the program to execute. Be aware that some of these directories require administrator privileges and will require your system password to access. In Mac OS X, the /usr/local/bin directory is listed in the PATH but the folder doesn’t usually exist until you create it.

Copy the scripts to the directory you have access to. Depending on your system and your comfort level, this can be performed using the command line or by “dragging-and-dropping” using the graphical user interface. On a Mac, we’ll do this by returning to our desktop and clicking on “Go” in the menu bar.

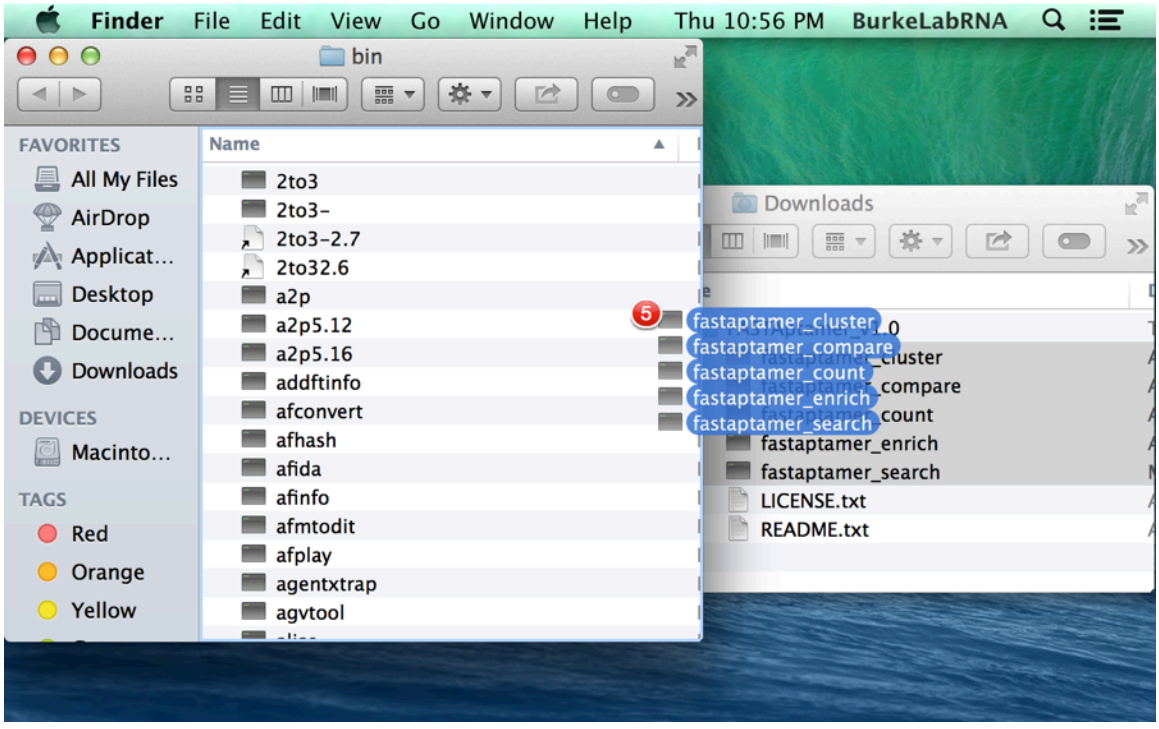


2c. Installing as Executable

Enter the directory you wish to copy or save the scripts to.

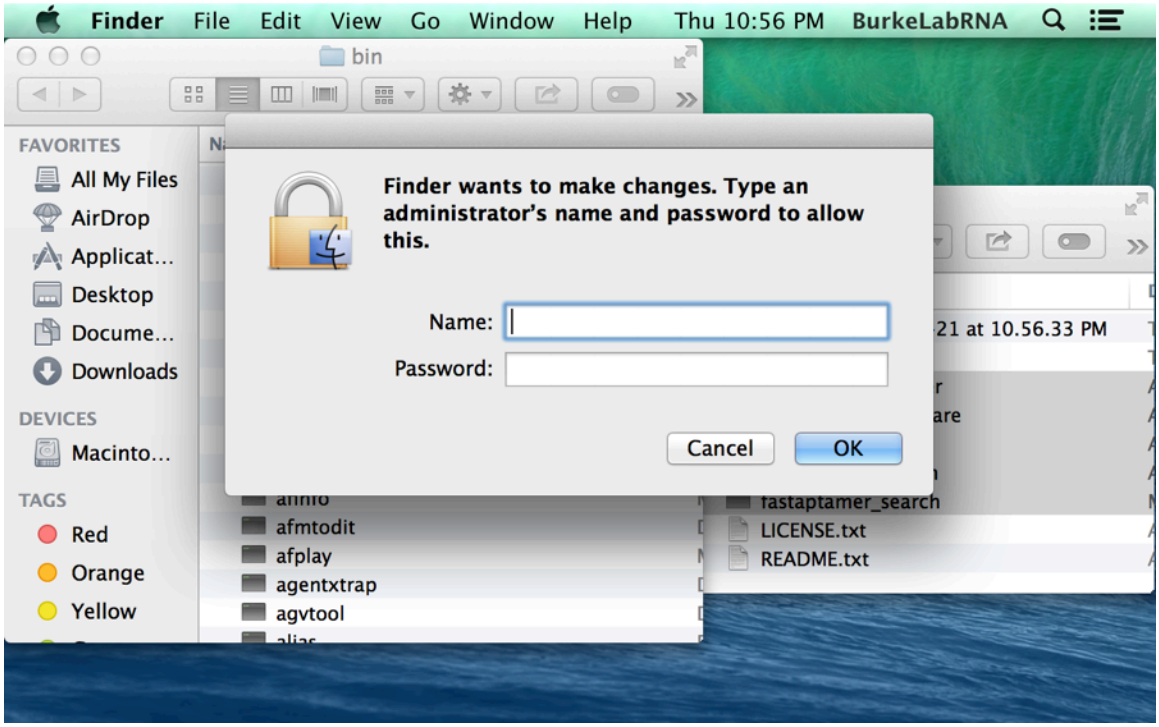


Copy or move the scripts into the executable directory.

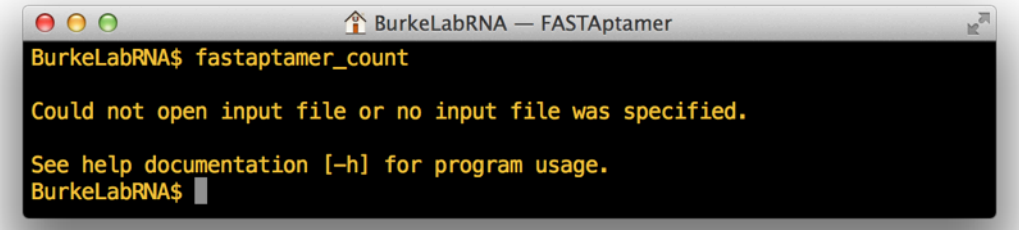


2c. Installing as Executable

If necessary, enter an administrator password to complete the installation.



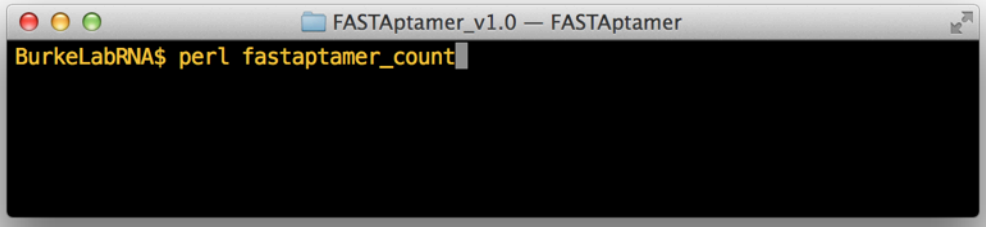
At this point, the toolkit should be ready to use and executable from anywhere in your directory. To test the installation, try to call up FASTAptamer-Count in the terminal by typing `fastaptamer_count` at the prompt and hitting enter.



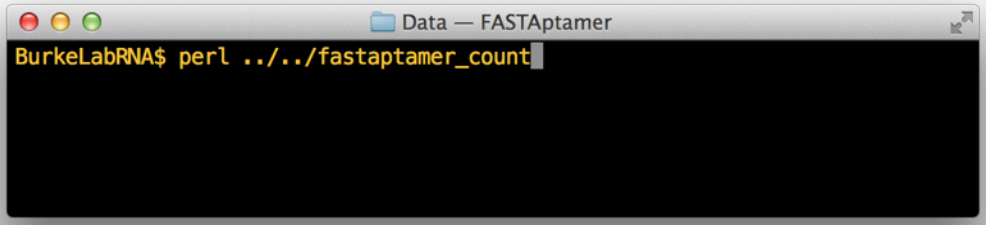
If you see an error message similar to the one above then you're ready to proceed to the tutorial in **section 3**. If you're having issues with the installation, try using FASTAptamer without installation.

2d. Use Without Installation

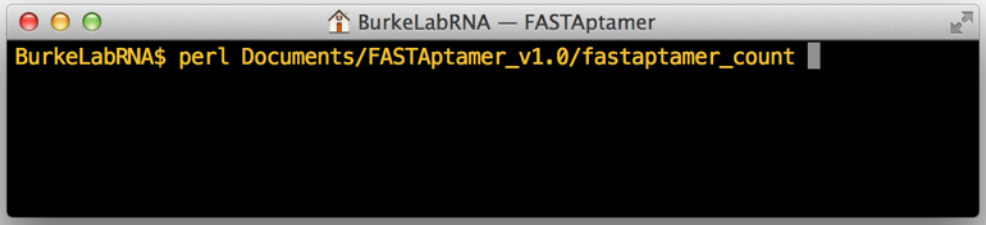
An easier, yet inelegant, solution to using FASTAptamer is by creating a folder where you'll be doing your data analysis and copying the scripts in that folder. Rather than entering the name of the script in the command line prompt, you'll have to first enter `perl`, followed by the script you wish to use.



If you navigate out of the directory containing the scripts you'll have to enter the relative path of the script so that Perl can find it. For example, if you're within a subdirectory of where the scripts are located you may have to enter something like the following:



Alternatively, if you're in a parent directory you'll have to enter something like this:



You should now be ready to use the FASTAptamer toolkit. If you're still having issues installing or using the software, feel free to contact us with a thorough description of what you've tried and what's happening.

3. Tutorial

The tutorials that follow are intended to get you familiar with FASTAptamer and the various options provided by the scripts. As a convenience, we've provided two quick reference tables (**section 4**) to refer to. We've also included a help screen for each script that can be invoked by using the `-h` command.

3a. Data Requirements & Pre-Processing

FASTAptamer-Count, the gateway to the toolkit, requires input files in the FASTQ format, the *de facto* standard for high-throughput DNA sequence files. SAM, BAM, or other file formats are not currently supported. If your sequencing information is, or will be, in one of these other formats, contact your sequencing provider and request FASTQ files, or use one of the several utilities listed in the resources (**section 4c**) to convert your files to FASTQ.

Although FASTAptamer-Count will accept any raw FASTQ file, it is prudent to ensure that the input file itself undergoes some level of pre-processing. Typically this involves the removal of 5' and/or 3' constant regions ("trimming") and quality filtering for only those reads whose bases have been called with high confidence. Several pre-processing tools are listed in the resources (**section 4c**).

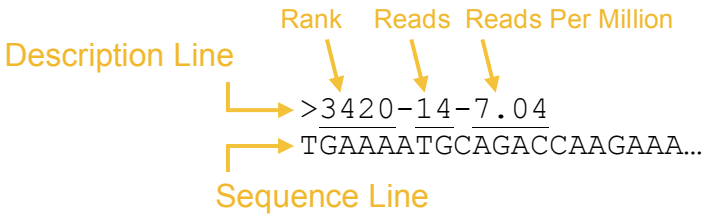
3b. Sample Data

Sample data for FASTAptamer can be downloaded as compressed FASTQ files from the Burke Lab website at http://burkelab.missouri.edu/fastapamer.html and through our GitHub site http://github.com/FASTAptamer.

The two population files (70HRT14.fastq.zip and 70HRT15.fastq.zip) are already pre-processed (trimmed and filtered) and can be used directly in FASTAptamer after decompression ("unzipping").

3c. FASTAptamer-Count

FASTAptamer-Count determines the abundance of each sequence in a population file. It also normalizes the reads for each sequence to RPM (reads per million), sorts by decreasing abundance, and rank sequences before sending the information to output. The FASTA formatted output file exploits the FASTA format by populating the description line with the rank, reads, and RPM of each unique sequence.



All data must first be processed through FASTAptamer-Count to generate a non-redundant FASTA file for use throughout the toolkit. Assuming you have already decompressed the sample data file and that the files are located in our current directory, let's process both rounds using FASTAptamer-Count.

The command for FASTAptamer-Count is `fastaptamer_count`.

Before we begin, recall that for all FASTAptamer scripts we can always call up a help screen using `-help` (or `-h`) to review requirements and options.

```
BurkeLabRNA$ fastaptamer_count -h

-------------------------------------------------------------------------------
                              FASTAptamer-Count
-------------------------------------------------------------------------------

Usage: fastaptamer_count [-h] [-q] [-i INFILE] [-o OUTFILE]

    [-h]            = Help screen.
    [-q]            = Suppress STDOUT of run report.
    [-i INFILE]     = FASTQ input file.
    [-o OUTFILE]    = FASTA output file.

FASTAptamer-Count serves as the gateway to the FASTAptamer toolkit.  For a given
.FASTQ input file, FASTAptamer-Count will determine the number of times each se-
quence was read, rank and sort sequences by decreasing total reads, and normali-
ze sequence frequency to reads per million. Output is generated as a non-redund-
ant FASTA file in the following format:

        >RANK-READS-RPM
        SEQUENCE

Summary report (total reads, unique reads, and execution time) is displayed as
STDOUT at program completion, unless [-q] is invoked.

BurkeLabRNA$
```

3c. FASTAptamer-Count

From the help screen we should be able to deduce that FASTAptamer-Count requires an input file in FASTQ format.

We'll begin with the 70HRT14.fastq file (which should already be "unzipped"). We can specify this file by using the `-i` flag. Our command should read:

```
fastaptamer_count -i 70HRT14.fastq
```

We'll also need to specify an output file using the `-o` flag. For simplicity sake we'll append _count to the current file name. Remember that the output file will now be in FASTA format.

```
fastaptamer_count -i 70HRT14.fastq -o 70HRT14_count.fasta
```

Execute the command. Your output should look like this:

```
BurkeLabRNA$ fastaptamer_count -i 70HRT14.fastq -o 70HRT14_count.fasta

2160216 total sequences.
72921 unique sequences.

Input file: "70HRT14.fastq".
Output file: "70HRT14_count.fasta".
Execution time: 12 s.

BurkeLabRNA$
```

You'll notice that FASTAptamer-Count provides a summary report showing the number of sequence entries in the FASTQ file, as well as the number of non-redundant entries created by FASTAptamer-Count.

This summary report can be suppressed by invoking the option `-q` on the command line prior to execution. Just like the help screen, the ability to suppress summary reports using the same command remains true for all FASTAptamer scripts.

For the next tutorials we'll need a FASTAptamer-Count file for the 70HRT15 population. We'll be able to use these two population files to compare the sequence distribution, cluster into sequence families, calculate fold-enrichment across the populations and search for the presence of sequence motifs.

Repeat these steps to generate a FASTAptamer-Count file called 70HRT15_count.fasta.

3d. FASTAptamer-Compare

FASTAptamer-Compare compares the sequence distribution between two populations files by generating a plain-text file with tab-separated values (.TSV) of each sequence present in both populations, along with their respective reads per million (RPM) and the calculated binary logarithm for the RPM (for each sequence, Log_2 RPM_y/RPM_x). FASTAptamer-Compare also facilitates the creation of a histogram by taking those binary logarithm values and calculating the number of times each value falls within one of its 102 bin buckets.

The command for FASTAptamer-Compare is `fastaptamer_compare`.

Pull up the help screen for FASTAptamer-Compare and review the usage and options provided by the script.

```
fastaptamer_compare -h
```

We'll need to specify two input files, called x and y, using the `-i` flag. We'll use our counted files generated using FASTAptamer-Count.

```
fastaptamer_compare -x 70HRT14_count.fasta -y 70HRT15_count.fasta
```

Before we execute the command, we'll have to specify where we want our output file to go and what we want to call it (using `-o`). For this example, let's call it 70HRT14_vs_15_compare.tsv.

```
fastaptamer_compare -x 70HRT14_count.fasta -y 70HRT15_count.fasta -o
70HRT14_vs_15_compare.tsv
```

Execute the command. Your output should look like this:

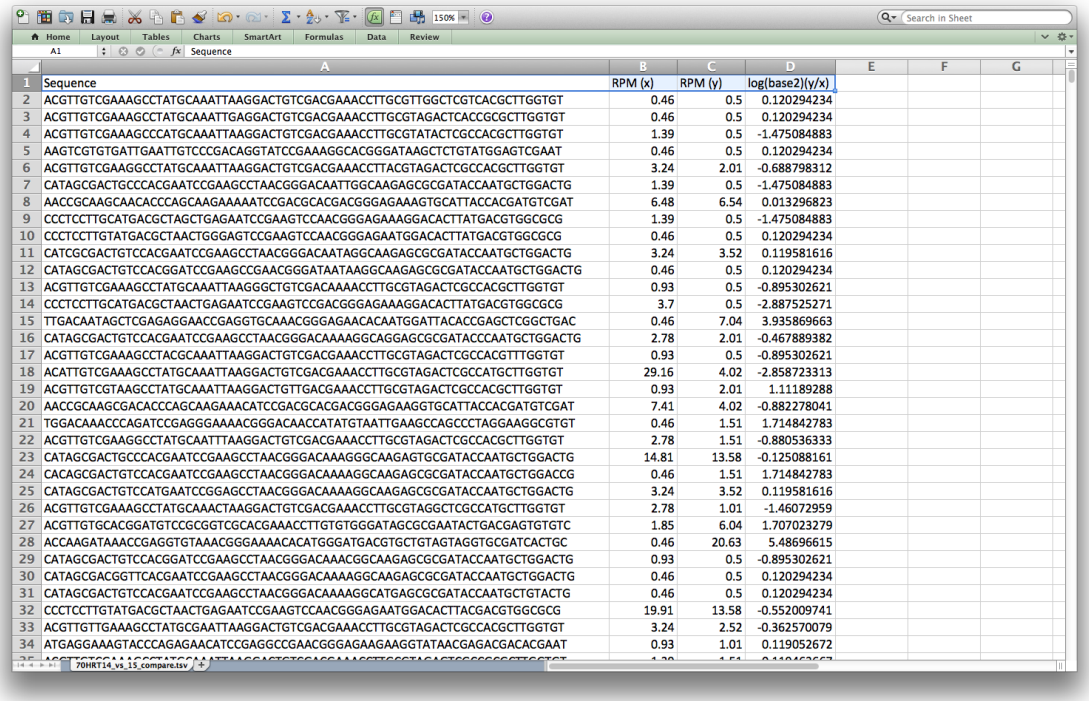
```
BurkeLabRNA$ fastaptamer_compare -x 70HRT14_count.fasta -y 70HRT15_count.fasta -o 70HRT1
4_vs_15_compare.tsv

Input file (x): "70HRT14_count.fasta".
Input file (y): - "70HRT15_count.fasta".
Output file: "70HRT14_vs_15_compare.tsv".
Execution time: 1 s.

BurkeLabRNA$
```

3d. FASTAptamer-Compare

FASTAptamer-Compare created a .TSV file that can be opened using any standard text editor or spreadsheet software.



	A	B	C	D
1	Sequence	RPM (x)	RPM (y)	log(base2)(y/x)
2	ACGTTGTCGAAAGCCTATGCAAATTAAGGACTGTCGACGAAACCTTGCGTTGGCTCGTCACGCTTGGTGT	0.46	0.5	0.120294234
3	ACGTTGTCGAAAGCCTATGCAAATTGAGGACTGTCGACGAAACCTTGCGTAGACTCACCGCGCTTGGTGT	0.46	0.5	0.120294234
4	ACGTTGTCGAAAGCCCATGCAAATTAAGGACTGTCGACGAAACCTTGCGTATACTCGCCACGCTTGGTGT	1.39	0.5	-1.475084883
5	AAGTCGTGTGATTGAATTGTCCCGACAGGTATCCGAAAGGCACGGGATAAGCTCTGTATGGAGTCGAAT	0.46	0.5	0.120294234
6	ACGTTGTCGAAGGCCTATGCAAATTAAGGACTGTCGACGAAACCTTACGTAGACTCGCCACGCTTGGTGT	3.24	2.01	-0.688798312
7	CATAGCGACTGCCCACGAATCCGAAGCCTAACGGGACAATTGGCAAGAGCGCGATACCAATGCTGGACTG	1.39	0.5	-1.475084883
8	AACCGCAAGCAACACCCAGCAAGAAAAATCCGACGCACGACGGGAGAAAGTGCATTACCACGATGTCGAT	6.48	6.54	0.013296823
9	CCCTCCTTGCATGACGCTAGCTGAGAATCCGAAGTCCAACGGGAGAAAGGACACTTATGACGTGGCGCG	1.39	0.5	-1.475084883
10	CCCTCCTTGTATGACGCTAACTGGGAGTCCGAAGTCCAACGGGAGAATGGACACTTATGACGTGGCGCG	0.46	0.5	0.120294234
11	CATCGCGACTGTCCACGAATCCGAAGCCTAACGGGACAATAGGCAAGAGCGCGATACCAATGCTGGACTG	3.24	3.52	0.119581616
12	CATAGCGACTGTCCACGGATCCGAAGCCGAACGGGATAATAAGGCAAGAGCGCGATACCAATGCTGGACTG	0.46	0.5	0.120294234
13	ACGTTGTCGAAAGCCTATGCAAATTAAGGGCTGTCGACAAAACCTTGCGTAGACTCGCCACGCTTGGTGT	0.93	0.5	-0.895302621
14	CCCTCCTTGCATGACGCTAACTGAGAATCCGAAGTCCGACGGGAGAAAGGACACTTATGACGTGGCGCG	3.7	0.5	-2.887525271
15	TTGACAATAGCTCGAGAGGAACCGAGGTGCAAACGGGAGAACACAATGGATTACACCGAGCTCGGCTGAC	0.46	7.04	3.935869663
16	CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAGGAGCGCGATACCCAATGCTGGACTG	2.78	2.01	-0.467889382
17	ACGTTGTCGAAAGCCTACGCAAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCACGTTTGGTGT	0.93	0.5	-0.895302621
18	ACATTGTCGAAAGCCTATGCAAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCATGCTTGGTGT	29.16	4.02	-2.858723313
19	ACGTTGTCGTAAGCCTATGCAAATTAAGGACTGTTGACGAAACCTTGCGTAGACTCGCCACGCTTGGTGT	0.93	2.01	1.11189288
20	AACCGCAAGCGACACCCAGCAAGAAACATCCGACGCACGACGGGAGAAGGTGCATTACCACGATGTCGAT	7.41	4.02	-0.882278041
21	TGGACAAACCCAGATCCGAGGGAAAACGGGACAACCATATGTAATTGAAGCCAGCCCTAGGAAGGCGTGT	0.46	1.51	1.714842783
22	ACGTTGTCGAAGGCCTATGCAATTTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCACGCTTGGTGT	2.78	1.51	-0.880536333
23	CATAGCGACTGCCCACGAATCCGAAGCCTAACGGGACAAAGGGCAAGAGTGCGATACCAATGCTGGACTG	14.81	13.58	-0.125088161
24	CACAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACCG	0.46	1.51	1.714842783
25	CATAGCGACTGTCCATGAATCCGGAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	3.24	3.52	0.119581616
26	ACGTTGTCGAAAGCCTATGCAAACTAAGGACTGTCGACGAAACCTTGCGTAGGCTCGCCATGCTTGGTGT	2.78	1.01	-1.46072959
27	ACGTTGTGCACGGATGTCCGCGGTCGCACGAAACCTTGTGTGGGATAGCGCGAATACTGACGAGTGTGTC	1.85	6.04	1.707023279
28	ACCAAGATAAACCGAGGTGTAAACGGGAAAACACATGGGATGACGTGCTGTAGTAGGTGCGATCACTGC	0.46	20.63	5.48696615
29	CATAGCGACTGTCCACGGATCCGAAGCCTAACGGGACAAACGGCAAGAGCGCGATACCAATGCTGGACTG	0.93	0.5	-0.895302621
30	CATAGCGACGGTTCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	0.46	0.5	0.120294234
31	CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCATGAGCGCGATACCAATGCTGTACTG	0.46	0.5	0.120294234
32	CCCTCCTTGTATGACGCTAACTGAGAATCCGAAGTCCAACGGGAGAATGGACACTTACGACGTGGCGCG	19.91	13.58	-0.552009741
33	ACGTTGTTGAAAGCCTATGCGAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCACGCTTGGTGT	3.24	2.52	-0.362570079
34	ATGAGGAAAGTACCCAGAGAACATCCGAGGCCGAACGGGAGAAGAAGGTATAACGAGACGACACGAAT	0.93	1.01	0.119052672

The histogram data is the last output generated and is available in the bottom-most rows of the document.

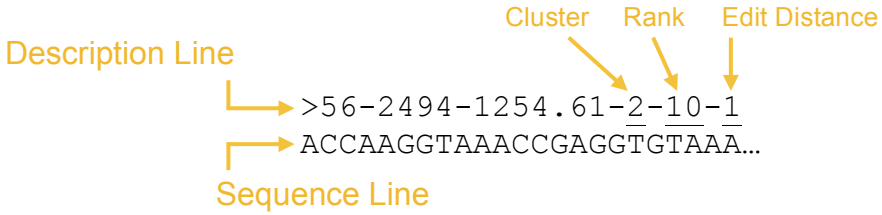
Recall that the default of FASTAptamer-Compare is to only output sequences that were present in BOTH population files. To output all the data, regardless of a match, invoke the option `-a` on the command line prior to execution. For these “unmatched” sequences, the output will leave out the RPM value for the population in which the sequence was not found. It will also not calculate the binary logarithm or send any additional values to the histogram bins for these sequences.

```
fastaptamer_compare -x 70HRT14_count.fasta -y 70HRT15_count.fasta -o
70HRT14_vs_15_compare.tsv -a
```

Lastly, the summary report listing the input and output files and execution time can be suppressed with the addition of `-q` to the command line.

3e. FASTAptamer-Cluster

FASTAptamer-Cluster can generate clusters of closely-related sequences using Levenshtein edit distance. The script preserves FASTA formatting and appends cluster identity information to the description line information provided by FASTAptamer-Count, including the cluster in which the sequence was grouped, the rank within that cluster (as determined by reads), and the edit distance from the "seed sequence" of the cluster.



The command for FASTAptamer-Cluster is `fastaptamer_cluster`.

If you call up the help screen (`-h`) you'll see that, similar to FASTAptamer-Count, we'll have to specify and input file (`-i`) and an output file (`-o`). The output file will remain in FASTA format, and for consistency's sake, we'll call the file 70HRT15_cluster.fasta (for the 70HRT15 population).

```
fastaptamer_cluster -i 70HRT15_count.fasta -o 70HRT15_cluster.fasta
```

We'll also have to specify the Levenshtein edit distance using the flag `-d`. Edit distance is the number of insertions, deletions, or substitutions required to transform one sequence string into another, for this reason, only integers can be used. For the 70HRT15 population we'll use an edit distance of 7.

```
fastaptamer_cluster -i 70HRT15_count.fasta -o 70HRT15_cluster.fasta -d 7
```

Clustering is a slow process and can take tens of hours for diverse populations. For our 70HRT14 population, if we were only interested in clustering sequences with an RPM ≥ 100, we can apply the filtering option using the command line option `-f`.

```
fastaptamer_cluster -i 70HRT15_count.fasta -o 70HRT15_cluster.fasta -d 7
-f 100
```

Execute the command.

3e. FASTAptamer-Cluster

```
BurkeLabRNA$ fastaptamer_cluster -i 70HRT14_count.fasta -d 7 -f 100 -o 70HRT14_cluster.f
asta

**************************************************
Clustering is a computationally intense process.
Please be patient as clustering can take several
hours for a single file.
**************************************************

Total number of sequences clustered: 518.

Cluster Unique Sequences        Reads   RPM
1       173     728636  337297.78
2       150     606204  280621.96
3       73      241127  111621.72
4       38      90011   41667.6
5       24      61551   28492.98
6       20      56698   26246.44
7       6       34223   15842.4
8       18      37399   17312.63
9       3       6697    3100.15
10      1       5369    2485.40
11      1       1975    914.26
```

You'll notice that FASTAptamer-Cluster provides additional information on the cluster size (in terms of unique sequences and total reads and RPM) as it finishes each cluster. This can be suppressed using the (-q) flag, or you may elect to 'redirect' the output to a new file using the standard Unix redirection command (>). This redirected output file will contain the cluster statistics in a tab-delimited format and can be used to create graphs of cluster sizes, useful for making informed decisions on which clusters to analyze further.

```
fastaptamer_cluster -i 70HRT15_count.fasta -o 70HRT15_cluster.fasta -d 7
-f 100 > 70HRT15_cluster_sizes.tsv
```

3f. FASTAptamer-Enrich

FASTAptamer-Enrich calculates fold-enrichment for individual sequences across populations. The script generates a plain-text file with tab-separated values for use in any standard spreadsheet software. Output information contains each sequence, the sequence length, the rank, reads and RPM for each sequence and population the sequence was detected in and cluster information (if available). Lastly, the output file contains the fold-enrichment ratio for each possible pairwise comparison (y/x, z/y and z/x).

FASTAptamer-Enrich requires two files, but can process up to 3 files simultaneously. Each file must come from either FASTAptamer-Count or FASTAptamer-Cluster. FASTAptamer-Enrich will adjust output accordingly to accommodate for populations with cluster information.

The command for FASTAptamer-Enrich is `fastaptamer_enrich`.

If you call up the help screen (`-h`) you'll notice that the input files will be designated using the flags (`-x`,`-y`, and an optional third file `-z`).

Let's provide FASTAptamer-Enrich with a 70HRT14 population file from FASTAptamer-Count and a FASTAptamer-Cluster file for the 70HRT15 population.

```
fastaptamer_enrich -x 70HRT14_count.fasta -y 70HRT15_cluster.fasta
```

We'll also have to specify a name and location for the output file (`-o`). We'll keep this file in the current directory and call it 70HRT14_vs_15_enrich.tsv.

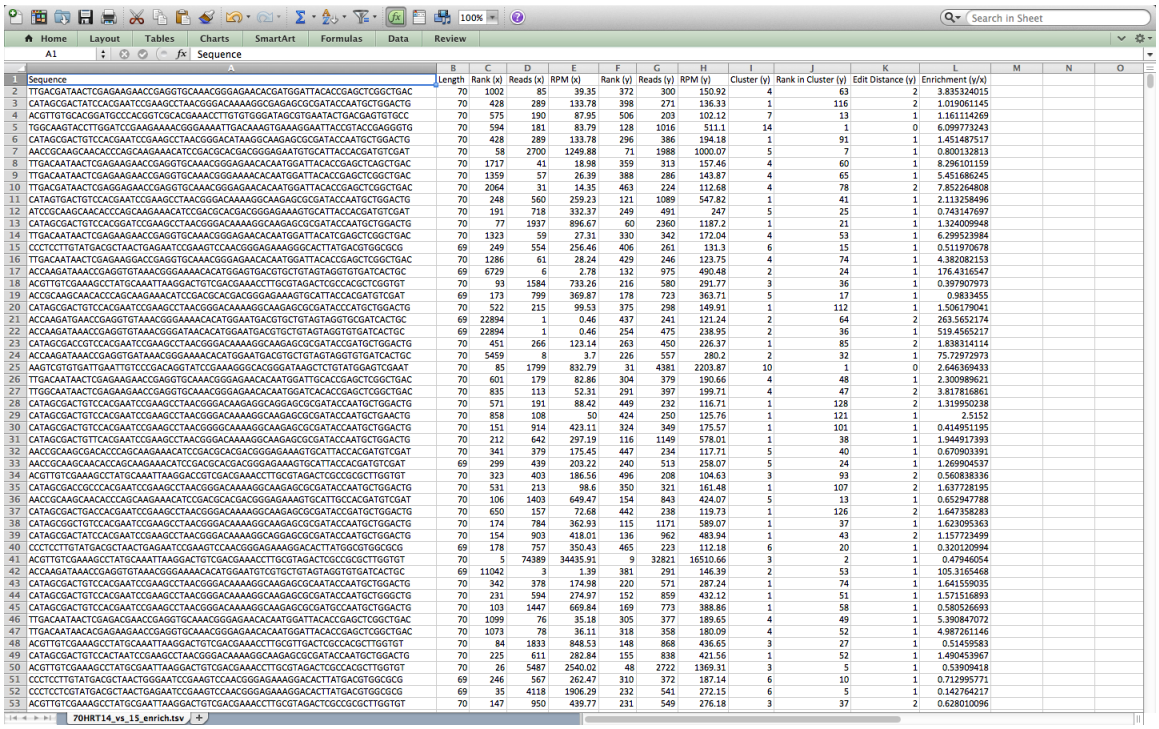
```
BurkeLabRNA$ fastaptamer_enrich -x 70HRT14_count.fasta -y 70HRT15_cluster.fasta -o 70HRT
14_vs_15_enrich.tsv

72921 sequences in "70HRT14_count.fasta".
516 sequences in "70HRT15_cluster.fasta".
Input file (x): "70HRT14_count.fasta".
Input file (y): - "70HRT15_cluster.fasta".
Output file: "70HRT14_vs_15_enrich.tsv".
Execution time: 1 s.

BurkeLabRNA$
```

3f. FASTAptamer-Enrich

We can open up this file in our favorite spreadsheet software and sort by metrics such as the rank in the final population or by overall enrichment. We have found that data from this step is useful for identifying candidate molecules for further investigation.



Sequence	Length	Rank (x)	Reads (x)	RPM (x)	Rank (y)	Reads (y)	RPM (y)	Cluster (y)	Rank in Cluster (y)	Edit Distance (y)	Enrichment (y/x)
TTGACGATAACTCGAGAAGAACCGAGGTGCAAACGGGAGAACACGATGGATTACACCGAGCTCGGCTGAC	70	1002	85	39.35	372	300	150.92	4	63	2	3.835324015
CATAGCGACTATCCACGAATCCGAAGCCTAACGGGACAAAAGGCGAGAGCGCGATACCAATGCTGGACTG	70	428	289	133.78	398	271	136.33	1	116	2	1.019061145
ACGTTGTGCACGGATGCCCACGGTCGACACGAAACCTTGTGTGGGATAGCGTGAATACTGACGAGTGTGCC	70	575	190	87.95	506	203	102.12	7	13	1	1.161114269
TGGCAAGTACCTTGGATCCGAAGAAAACGGGAAAATTGACAAAGTGAAAGGAATTACCGTACCGAGGGTG	70	594	181	83.79	128	1016	511.1	14	1	0	6.099773243
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACATAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	428	289	133.78	296	386	194.18	1	91	1	1.451487517
AACCGCAAGCAACACCAGCAAGAAACATCCGACGCACGACGGGAGAATGTGCATTACCACGATGTCGAT	70	58	2700	1249.88	71	1988	1000.07	5	7	1	0.800132813
TTGACAATAACTCGAGAAGAACCGAGGTGCAAACGGGAGAACACAATGGATTACACCGAGCTCGGCTGAC	70	1717	41	18.98	359	313	157.46	4	60	1	8.296101159
TTGACAATAACTCGAGAAGAACCGAGGTGCAAACGGGAAAACACAATGGATTACACCGAGCTCGGCTGAC	70	1359	57	26.39	388	286	143.87	4	65	1	5.451686245
TTGACGATAACTCGAGGAGAACCGAGGTGCAAACGGGAGAACACAATGGATTACACCGAGCTCGGCTGAC	70	2064	31	14.35	463	224	112.68	4	78	2	7.852264808
CATAGTGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	248	560	259.23	121	1089	547.82	1	41	1	2.113258496
ATCCGCAAGCAACACCCAGCAAGAAACATCCGACGCACGACGGGAGAAAGTGCATTACCACGATGTCGAT	70	191	718	332.37	249	491	247	5	25	1	0.743147697
CATAGCGACTGTCCACGGATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	77	1937	896.67	60	2360	1187.2	1	21	1	1.324009948
TTGACAATAACTCGAGAAGAACCGAGGTGCAAACGGGAGAACACAATGGATTACATCGAGCTCGGCTGAC	70	1323	59	27.31	330	342	172.04	4	53	1	6.299523984
CCCTCCTTGTATGACGCTAACTGAGAATCCGAAGTCCAACGGGAGAAAGGGCACTTATGACGTGGCGCG	69	249	554	256.46	406	261	131.3	6	15	1	0.511970678
TTGACAATAACTCGAGAAGGACCGAGGTGCAAACGGGAGAACACAATGGATTACACCGAGCTCGGCTGAC	70	1286	61	28.24	429	246	123.75	4	74	1	4.382082153
ACCAAGATAAACCGAGGTGTAAACGGGAAAACACATGGAGTGACGTGCTGTAGTAGGTGTGATCACTGC	69	6729	6	2.78	132	975	490.48	2	24	1	176.4316547
ACGTTGTCGAAAGCCTATGCAAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCACGCTCGGTGT	70	93	1584	733.26	216	580	291.77	3	36	1	0.397907973
ACCGCAAGCAACACCCAGCAAGAAACATCCGACGCACGACGGGAGAAAGTGCATTACCACGATGTCGAT	69	173	799	369.87	178	723	363.71	5	17	1	0.9833455
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCCATGCTGGACTG	70	522	215	99.53	375	298	149.91	1	112	1	1.506179041
ACCAAGATGAACCGAGGTGTAAACGGGAAAACACATGGAATGACGTGCTGTAGTAGGTGCGATCACTGC	69	22894	1	0.46	437	241	121.24	2	64	2	263.5652174
ACCAAGATAAACCGAGGTGTAAACGGGATAACACATGGAATGACGTGCTGTAGTAGGTGTGATCACTGC	69	22894	1	0.46	254	475	238.95	2	36	1	519.4565217
CATAGCGACCGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCGATGCTGGACTG	70	451	266	123.14	263	450	226.37	1	85	2	1.838314114
ACCAAGATAAACCGAGGTGATAAACGGGAAAACACATGGAATGACGTGCTGTAGTAGGTGTGATCACTGC	70	5459	8	3.7	226	557	280.2	2	32	1	75.72972973
AAGTCGTGTGATTGAATTGTCCCGACAGGTATCCGAAAGGGCACGGGATAAGCTCTGTATGGAGTCGAAT	70	85	1799	832.79	31	4381	2203.87	10	1	0	2.646369433
TTGACAATAACTCGAGAAGAACCGAGGTGCAAACGGGAGAACACAATGGATTGCACCGAGCTCGGCTGAC	70	601	179	82.86	304	379	190.66	4	48	1	2.300989621
TTGGCAATAACTCGAGAAGAACCGAGGTGCAAACGGGAGAACACAATGGATCACACCGAGCTCGGCTGAC	70	835	113	52.31	291	397	199.71	4	47	2	3.817816861
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAGAGGCAGGAGCGCGATACCAATGCTGGACTG	70	571	191	88.42	449	232	116.71	1	128	2	1.319950238
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGAACTG	70	858	108	50	424	250	125.76	1	121	1	2.5152
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGGCAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	151	914	423.11	324	349	175.57	1	101	1	0.414951195
CATAGCGACTGTTCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	212	642	297.19	116	1149	578.01	1	38	1	1.944917393
AACCGCAAGCGACACCCAGCAAGAAACATCCGACGCACGACGGGAGAAAGTGCATTACCACGATGTCGAT	70	341	379	175.45	447	234	117.71	5	40	1	0.670903391
AACCGCAAGCAACACCAGCAAGAAACATCCGACGCACGACGGGAGAAAGTGCATTACCACGATGTCGAT	69	299	439	203.22	240	513	258.07	5	24	1	1.269904537
ACGTTGTCGAAAGCCTATGCAAATTAAGGACCGTCGACGAAACCTTGCGTAGACTCGCCGCGCTTGGTGT	70	323	403	186.56	496	208	104.63	3	93	2	0.560838336
CATAGCGACCGCCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	531	213	98.6	350	321	161.48	1	107	2	1.637728195
AACCGCAAGCAACACCCAGCAAGAAACATCCGACGCACGACGGGAGAAAGTGCATTGCCACGATGTCGAT	70	106	1403	649.47	154	843	424.07	5	13	1	0.652947788
CATAGCGACTGACCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCGATGCTGGACTG	70	650	157	72.68	442	238	119.73	1	126	2	1.647358283
CATAGCGGCTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	174	784	362.93	115	1171	589.07	1	37	1	1.623095363
CATAGCGACTATCCACGAATCCGAAGCCTAACGGGACAAAAGGCAGGAGCGCGATACCAATGCTGGACTG	70	154	903	418.01	136	962	483.94	1	43	2	1.157723499
CCCTCCTTGTATGACGCTAACTGAGAATCCGAAGTCCAACGGGAGAAAGGACACTTATGGCGTGGCGCG	69	178	757	350.43	465	223	112.18	6	20	1	0.320120994
ACGTTGTCGAAAGCCTATGCAAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCGCGCTTGGTGT	70	5	74389	34435.91	9	32821	16510.66	3	2	1	0.47946054
ACCAAGATAAACCGAGGTGTAAACGGGAAAACACATGGAATGACGTGCTGTAGTAGGTGTGATCACTGC	69	11042	3	1.39	381	291	146.39	2	53	1	105.3165468
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATACCAATGCTGGACTG	70	342	378	174.98	220	571	287.24	1	74	1	1.641559035
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGGCTG	70	231	594	274.97	152	859	432.12	1	51	1	1.571516893
CATAGCGACTGTCCACGAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATGCCAATGCTGGACTG	70	103	1447	669.84	169	773	388.86	1	58	1	0.580526693
TTGACAATAACTCGAGACGAACCGAGGTGCAAACGGGAGAACACAATGGATTACACCGAGCTCGGCTGAC	70	1099	76	35.18	305	377	189.65	4	49	1	5.390847072
TTGACAATAACACGAGAAGAACCGAGGTGCAAACGGGAGAACACAATGGATTACACCGAGCTCGGCTGAC	70	1073	78	36.11	318	358	180.09	4	52	1	4.987261146
ACGTTGTCGAAAGCCTATGCAAATTAAGGACTGTCGACGAAACCTTGCGTTGACTCGCCACGCTTGGTGT	70	84	1833	848.53	148	868	436.65	3	27	1	0.51459583
CATAGCGACTGTCCACTAATCCGAAGCCTAACGGGACAAAAGGCAAGAGCGCGATACCAATGCTGGACTG	70	225	611	282.84	155	838	421.56	1	52	1	1.490453967
ACGTTGTCGAAAGCCTATGCGAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCACGCTTGGTGT	70	26	5487	2540.02	48	2722	1369.31	3	5	1	0.53909418
CCCTCCTTGTATGACGCTAACTGGGAATCCGAAGTCCAACGGGAGAAAGGACACTTATGACGTGGCGCG	69	246	567	262.47	310	372	187.14	6	10	1	0.712995771
CCCTCCTCGTATGACGCTAACTGAGAATCCGAAGTCCAACGGGAGAAAGGACACTTATGACGTGGCGCG	69	35	4118	1906.29	232	541	272.15	6	5	1	0.142764217
ACGTTGTCGAAAGCCTATGCGAATTAAGGACTGTCGACGAAACCTTGCGTAGACTCGCCGCGCTTGGTGT	70	147	950	439.77	231	549	276.18	3	37	2	0.628010096

FASTAptamer-Enrich also provides a tool to filter output to only a subset of sequences that were highly sampled. To invoke this command use the command line flag `-f` and specify a RPM value that sequences must meet or exceed to get sent to output. The script will tally the RPM values for each sequence across all populations it was present in to determine whether the criterion is being met.

Like all FASTAptamer scripts, the summary report can be suppressed by invoking `-q` at the command line prior to execution.

3g. FASTAptamer-Search

FASTAptamer-Search is a script that allows for degenerate motif searches using IUPAC-IUBMB single nucleotide codes. Keep in mind that the use of T and U are interchangeable.

A/T/G/C/U	single bases
R	puRines (A/G)
Y	pYrimidines (C/T)
W	Weak (A/T)
S	Strong (G/C)
M	aMino (A/C)
K	Keto (G/T)
B	not A
D	not C
H	not G
V	not T
N	aNy base (not a gap)

With FASTAptamer-Search you can search for the co-occurrence of more than one motif and across multiple files. FASTAptamer-Search will generate an output file containing only those sequences that matched the patterns.

The command for FASTAptamer-Search is `fastaptamer_search`.

A notable difference between this script and the others in the toolkit is that the help screen requires the use of the full command line flag `-help`, to avoid ambiguity with the `-highlight` flag which allows us to place parentheses around matched patterns for easy visualization.

For the both the 70HRT14 and 70HRT15 clustered population, we'll search for the presence of the dominant family 1 pseudoknot motif. This motif contains two patterns, UCCG and CGGGANAA. For each input file we'll have to use an input flag (`-i`), and for each pattern we'll have to use a pattern flag (`-p`).

```
fastaptamer_search -i 70HRT14_cluster.fasta -i 70HRT15_cluster.fasta
-p UCCG -p CGGANAA -o f1pk_motif_search.fasta
```

Execute the search. If you find that the output is being displayed on screen, you'll be aware that the script can be used to quickly find a motif, or can create an output file of matches using the `-o` flag.

3g. FASTAptamer-Search

```
BurkeLabRNA$ fastaptamer_search -i 70HRT14_cluster.fasta -i 70HRT15_cluster.fasta -p UCC
G -p CGGGANAA -o f1pk_motif_search.fasta


--------------------------------------------------------------------------------
>> SEARCH RESULT SUMMARY
--------------------------------------------------------------------------------

Searched for 2 patterns:
UCCG
CGGGANAA

across the following 2 input files:
70HRT14_cluster.fasta
70HRT15_cluster.fasta
Overall, 509 sequences were matched.

Your search took 0 seconds.

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

BurkeLabRNA$
```

Summary reports listing the number of matched sequences can be suppressed using the `-q` flag.

You may notice that the number of matched sequences seems rather low. If we revist the process that we took to get to these steps you'll notice that we're using clustered files, and that during the clustering process for this tutorial, we restricted our output to only those sequences that were sampled with 100 RPM or more.

4a. FASTAptamer Input/Output

Script	Input	Output
fastaptamer_count	FASTQ	FASTA
fastaptamer_compare	2 FASTA files (from FASTAptamer-Count)	Tab separated values plain text
fastaptamer_cluster	FASTA (from FASTAptamer-Count)	FASTA
fastaptamer_enrich	2 or 3 FASTA files (from FASTAptamer-Count or FASTAptamer-Cluster)	Tab separated values plain text
fastaptamer_search	FASTA	FASTA FASTA

FASTAptamer

4b. FASTAptamer Command Line Options

Script	Function	Command Line Flags	Interpretation
fastaptamer_count	Determines abundance of each sequence, normalizes value to total reads per million, ranks and sorts by decreasing abundance.	-i	Input file (.FASTQ)*
		-o	Output file (.FASTA)*
		-h	Help screen
		-q	Quiet mode - suppresses summary report
		-v	Display version
fastaptamer_compare	Calculates $\log_2$ values of RPM y/x, generates table containing RPM for each sequence in both files, generates and fills values for histogram of sequence distribution.	-x	Input file 1 (.FASTA from FASTAptamer-Count or FASTAptamer-Cluster)*
		-y	Input file 2 (.FASTA from FASTAptamer-Count or FASTAptamer-Cluster)*
		-o	Output file (.TSV)*
		-h	Help screen
		-a	Output all sequences
		-q	Quiet mode - suppresses summary report
		-v	Display version
fastaptamer_cluster	Generates sequence clusters based on a user-defined Levenshtein edit distance.	-i	Input file (.FASTA from FASTAptamer-Count)*
		-o	Output file (.FASTA)*
		-h	Help screen
		-d	Edit Distance*
		-f	Read filter
		-q	Quiet mode - suppresses summary report
		-v	Display version
fastaptamer_enrich	Calculates fold-enrichment values for each sequence in 2 or 3 populations.	-x	Input file 1 (.FASTA from FASTAptamer-Count or FASTAptamer-Cluster)*
		-y	Input file 2 (.FASTA from FASTAptamer-Count or FASTAptamer-Cluster)*
		-z	Input file 3 (optional - .FASTA from FASTAptamer-Count or FASTAptamer-Cluster)
		-o	Output file (.TSV)*
		-h	Help screen
		-f	RPM threshold filter
		-q	Quiet mode - suppresses summary report
		-v	Display version
fastaptamer_search	Degenerately searches for multiple sequence patterns across several files.	-i	Input files(s) (.FASTA from FASTAptamer-Count, FASTAptamer-Cluster or other)*
		-o	Output file (.FASTA)
		-p	Pattern(s) *
		-help	Help screen
		-highlight	Highlight matched motifs
		-q	Quiet mode - suppresses summary report
		-v	Display version

***Required**

4c. Resources and Links

Learning Command Line:

- Web Resources
  - **LinuxCommand.Org** (http://linuxcommand.org) is a highly recommended resource and includes a free downloadable PDF.
  - A pair of frequently recommended online tutorials can be found at http://ryanstutorials.net/linuxtutorial/ and http://cli.learncodethehardway.org.

- Books
  - UNIX and Perl to the Rescue!: A Field Guide for the Life Sciences (and Other Data-Rich Pursuits) by Keith Bradnam and Ian Korf.
  - Practical Computing for Biologists by Steven Haddock and Casey Dunn

Data pre-processing (trimming, filtering for quality, etc.):

- **cutadapt** (http://code.google.com/p/cutadapt/) - what we use to trim constant regions from our sequencing data.
- **FASTX-Toolkit** (http://hannonlab.cshl.edu/fastx_toolkit/) - our go to set of tools for quality analysis and filtering.

OMICtools (http://omictools.com/common-tools-c1219-p1.html) provides a long list of common software for high-throughput sequence analysis. Some other tools we keep coming across are listed below.

- **PRINSEQ** (http://prinseq.sourceforge.net)
- **FastQC** (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
- **SolexaQA** (http://solexaqa.sourceforge.net)
- **ea-utils** (http://code.google.com/p/ea-utils/)
- **Trimmomatic** (http://www.usadellab.org/cms/?page=trimmomatic)