AbMining ToolBox Installation and User Guide

Csaba Kiss

Oct 31, 2013

Contents

General comments

Before you install the software package, make sure that your system has all the required software installed and tested. The required software install guide will help you with the installation of the required packages for all popular operating systems.

http://sourceforge.net/projects/abmining/files/AbMTb_req_software.pdf/download

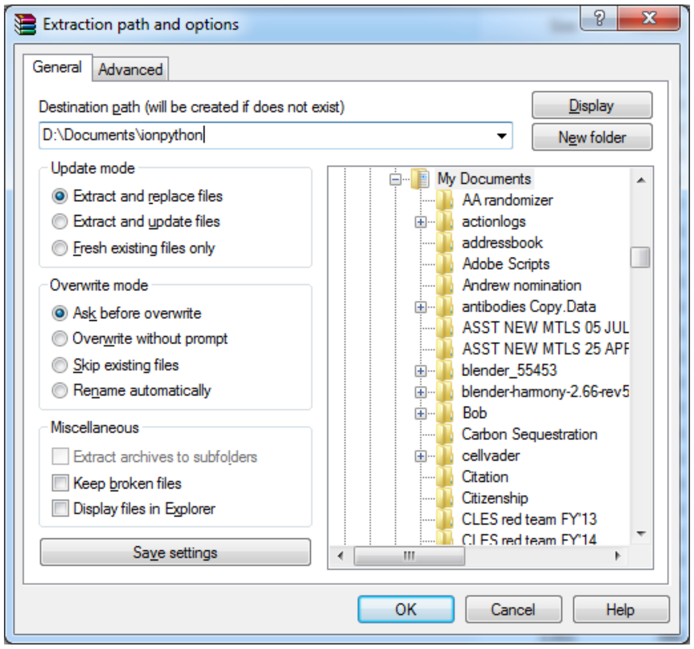
This document will demonstrate the installation and use of all the scripts included in the software package using two sequence sample files (sample1.fastq, sample2.fastq). An MS Windows 7 operating system will be used for the tutorial, but other systems could be used just as easily.

Installation of the AbMining ToolBox software package on Windows 7

Download the AbMining ToolBox.zip file to your computer by clicking on the link:

http://sourceforge.net/projects/abmining/files/AbMiningToolbox.zip/download

Create an arbitrary directory on your computer. For the demonstration I will create the directory in the **My Documents** folder called ionpython. Open the AbMining ToolBox.zip file and extract its contents into the AbMining ToolBox directory.



Destination path (will be created if does not exist)
D:\Documents\ionpython

Update mode
[✓ Extract and replace files]
Extract and update files
Fresh existing files only

Overwrite mode
[✓ Ask before overwrite]
Overwrite without prompt
Skip existing files
Rename automatically

Miscellaneous
Extract archives to subfolders
Keep broken files
Display files in Explorer

My Documents
AA randomizer
actionlogs
addressbook
Adobe Scripts
Andrew nomination
antibodies Copy.Data
ASST NEW MTLS 05 JUL
ASST NEW MTLS 25 APF
blender_55453
blender-harmony-2.66-rev5
Bob
Carbon Sequestration
cellvader
Citation
Citizenship
CLES red team FY'13
CLES red team FY'14

Figure 1: WinRar is used to extract all the AbMining ToolBox files into the chosen directory

Now open a Command prompt (terminal window on Mac OSX or linux) by typing

```
>⊞-R cmd ([Windows-key]-R) open the run window and type cmd in there.
```

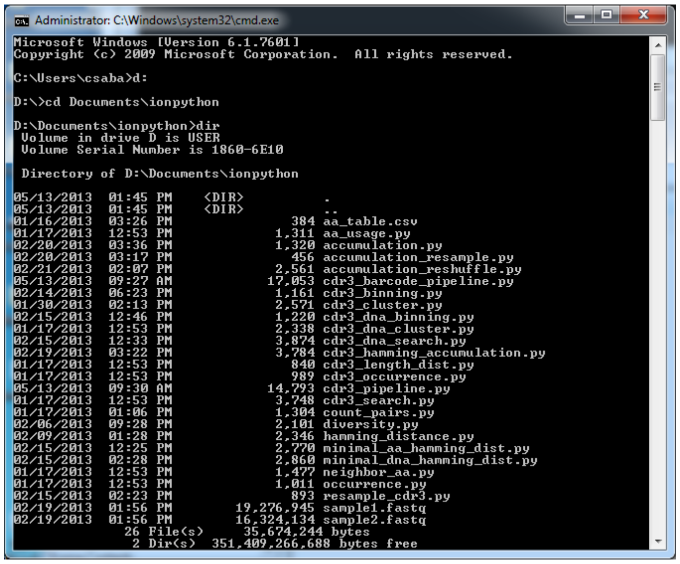
Navigate to the directory where you extracted the AbMining ToolBox script files

```
>d:
>cd Documents\ionpython
```

Check the contents of the directory

```
>dir
```

(ls -la for Mac OSX and linux)



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\csaba>d:

D:\>cd Documents\ionpython

D:\Documents\ionpython>dir
 Volume in drive D is USER
 Volume Serial Number is 1860-6E10

 Directory of D:\Documents\ionpython

05/13/2013  01:45 PM    <DIR>          .
05/13/2013  01:45 PM    <DIR>          ..
01/16/2013  03:26 PM               384 aa_table.csv
01/17/2013  12:53 PM             1,311 aa_usage.py
02/20/2013  03:36 PM             1,320 accumulation.py
02/20/2013  03:17 PM               456 accumulation_resample.py
02/21/2013  02:07 PM             2,561 accumulation_reshuffle.py
05/13/2013  09:27 AM            17,053 cdr3_barcode_pipeline.py
02/14/2013  06:23 PM             1,161 cdr3_binning.py
01/30/2013  02:13 PM             2,571 cdr3_cluster.py
02/15/2013  12:46 PM             1,220 cdr3_dna_binning.py
01/17/2013  12:53 PM             2,338 cdr3_dna_cluster.py
02/15/2013  12:33 PM             3,874 cdr3_dna_search.py
02/19/2013  03:22 PM             3,784 cdr3_hamming_accumulation.py
01/17/2013  12:53 PM               840 cdr3_length_dist.py
01/17/2013  12:53 PM               989 cdr3_occurrence.py
05/13/2013  09:30 AM            14,793 cdr3_pipeline.py
01/17/2013  12:53 PM             3,748 cdr3_search.py
01/17/2013  01:06 PM             1,304 count_pairs.py
02/06/2013  09:28 PM             2,101 diversity.py
02/09/2013  01:28 PM             2,346 hamming_distance.py
02/15/2013  12:25 PM             2,770 minimal_aa_hamming_dist.py
02/15/2013  02:28 PM             2,860 minimal_dna_hamming_dist.py
01/17/2013  12:53 PM             1,477 neighbor_aa.py
01/17/2013  12:53 PM             1,011 occurrence.py
02/15/2013  02:23 PM               893 resample_cdr3.py
02/19/2013  01:56 PM        19,276,945 sample1.fastq
02/19/2013  01:56 PM        16,324,134 sample2.fastq
              26 File(s)     35,674,244 bytes
               2 Dir(s)  351,409,266,688 bytes free
```

Figure 2: The AbMining ToolBox directory containing all the necessary files

Now you are ready to use the software.

Extra Mac and linux instructions

In order to use the scripts on a Mac or a linux system you need to give the python scripts executable rights and add the AbMining ToolBox to the system path. To do that, please follow these instructions. Open up a terminal window and navigate to the folder where you extracted the files from AbMining ToolBox.zip

```
>cd /Volumes/User/csaba/Documents/ionpython
```

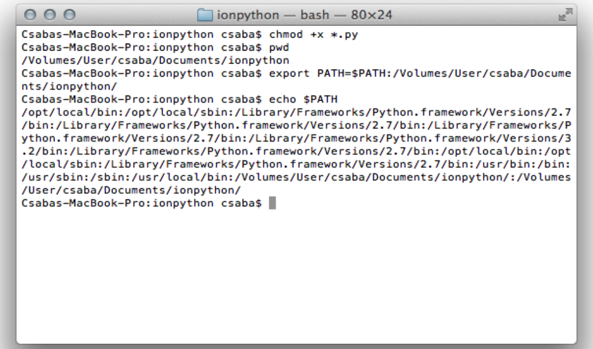
(your location will be different from the above line), then type

```
>chmod +x *.py
>pwd
```

This will print out the current working directory. That is the path that you need to add to your system path. Note it and type again

```
>export $PATH=$PATH:/ionpython
>echo $PATH
```

You should see the AbMining ToolBox folder added at the end of the list of folders. Your screen should look something similar to this:



```
Csabas-MacBook-Pro:ionpython csaba$ chmod +x *.py
Csabas-MacBook-Pro:ionpython csaba$ pwd
/Volumes/User/csaba/Documents/ionpython
Csabas-MacBook-Pro:ionpython csaba$ export PATH=$PATH:/Volumes/User/csaba/Docume
nts/ionpython/
Csabas-MacBook-Pro:ionpython csaba$ echo $PATH
/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/2.7
/bin:/Library/Frameworks/Python.framework/Versions/2.7/bin:/Library/Frameworks/P
ython.framework/Versions/2.7/bin:/Library/Frameworks/Python.framework/Versions/3
.2/bin:/Library/Frameworks/Python.framework/Versions/2.7/bin:/opt/local/bin:/opt
/local/sbin:/Library/Frameworks/Python.framework/Versions/2.7/bin:/usr/bin:/bin:
/usr/sbin:/sbin:/usr/local/bin:/Volumes/User/csaba/Documents/ionpython/:/Volumes
/User/csaba/Documents/ionpython/
Csabas-MacBook-Pro:ionpython csaba$ 
```

Figure 3: Adding the AbMining ToolBox directory to the system path.

If you need to add the folder permanently to your system path, please follow this guide:
http://blog.just2us.com/2011/05/setting-path-variable-in-mac-permanently/
Now you are ready to use the software on your computer.

User Guide of AbMining ToolBox

This guide will explain how to use each script in the software package using the provided sample sequence files. Sample1 and 2 contains 50,000 and 30,000 random antibody Ion Torrent raw sequence reads. The tutorial will follow a workflow that you might use in your laboratory. Follow the step-by-step instructions and later use whichever scripts you need in your analysis.

cdr3_pipeline.py

This is the main script for the analysis of antibody library deep sequencing data output from all three platforms (454, Ion Torrent, MiSeq). The script can use the raw sff file that 454 and Ion Torrent sequencers produce or the processed fastq files of Ion Torrent and Miseq. The script will ask for quality trimming settings. The default values that have been determined experimentally by us are recommended, but using different settings is encouraged. The script produces 6 files (The * indicates the sample file name containing the sequencing reads):

- The *.trim.fasta file contains the quality trimmed DNA sequences.
- The *.dna.cdr3 file contains the list of all the CDR3 encoding DNA sequences.
- The *.cdr3 file contains all the CDR3 peptide sequences.
- The bins_*.dna.cdr3.csv file contains the unique CDR3 DNA sequences with their abundances in a csv format.
- The bins_*.cdr3.csv contains the unique CDR3 peptide sequences with the corresponding occurrences in a csv format.
- The cdr3_(date).log file contains the log file of the screen output using the current date as the name.

Open a Command prompt (terminal window on Mac OSX or linux) by typing

```
>⊞-R cmd ([Windows-key]-R open the run window and type cmd in there.
```

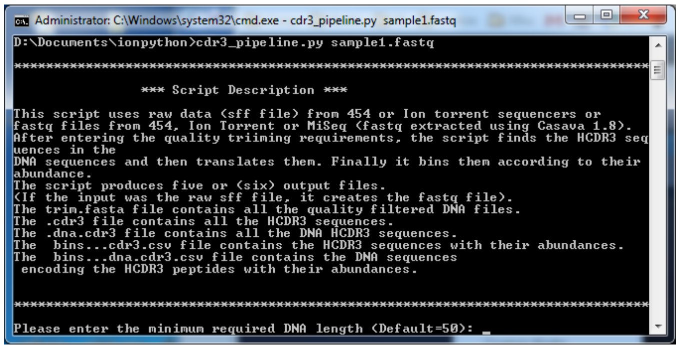
Navigate to the directory where you extracted the AbMining ToolBox script files

```
>d:
>cd Documents\abmining
```

Type the following command

```
>cdr3_pipeline.py sample1.fastq
```

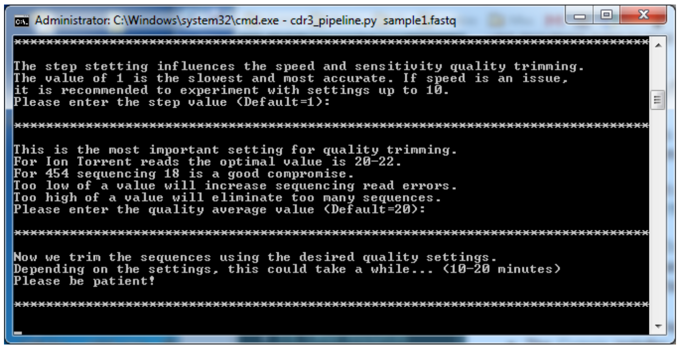
The script should start with a description and questions:



```
Administrator: C:\Windows\system32\cmd.exe - cdr3_pipeline.py  sample1.fastq
D:\Documents\ionpython>cdr3_pipeline.py sample1.fastq

********************************************************************************

                *** Script Description ***

This script uses raw data (sff file) from 454 or Ion torrent sequencers or
fastq files from 454, Ion Torrent or MiSeq (fastq extracted using Casava 1.8).
After entering the quality triiming requirements, the script finds the HCDR3 seq
uences in the
DNA sequences and then translates them. Finally it bins them according to their
abundance.
The script produces five or (six) output files.
(If the input was the raw sff file, it creates the fastq file).
The trim.fasta file contains all the quality filtered DNA files.
The .cdr3 file contains all the HCDR3 sequences.
The .dna.cdr3 file contains all the DNA HCDR3 sequences.
The  bins...cdr3.csv file contains the HCDR3 sequences with their abundances.
The  bins...dna.cdr3.csv file contains the DNA sequences
 encoding the HCDR3 peptides with their abundances.


********************************************************************************

Please enter the minimum required DNA length (Default=50): _
```

Figure 4: The cdr3_pipeline.py script startup screen.

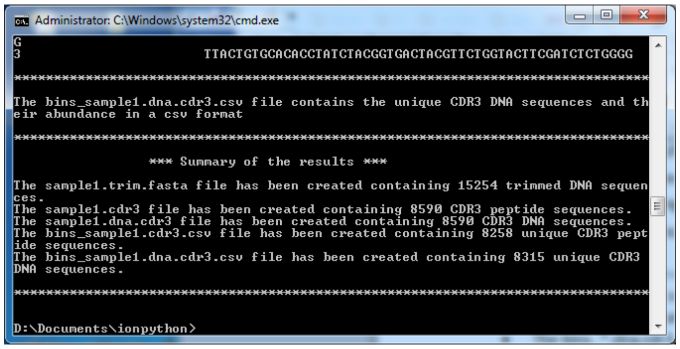
If you get an error, you should check the python documentation for help
http://docs.python.org/2/using/windows.html
The scripts will ask quality trimming questions. We recommend using the default values and just push Enter 5 times. Later, you are encouraged to experiment with various quality settings. The script will start running:



```
Administrator: C:\Windows\system32\cmd.exe - cdr3_pipeline.py  sample1.fastq
********************************************************************************

The step stetting influences the speed and sensitivity quality trimming.
The value of 1 is the slowest and most accurate. If speed is an issue,
it is recommended to experiment with settings up to 10.
Please enter the step value (Default=1):

********************************************************************************

This is the most important setting for quality trimming.
For Ion Torrent reads the optimal value is 20-22.
For 454 sequencing 18 is a good compromise.
Too low of a value will increase sequencing read errors.
Too high of a value will eliminate too many sequences.
Please enter the quality average value (Default=20):

********************************************************************************

Now we trim the sequences using the desired quality settings.
Depending on the settings, this could take a while... (10-20 minutes)
Please be patient!

********************************************************************************

_
```

Figure 5: The cdr3_pipeline.py script is running

At the end of the run you should see a similar screen as below



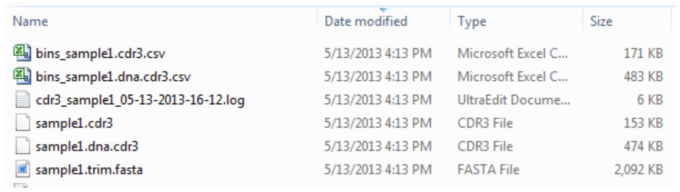
```
Administrator: C:\Windows\system32\cmd.exe
G
3                       TTACTGTGCACACCTATCTACGGTGACTACGTTCTGGTACTTCGATCTCTGGGG

********************************************************************************

The bins_sample1.dna.cdr3.csv file contains the unique CDR3 DNA sequences and th
eir abundance in a csv format

********************************************************************************

                 *** Summary of the results ***

The sample1.trim.fasta file has been created containing 15254 trimmed DNA sequen
ces.
The sample1.cdr3 file has been created containing 8590 CDR3 peptide sequences.
The sample1.dna.cdr3 file has been created containing 8590 CDR3 DNA sequences.
The bins_sample1.cdr3.csv file has been created containing 8258 unique CDR3 pept
ide sequences.
The bins_sample1.dna.cdr3.csv file has been created containing 8315 unique CDR3
DNA sequences.

********************************************************************************


D:\Documents\ionpython>
```

*Figure 6:*The end result of the cdr3_pipeline.py on sample1.fastq

Sample1 should produce a result of

- 15,254 trimmed DNA sequences
- 8,590 CDR3 peptide sequences
- 8,590 CDR3 DNA sequences
- 8,258 unique CDR3 peptide sequences
- 8,315 unique CDR3 DNA sequences.

There are also 6 new files in the AbMining ToolBox directory containing the sequences above, plus a log file logging the screen output into a file.



Name	Date modified	Type	Size
bins_sample1.cdr3.csv	5/13/2013 4:13 PM	Microsoft Excel C...	171 KB
bins_sample1.dna.cdr3.csv	5/13/2013 4:13 PM	Microsoft Excel C...	483 KB
cdr3_sample1_05-13-2013-16-12.log	5/13/2013 4:13 PM	UltraEdit Docume...	6 KB
sample1.cdr3	5/13/2013 4:13 PM	CDR3 File	153 KB
sample1.dna.cdr3	5/13/2013 4:13 PM	CDR3 File	474 KB
sample1.trim.fasta	5/13/2013 4:13 PM	FASTA File	2,092 KB

Figure 7: 6 new files have been created in the AbMining ToolBox directory

Repeat the process with the other sample file

```
>cdr3_pipeline.py sample2.fastq
```

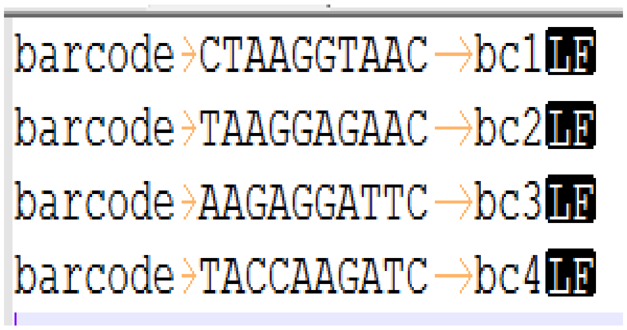
Sample2 should produce a result of

- 22,820 trimmed DNA sequences
- 10,513 CDR3 peptide sequences
- 10,513 CDR3 DNA sequences
- 9,795 unique CDR3 peptide sequences
- 9,902 unique CDR3 DNA sequences.

The output files can be imported into spreadsheet software (Excel) and analyzed further.

cdr3_barcode_pipeline.py

The barcode script is used mainly for antibody selections, when multiple samples are sequenced simultaneously. The script performs exactly the same functions as the `cdr3_pipeline.py` for each barcoded sample. The script will parse the raw sequences into separate files by the DNA barcodes with the sample names in the file names. The script will produce the same 5 files as the cdr3_pipeline.py script for each barcoded sample and a log file with the screen output. The script will ask for a barcode file containing the sample barcodes and the sample names in the following format:
`barcode[TAB]CTAAGGTAAC[TAB]sample_name`
see barcode.txt for an example



```
barcode	CTAAGGTAAC	bc1
barcode	TAAGGAGAAC	bc2
barcode	AAGAGGATTC	bc3
barcode	TACCAAGATC	bc4
```

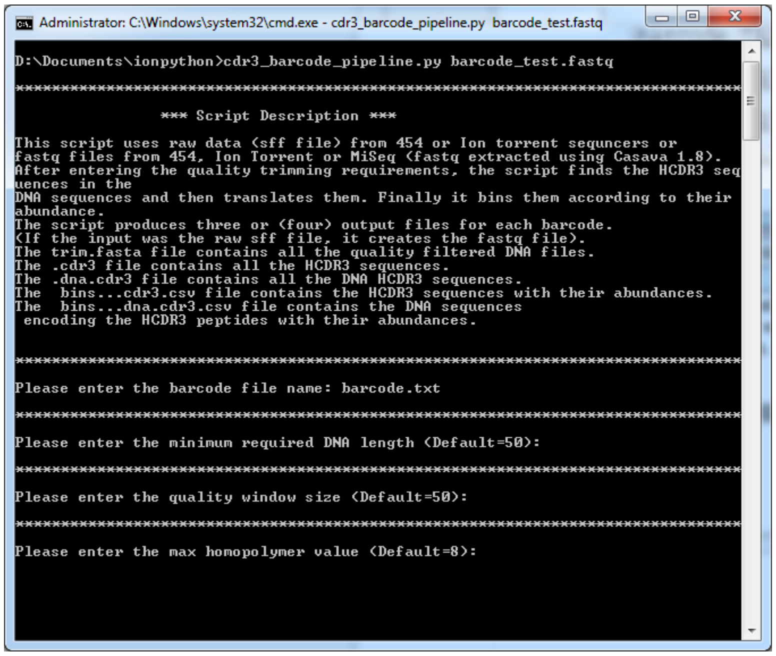
Figure 8: A sample barcode file

To test the script, type

```
>cdr3_barcode_pipeline.py barcode_test.fastq
```

when the script asks for the barcode file type

```
barcode.txt
```



Administrator: C:\Windows\system32\cmd.exe - cdr3_barcode_pipeline.py barcode_test.fastq

```
D:\Documents\ionpython>cdr3_barcode_pipeline.py barcode_test.fastq

********************************************************************************

                *** Script Description ***

This script uses raw data (sff file) from 454 or Ion torrent sequncers or
fastq files from 454, Ion Torrent or MiSeq (fastq extracted using Casava 1.8).
After entering the quality trimming requirements, the script finds the HCDR3 seq
uences in the
DNA sequences and then translates them. Finally it bins them according to their
abundance.
The script produces three or (four) output files for each barcode.
(If the input was the raw sff file, it creates the fastq file).
The trim.fasta file contains all the quality filtered DNA files.
The .cdr3 file contains all the HCDR3 sequences.
The .dna.cdr3 file contains all the DNA HCDR3 sequences.
The  bins...cdr3.csv file contains the HCDR3 sequences with their abundances.
The  bins...dna.cdr3.csv file contains the DNA sequences
 encoding the HCDR3 peptides with their abundances.


********************************************************************************

Please enter the barcode file name: barcode.txt

********************************************************************************

Please enter the minimum required DNA length (Default=50):

********************************************************************************

Please enter the quality window size (Default=50):

********************************************************************************

Please enter the max homopolymer value (Default=8):
```

Figure 9: The cdr3_barcode_pipeline.py begins to run

Then just keep hitting enter for the default value for quality trimming. The test barcode file contains 4 barcodes with sample names of bc1, bc2, bc3 and bc4. At the end of the run you will end up with 4 .trim.fasta, 4 .cdr3, 4 .dna.cdr3, 4 bins_...cdr3 files for each barcoded sample and also a barcode....log file containing all the information on the screen logged in a file.

cdr3_cluster.py

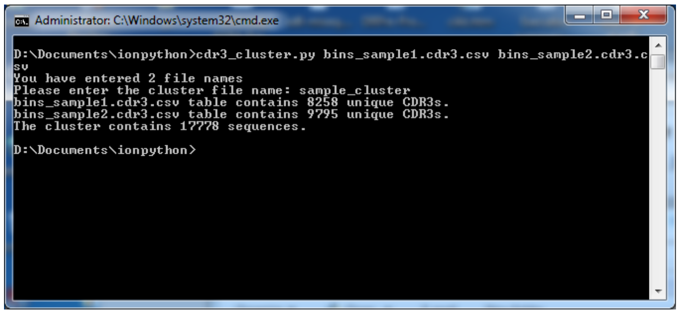
This script is used mainly for tracking selection outputs of antibodies. It requires the bins_.* input files produced by the `cdr3_pipeline.py` or the `cdr3_binning.py` scripts. It can use any number of input files. The script compares each sample file using the unique CDR3 sequences and creates a csv file containing the sequences with their number of occurrences in each sample.

```
>cdr3_cluster.py bins_sample1.cdr3 bins_sample2.cdr3
```

Give your cluster a name, say sample_cluster.

The script will create a csv file named `sample_cluster.csv`. The screen should look similar to the screenshot below.

The result of the clustering can be imported into a spreadsheet software for further analysis.



```
D:\Documents\ionpython>cdr3_cluster.py bins_sample1.cdr3.csv bins_sample2.cdr3.c
sv
You have entered 2 file names
Please enter the cluster file name: sample_cluster
bins_sample1.cdr3.csv table contains 8258 unique CDR3s.
bins_sample2.cdr3.csv table contains 9795 unique CDR3s.
The cluster contains 17778 sequences.

D:\Documents\ionpython>
```

Figure 10: The result of the cdr3_cluster.py script

cdr3_search.py

This script performs the CDR3 recognition part of the pipeline script. If you have a fasta file containing already processed sequences, it's quicker to run this script to search for CDR3s. The script produces one output file with a .cdr3 extension. Try it with this command

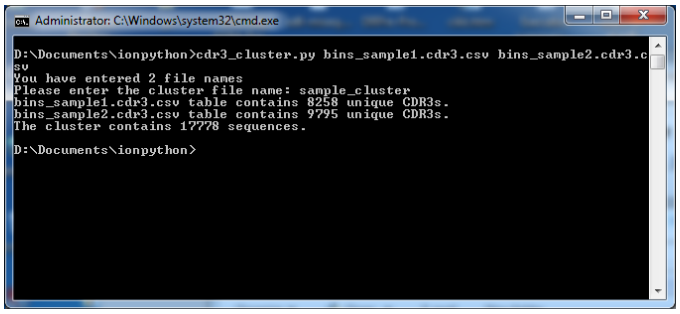
```
>cdr3_search.py sample1.trim.fasta
```

This will produce a file named `sample1.trim.fasta.cdr3` containing 8,590 CDR3s.

cdr3_binning.py

This script performs the binning part of the pipeline script using a text file containing CDR3 sequences in each line of the file. The script bins the unique CDR3s from the cdr3_search.py or the cdr3_pipeline.py script outputs (*.cdr3) and creates a csv file containing the unique sequences with their occurrences.

```
>cdr3_binning.py sample1.cdr3
```



```
D:\Documents\ionpython>cdr3_cluster.py bins_sample1.cdr3.csv bins_sample2.cdr3.c
sv
You have entered 2 file names
Please enter the cluster file name: sample_cluster
bins_sample1.cdr3.csv table contains 8258 unique CDR3s.
bins_sample2.cdr3.csv table contains 9795 unique CDR3s.
The cluster contains 17778 sequences.

D:\Documents\ionpython>
```

Figure 11: The result of the cdr3_cluster.py script

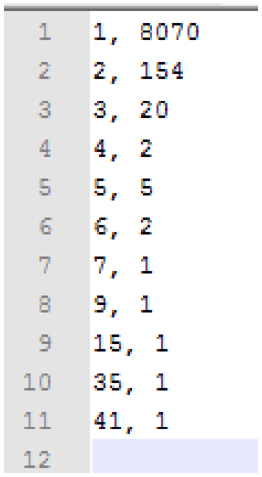
This will produce the `bins_sample1.cdr3` file containing 8,258 unique CDR3s.

cdr3_occurrence.py

The script counts the number of occurrences a unique CDR3 is present in a dataset. It uses the output of the cdr3_binning.py script.

```
>cdr3_occurences.py bins_sample1.cdr3
```

The script produces the bins_sample1.cdr3.occurences text file, which contains the following numbers:



```
1, 8070
2, 154
3, 20
4, 2
5, 5
6, 2
7, 1
9, 1
15, 1
35, 1
41, 1
```

Figure 12: The output of cdr3_occurrences.py

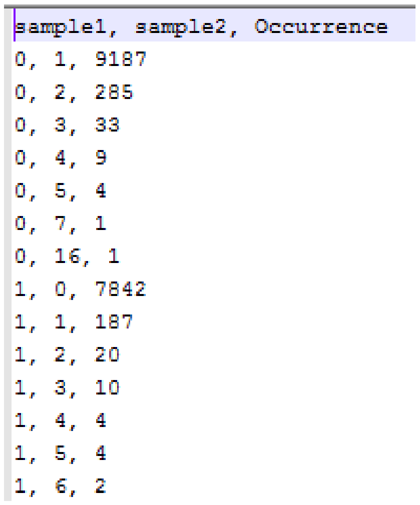
The interpretation of the output is as follows: Sample1 contains 8070 CDR3s that are present only once, 154 that are present two times … and there is a single CDR3 that is present 41 times.

count_pairs.py

This script is similar to the `cdr3_occurrence.py` script. It counts the occurrences of frequencies of pairs of CDR3s from two samples. It uses a cluster file generated by `cdr3_cluster.py` using **two** binned samples. The output of the file can be used in a graphical representation of how diverse two samples are from each other. This script was used to create **Figure 2A, Figure 3A-E.**

```
>cdr_count.py sample_cluster.csv
```

The script creates the csv file sample_cluster.count:



```
sample1, sample2, Occurrence
0, 1, 9187
0, 2, 285
0, 3, 33
0, 4, 9
0, 5, 4
0, 7, 1
0, 16, 1
1, 0, 7842
1, 1, 187
1, 2, 20
1, 3, 10
1, 4, 4
1, 5, 4
1, 6, 2
```

Figure 13: The output of count_pairs.py script.

The interpretation of the output is as follows: The first line of the file means that there are 9187 CDR3s that are present in Sample 2 but absent in Sample 1. Line 2 shows that there are 285 CDR3s that are present twice in Sample 2 but absent in Sample 1 and so on.

accumulation.py

This script estimates the size of the library using the cdr3 output of the cdr3_pipeline.py script. It creates a csv output that was used to create **Figure 5**.

```
>accumulation.py sample1.cdr3
```

The script produces the `sample1.accumulation` file. The first columns contains the number of CDR3s, the second column contains the number of unique CDR3s.

cdr3_resample.py

The script randomly resamples the input files for a desired depth. The script uses any text file. This script was used to resample the datasets for Supplementary Figures 4A-E, and 5.

```
>cdr3_resample.py bins_sample1.cdr3.csv
```

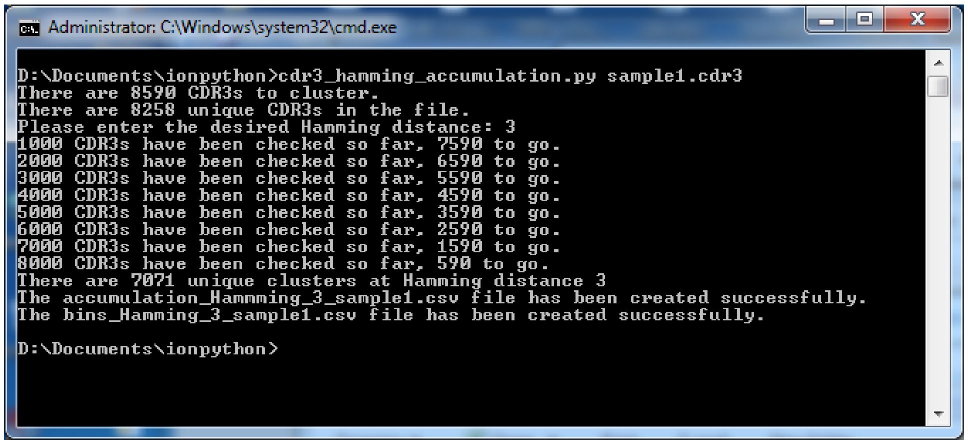
The script produces the bins_`sample1.cdr3.csv.sample` file containing the desired number of random lines of the input file.

cdr3_hamming_accumulation.py

This script takes the .cdr3 output of the `cdr3_pipeline.py` or `cdr3_search.py` scripts and bins them at a given Hamming distance. The script also calculates the accumulation curve at the given Hamming distance.

```
>cdr3_hamming_accumulation.py sample1.cdr3
```

The script can run for a **very long** time depending on the number of cdr3s in the sample:



```
D:\Documents\ionpython>cdr3_hamming_accumulation.py sample1.cdr3
There are 8590 CDR3s to cluster.
There are 8258 unique CDR3s in the file.
Please enter the desired Hamming distance: 3
1000 CDR3s have been checked so far, 7590 to go.
2000 CDR3s have been checked so far, 6590 to go.
3000 CDR3s have been checked so far, 5590 to go.
4000 CDR3s have been checked so far, 4590 to go.
5000 CDR3s have been checked so far, 3590 to go.
6000 CDR3s have been checked so far, 2590 to go.
7000 CDR3s have been checked so far, 1590 to go.
8000 CDR3s have been checked so far, 590 to go.
There are 7071 unique clusters at Hamming distance 3
The accumulation_Hammming_3_sample1.csv file has been created successfully.
The bins_Hamming_3_sample1.csv file has been created successfully.

D:\Documents\ionpython>
```

Figure 14: The cdr3_hamming_accumulation.py is running with Hamming distance 3

The script produces two files. The `accumulation_Hamming_X_sample1.csv` contains the accumulation data at a given Hamming distance. The first column is the number of CDR3s and the second column contains the unique clusters at the given Hamming distance. The `bins_Hamming_X_sample1.csv` contains the 'unique' CDR3s clustered at a given Hamming distance.

aa_usage.py

This script analyzes the CDR3s for their amino-acid usage for each length of CDR3s at all positions in the peptide. This script takes the .cdr3 output of the `cdr3_pipeline.py` or the `cdr3_search.py` scripts.

```
>aa_usage.py sample1.cdr3
```

The script creates the `sample1.cdr3.aa` csv file containing the following columns:

- Length = CDR3 length in aa
- Position = CDR3 Position along the peptide
- aa_codon = a particular amino-acid at the particular position at that particular length
- aa_count = The number of occurrences of the particular aminoacid at that position at that particular length.

This script has been used to calculate the amino-acid usage of the CDR3s using Excel pivot tables and created **Figure 2D and 4C**.

cdr3_length_dist.py

This script calculates the distribution of the length of the unique CDR3. The script uses the `bins...cdr3` file as the input.

```
>cdr3_length_dist.py bins_sample1.cdr3
```

The script creates a `length_bins_sample1.cdr3 csv` file containing two columns. The first column contains the CDR3 lengths followed by their number of occurrences. See **Figure 2C**.

pi.py

The script calculates the approximate pI of the CDR3s according to [1,2]. It uses the unique CDR3 file bins_*.cdr3 as the input file and produces a .pi file in csv format containing the unique CDR3 sequences and the pI value.

```
>pi.py bins_sample1.cdr3
```

hydro.py

The script calculates the average hydrophobicity value of the CDR3s using the CCS (combined consensus hydrophobicity) scale according to [3]. It uses the unique CDR3 file bins_*.cdr3 as the input file and produces a .hydro file in csv format containing the unique CDR3 sequences and the average hydrophobicity value.

Conclusions

All the scripts contain comments and explanations. You are encouraged to look into them and experiment with them. If you need help you could always contact us by email at csakis[at]lanl[dot]gov. If you request other scripts, we are happy to collaborate.

References

1. Bjellqvist B, Basse B, Olsen E, Celis JE (1994) Reference points for comparisons of two dimensional maps of proteins from different human cell types defined in a pH scale where isoelectric points correlate with polypeptide compositions. Electrophoresis 15: 529–539.
2. Bjellqvist B, Hughes GJ, Pasquali C, Paquet N, Ravier F, et al. (1993) The focusing positions of polypeptides in immobilized pH gradients can be predicted from their amino acid sequences. Electrophoresis 14: 1023–1031.
3. Tossi A, Sandri L, Giangaspero A (2002) New consensus hydrophobicity scale extended to non-proteinogenic amino acids. Peptides: 416–417.