

Antibase Cytoscape Networking

Ryan Nguyen

March 2018

1 Introduction

The goal of these set of scripts is to go through your mass spec data (in mzXML file format) and see if any Antibase compounds or their adducts are present in your sample. This will be visualized in a custom cytoscape network and you can see which scans in your mzXML hypothetically maps to an Antibase compound. If you have any questions, you can contact me at **rnguyen2018@berkeley.edu**. If it is after May 2018, then contact me at **ryan.nguyen@yale.edu**.

2 Download Instructions

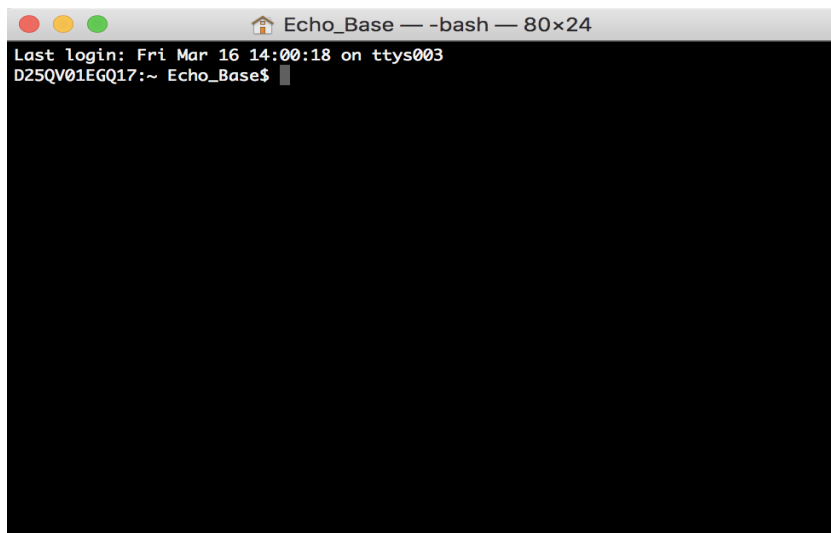
In order to use these scripts, you will need a couple of things (links to download are listed as well):

- Python 3: <https://www.python.org/downloads/release/python-360/>
- Git: <https://git-scm.com/downloads>

Pick the download that works for your system (MacOS or Windows). Both of these are already downloaded onto the Mac in the microscope room. If these downloaded onto your computer successfully, you should be able to open them up on terminal (Mac) or Command Prompt (Windows).

2.1 Making sure Python works on Mac

For Mac users, click on the magnifying glass in the upper right hand corner of your screen and type in "Terminal". Click on that. It should look something like this.



If you type in **python3**, you should get this on your screen.

```
Echo_Base — Python — 80x24
Last login: Fri Mar 16 14:00:18 on ttys003
D25QV01EGQ17:~ Echo_Base$ python3
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct 3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Hooray! It works. Now, press Ctrl-D to quit python.

2.2 Making Sure Git works on your Mac

To make sure that git works, type in **git** into your Terminal. You should see this on your screen:

```
Echo_Base — -bash — 115x47
Last login: Fri Mar 16 14:05:49 on ttys004
D25QV01EGQ17:~ Echo_Base$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
       [--exec-path<path>] [--html-path] [--man-path] [--info-path]
       [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
       [--git-dir<path>] [--work-tree<path>] [--namespace<name>]
       <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
D25QV01EGQ17:~ Echo_Base$
```

Hooray! It works.

2.3 Downloading the scripts

So this step is to put a folder on your desktop where all the scripts will be stored. To do this, open up your Terminal and type in this line by line:

```
cd ~/Desktop (make sure there is a space between cd and the following statement)
mkdir antiBase
cd antiBase
```

DO NOT EXIT YOUR TERMINAL. On your desktop, there should now be a folder called **antiBase**. Now, type in this:

```
git init
git clone https://github.com/Traxlab/antibase.git
```

If no errors popped up, then you have downloaded it correctly. Congrats!

3 Running the Scripts

Okay, open up the antiBase folder on your desktop and make sure there are no mzXML files in there. Once you've done that, put your mzXML file of interest into the folder. You are now ready to run the scripts.

NOTE: If you plan to use the MS-Dial output as your source, please make sure that it is a csv file and that the filename starts with "avg". Before running, make sure there are no other MS-DIAL csv files in the folder.

Open terminal and run these lines of code:

```
cd Desktop/antiBase
python3 antibaseComp.py
```

DO NOT EXIT TERMINAL. This will take about 5-6 minutes to run, and you will know when it's done when it prints out the amount of seconds it took to run. Your Json file with all of your data has now been made! To visualize this, run this line of code:

```
python3 -m http.server
```

You might be asked to allow Python to run a server. Say allow. Then, open Google Chrome and search

localhost:8000

You might need to scroll around the webpage a little to find it, but your visualization will be there. On any given network, you will see the chemical formula for the antibase compound and its molecular weight. Connected to that will be a node(s) that contains its scan number in your mzXML, the m/z value of that scan, and what adduct it can map to. To stop the visualization, simply go to your Terminal and press Ctrl-C.

If you would like to view this again item at a later time, all you have to do is this :

```
cd Desktop/antiBase
python3 -m http.server
```

Then navigate again to **localhost:8000**

4 Example Output

