DTarray_pro User Guide

Aaron Maurais

25 September 2018

Contents

1 Introduction

DTarray_pro extracts Uniprot ID numbers, molecular weights, and spectral counts from DTASelect-filter files. Protein data is combined into one dataset and written to the working directory as a tab delimitated text file (.tsv). This document will describe how to install and use the latest version of DTarray_pro step by step. Some experience using a unix shell is assumed.

2 Installation

DTarray_pro is hosted at GitHub, which is a free hosting service for distributed version control in software development. The latest stable version of DTarray_pro will be posted at https://github.com/ajmaurais/DTarray_pro/releases

2.1 Download and unpack DTarray_pro archive file from GitHub

- Navigate to the releases tab on the DTarray_pro GitHub page.
- The files for the latest release should be at the top of the page.
- Download the file: `Source code (tar.gz)` for the latest release, to you computer.
- DTarray_pro expects to be installed in `~/local`. The program needs data stored in text files in `~/local/DTarray_pro-1.7.4/db` for some features to work. First make the directory `~/local` on your `pleiades` account if it doesn't already exist.
- Transfer the source code archive (should be named something like `DTarray_pro-1.7.4.tar`) to your `pleiades` account using your FTP client of choice.
- The source code archive has to be unpacked before you can access it. To unpack the `.tar` type the following commands in your terminal.

```
$ cd ~/local
$ tar -xfv DTarray_pro-1.7.4.tar
```

- As a result, a new directory should be created in `/local` named `DTarray_pro-1.7.4`
- Once you have unpacked the archive, you no longer need the `.tar` file and can delete if of you wish.

2.2 Build DTarray_pro executable

- Before you can use `DTarray_pro`, you have to build the executable from source. Fortunately `DTarray_pro` is configured to work with a build automation tool called `make` so the process should be straightforward.
- To build `DTarray_pro` run the following commands in your terminal.

```
$ cd ~/local/DTarray_pro-1.7.4/
$ ./configure
$ make
```

- After you have run `make`, there should be several new files in the `DTarray_pro-1.7.4` directory. If everything worked, the executable file should be located at `DTarray_pro-1.7.4/bin/DTarray`

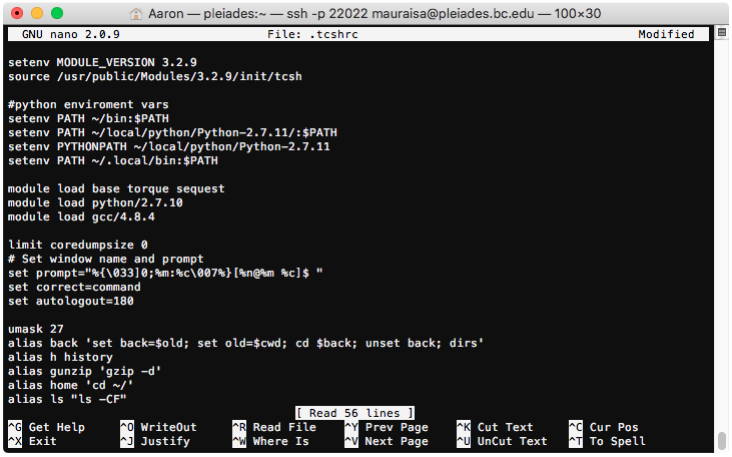
2.3 Adding a shortcut for DTarray_pro (optional)

To run `DTarray_pro` you have to navigate on your terminal to a folder which contains DTASelect-filter files then type the full path to the executable file relative from the directory you are currently in. Its possible to install a program system wide so you don't have to type the path every time, but without administrative privileges, its a bit complicated. A workaround is to create a shortcut or alias to the executable file. This section will explain how to add an alias for `DTarray_pro` to your shell profile on `pleiades`

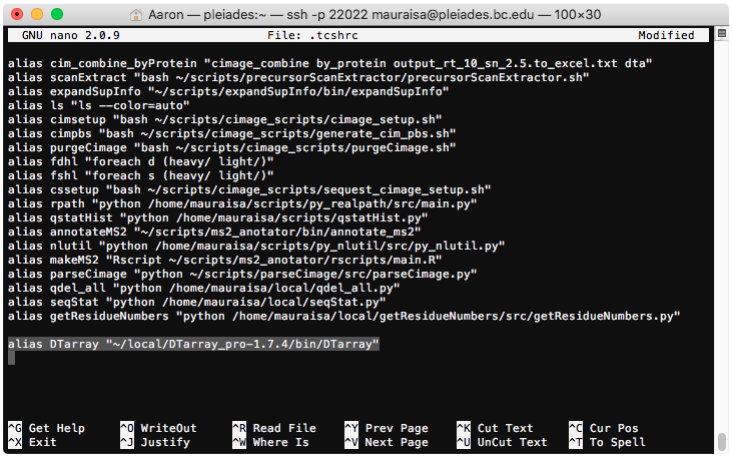
- To add an alias for `DTarray_pro`, you will have to edit your shell profile, which is a file stored in your home directory named `.tcshrc`.
- To edit your shell profile, you will use a command line text editor called `nano`. To open `.tcshrc` in `nano`, type:

```
$ cd
$ nano .tcshrc
```

- After starting `nano`, your terminal window should look something like this:



- Scroll to the bottom of the file and add the line:
`alias DTarray "~/local/DTarray_pro-1.7.4/bin/DTarray"`



- To save and exit the file, hit `^+ X`. A dialog should show up at the bottom which says:

```
Save modified buffer (ANSWERING "No" WILL DESTROY
   CHANGES) ?
```

- Hit `y`

- Next a dialog should show up at the bottom which says

```
File name to write: .tcshrc
```

- Hit `enter` to exit.
- Finally you have to tell the computer to reload your shell profile after you have modified it with the command:

```
$ source .tcshrc
```

- You can test your alias by typing the following command from your home directory:

```
$ DTarray --version
```

- If the alias is recognized by the computer, it should display something like:

```
DTarray_pro 1.7
Last git commit: Sat Sep 22 20:28:17 2018
git revision: 04d30f4fd7790abfca60197d85cefc9d2a
```

3 Usage

3.1 Input format options

Users have two options for how the input DTASelect-filter files read in by `DTarray_pro` are structured.

3.1.1 std mode

In `std` mode, DTASelect-filter are stored in a common directory and the files are named with the format `<sample_name>.dtafilter` (This is the default input mode for `DTarray_pro`.)

```
$ ls
sample_1.dtafilter                sample_2.dtafilter
sample_3.dtafilter                sample_4.dtafilter
```

The user can either manually setup the folder, or use DTsetup to automatically generate the folder. To run `DTarray_pro` in `std` mode, run the commands:

```
$ cd <path_to_directory_with_filterfiles>
$ DTarray

DTarray_pro v1.7

Adding sample_1...
Adding sample_2...
Adding sample_3...
Adding sample_4...

4 files combined.

Writing protein data... done!
Protein data written in wide format to: DTarray_pro.tsv
```

If `DTarray_pro` was successful, two files should have been generated. `DTarray_pro.tsv` is a tab delimitated text file (.tsv) which contains a row for each protein and a column for the spectral counts of that protein in each sample. `dtarray_pro_flist.txt` is a list of valid `.dtafilter` files found in the directory. If a file list already exists in the folder, the existing file list is used. The user can edit the file list to change the order of the columns in the output file and add additional files to the list.

3.1.2 subidr mode

In `subdir` mode, DTASelect-filter files are stored in a separate directory, the directory name is the sample name, and the filter files are named `DTASelect-filter.txt` (This is the default input mode for `dtarray.pl`.)

```
$ find */*.txt
sample_1/DTASelect-filter.txt
sample_2/DTASelect-filter.txt
sample_3/DTASelect-filter.txt
sample_4/DTASelect-filter.txt
```

To run `DTarray_pro` in `subdir` mode, the user has to include the `-i subdir` option, because `subdir` is not the default behavior.

```
$ cd <path_to_parent_directory>
$ DTarray -i subdir
```

Upon completion, a files named `DTarray_pro.tsv` and `dtarray_pro_flist.txt` are generated formatted the same as in `std` mode.

3.2 Output file options

This document will not provide an exhaustive list of options for `DTarray_pro`. Instead, this section will provide examples of options will likely find useful. For the full list of optional arguments see `DTarray_pro-1.7.4/helpFile.pdf` or use `DTarray -h` to see the help file from the terminal.

3.2.1 Get spectral counts for unique peptides by protein

the `-u` option is used to include a column for the total counts for unique peptides in the `DTarray_pro.tsv` output file.

```
$ DTarray -u
```

By default, the columns for spectral counts and unique peptide spectral counts are grouped by sample. See section 3.2.2 to change this behavior.

3.2.2 Specify how to group supplementary information columns in output file

The examples in this section are use the `-u` option (section 3.2.1), but the `-u` option is also compatible with other `DTarray_pro` options including the `-lr` option (see section 3.2.5)

By default, the columns for sample specific supplementary information are grouped by sample as shown in table 1.

The `-s` option can be used to control this behavior. To group the columns by sample (table 2, then observation add the `-s 1` option.

```
$ DTarray -u -s 1
```

	Sample_1		Sample_2	
Protein	SC	Unique_SC	SC	Unique_SC
ALBU	2149	2149	3092	3092
TRFE	661	661	698	698
IGHG1	573	152	382	52

Table 1: Default column arrangement

	SC		Unique_SC	
Protein	Sample_1	Sample_2	Sample_1	Sample_2
ALBU	2149	3092	2149	3092
TRFE	661	698	661	698
IGHG1	573	382	152	52

Table 2: Grouping columns by sample

3.2.3 Get spectral counts for peptides

By default no peptide file is generated. To also generate a file containing spectral counts for peptides in each sample, use the `-p 1` option.

```
$ DTarray -p 1
```

An additional file should be generated named `peptideList.tsv` containing peptide data.

3.2.4 Modify how peptides are grouped in output file

By default, peptides are grouped by sequence and parent protein. A separate entry for each charge state of a given peptide will be included in peptide output files. If the `-g 2` option is set, peptides will also be grouped by charge; i.e., the spectral counts for each peptide will be the sum of all charge states identified for that peptide.

```
$ DTarray -p 1 -g 2
```

The `-modG <group_method>` specifies how to group modified peptides in `peptideList.tsv`. By default peptides with the same sequence, but different modification status will not be grouped. A separate entry will be included for

each modification status found for a peptide. To ignore modification status when grouping peptides, use the `-modG 1` option.

```
$ DTarray -p 1 -modG 0
```

If the `-p 1` option is not set, the `-g` and `-modG` will be ignored. The `-g` and `-modG` options can also be combined as desired.

3.2.5 Get subcelluar location data for proteins

`DTarray_pro` can use `DTarray_pro-1.7.4/db/humanLoc.tsv` to lookup subcelluar localization information for proteins. `humanLoc.tsv` contains Uniprot annotations for subcelluar localization by Uniprot ID, updated as of Jan 18 2017. Currently, sub cell location information is available for human proteins only.

There are two ways in which subcelluar location information can be compiled.

1. The `-loc` option will add a column for the location of each protein in `DTarray_pro.tsv`. To include the subcelluar location column in `DTarray_pro.tsv` run the command:

   ```
   $ DTarray -loc
   ```

2. The `-lr 1` option will create a file named `loc_summary.tsv` with the sum of spectral counts, sequences, and proteins identified for each subcelluar location. To generate `loc_summary.tsv` run the command:

   ```
   $ DTarray -lr 1
   ```

   By default, columns are arranged by sample. The `-s 1` option can be used to arrange the columns by observation. See section 3.2.2 for an explanation of the `-s` option.

3.2.6 Calculate molecular weights for peptides and proteins

`DTarray_pro` will calculate protein/peptide molecular weights and molecular formulas when the `-mw` option is provided. Columns will be included in output files for average mass, monoisotopic mass and molecular formula.

Three files are required for the calculation:

1. An atom count table, named `atomCountTable.txt` which contains the number and types of atoms found in each amino acid (similar to `Cimage` table).

2. An atom mass table, located at `DTarray_pro-1.7.4/db/atomMasses.txt`, containing the masses of each atom.

3. A `.fasta` file to lookup protein sequences located at `DTarray_pro-1.7.4/db/humanProteome.fasta` (required for proteins only) Currently, the `-mw` option is supported for human proteins only.

By default the atom count table located at `DTarray_pro-1.7.4/db/atomCountTable.txt` is used. The default atom count table includes a static modification for iodoacetamide alkylation. To calculate peptide and protein masses with default residue masses, run:

```
$ DTarray -p 1 -mw
```

The user can also supply a custom `atomCountTable.txt` file with the `-act <file_name>` option. A copy of the default atom count table can be generated in the working directory with the `-mact` option.

```
# make default atomCountTable.txt in working directory
$ DTarray -mact
# run DTarray with custom atomCountTable.txt
$ DTarray -act ./atomCountTable.txt -p 1 -mw
```

The user can also edit the default atom count table as at `DTarray_pro-1.7.4/db/atomCountTable.txt` as desired, but editing `DTarray_pro-1.7.4/db/atomMasses.txt` is not recomended.