A Manual for BEM Online Update Tools

For PseAAC-Builder version 3.0

(BOUTs)

Contents

Pufeng Du, Ph. D.

Mar. 20th, 2013

1 Format conventions

In this manual, you will see a number of sections and words are marked with deep-blue background and white characters, like `this`. These parts are computer program name, command line options, screen displays or URLs. These parts, if not italic, should be either input or output directly on the screen as it is. However, for those parts with the above color scheme and italic fonts, such as *`this`*, they should be replaced with real filenames, options or parameter values in practice.

2 General information

Binary Extension Module (BEM) is a type of add-on for PseAAC-Builder version 3.0. The Gene Ontology and the Functional Domain modes of the general form pseudo-amino acid composition (PseAAC) are built by the BEMs. The BEMs contain information from the UniProtKB. They should be updated when the UniProtKB is updated. The BEM updates are provided on a monthly basis. If the users needed the most up-to-date BEMs, they would run the BEM Online Update Tools (BOUTs).

BOUTs are provided as helper programs for the PseAAC-Builder (PSEB). There are four programs in the BOUT, including the BEM Compiler (`bmc`), the BEM Viewer (`bmv`), the Pre-processor (`GPAList.pl`) and the driver script (`update-bems.sh`). The BEM Compiler and the BEM Viewer are written in C++. The Pre-processor is a Perl script. The driver script is a Bash shell script.

By executing the driver script, records in UniProtKB are converted into BEMs. The conversion includes three steps. In the first step, a tabular file is downloaded from UniProtKB. This file is converted into several temporary files by the Pre-processor. In the second step, the temporary files are compiled by the BEM Compiler and a binary file is built. In the final step, GZip program compresses the binary file to reduce its size. The BEM file is produced by changing the filename from ".gz" to ".bem".

3 A quick start guide

To make BOUTs working properly, the Perl interpreter and the bash shell are needed as well as the GNU C++ Compiler and the GNU C Library. Use the following command in the directory "`bmc`" to build and install the BEM compiler.

```
make && make install
```

Use the following command to launch the update process.

```
update-bems go
```

Please ensure that an internet connection is available for accessing the UniProtKB at http://www.uniprot.org/. Please also make sure that the writing privilege of current directory has been assigned to the user who launched the update process. The update process would last for several minutes to several hours, depending on the speed and quality of the internet connection. Two files would be created after the updates. One file is a ".bem" file that is usually named like “uniprot-go-MMDDYYYY.bem”, where MMDDYYYY is a string representing current date. The other file is a tar-ball, which is usually named like “uniprot-MMDDYYYY.tar.gz”. This file contains the original tabular data downloaded from UniProtKB.

The following command can check the integrity of a BEM file.

```
bmv -i -m bem_file
```

If the BEM file is correct, a statistical report will be written on the screen like the follows:

```
Total features: 20660
Total instances: 539616
Total ones in feature matrix: 2399431
```

The meaning of the above report will be explained later. If the BEM file is corrupted, the result is unpredicted theoretically. However, in most cases, an error message of “Segmentation fault” would be generated.

The BEM file can be loaded into PseAAC-Builder via the option “–b”. A sample of the command is:

```
pseb -i fasta_file -t 2 -b bem_file
```

The usage of pseb is described in the Manual for PseAAC-Builder 3.0.

4 Command line usages

4.1 The driver script - `update-bems.sh`

The driver script update-bems.sh is the only script that should be directly executed by the users. The usage of the script is:

```
update-bems.sh {go|ipr}
```

Only one parameter is needed by the script. The parameter can be either “go” or “ipr”. If the parameter is “go”, the program will generate BEM file for Gene Ontology annotations. If the parameter is “ipr”, the program will generate BEM file for protein domain compositions.

4.2 The BEM compiler - `bmc`

The BEM compiler should NOT be launched by the users. For those users want to generate the binary form representation other than the Gene Ontology and the Protein Domain Compositions, this program should be used. The command line syntax of this program is:

```
bmc -o output_file -f feature_list -p protein_list -m association_map
```

There are four options above. All four options are mandatory. The order of the options can vary. `output_file` specifies the result filename. `feature_list`, `protein_list` and `association_map` are files that are usually generated by the BEM pre-processor program `GPAList.pl`. The format of the four files will be described later. There is NO built-in format checking functions in `bmc`. If the format of the files is incorrect, the result will be unpredictable theoretically. In most cases, this will result in an error message "Segmentation fault".

The output file can NOT be used as the BEM directly. It must be compressed by GZip program before it can be used in PseAAC-Builder. To do this, simply type the following command.

```
gzip output_file
```

The compressed file will come with a ".gz" suffix. A rename from ".gz" to ".bem" can make it more like a BEM file. However, the rename is not mandatory as the PseAAC-Builder do not check the filename.

4.3 The pre-processor - `GPAList.pl`

GPAList.pl is a Perl script. This program read the tabular file, which is downloaded from UniProtKB, to generate three temporary files for the BEM compiler program **`bmc`**. The command line usage is:

```
GPAList.pl uniprot_file column_id column_term
```

Three parameters must be entered with the order above. The `uniprot_file` is the file downloaded from UniProtKB. The `column_id` and `column_term` are zero-based indices that indicate the columns in `uniprot_file` that contain the protein id and the features, respectively.
For example, the following command will process a file named `test.tab`. The first column of this file contains the protein id. The fourth column contains the features. Please note that the column indices are zero-based.

```
GPAList.pl test.tab 0 3
```

If there was no error, the message "`GPALIST INFO: First stage compile finished`" will be shown on screen. Three files will be created with filenames "`test.tab-gol`", "`test.tab-prl`" and "`test.tab-mas`", which can be used in `bmc` as the `feature_list`, `protein_list` and `association_map`, respectively.

4.4 The BEM viewer - `bmv`

The BEM viewer can extract and display information from BEM files. The usage of the BEM viewer can be obtained by typing "**`bmv --help`**" on the command line.

```
    Usage: bmv [OPTION...]
    BMV - A program for viewing the content of binary extension modules in
    pseaac-builder.

    -i, --info                   Display the statistic summary of loaded
                                   MODULE_FILE.
    -l, --list={f|p}             List every possible feature/instance names.
    -m, --module=MODULE_FILE     Load the MODULE_FILE.
    -q, --query=INSTANCE_ID      Display  the  features  corresponding  to  the
instance
                                 with    INSTANCE_ID    in    the    MODULE_FILE.
INSTANCE_ID
                                 can be multiple ids that are separated by comma.
    -?, --help                   Give this help list
        --usage                  Give a short usage message
    -V, --version                Print program version
```

This program has four options. The “-m” option is mandatory. A BEM file name should be used as its value. The “-i” option will give a report as we have shown earlier. In this report, three lines will be included. The first line gives the number of “Total features”. This is the number of all Gene Ontology terms or the number of all Protein Domains. The second line gives the number of “Total Proteins”. The third line gives the number of “Total ones”. This is the number of all ones in all binary feature vectors of all proteins in the BEM file.

For example, if a BEM file contains two proteins, and the total number of features 100. Then every protein will be represented as a 100-dimension binary vector. If the first protein got ones in the 24th and 96th dimensions and the send protein 35th and 67th dimensions, the number of “Total ones” would be four.

The option “-l” is used to display a list of all proteins or all features, depending on its value “p” or “f”. The value of options “-q” can be a string containing a serial of protein ids that are separated by commas, like “CMOB_SALPA,FADR_VIBVU”. It will display the feature of every protein in the following format.

```
    CMOB_SALPA
       GO:0016300
       GO:0002098
    FADR_VIBVU
       GO:0003677
       GO:0005737
       GO:0006631
       GO:0000062
       GO:0019217
       GO:0003700
       GO:0006351
```

There is no special format checking to the BEM files in `bmv`. If the BEM file is corrupted, the result is unpredictable theoretically. However, in most cases, an error message “Segmentation fault” will be shown.

5 File format specifications

5.1 BEM files

The BEM file is a GZip compressed binary file. The uncompressed BEM file is a composite binary table that contains three parts.

The first part contains the name of all features. First four bytes is an integer, indicating the total number of features. Every feature name will occupy 16 bytes continuously from the fifth byte.

The second part contains index structures that make the position of the third part related with the name of every protein. The first four bytes is an integer, indicating the total number of proteins. After this integer, every record in this part occupies 24 bytes. The first 16 bytes of every record is the name of the proteins. The 8 bytes followed are two integers that point out a position in the third part and the length of the records in the third part.

The third part contains a serial of integers that indicates the position of features in the first part. For example, if a protein can be found in the second part, with integers: 16 and 3, then in the third part the 16th, 17th and 18th integers would be used to find three features from the first part.

To get a more comprehensive understanding to the file format, the code in bems.cpp should be read.

5.2 UniProt tables

The UniProt Table is an ASCII coded text file. This file should be downloaded from UniProtKB with the online query functions. The standard query is

```
    reviewed:yes&format=tab&columns=id,entry%20name,protein%20names,go-id,
interpro&compress=yes
```

There are 5 columns in this table, including ID, Entry name, Protein Names, GO IDs and InterPro IDs. The columns are separated by the table character. In the GO IDs and InterPro IDs columns, different terms should be separated by semi-colons. Every line in this table should have a unique identifier in either the ID column or the Entry name column, which should be used in the GPAList.pl as the ID column.

5.3 Other files

The other files, which are used in the BOUTs, are the Protein list file, feature list file and the association matrix file. These files are only intermediary between the UniProt Table and the BEM file. The users do not need to access or modify these files manually. Thus, the format specifications of these files would not be discussed here. As the GPAList.pl is provided in the form of source code, it will be easy to understand the format of these file by analyzing its code.

6 Inquiries and bug reports

The manual and the BOUTs are written in a hurry. Every pieces of bug reports, complains and suggestions are welcome. They should be sent directly to the author Dr. Pufeng Du through Email PufengDu@gmail.com. If you want to participate in the project of PseAAC-Builder, you can also send emails to Dr. Pufeng Du.