A Manual for PseAAC-Builder version 3.0

For Linux/Cygwin environments

(PSEB)

Contents

Pufeng Du, Ph. D.

Copyright © 2013 Tianjin University

Mar. 20th, 2013

1 Format conventions

In this manual, you will see a number of sections and words are marked with light-gray 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

PseAAC-Builder (PSEB) is a program that can generate the pseudo-amino acid composition (PseAAC) from the primary sequence of a protein. The version 1.0 of this program supports two different types of PseAACs, which are known as the classic PseAAC and the amphiphilic PseAAC. The founder of the PseAAC, Prof. Kuo-Chen Chou, proposed the general form PseAAC in the year 2012. Thus, in this new version of PSEB, we added the support of general form PseAAC.

The code of the PSEB has been rewritten from scratch since version 2.0. The version 3.0 can process sequences 100 times fast of the version 1.0. Since the version 3.0 is based on the version 2.0's code, it may be useful to read the documents of version 2.0 if you encountered some problems in using version 3.0.

There is no dependency requirement like .NET Framework or Mono Framework in version 3.0 PSEB. However, to compile the code on your platform, the following libraries should be installed before the building process: zlib (-lz), dlopen (-ldl) and math library (-lm). A static linked binary executable was provided, which should work with Ubuntu Linux 12.04.

3 Installation

3.1 Linux/UNIX

To install the PseAAC-Builder on Linux/UNIX, one can download the source code package, unpack it to a directory like "~/pseb", navigate to that directory and use the common routine as:

```
make && make install
```

to setup them.

The compiled binary files will be put in a directory like "~/pseb/bin". You can copy them to the directory you like. Typically, you should get three binary files with the name "pseb", "bmc" and "bmv". The "pseb" is the main program of PseAAC-Builder. The "bmc" and "bmv" are parts of the online-update tools. You may find the document for these programs in "~/pseb/tools".

If you get error message, you may need to check whether that is caused by missing some libraries. If so, you need to find those libraries for your platform yourself.

For Ubuntu environment, we provided static linked binary executable programs within the package (in directory prebin). You can try to use them directly. For other Linux based systems, you may also try these binary files directly.

3.2 Windows

The support of Windows platform depends on the Cygwin environment. The Windows platform support may not be continued in the future versions. Please refer to the Cygwin version package for more details.

3.3 Others

Currently, PseAAC-Builder does not formally support other platforms. However, if the platform has a ported GNU GCC and a GNU C library, the source code should be compiled without errors and with dependencies, such as zlib and math library.

4 Command line usages

The PSEB v3.0 program can accept 17 different options and parameters. The usage of these options and parameters can be displayed by using the following command.

```
pseb --help
```

Please note that almost every option has two different forms: the long form and the short form. The effects of using long form and short form are exactly the same. The long form requires the "=" symbol between the values and the options, while the short form does not. The complete explanations of the options are in Table 1.

Table 1 – Command line parameters

Options	Interpretations
-a, --aminoacid	This option is used to indicate that the amino acid compositions should be included in the model. If this option is omitted, the amino acid compositions will not be included. If the -d option is omitted also, the -w option will also be ignored.
-b, --bem-file=BEM_FILE	BEM_FILE can be any valid BEM format file that was provided with the PseAAC-Builder or created using BOUTs toolset. Different BEM_FILE will generate different types of general form PseAAC. If -t option was not set to 2, this option will be ignored.
-d, --dipeptide	This option is used to indicate that the dipeptide composition should be included in the model. If this option is omitted, the dipeptide composition will not be included. If the -a option is omitted also, the -w option will be ignored.
-e, --user-ext=LUA_FILE	LUA_FILE is a Lua script that follows the rules of PSEB User Extension rules.
-i, --input=INPUT_FILE	INPUT_FILE should be a valid FASTA format file. The additional restriction is in the annotation line, which was always started by ">". Every annotation line in the input FASTA file must contain two and only two fields, which are separated by "|". The first field is the id of the sequences, which can be gi numbers, uniprot id, accession numbers or anything that can identify the sequence uniquely in the FASTA file. The second field is an integer, which is recognized as the class label when you export the result. If you do not need a class label, simply put a "0" in this field. If you do not specify the INPUT_FILE, please omit the -i option completely. In this case, the STDIN will be used as the input file.
-l, --lambda=LAMBDA	The LAMBDA parameter in the PseAAC algorithm can be any integer in the range (0,20]. The default value for LAMBDA is 10.

	When -s is used, LAMBDA will be used for every segment separately. Please bear in mind that LAMBDA must be less than the length of every segment. This is the same restriction as the PseAAC server. However, PseAAC-Builder would not complain the invalid value of LAMBDA. But the result for too big LAMBDA is completely meaningless.
-m, --output-format={svm|csv|tab}	The output format can be selected from the following three. (1) svm: The libSVM training data format. (2) csv: The format that can be loaded into spreadsheet program. (3) tab: Simple format delimited by TAB. If this option is omitted, the tab format will be default.
-n, --ext-options=VAR_STRING	VAR_STRING is a set of Lua variables like "name1=value1; name2=value2; ..." that are separated by semicolon, these variables will be executed only once at the beginning of the script.
-o, --output=OUTPUT_FILE	The OUTPUT_FILE can be any valid file that can be written with your own privilege on your system. The program will overwrite the file if it has already existed. If you do not specify the OUTPUT_FILE, please omit this option completely. In this case, the STDOUT will be used as the output file.
-q, --query	Since PseAAC-Builder includes all AAIndex database inside the program, this option is used to export a list of physicochemical property names for your choice. This option is an immediate option. When this option appears on the command line, all other options will be ignored. When this option appears with the other immediate option, the function of the first option (from left to right) will be executed.
-s, --segment=SEG	SEG should be an integer in the range [1,4]. The sequence will be segmented into SEG segments before constructing pseudo-amino acid compositions. The feature vector each segments will be connected. The default value of this option is 1.
-t, --type=PSE_TYPE	PSE_TYPE can be 0, 1, 2 or 3. 0 - classic pseudo-amino acid composition. 1 - amphiphilic pseudo-amino acid composition. 2 - the BEMs based features. 3 - User defined Lua scripts. If this option is omitted, the -w option will be ignored. If this option is set to 2, -a -l -x and -w options will be ignored and the -b option become mandatory.
-w, --weight=WEIGHT	The WEIGHT parameter in the pseudo-amino acid composition algorithm can be a value between (0,1]. If you do not specify the WEIGHT parameter, the program will not use w in the algorithm to balance the pseudo-factors and the real-factors. If the -d and the -a options are both used, the WEIGHT will be applied to balance all first 420 dimensions of the feature vector. When -s is used, w will be

	applied to every segment separately.
-x, --select-aaindex=SELECTION_FILE	SELECTION_FILE is a text file that contains the names of selected physicochemical properties. Every line of this file contains the name of one physicochemical property. These names should be from the list that can be produced by the -q option. The physicochemical properties that are listed in this file will be used for calculating the pseudo-amino acid compositions. If this option is omitted, the choice of the physicochemical properties will be decided by the -t option. The physicochemical properties of original form of pseudo-amino acid composition will be used.
-?, --help	Display a help screen.
--usage	Give a short usage message.
-V, --version	Print program version.

5 General Mode Modules

PSEB v3.0 supports two types of add-on modules for constructing the general form PseAAC. They are known as the Binary Extension Modules and the Embedded Script Procedures.

5.1 Binary Extension Modules

The Binary Extension Modules (BEMs) are used to generate PseAAC in Gene Ontology mode and the Functional Domain mode. Basically, the BEMs are extracted and compressed information from the UniProtKB/SwissProt database. They can be downloaded from the Source-Forge web site of PSEB. The name of a BEM file contains three fields, which are separated by the hyphen. For example, a BEM file may have a name like "`uniprot-go-03202013.bem`". The first field is the name of the source database. Currently, it can only be "uniprot". The second field is the mode of PseAAC. Currently, it can be either "go" or "ipr", where "go" means Gene Ontology module, and "ipr" Functional Domain module. The last field is the building date of the module. The value "03202013" indicates the module was built on Mar. 20th, 2013. As the BEMs are binary representations of protein annotations. These annotations need to be updated when the UniProtKB/SwissProt database is updated. Therefore, we provide the updates for BEMs on a monthly basis. For the user who needs the most up-to-date module, we provide a set of tools, which are known as the BEM Online Update Tools (BOUTs), for constructing the BEMs for their own. Especially, the BEMs are not restricted to represent only the Gene Ontology terms or the Functional Domain compositions. They can be used to describe any kind of sequence annotations. The users can built their own BEMs for various features by modifying and using BOUTs.

To load a BEM file into PSEB, the option "-t" and "-b" must be specified together on the command line. For example, the following command will compute the Gene Ontology mode PseAAC for the FASTA file "`test.fas`", with the FASTA file "`test.fas`" and the BEM file "`uniprot-go-03202013.bem`" in the current directory.

```
pseb –i ./test.fas –t 2 –b ./uniprot-go-03202013.bem
```

In the above command, "-t 2" indicates the BEM file should be loaded. The "-b" option give the filename of a BEM file. To produce the Functional Domain mode, the users only need to load another BEM file, like `uniprot-ipr-03202013.bem`. The "-i" option give the name of your FASTA file.

5.2 Embedded Script Procedures

The general form PseAAC has too many variations. It is impossible to include all modes in a single program. We decided to let the users to develop their own forms of PseAAC. Therefore, we embedded the Lua script (http://www.lua.org) engine inside the PSEB v3.0 program. The Lua script, which is a light-weighted and easy-to-learn script language, has been widely applied in recent years, from online gaming to the genomics research. The manual of Lua can be found on its official website.

The Lua script can access the internal data structure of PSEB v3.0. It can read the command line parameters. It can implement many user-defined algorithms in modeling protein sequences. The scripts would be released as add-ons and updated from time to time by all the users of PSEB as well as our team.

The PSEB program will handle the data input, command-line parsing, flow controlling, error reporting, result output and formatting. The Lua script only needs to focus on how to process a single sequence. As an example, the Pseudo-Positional Specific Scoring Matrix (PsePSSM) mode was implemented by Lua. Propy (http://code.google.com/p/protpy/) was also cloned by Lua. To help the users to design their own scripts, we will introduce the interface between Lua and the PSEB internal data structures.

5.3 Lua script interface

The global scope of the script will be executed once and only once before the protein sequences are loaded. After the loading of all protein sequences, the PSEB program will call a Lua procedure with the following signature on every sequence:

```
function pseb3_seq_proc (pr_id, pr_seq)
```

There are two parameters in this function. The "`pr_id`" is a string, which is a sequence ID in the FASTA file. The "`pr_seq`" is the corresponding protein sequence as a string. This function must return two values. One is the number of dimensions of the PseAAC, and the other is a list object containing the PseAAC. Any script error in this function or the function called by this function will cause the PSEB to abort the process and exit abnormally.

5.4 Accessing the internal data structures

The internal data structures can be accessed by the following two functions. No additional libraries or modules are required for calling these two functions. They are built-in procedures of the PSEB.

```
function psebGetPCList (pc_id)

function psebGetOptions ()
```

The `psebGetPCList` function returned a list containing normalized values of a type of physicochemical property. The parameter "pc_id" must be a valid name in AAIndex database. The returned value of `psebGetPCList` is a table object. For example,

```
local t = psebGetPCList(prop_name),
```

where t is the returned table object and prop_name a string indicating the name of a physicochemical properties.

The following properties can be accessed in the returned table object t.

(1) `name` property. The "name" property is the ID provided.

(2) `desc` property. The "desc" property is the description of the physicochemical property, such as "Average volumes of residues (Pontius et al., 1996)" or "Hydrophobicity index (Wolfenden et al., 1979)".

(3) The normalized values of every amino acid. These values can be accessed like `t["A"].` These values have been normalized before they are returned. Only 20 standard amino acid letters can be used as the index.

The `psebGetOptions` function will return a table object containing two properties, including "l" and "w", which correspond to the value of command line options "-l" and "-w", respectively. If the command line options are not specified, the internal default values of these options will be applied. The default values of "l" and "w" are 10 and 0.05, respectively.

If a user-defined procedure access the physicochemical properties every time it is called, it will be very inefficient. The following is a little trick in writing code when the physicochemical properties are needed to be accessed for every sequence.

The physicochemical properties should be buffered as a global variable in the Lua script. The global variables would not be changed between the process of one sequence and another. Therefore, when accessing the physicochemical properties, the following structures should be used.

```
--A global variable buffer to avoid crossing C/Lua boundary every time the script
--is called
tempPC = {}
pseb_opt = nil
pseb_opt_w = nil
pseb_opt_l = nil

--C/Lua interface
function pseb3_seq_proc (pr_id, pr_seq)
    ...
if not tempPC[prop_name] then
        tempPC[prop_name] = psebGetPCList(prop_name)
    end
    if not pseb_opt then
       pseb_opt = psebGetOptions()
       if pseb_opt then
          pseb_opt_l = pseb_opt["l"]
          pseb_opt_w = pseb_opt["w"]
       end
    end

    ...

    return ...
end
```

When the above procedure is called by the PSEB for the first time, it tests whether the `prop_name` can be found in the `tempPC` table. If the `prop_name` is found, it uses the value in `tempPC` table. If not, it calls the `psebGetPCList` to get the physicochemical properties and buffered them in the `tempPC table`. When it is called again, it will not need to call `psebGetPCList` again. This also applies to the `psebGetOptions` function.

5.5 Additional command-line parameters

Since version 1.0, one of promising features in PSEB program is the parameter scanning function. If the users defined their own procedures with Lua script, the parameters for the script can also be specified on the command line. This is achieved via the option "-n". For example, the following command will let the script read two parameters, "foo" and "sig".

```
pseb –i test.fas –t 3 –e ./test.lua –n "foo=1;sig=STOP"
```

```
```

Please note that the value of "-n" options must be quoted. Every parameter must be specified in the form of "X=Y", where X is a valid variable name in Lua and Y can be anything valid on the command line. If there are two or more parameters, they must be separated by the semi-colon.

In the above example, the parameter "foo" is an integer, while the parameter "sig" is a string. However, in the Lua script, both parameters are recognized as strings. The users are responsible to remember and convert the type of the parameters.

These parameters, once set on the command line, can be accessed in the Lua script as global variables. Please note that, these global variables are set even before the script file is loaded. Therefore, if the script set the same global variable, the values of the command line parameters will be overwritten.

5.6 Downloading and using the Lua script add-ons

Some Lua scripts can be downloaded from the official sites of PSEB v3.0 in SourceForge. The Lua scripts are usually provided as a zip package, containing a Lua script and a text file named README. The users should read README file before they load and run the Lua scripts.

There is no installation procedure for these add-on modules. Users can simply extract the Lua file to somewhere they like. To load and run a Lua script, the user must use the "-t 3" and "-e" options on the command line in the following format:

```
pseb –i your_fasta_file –t 3 –e your_lua_script
```

5.6.1 Using the PsePSSM mode

The PsePSSM mode was provided as a Lua script. The execution of this script needs some further configurations. Please refer to the README file in the PsePSSM add-on package.

5.6.2 Using the Propy clone mode

The Propy program was cloned as a Lua script. The execution of this script needs further configurations, Please refer to the README file in the Propy add-on package.

6 Other Information

6.1 About the GUI

The GUI part is never a key part in PseAAC-Builder. Since version 2.0, the GUI of PseAAC-Builder was written in Tcl/TK. In version 3.0, the GUI part was not updated. Therefore, only the functions available in version 2.0 can be used through the GUI. For the users who need to use the GUI and the function of version 3.0, please send an email to PufengDu@gmail.com. We will decide whether to improve the GUI function according to the number of requests.

6.2 Submitting your add-ons

If you want to put your add-on modules together with the PseAAC-Builder, you need to send them to Dr. Pufeng Du for a review. Please make sure you included all the information you need to put online in the package. Especially, please do not forget to sign your name and give the citation requirement in the README file. We are very happy to put your add-ons together with the PseAAC-Builder and "advertise" your work on the site of PseAAC-Builder.

6.3 Inquiries and bug reports

The manual and the PseAAC-Builder 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.