ConSAT user manual

Version 1.0 – March 2014

Alfonso E. Romero

Department of Computer Science, Centre for Systems and Synthetic Biology

Royal Holloway, University of London – Egham Hill, Egham, TW20 0EX

Table of Contents

1. Overview of ConSAT

ConSAT stands for "Consensus Signature Architecture Tool". Using ConSAT on a set of protein sequences you can get their corresponding protein families. The protein families, for our purpose, are *consensus domain architectures*. Proteins within the same family are assumed to share many properties among them as they are assumed to descend from a common ancestor –that is, they are evolutionarily related. Thus, protein families can be of a good help in the study of large protein sets, moving us from the study of single sequences to the study of the set of families (the number of families found will be lower than the set of individual sequences).

The way ConSAT builds the architectures is by combining two sources of data: (1) domain assignments from InterPro, and (2) domain assignments from GFams (see section 7. The GFams dataset). The combination is done in a way that no overlapping domains will be allowed, and maximising the sequence coverage. See section 6. The ConSAT algorithm for more details.

ConSAT is written in Python and has a few requirements to run properly (see section 2.1 Requirements). You can check, as well about its installation (section 2.2 Installation) or its set of features (2.3 Features). To learn how to run it, there is a quick introduction in section 3. A quick introduction to ConSAT, and a more detailed manual in 4. How to use ConSAT. Details on the output files are given in 5. Output files. The final section of this manual, 8. Version History of this software, shows the historical changes to the ConSAT software for the different versions of it.

ConSAT has been mainly developed by Alfonso E. Romero <aeromero @ cs . rhul . ac . uk> with the contribution of Tamás Nepusz, Rajkumar Sasidharan, and Alberto Paccanaro. This project has been funded by the Biotechnology and Biological Sciences Research Council (BBSRC).

2. The Software

2.1 Requirements

In order to run properly, ConSAT requires the following programs to be installed in your machine:

- Python 2.7.
- NCBI Blast+.
- HMMer version 3.0 (or higher).
- Scipy: is not needed by recommended for efficiency purposes.

2.2 Installation

First of all, download the sources of the stable release from http://paccanarolab.org/software/consat or check the unstable release at https://github.com/alfonsoeromero/ConSAT/ (they eventually might be the same).

To install this software the same procedure for installing any Python module (See http://docs.python.org/2/install/#inst-new-standard for more details) can be followed:

- Unpack the sources
- Go to the newly created folder
- Run the command `python setup.py install`
- Sometimes the previous step requires superuser (root) permissions. For Linux systems like Ubuntu this can be done by pre-pending `sudo` to the previous command, or entering a super

user terminal (executing `su`).

After installation, the command `consat` should be available at any directory of your system. Please check very carefully that this is the case by opening a terminal, running `consat` and check that no errors other than the lack of parameters are produced.

2.3 Features

2.3.1 Software architecture

ConSAT is a set of Python *basic* scripts, each one prepared to perform a single task, and a set of *master* scripts to run the whole pipeline. The *basic* scripts take a set of input text files and produce another set of output files. The *master* scripts take a configuration file with the routes of all the input files needed and the value of the parameters and run a whole ConSAT computation. The *master* script is relying on `modula`, a very powerful Python coded by Tamás Nepusz which avoids doing all the calculations (in case something goes wrong) by storing several intermediate files. Thus, any ConSAT calculation requires a `work` directory (where the intermediate files will be created) and an `output` directory (where the output files will be written). After any typical ConSAT execution you will only be intersted in looking at the `output` directory.

2.3.2 Hardware requirements

To process an average set of sequences (20-30,000 protein sequences) you will need nothing more than a regular machine. Any nowadays computer with a few gigabytes of hard drive, and 4-8 gigabytes of RAM will do the trick. We will try to use several processors if they are available in the parts of the computation which can be run in parallel.

If you want to run ConSAT in a larger set of sequences (>100,000 sequences) you will probably be able to do this, but the RAM consumption would be increased a bit. Depending on your dataset you may need 16 or 32 gigabytes of RAM to make ConSAT run properly. Some tasks are done in memory as this is a compromise between efficiency and space usage (and we prefer the latter). However, if you want to run ConSAT in a single organism it is very uncommon that you need more than 8 GB of RAM.

2.3.3 License

ConSAT is released under the GNU General Public License (GPL) v3 license. You can check this license online here: http://www.gnu.org/licenses/gpl.html. A copy of the license has been provided in the ConSAT sources.

3. A quick introduction to ConSAT

`consat` is driven by a master configuration file. You can create an empty configuration `consat` file by typing:

```
$ consat init
```

This will produce a file `consat.cfg` in the same directory where you run `consat`. In order to properly run the ConSAT pipeline, open the configuration file and modify the fields according to your needs. There is a proper description of each field in that file, explaining what it is for and what values you can put it there. After this, you can re-run ConSAT with your own configuration file

(assuming this is `config.cfg` and still is located in the same directory) by doing:

```
$ consat -c config.cfg
```

This will run ConSAT and will show in the standard output the progress of the calculations. Depending on the length of your set of protein sequences this could take from a few minutes to a few hours.

4. How to use ConSAT

This is the "long" version of the mini-manual presented in section 3. A quick introduction to ConSAT. Here we will go into deeper detail and will explain further options beyond the basic ones.

4.1 The ConSAT modes: *manual* and *automatic*

ConSAT can be used in two different modes: the *manual* and the *automatic* one. In the *manual* mode, the user will have to proceed as stated in section 3. A quick introduction to ConSAT: a configuration file will have to be created, the parameters will have to be filled in, and finally, ConSAT will be run with that configuration file. Assuming success in the execution, the output files will be located in the output folder.

There is even an easier way to run ConSAT, and this is the *automatic* mode. While in the manual mode the user could specify the values of the parameters, in the automatic mode there is no such possibility (except for a few options). However, in the *automatic* mode ConSAT will download almost every needed data file, and will free the user from this tedious task.

To run ConSAT in the automatic mode, you will use the script `automated_consat`. For instance, having a FASTA file of proteins named `proteins.fasta`, an InterPro annotation of these proteins in `proteins.interpro` and a GFams release in `gfams_1.0.hmm`, we can run the following command:

```
$ automated_consat -s proteins.fasta -i proteins.interpro -m
gfams_1.0.hmm consat result
```

The work files, the downloaded data and the output of ConSAT will be written to the `result` directory (last parameter). With the parameter "consat" we are noting that the automated script will run ConSAT. There is, as well, the possibility of an "automated GFam mode" (see next section).

All the needed files will be downloaded in their more recent versions: the Gene Ontology file, the translation table between InterPro terms and GO terms (interpro2go), the descriptions for the domains from other databases (Pfam, Panther, etc.), and the hierarchical relations between the different InterPro domains. In the user-specified result directory, a `data` directory will be created containing the download files, and two directories (`work` and `output`) will contain the intermediate and the final produced files, respectively. A configuration file will be created, as well, in the user-specified result directory. Thus, if the same command is run in a different moment within a different result directory, it is not guaranteed that the results are 100% the same (because new data files, different from the first ones, could have been downloaded into your data directory, therefore changing your final results).

If you want to reproduce your results from a previously run experiment in automatic mode you should do the following:

- Get your `data` directory and your `consat.cfg` file from the result folder and move them into a different place (or even, different computer).
- Run ConSAT with that configuration file, making sure that the routes in that file correspond to the routes of the data folder, wherever you have placed it.

4.2 Running GFam

Having installed ConSAT, it is possible to run Gfam (http://www.paccanarolab.org/software/gfam/) almost as in the original way. After the installation, the executable `gfam` should be available. This will allow to run GFam on your set of sequences with the following differences:

- The efficiency of the process should have improved with respect to the original version.
- Some bugs have been changed from the original release (eventually they should be corrected in the GFam software as well).
- The treatment of domain insertions is completely new, and they are explicitly represented as insertions in the corresponding output files.
- HMMs are learnt from the clusters found. In order to do so, the software packages clustal-omega and HMMer are needed.
- A method for associating functions transferred from proteins in a GOA file having the same architecture has been developed. Therefore, new options in the configuration file have been added.
- A method for combining the functional labels obtained by over representation of GO terms obtained from InterPro-assigned domains to the architecture plus those obtained from transferred from a GOA file is available (you would have an extra file in the output folder).

If you want, you can run GFam in an automatic mode, using the same `automated_consat` script with the following format:

```
$ automated_consat -s proteins.fasta -i proteins.interpro gfam result
```

Note that, in this case, no models should have to be provided (GFam does not need the Gfams HMM models), and the mode will be now `gfam` (the parameter before the last). Everything which we pointed out for the "automatic" mode of ConSAT holds too for this GFam mode.

4.3 Understanding the representation of an architecture

In all the output files regarding protein architectures we give a textual representation of the architecture. An architecture is either `NO_ASSIGNMENT` (no architecture was found for that protein sequence) or a protein architecture expression.

Protein architecture expressions are composed of three things (being the second and the third optional):

1. Protein domain names (e.g. IPR00001, PTH024420, GFAM00010).
2. Semicolons (";")
3. Curly braces ("{" and "}").

Semicolons separate protein domains found one next to the other. Thus, the architecture IPR00001;PTH024420 can be understood as the domain IPR00001 followed by the domain PTH024420. Semicolons mark domain juxtaposition, and should be used between any two domains appearing in the protein domain architecture.

Curly braces mark that the containing domain (or list of domains) was found inside the preceding domain. If we find an architecture like this: IPR00001;PTH024420{IPR00007} this means that the domain IPR00001 is followed by the domain PTH024420, inside which a hit of the domain IPR0007 was found.

For computer scientists, this is the BNF grammar for protein domains we are using in ConSAT:

```
<protein_architecture>::= “NO_ASSIGNMENT” | <domain_expression>
<domain_expression> ::= <domain> [“;” <domain_expression> ]
<domain> ::= <domain_name> [ “{” <domain_expression> “}”]
<domain_name> ::= any domain name from the different domain Dbs
```

4.4 Associating functional labels with ConSAT

We provide Gene Ontology terms for the protein architectures found. These terms come from two different methods:

- **Over represented terms from InterPro domains:** once the architecture has been found, the set of GO terms associated to the InterPro terms in the architecture is retrieved (using the InterPro2GO table). Once we get this set, an over-representation analysis is carried out over it, and only significant terms are kept (the significance level, by default is 0.05). The over representation analysis is carried out against the InterPro2GO table (this defines de prior distribution for any GO term).
- **Transferred terms from a GOA file:** if we provide a GOA file with functions for some of our proteins, we can use this to assign GO terms to our sequences. The algorithm is the following:

    For each architecture, we retrieve the set of proteins matching it

        For each protein in that architecture, we retrieve its set of assigned GO terms

            For each GO term, we carry out an over representation analysis with respect to the GOA file (that defines the background distribution for each GO term)

                If the test is passed at a certain alpha, the term is assigned to the architecture

    Again, we use alpha equal to 0.05.

We provided, as well, a method combining both set of assigned GO terms. The combined *p*-value for each GO term is obtained by using Fisher's method (you can check http://en.wikipedia.org/wiki/Fisher's_method for details). Note that, when a GO term is associated by only one of the methods it will get that same *p*-value as the combined one.

5. Output files

We provide here a brief description of the most important output files produced by a normal

ConSAT execution.

File `domain_architecture_details.txt`:

In this file, a detail of the architecture assignment for each individual protein sequence is given. The file is in a very easy format, with a few lines for each protein sequence, like this example where two proteins are presented (one with some assignments in its architecture, and another without any architecture assigned). Note that both the original domain and the one assigned by InterPro are given:

```
N1VAX8
    Primary assignment source: HMMPfam
    Number of data sources used: 1
    Data sources: HMMPfam
    Coverage: 0.655
    Coverage w/o novel domains: 0.655
     15- 257: PTHR24220 (HMMPfam, stage: 1)
     17- 255: SSF52540 (HMMPfam, stage: 1) (InterPro ID: IPR027417)
               P-loop containing nucleoside triphosphate hydrolase
    34- 179: PF00005 (HMMPfam, stage: 1) (InterPro ID: IPR003439 --> IPR027417)
               P-loop containing nucleoside triphosphate hydrolase
N1VAX9
    Primary assignment source: None
    Number of data sources used: 0
    Data sources:
    Coverage: 0.000
    Coverage w/o novel domains: 0.000
```

File `domain_architectures.tab`:

Tab separated file with the following fields for each protein:

- Protein id
- Starting position covered by the architecture
- Ending position covered by the architecture
- Length covered by the architecture
- Architecture specifying the position of each domain (shown with an interval in parentheses).
- Description of the architecture formed by the concatenation of the individual domain descriptions (if any).

File `overrep_by_arch.txt`:

For each architecture, it shows the functional prediction (GO terms) together with the *p*-value, which is associated to each architecture via the overrepresentation of GO terms associated by InterPro to each domain.

A line with the architecture is shown, followed by the functional predictions for that architecture, indented (if any). Architectures are separated among each other with a double carriage return.

For this kind of files, the prediction is shown at the predefined significance level (0.05 by default), but if the file ends in "_unfiltered", all predictions are shown regardless of their *p*-value.

File `transfer_by_arch.txt`:

Same kind of file that the overrep_by_arch.txt. Functional labels are obtained now by transferring experimental GOA terms from proteins with the same architecture and then running overrepresentation on the set of transferred labels. The format is the same as in the mentioned file.

File `combined_prediction_by_arch.txt`:

It gives the predictions given in the files transfer_by_arch and overrep_by_arch combined using Fisher's method. The format is the same as in those files.

File `weight_file_per_arch.txt`:

It is a tab separated file with two columns. In the first one we have architectures, and in the second we have a vector of assigned terms (identified by their integer id) and weights, separated by space. For example, this could be a line:

IPR007541;IPR000772{G3DSA:2.80.10.50} 3330:6.32311 3:4.48

The mentioned architecture has two terms assigned, the 3330 and the 3, weighting 6.32311 and 4.48, respectively. The mapping between term identifiers and term strings is given by the lexicon file.

File `text/lexicon`:

Maps the words associated to architectures and their identifiers. For each line it shows a triplet:

<word> <word_id> <doc_freq>

Where <word> is the word itself, <word_id> is a positive number, unique for each word, and <doc_freq> is the number of documents (sequences) containing this word.

6. The ConSAT algorithm

We give here the pseudocode of the ConSAT algorithm

```
FOR each protein p:
     architecture = {}
     assignment_length = {}
```

```
        sorted_p = sort_by_length(assignments(p))
        FOR each InterPro Assignment a in sorted_p:
             IF not overlaps(a, assignment_length[a.source]):
                  assignment_length[a.source].add(a)
        primary_source = arg_max(assignment_length)
        architecture.add_all(assignment_length[primary_source])
        FOR each stage in Stages:
             sources = get_sources(stage)
             s = filter_by_source(assignments(p), sources)
             s = sort_by_length(s)
             FOR each InterPro assignment a in s – architecture:
                  IF not overlaps(a, architecture):
                       architecture.add(a)
        r = find_unassigned_regions(p, architecture)
        r = r – low_complexity_regions(p)
        FOR each region in r:
             hits = hmm_scan(GFams, r)
             hits = sort_hits_by_evalue(hits)
             FOR each hit in hits:
                  IF not overlaps(hit, architecture):
                       architecture.add(hit)
    return architecture
```

7. The GFams dataset

The GFams dataset is a set of putative domains found in UniProt. We say they are *putative* because they have been obtained by purely computational methods, although we believe they could be protein domains from a computational point of view. The GFams dataset is obtained by the following procedure:

1. GFam is applied on a certain SwissProt release.
2. The clusters found (see GFam paper) are chosen as domains.
3. A HMM is learnt in each cluster to predict future occurrences of this domain.

The GFams database has its own release number and we plan to rebuild them from time to time, but not so frequently as the ConSAT database. Given that the rate of growth of SwissProt is much less than the one of the whole UniProt, we think that we do not need to relearn SwissProt with every UniProt release.

The HMMs for the GFams are needed for running ConSAT and can be downloaded from the same website of the ConSAT web server: http://paccanarolab.org/consat.

8. Version History of this software

1.0 (March 2014):

- first version of ConSAT.