AACon Documentation

Release 1.1

Peter Troshin, et al.

Apr 21, 2017

CONTENTS:

1 Getting Started **3**
- 1.1 Benefits . . . 3
- 1.2 Distributions . . . 3
  - 1.2.1 Jalview and JABAWS . . . 4
  - 1.2.2 Standalone Client . . . 4
  - 1.2.3 Web service . . . 4

2 Included Methods **5**
- 2.1 Vadar . . . 5
- 2.2 SMERFS . . . 5
  - 2.2.1 References . . . 6

3 Standalone Client **7**
- 3.1 Executing AACon . . . 7
- 3.2 Input . . . 7
- 3.3 Output . . . 8
- 3.4 Calculating conservation . . . 9
- 3.5 Custom gap character . . . 10
- 3.6 Running SMERFS with custom parameters . . . 10
- 3.7 Outputing to a file . . . 10
- 3.8 Execution details . . . 11
- 3.9 Results normalization . . . 11
- 3.10 Conservation for large alignments . . . 11

4 Web Service **13**
- 4.1 Web Service Details . . . 13
- 4.2 Using Web Service CLI Client . . . 13
- 4.3 Structure of the command line client . . . 14
- 4.4 AACon web service functions overview . . . 14
- 4.5 Building web services artifacts . . . 15
- 4.6 Connecting to AAConWS . . . 15
- 4.7 Calculating conservation . . . 15
- 4.8 Checking the status of the calculation . . . 16
- 4.9 Calculating conservation with presets . . . 16
- 4.10 Calculating conservation with custom parameters . . . 16
- 4.11 A complete AAConWS web service client example . . . 17

5 Java Library **21**
- 5.1 Using the library . . . 21
- 5.2 Loading data . . . 21

5.3 Initialising the executor . . . 22
5.4 Normalisation . . . 22
5.5 Setting the conservation methods . . . 22
5.6 Output the results . . . 23
5.7 Example . . . 23

6 Citations **25**

7 Changelog **27**
7.1 Version 1.1 (Released 19 April 2017) . . . 27
7.2 Version 1.0 (Released 16th Dec 2011) . . . 27

Note: Some of the links might not work properly in the pdf version...

CHAPTER

ONE

GETTING STARTED

AACon is a set of tools implementing 17 different conservation scores reviewed by Valdar as well as the more complex SMERFS algorithm for predicting protein functional sites (list of included methods). AACon has been written with efficiency in mind and takes less than a second to calculate conservation by all 18 methods for an alignment of 500 sequences 350 residues long on a single CPU. AACon exploits parallelism for the more demanding methods and to allow multiple methods to run simultaneously. The parallel code gives close to linear speedup with the number of processors, thus making it suitable for server applications or other demanding environments.

AACon is available as a SOAP web service, a stand alone Java executable and a Java library with concise API for accessing all the conservation methods programmatically. The executable runs on all platforms that support Java version 6 and above (Windows, Unix/Linux, Mac).

1.1 Benefits

- Implements 17 different conservation score metrics from Valdar's paper in addition to SMERFS.
- Can be deployed on most operating systems, as a Tomcat Java Web Application or Standalone executable client.
- Can be accessed from Jalview using its graphical client, or using the JABAWS command line client, in addition to the AACon standalone client.
- Takes advantage of parallelisation techniques to improve the speed of demanding analysis.
- Local or intranet installation eliminates any security concerns you might have about sending sensitive data over the internet.

1.2 Distributions

Tip: To help you choose the AACon distribution that better suits your needs and read on the quick-start guides below.

I want to use AACon for...

- *Jalview and JABAWS* - Running AACon services through Jalview via a *public* or *private* JABAWS server
- *Standalone Client* - Running AACon from the command line using the AACon Standalone Client
- *Web service* - Running AACon Web Service Client

1.2.1 Jalview and JABAWS

Jalview is a multiple sequence alignment and analysis application that can be used as a graphical AACon client (via JABAWS). Jalview provides the same AACon functionality as the JABAWS Command Line Interface (CLI) client. Simply launch Jalview and run any of the methods provided under the 'Web Service >> Conservation' menu. Jalview uses the *public* JABAWS server by default, but by running the JABAWS Virtual Appliance (VA) or JABAWS Web Application aRchive (WAR) in your local infrastructure, a *private* server can also be used. Alternatively, AACon can be run using its dedicated Standalone Client or Web Service Client.

1.2.2 Standalone Client

The standalone executable is for calculating conservation from the command line. The Java library provides simple API for the Java developers who wants to have AACon functionality from their applications. The executable and the library comes from the same jar file and will work on any operation system that supports Java.

Command line executable and the Java library: download

Help with a command line client is available from standalone executable page. The library is described on the library page.

1.2.3 Web service

The AACon web service, is a standard SOAP web service and thus you can use your favorite programming language to access it. That's more, to simplify the use of web service even further, we offer a Java web service client. This client can submit the tasks to a web service, retrieve the results and thus can be used in place of the command line client. Alternatively, it can serve as an example of coding against AACon web service.

Web service client: download

Please refer to web service help page for a help on using AACon web service.

CHAPTER

TWO

INCLUDED METHODS

2.1 Vadar

Protein Conservation Metrics described in the paper by Vadar[1]:

- KABAT (*Wu and Kabat, 1970*)
- JORES (*Jores et al., 1990*)
- SCHNEIDER (*Schneider, 1997*)
- SHENKIN (*Shenkin et al., 1991*)
- GERSTEIN (*Gerstein and Altman, 1995*)
- TAYLOR_GAPS (*Taylor, 1986*)
- TAYLOR_NO_GAPS (*Taylor, 1986*)
- ZVELIBIL (*Zvelibil et al., 1987*)
- KARLIN (*Mirny and Shakhnovich, 2001*)
- ARMON (*Armon et al., 2001*)
- THOMPSON (*Thompson et al., 1997*)
- NOT_LANCET (*Pilpel and Lancet, 1999*)
- MIRNY (*Mirny and Shakhnovich, 1999*)
- WILLIAMSON (*Williamson, 1995*)
- LANDGRAF (*Landgraf et al., 1999*)
- SANDER (*Sander and Schneider, 1991*)
- VALDAR (*Valdar and Thornton, 2001*)

2.2 SMERFS

A method for predicting protein functional sites developed in the Barton Group[2]:

[1] Valdar, W. S. J. Scoring residue conservation. Proteins: Structure, Function, and Genetics 48, 227–241 (2002). DOI: 10.1002/prot.10146

[2] Manning, J. R., Jefferson, E. R. & Barton, G. J. The contrasting properties of conservation and correlated phylogeny in protein functional residue prediction. BMC Bioinformatics 9, 51 (2008). DOI: 10.1186/1471-2105-9-51

- SMERFS (*Manning et al., 2008*, more about available custom parameters here)

2.2.1 References

CHAPTER

THREE

STANDALONE CLIENT

This page provides some tips on using AACon command line tool.

3.1 Executing AACon

Type java -jar <AACon jar file name>

```
java -jar compbio-conservation-1.1.jar
```

This will print AACon help.

3.2 Input

Input to AACon can be either

- The list of aligned FASTA formatted sequences
- The Clustal formatted alignment

FASTA example

```
>UniRef90_Q5VFX
MEWLKIALVAFSGRGNDVDYPCSILGRVANNDKELVNILRNGFFLLTYRMNFSPLPHSSVTSDKGWGCLVRS
SQMLLAHALWRYSANDCRLDHFRDMDTEDSTPFSLHKMVRAVMKKADVFRPEYWTPSQGCEAIRCCVNNAVD
RKLIPPIRVVVCSQGCLLAREICSNLEFGTVLILAPMRCGASRRMTQMMFFSLEHLLHSSACIGVVGGVPQR
SYYILGTSGQRLLYLDPHCMTQEALVSSHAEKAGVVTVTASLVKSVRWDCVDTSCFLGFLVDSFAEWLELRT
CLEELQRRGMEQLLCVDDGVAVGLVDEEIAGWPSEEDVAE
>UniRef90_Q4VBK1
---------------PDTDEPVWILGACYNVKTKKSELLSDSRLWFTYRKKFSPIGGTGPSSDAGWGCMLRC
GQMILAQALWRWDPEKHQPKEYQRFLDKKDSCYSIHQMAQMGVGE-GKSVGEWYGPNTVAQVLKKL----AL
FDDWNSLSVYVSMDNTVVIEDIKKLCDWRPLLLVIPLRMGINS-INPVYIQALKECFKMPQSCGVLGGKPNL
AYYFIGFIDDELIYLDPHTTQQA-VDTESGSAVDDQSFHCQRPHRMKITSLDPSVALGFFCKSEEDFDSWCD
LVQ-------QELLKKRNLRMFELVEKHPSHWPPF-----
>UniRef90_A7RTH8
---MDAACVTYEGVSHETEEDVWILGKRYNIDMGYLNTDVRSRIWLTYRKNFPKIGGTGPTTDSGWGCMLRC
GQMMLAQALWQWDPENEYMQILEAFLDKKDSLYSIHQIAQMGVSE-GKAVGSWFGPNTVAQVLKKL----SA
FDDWSSLCLHVAMDNTVIIEDISN---WRPLVLFIPLRLGLTEM-NVVYNEPLKACFTFKQSLGIIGGRPNH
ATYFIGYFGNNLVYLDPHTTQQTVN-PDELSRIPDGSFHCVYPCRMNIADVDPSVALGFFCKSEEDFDDLCQ
QIKKIIDGKSRPMFEIAK--------DRPQHWPVLE----
```

Clustal example

```
CLUSTAL

QUERY/1-328 MEWLKIALVAFSGRGNDVDYPCSILGRVANNDKELVNILRNGFFLLTYRMNFSPLPHSSV
UniRef90_Q4VBK1/1-294 ---------------PDTDEPVWILGACYNVKTKKSELLSDSRLWFTYRKKFSPIGGTGP
UniRef90_A7RTH8/1-303 ---MDAACVTYEGVSHETEEDVWILGKRYNIDMGYLNTDVRSRIWLTYRKNFPKIGGTGP

QUERY/1-328 TSDKGWGCLVRSSQMLLAHALWRYSANDCRLDHFRDMDTEDSTPFSLHKMVRAVMKKADV
UniRef90_Q4VBK1/1-294 SSDAGWGCMLRCGQMILAQALWRWDPEKHQPKEYQRFLDKKDSCYSIHQMAQMGVGEGKS
UniRef90_A7RTH8/1-303 TTDSGWGCMLRCGQMMLAQALWQWDPENEYMQILEAFLDKKDSLYSIHQIAQMGVSEGKA

QUERY/1-328 FRPEYWTPSQGCEAIRCCVNNAVDRKLIPPIRVVVCSQGCLLAREICSNLEFGTVLILAP
UniRef90_Q4VBK1/1-294 VG-EWYGPN----TVAQVLKKLALFDDWNSLSVYVSMDNTVVIEDIKKLCDWRPLLLVIP
UniRef90_A7RTH8/1-303 VG-SWFGPN----TVAQVLKKLSAFDDWSSLCLHVAMDNTVIIEDIS---NWRPLVLFIP

QUERY/1-328 MRCGASRRMTQMMFFSLEHLLHSSACIGVVGGVPQRSYYILGTSGQRLLYLDPHCMTQEA
UniRef90_Q4VBK1/1-294 LRMGIN-SINPVYIQALKECFKMPQSCGVLGGKPNLAYYFIGFIDDELIYLDPH-TTQQA
UniRef90_A7RTH8/1-303 LRLGLT-EMNVVYNEPLKACFTFKQSLGIIGGRPNHATYFIGYFGNNLVYLDPH-TTQQT

QUERY/1-328 LVSSHAEKAGVVTVTASLVKSVRWDCVDTSCFLGFLVDSFAEWLELRTCLEELQRRGMEQ
UniRef90_Q4VBK1/1-294 VDTESGSAVDDQSFHCQRPHRMKITSLDPSVALGFFCKSEEDFDSWCDLVQQ-------E
UniRef90_A7RTH8/1-303 VNPDELSRIPDGSFHCVYPCRMNIADVDPSVALGFFCKSEEDFDDLCQQIK--------K

QUERY/1-328 LLCVDDGVAVGLVDEEIAGWPSEEDVAE
UniRef90_Q4VBK1/1-294 LLKKRNLRMFELVEKHPSHWPPF-----
UniRef90_A7RTH8/1-303 IIDGKSRPMFEIAKDRPQHWPVLE----
```

The gaps in the alignment can be represented by *, -, space character, X and . (a dot). Other, custom gap characters can also be defined.

3.3 Output

AACon supports two modes of the output the first and the default mode is *RESULT_NO_ALIGNMENT*. Only the method names and the conservation score for each position in the alignment are presented. The second output format is *RESULT_WITH_ALIGNMEN*, where the input, in the form of FASTA formatted alignment, is included at the beginning of the file, followed by the same output as *RESULT_NO_ALIGNMENT*.

RESULT_NO_ALIGNMENT example is below:

```
#KABAT 0.75 0.75 0.75 0.375 0.375 0.375 0.938 0.375 0.938 0.375 0.37 0.3
#JORES 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
#SCHNEIDER 0.667 0.667 0.667 0.333 0.333 0.333 0.754 0.333 0.754 0.333 0
#SHENKIN 0.779 0.779 0.779 0.46 0.46 0.46 0.845 0.46 0.845 0.46 0.46 0.4
#GERSTEIN 0.667 0.667 0.667 0.333 0.333 0.333 0.754 0.333 0.754 0.33 0.3
#TAYLOR_GAPS 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.632 1 0 1 0.526 0.842 0 0.5
#TAYLOR_NO_GAPS 0 1 0.526 0 0.895 0 0.947 0 0 0.789 0 0.842 0.947 0 0.94
#ZVELIBIL 0.1 0.3 0.4 0.1 0.2 0.1 0.3 0.1 0.3 0.2 0.2 0.1 0.3 0 0.2 0.2
#KARLIN 0.338 0.338 0.338 0.433 0.31 0.264 0.558 0.288 0.558 0.338 0.431
#ARMON 0.617 0.617 0.617 0.208 0.103 0.063 0.617 0.129 0.617 0.177 0.204
#THOMPSON 0.719 0.727 0.708 0.451 0.394 0.384 0.471 0.374 0.49 0.428 0.4
#NOT_LANCET 0.011 0.005 0.054 0.172 0.14 0.108 0.194 0.145 0.194 0.1 0.2
#MIRNY 0.667 0.667 0.667 0.754 0.333 0.754 0.754 0.754 0.754 0.333 0.754
#WILLIAMSON 0.884 0.953 0.971 0.503 0.511 0.445 0.645 0.552 0.503 0.522
#LANDGRAF 0.752 0.726 0.847 0.539 0.436 0.25 0.433 0.476 0.547 0.308 0.6
```

```
#SANDER 0 0 0 0.16 0.107 0.093 0.133 0.027 0.16 0.12 0.2 0.107 0.17 0.01
#VALDAR 0 0 0 0.187 0.115 0.115 0.256 0.046 0.256 0.164 0.233 0.092 0.25
#SMERFS 0.625 0.625 0.625 0.625 0.446 0.3 0.232 0.232 0.232 0.232 0.232
```

RESULT_WITH_ALIGNMENT output example is below:

```
>QUERY
MEWLKIALVAFSGRGNDVDYPCSILGRVANNDKELVNILRNGFFLLTYRMNFSPLPHSSVTSDKGWGCLVRSSQMLLAHA
LWRYSANDCRLDHFRDMDTEDSTPFSLHKMVRAVMKKADVFRPEYWTPSQGCEAIRCCVNNAVDRKLIPPIRVVVCSQGC
LLAREICSNLEFGTVLILAPMRCGASRRMTQMMFFSLEHLLHSSACIGVVGGVPQRSYYILGTSGQRLLYLDPHCMTQEA
LVSSHAEKAGVVTVTASLVKSVRWDCVDTSCFLGFLVDSFAEWLELRTCLEELQRRGMEQLLCVDDGVAVGLVDEEIAGW
PSEEDVAE
>UniRef90_Q4VBK1
---------------PDTDEPVWILGACYNVKTKKSELLSDSRLWFTYRKKFSPIGGTGPSSDAGWGCMLRCGQMILAQA
LWRWDPEKHQPKEYQRFLDKKDSCYSIHQMAQMGVGE-GKSVGEWYGPNTVAQVLKKL----ALFDDWNSLSVYVSMDNT
VVIEDIKKLCDWRPLLLVIPLRMGINS-INPVYIQALKECFKMPQSCGVLGGKPNLAYYFIGFIDDELIYLDPHTTQQA-
VDTESGSAVDDQSFHCQRPHRMKITSLDPSVALGFFCKSEEDFDSWCDLVQ-------QELLKKRNLRMFELVEKHPSHW
PPF-----
>UniRef90_A7RTH8
---MDAACVTYEGVSHETEEDVWILGKRYNIDMGYLNTDVRSRIWLTYRKNFPKIGGTGPTTDSGWGCMLRCGQMMLAQA
LWQWDPENEYMQILEAFLDKKDSLYSIHQIAQMGVSE-GKAVGSWFGPNTVAQVLKKL----SAFDDWSSLCLHVAMDNT
VIIEDISN---WRPLVLFIPLRLGLTEM-NVVYNEPLKACFTFKQSLGIIGGRPNHATYFIGYFGNNLVYLDPHTTQQTV
N-PDELSRIPDGSFHCVYPCRMNIADVDPSVALGFFCKSEEDFDDLCQQIKKIIDGKSRPMFEIAK--------DRPQHW
PVLE----
#KABAT 3 3 3 6 6 6 1.5 6 1.5 6 6 6 1.5 6 6 9 3 3 3 3 3 3 1 1 1 9 9 3 1 9 3 9 9 9 9␣
↪3 9 3 9 9 3 3 6 6 6 6 6 9 9 3 3 9 9 9 9 6 6 6 6 6 1.5 1.5 6 9 9 3 9 3 1 1 9 9 1.5 3␣
↪3 3 3
#SANDER -72 -72 -72 -36 -48 -51 -42 -66 -36 -45 -27 -48 -33 -54 -45 3 30 9 30 -12 12␣
↪0 39 45 54 45 -18 -36 -72 -72 -72 -72
```

Warning: Please note that only a small part of the output is presented in all examples that follow for the sake of simplicity. Therefore the results are for illustration only. Normally each position of the alignment is given a conservation score by each calculation method.

`-f` option can be used to specify the output format. For example the command

```
java -jar compbio-conservation-1.1.jar -i=data.align -m=KABAT -f=RESULT_WITH_ALIGNMENT
```

outputs the input alignment into the result file.

3.4 Calculating conservation

Calculate conservation using KABAT conservation method

The alignment is read from the data.align file.

```
java -jar compbio-conservation-1.1.jar -i=data.align -m=KABAT
```

The above command will print the results to the console. You should see something like this:

```
#KABAT 3 3 3 6 6 6 1.5 6 1.5 6 6 6 1.5 6 6 9 3 3 3 3 3 3 1 1 1 9 9
```

Calculate conservation using two or more methods, for example SANDER and KABAT

Just specify the list of comma separated methods after `-m` switch like this:

java -jar compbio-conservation-1.1.jar -i=data.align -m=SANDER,KABAT

The results will look something like this:

```
#KABAT 3 3 3 6 6 6 1.5 6 1.5 6 6 6 1.5 6 6 9 3 3 3 3 3 3 3 1 1 1 9 9 3
#SANDER -72 -72 -72 -36 -48 -51 -42 -66 -36 -45 -27 -48 -33 -54 -45 3 30
```

Calculate conservation using all supported conservation methods and make results comparable

```
java -jar compbio-conservation-1.1.jar -i=data.align -n
```

The results will be printed to the console. Where `-n` normalizes all results.

3.5 Custom gap character

Assuming that the gaps in the alignment are represented by '-', '_' and '*x*' symbols the following command will interpret them correctly.

```
java -jar compbio-conservation-1.1.jar -i=data.align -m=KABAT -g=-,_,x
```

3.6 Running SMERFS with custom parameters

Unlike other methods, SMERFS supports a few custom parameters, in particular

1. The window width - an integer and an odd number
2. Two methods of window scores to columns allocation:

   MID_SCORE - gives the window score to the middle column *MAX_SCORE* - gives the column the highest score of all the windows it belongs to
3. A gap percentage cutoff - a float greater than 0 and smaller or equal 1

For example:

```
java -jar compbio-conservation-1.1.jar -i=data.align -m=SMERFS -s=5,MID_SCORE,0.1
```

3.7 Outputing to a file

Use -o option to print the results to the file instead of a console. For example

```
java -jar compbio-conservation-1.1.jar -i=data.align -n -o=outfile.txt
```

will produce output.txt results file. Nothing will be printed to the console.

3.8 Execution details

Use -d option to tell AACon to output its execution details.

```
java -jar compbio-conservation-1.1.jar -i=data.align -n -d=stat.out -f=RESULT_WITH_
↪ALIGNMENT
```

The output of the previous command, the context of the stat.out file is below

```
No methods are request assuming all are required.
No output file is provided, writing results to the standard output.
Setting output format to RESULT_WITH_ALIGNMENT
Using 4 CPUs
Start time: 2010/12/08 12:02:35
Alignment loaded in: 115 ms
Alignment has: 3 sequences.
Alignment length is: 328
KARLIN 15 ms
LANDGRAF 25 ms
SANDER 2 ms
VALDAR 2 ms
SMERFS 3 ms
SCHNEIDER 1 ms
KABAT 1 ms
SHENKIN 3 ms
JORES 5 ms
GERSTEIN 3 ms
ARMON 1 ms
THOMPSON 4 ms
NOT_LANCET 4 ms
ZVELIBIL 13 ms
MIRNY 4 ms
TAYLOR_GAPS 20 ms
WILLIAMSON 4 ms
TAYLOR_NO_GAPS 21 ms
Total calculation time: 0 s
End time: 2010/12/08 12:02:36
```

3.9 Results normalization

Different conservation algorithms produce vastly different values. To compare them meaningfully one need to bring the results into the same range. AACon bring the results to the range between 0 and 1 if `-n` switch is used.

3.10 Conservation for large alignments

Attention: Troubleshooting Java VM running out of memory

To enable AACon to deal with large alignments (thousands of sequences) let the Java Virtual Machine use more memory with the flag `-Xmx<AMOUNT_OF_MEMORY>` to your command line as follows:

```
java -Xmx1G -jar aacon.jar <options>
```

Where `1G = 1` gigabyte of memory, the same result can be achieved with the following instruction `-Xmx1000M`. However, floating point numbers like `-Xmx1.5G` are not allowed.

CHAPTER

FOUR

WEB SERVICE

This page provides some tips on using AACon web service. Please note that for now all the examples are in Java. Other languages will follow given a sufficient demand. Please also note that AACon web service client requires Java 6 or later.

4.1 Web Service Details

Feature	Description
Operated by	The University of Dundee, Computational Biology Group headed by Prof Geoff Barton. It is backed up by the College of Life Sciences HPC cluster.
Service web site	http://www.compbio.dundee.ac.uk/aacon
Execution limits	Tasks exceeding 5000×1000 (sequences per letters) will not be accepted for alignment. If you would like to work with bigger alignments consider using a command line version of AACon.
AACon web service WSDL	http://www.compbio.dundee.ac.uk/aacon/AAConWS?wsdl

Tip: For a more detailed description of all available types and their functions please refer to the data model javadoc.

4.2 Using Web Service CLI Client

The command client can be used to calculate conservation using public AACon web service. The client is operating system independent and supports most of the AACon web service functions. Using this client you could calculate conservation using web service features such as presets and parameters. Below is the list of options supported by the command line client.

```
Usage: java -jar <path_to_jar_file> ACTION [OPTIONS]
ACTIONS:
-i=<inputFile> - full path to Fasta or Clustal formatted alignment file
-parameters - lists parameters supported by web service
-presets - lists presets supported by web service
-limits - lists web services limits
Please note that if input file is specified other actions are ignored
OPTIONS: (only for use with -i action):
```

```
-r=<presetName> - name of the preset to use
-o=<outputFile> - full path to the file where to write the results
-f=<parameterInputFile> - the name of the file with the list of parameters to use.
Please note that -r and -f options cannot be used together. Conservation can be␣
↪calculated with either a preset or the parameters from the file, but not both!
```

Using the command line web service client is easy. For example to calculate conservation for the alignment from input.fasta file using AACon web service with default settings and print the results to the console. By default, the conservation is calculated by *SHENKIN* method.

```
java -jar aacon-client.jar -i=d:\input.fasta
```

Content of input.fasta file is show below (please note sequences has been trimmed for clarity)

```
>QUERY
MEWLKIALVAFSGRGNDVDYPCSILGRVANNDKELVNIL
>Q4VBK1
----------------PDTDEPVWILGACYNVKTKKSELL
>A7RTH8
---MDAACVTYEGVSHETEEDVWILGKRYNIDMGYLNTD
```

Calculate conservation using 12 fast conservation methods (using preset “Quick conservation”), write output results to the result.txt file

```
java -jar aacon-client.jar -i=d:\input.fasta -o=d:\result.txt -r="Quick conservation"
```

Get the list of presets supported by AACon web service

```
java -jar aacon-client.jar -presets
```

4.3 Structure of the command line client

Packages	Classes and Interfaces
compbio.data.msa	Annotation interface for all scoring methods
compbio.data.sequence	AAConWS data types
compbio.metadata	AAConWS meta data types
compbio.ws.client	AAConWS command line client package with AAConClient.java class containing the main method

4.4 AACon web service functions overview

Functions for conservation calculation

```
String id = analize(List<FastaSequence> list)
String id = customAnalize(List<FastaSequence> sequenceList, List<Option> optionList)
String id = presetAnalize(List<FastaSequence> sequenceList, Preset preset)
```

Functions pertaining to job monitoring and control

```
JobStatus status = getJobStatus(String id)
HashSet<Score> conservation = getConservation(String id)
boolean cancelled = cancelJob(String id)
ChunkHolder chunk = pullExecStatistics(String id, long marker)
```

Functions relating to service features discovery

```
RunnerConfig rc = getRunnerOptions()
Limit limit = getLimit(String name)
LimitsManager lm = getLimits()
PresetManager pm = getPresets()
```

4.5 Building web services artifacts

AAConWS are the standard JAX-WS SOAP web services, which are WS-I basic profile compatible. This means that you could use your favorite programming language to work with AAConWS. Below is how you can generate portable artifacts to work with AAConWS from Java. However, if programming in Java we recommend using our client library as it provides a handful of useful methods in addition to plain data types.

wsimport -keep http://www.compbio.dundee.ac.uk/aacon/AAConWS?wsdl

---

4.6 Connecting to AAConWS

All the examples below assume that AACon web service command line client is in the classpath. You can download it from the download page. The code excerpt below will connect your program to AAConWS web service deployed in the University of Dundee.

```
Annotation<AAConWS> client = AAConWSClient.connect();
```

If you want to work with the generated artifacts directly you can inspect AAConWS command line client source code and use it as a template for building your own custom AAConWS clients.

4.7 Calculating conservation

Given that `client` is web service proxy, created as described in "Connecting to AAConWS" section, the conservation scores can be obtained as follows:

```
1) List<FastaSequence> fsl = SequenceUtil.readFasta(new FileInputStream("alignment.
↪fasta"));
2) String jobId = client.analize(fsl);
3) HashSet<Score> result = client.getAnnotation(jobId);
```

Line one loads sequence alignment from the file Line two submits them to web service represented by AAConWS proxy Line three retrieves the conservation scores calculated according to Shenkin algorithm. This line blocks the execution until the result is available. Use this with caution. In general, you should make sure that the calculation has been completed before attempting retrieving results. This is to avoid keeping the connection to the server on hold for a prolonged periods of time. While this may be ok with your local server, our public server (www.compbio.dundee.ac.uk/aacon) will not let you hold the connection for longer than 10 minutes. This is done to prevent excessive load on the server. The next section describes how to check the status of the calculation. Methods and classes mentioned in the excerpt are available from the AAConWS client library.

4.8 Checking the status of the calculation

You may have noticed that there was no pause between submitting the job and retrieving the results. This is because getAnnotation(jobId) method block the processing until the calculation is completed. However, taking into account that the connection holds server resources, our public server (www.compbio.dundee.ac.uk/aacon) is configured to reset the connection after 10 minutes of waiting. To work around the connection reset you are encouraged to check whether the calculation has been completed before accessing the results. You can do it like this:

```
while (client.getJobStatus(jobId) != JobStatus.FINISHED) {
    Thread.sleep(2000); // wait two seconds, then recheck the status
}
```

4.9 Calculating conservation with presets

```
1) PresetManager<AACon> presets = client.getPresets();
2) String jobId = client.presetAnalize(fsl, presets.getPresetByName("Quick␣
↪conservation"));
3) HashSet<Score> result = client.getAnnotation(jobId);
```

Line one obtains the lists of presets supported by a web service. Line two calls web service `presetAnalise` method with a chosen preset. This call returns a job identifier. Lines three retrieves the results using job identifier.

Available presets are:

- "Quick conservation" (a collection of 12 fast conservation algorithms includes KABAT, JORES, SCHNEIDER, SHENKIN, GERSTEIN, TAYLOR_GAPS, TAYLOR_NO_GAPS, ZVELIBIL,ARMON, THOMPSON, NOT_LANCET, MIRNY, WILLIAMSON)
- "Slow conservation" (a collection of time consuming conservation algorithms includes LANDGRAF, KARLIN, SANDER, VALDAR and SMERFS)
- "Complete conservation" (all available algorithms)

4.10 Calculating conservation with custom parameters

Below is the example of using custom parameters for SMERFS method.

```
// Using options
1) RunnerConfig<AACon> options = client.getRunnerOptions();
2) options.getArgument("Calculation method").setDefaultValue("SMERFS");
3) options.getArgument("SMERFS Column Scoring Method").setDefaultValue("MAX_SCORE");
4) options.getArgument("SMERFS Gap Threshhold").setDefaultValue("1");
5) String jobId = client.customAnalize(fsl, options.getArguments());
6) HashSet<Score> result = client.getAnnotation(jobId);
```

Line one obtains the RunnerConfig object that holds information on supported parameters and their values Line two retrieve a particular parameter from the holder by its name and sets the new value for this parameter. Lines three and four do the same but for two more parameters Line five submit a job to a web service Line six retrieves the results of the analysis. The names of all the parameters supported by a web service can be obtained using options.getArguments() method. Further details on the methods available from RunnerConfig object are available from the javadoc.

4.11 A complete AAConWS web service client example

Finally, a complete example of the program that connects to AAConWS service is below. The text notes are commented by block style comments e.g. /* comment */, the code alternatives are commented out with the line comments - "//". You may want to remove line style comments to test alternatives of the functions. All you need for this to work is a AAConWS binary client. Please make sure that the client is in the Java class path before running this example.

```
import java.io.ByteArrayInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;
import java.util.Set;

import compbio.data.msa.Annotation;
import compbio.data.sequence.FastaSequence;
import compbio.data.sequence.Score;
import compbio.data.sequence.SequenceUtil;
import compbio.metadata.JobSubmissionException;
import compbio.metadata.Preset;
import compbio.metadata.PresetManager;
import compbio.metadata.ResultNotAvailableException;
import compbio.metadata.UnsupportedRuntimeException;
import compbio.metadata.WrongParameterException;
import compbio.runner.conservation.AACon;


/**
 * AAConWS client example
 */
public class AAConWSClientExample {

    /*
     * Input sequences for alignment. For the simplicity keep them in the class
     */
    static final String input = ">Foo      \r\n"
    + "MTADGPRELLQLRAAVRHRPQDFVAWLMLADAELGMGDTTAGEMAVQRGLALHPGHPEAV\r\n"
    + "ARLGRVRWTQQRHAEAAVLLQQASDAAPEHPGIALWLGHALEDAGQAEAAAAAYTRAHQL\r\n"
    + "LPEEPYITAQLLNWRRRLCDWRALDVLSAQVRAAVAQGVGAVEPFAFLSEDASAAEQLAC\r\n"
    + "ARTRAQAIAASVRPLAPTRVRSKGPLRVGFVSNGFGAHPTGLLTVALFEALQRRQPDLQM\r\n"
```

```
    + "HLFATSGDDGSTLRTRLAQASTLHDVTALGHLATAKHIRHHGIDLLFDLRGWGGGGRPEV\r\n"
    + "FALRPAPVQVNWLAYPGTSGAPWMDYVLGDAFALPPALEPFYSEHVLRLQGAFQPSDTSR\r\n"
    + "VVAEPPSRTQCGLPEQGVVLCCFNNSYKLNPQSMARMLAVLREVPDSVLWLLSGPGEADA\r\n"
    + "RLRAFAHAQGVDAQRLVFMPKLPHPQYLARYRHADLFLDTHPYNAHTTASDALWTGCPVL\r\n"
    + "TTPGETFAARVAGSLNHHLGLDEMNVADDAAFVAKAVALASDPAALTALHARVDVLRRES\r\n"
    + "GVFEMDGFADDFGALLQALARRHGWLGI\r\n"
    + "\r\n"
    + ">Bar                    \r\n"
    + "----------------------------------MGDTTAGEMAVQRGLALH-------\r\n"
    + "---------QQRHAEAAVLLQQASDAAPEHPGIALWL-HALEDAGQAEAAAA-YTRAHQL\r\n"
    + "LPEEPYITAQLLN--------------------AVAQGVGAVEPFAFLSEDASAAE----\r\n"
    + "----------SVRPLAPTRVRSKGPLRVGFVSNGFGAHPTGLLTVALFEALQRRQPDLQM\r\n"
    + "HLFATSGDDGSTLRTRLAQASTLHDVTALGHLATAKHIRHHGIDLLFDLRGWGGGGRPEV\r\n"
    + "FALRPAPVQVNWLAYPGTSGAPWMDYVLGDAFALPPALEPFYSEHVLRLQGAFQPSDTSR\r\n"
    + "VVAEPPSRTQCGLPEQGVVLCCFNNSYKLNPQSMARMLAVLREVPDSVLWLLSGPGEADA\r\n"
    + "RLRAFAHAQGVDAQRLVFMPKLPHPQYLARYRHADLFLDTHPYNAHTTASDALWTGCPVL\r\n"
    + "TTPGETFAARVAGSLNHHLGLDEMNVADDAAFVAKAVALASDPAALTALHARVDVLRRES\r\n"
    + "GVFEMDGFADDFGALLQALARRHGWLGI\r\n"
    + "\r\n"
    + ">Noname             \r\n"
    + "-MTADGPRELLQLRAAVRHRPQDVAWLMLADAELGMGDTTAGEMAVQRGLALHPGHPEAV\r\n"
    + "ARLGRVRWTQQRHAEAAVLLQQASDAAPEHPGIALWLGHALED--------------HQL\r\n"
    + "LPEEPYITAQLDVLSAQVR-------------AAVAQGVGAVEPFAFLSEDASAAEQLAC\r\n"
    + "ARTRAQAIAASVRPLAPTRVRSKGPLRVGFVSNGFGAHPTGLLTVALFEALQRRQPDLQM\r\n"
    + "HLFATSGDDGSTLRTRLAQASTLHDVTALGHLATAKHIRHHGIDLLFDLRGWGGGGRPEV\r\n"
    + "FALRPAPVQVNWLAYPGTSGAPWMDYVLGDAFALPPALEPFYSEHVLRLQGAFQPSDTSR\r\n"
    + "VVAEPPSRTQCGLPEQGVVLCCFNNSYKLNPQSMARMLAVLREVPDSVLWLLSGPGEADA\r\n"
    + "RLRAFAHAQGVDAQRLVFMPKLPHPQYLARYRHADLFLDTHPYNAHTTASDALWTGCPVL\r\n"
    + "TTPGETFAARVAGSLNHHLGLDEMNVADDAAFVAKAVALASDPAALTALHARVDVLRRES\r\n"
    + "I---------------------------";

    public static void main(String[] args) throws UnsupportedRuntimeException,
                    JobSubmissionException, WrongParameterException,
                    FileNotFoundException, IOException, ResultNotAvailableException,
                    InterruptedException {

            /*
             * Annotation interface for AAConWS web service instance
             */
            Annotation<AACon> client = AAConClient.connect();

            /* Get the list of available presets */
            PresetManager presetman = client.getPresets();

            /* Get the Preset object by preset name */
            Preset preset = presetman.getPresetByName("Complete conservation");

            /*
             * Load sequences in FASTA format from the file You can use something
             * like new FileInputStream() to load sequence from the file
             */
            List<FastaSequence> fastalist = SequenceUtil
                            .readFasta(new ByteArrayInputStream(input.getBytes()));

            /*
             * Submit loaded sequences for an alignment using preset. The job
             * identifier is returned by this method, you can retrieve the results
             * with it sometime later.
```

```
             */
            String jobId = client.presetAnalize(fastalist, preset);

            /* This method will block for the duration of the calculation */
            Set<Score> result = client.getAnnotation(jobId);

            /*
             * This is a better way of obtaining results, it does not involve
             * holding the connection open for the duration of the calculation,
             * Besides, as the University of Dundee public server will reset the
             * connection after 10 minutes of idling, this is the only way to obtain
             * the results of long running task from our public server.
             */
            // while (client.getJobStatus(jobId) != JobStatus.FINISHED) {
            // Thread.sleep(1000); // wait a second, then recheck the status
            // }

            /* Output the alignment to standard out */
            Score.write(result, System.out);

            /* Alternatively, you can record retrieved alignment into the file */
            // FileOutputStream out = new FileOutputStream("result.txt");
            // Score.write(result, out);
            // out.close();
    }
}
```

For a more detailed description of all available types and their functions please refer to the data model javadoc.

CHAPTER

FIVE

JAVA LIBRARY

This page provides information pertaining to AACon Java library usage.

5.1 Using the library

Note: For impatient please find the complete example below!

To calculate conservation using AACon follow the steps:

1. *Loading data* - Load data
2. *Initialising the executor* - Initialise the executor
3. *Normalisation* - Decide on normalisation
4. *Setting the conservation methods* - Set conservation methods
5. *Output the results* - Output the results

All these steps are discussed in greater details below.

5.2 Loading data

AACon methods require a List of FastaSequence objects as input. Below is an example of how to load the Fasta formatted sequence file or the Clustal alignment file. For this call to succeed all the sequences must be of the same length (as this is an alignment).

```
List<FastaSequence> sequences = CmdParser.openInputStream(<PATH_TO_INPUT_FILE>);
```

For the examples of the input file please see AACon standalone executable help page.

5.3 Initialising the executor

AACon methods require the ExecutorService object which is used to parallel the calculations. The ExecutorService initialised with the number of threads equals to the number of cores on the executing machine is recommended for a faster calculation and can be obtained as follows:

```
int corenum = Runtime.getRuntime().availableProcessors();
ExecutorService executor = Executors.newFixedThreadPool(corenum);
```

Please take care to initialise and pass only one executor to all the methods to avoid the waist of resources. After use, the executor must be disposed of, it can be done as follows:

```
executor.shutdown();
```

5.4 Normalisation

AACon methods require the boolean parameter telling the system whether the results should be normalised or not. Normalised results have values between 0 and 1. Please note however, that some results cannot be normalised. In such a case, the system returns not normalised values, and log the issue to the standard error stream. The following formula is used for normalisation

```
n = (d - dmin)/(dmax - dmin)
```

Negative results first converted to positive by adding an absolute value of the most negative result.

5.5 Setting the conservation methods

AACon `ConservationCalculator.getConservation()` method takes a Set of `ConservationMethod` to use for calculating the conservation. Below is a few examples of making such sets.

- To calculate conservation according to KABAT method construct a set in the following way -

  ```
  EnumSet.of(ConservationMethod.KABAT).
  ```

- For all the methods use the following construct -

  ```
  EnumSet.allOf(ConservationMethod.class)
  ```

- For a set of methods including KABAT, JORES, SCHNEIDER, SHENKIN and GERSTEIN use the following construct -

  ```
  EnumSet.range(ConservationMethod.KABAT, ConservationMethod.GERSTEIN)
  ```

5.6 Output the results

The results can be output using one of the following methods:

1. *ConservationFormatter.formatResults(Map<ConservationMethod, double[]> scores, OutputStream outStream)*
2. *ConservationFormatter.formatResults(Map<ConservationMethod, double[]> scores, String outFilePath, Format format, List<FastaSequence> alignment)*

Use the first method to output the results of the calculation without an alignment to any OutputStream. Use the second method to output results with the alignment.

First method usage example:

```
// Usage example - printing results to the console
ConservationFormatter.outputScoreLine(result, System.out)
// printing results to the file
FileOutputStream outfile = new FileOutputStream("results.txt");
ConservationFormatter.formatResults(result, outfile);
outfile.close();
```

Second method usage example:

```
// Usage example - printing results with alignment in Fasta format to the file called␣
↪output.txt
ConservationFormatter.formatResults(result, "test.txt", Format.RESULT_WITH_ALIGNMENT,␣
↪sequences);
```

For more information on the output formats please see standalone AACon help.

5.7 Example

Using AACon library for calculating conservation:

```
// Determine the number of CPU cores available on the system.
int corenum = Runtime.getRuntime().availableProcessors();
// Initialize the Executor instance with a number of cores
ExecutorService executor = Executors.newFixedThreadPool(corenum);

// Load the data from the file containing either Clustal formatted alignment
// or a list of FASTA formatted sequences. Assuming that small.align file is
// in the same directory as this program
List<FastaSequence> sequences = CmdParser.openInputStream("small.align");

// Calculate conservation scores using all methods.
Map<ConservationMethod, double[]> result = getConservation(sequences, true,
                         EnumSet.allOf(ConservationMethod.class)), executor);

// Print the results to the console.
ConservationFormatter.formatResults(result, System.out);
```

CHAPTER

SIX

CITATIONS

Note: It is important that you cite AACon when you use it. Citing us helps us funding the work we do and allow us to continue to improve the project further.

Peter Troshin, Agnieszka Golicz, David Martin, Fábio Madeira and Geoffrey J. Barton - **AACon: Amino Acid Conservation Service.** Paper in preparation. 2017.

CHAPTER

SEVEN

CHANGELOG

7.1 Version 1.1 (Released 19 April 2017)

The website and documentation were improved:

- Sphinx is now used to generate our documentation pages.
- Documentation was updated to reflect the latest changes introduced in the project.

Several bugs have been fixed including:

- Metric X
- Metric Y
- Metric Z

7.2 Version 1.0 (Released 16th Dec 2011)

First version of AACon. This service has been in production in JABAWS v2, since 2011 and publicly accessible via the JABAWS clients as well as in Jalview. Additionally, AACon has been available for download since the same time from http://www.compbio.dundee.ac.uk/aacon/.

Note: This is an open source project. If you want to contribute or report an issue have a look at our Github repository.