IgCAT: a Prefix Tree-Based Immunoglobulin Annotation System

User Guide

Sergey Knyazev

July 2014

1 Introduction

This guide will help you to install IgCAT[3], to run, and to test it.

2 Installation

2.1 Dependencies

In order to install IgCAT, you need Java Development Kit (JDK) [1] and Appache Maven project management tool [2] installed. To install JDK, open download page [3], find Java SE 7u65 (or better) version, then follow the instructions [4]. In case you install Maven, you will download its distributive version 3.1.1 (or better) from download page [5] and will install it according to the instructions.

IgCAT has some helpful scripts written in Python, so if you want to run them, you will install python version 2.7.6 or higher [6] with additional libraries such as BioPython [7].

2.2 IgCAT Installation

To install IgCAT, just run following command from IgCAT home directory ($IGCAT_HOME):

```
$ mvn clean package
```

When installation is complete an executable jar file (ig-regions-1.0-SNAPSHOT.jar) will be at:

[1] http://www.oracle.com/technetwork/java/javase/overview/index.html
[2] http://maven.apache.org/
[3] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[4] http://docs.oracle.com/javase/7/docs/webnotes/install/index.html
[5] http://maven.apache.org/download.cgi
[6] https://www.python.org/downloads/
[7] http://biopython.org/wiki/Download

```
ig-regions/target/
```

3 Running

To run IgCAT you should implement following command from IgCAT home directory:

```
$ java -Dlogback.configurationFile=logback.xml -jar ig-regions/target/ig-regions-1.0-SNAPSHOT.jar
[args]
```

Where args are user arguments which allow to adjust runs for user needs. You can specify parameters such as fasta file names, alignment matrices location, alignment type, references database, and so on (see section 3.2).

Simple run can be started by using commands from $IGCAT_HOME:

```
$ mkdir test/simple_example/out
```

```
$ java -Dlogback.configurationFile=logback.xml -jar ig-regions/target/ig-regions-1.0-SNAPSHOT.jar
-r data/germline/human/vl.fasta -m data/nomenclature/human/vl.kabat -s test/simple_example/ig.fasta
--amino --igblast-like --outdir test/simple_example/out
```

This command will find immunoglobulin (IG) regions location in ig.fasta with using vl.fasta and vl.kabat as library. Results will be in test/simple_example/out directory (see section 3.1 for details).

3.1 Output files

IgCAT generates 5 files into folder specified by user. These files contain regions bounds prediction. All files provides almost the same information, but in different formats, and one of them is in IgBLAST format[4].

3.2 Options

This section provides a full list of IgCAT arguments.

There is mandatory arguments you should always specify:

```
-s <value> | --source <value>
```

this option specifies a fasta file with IG sequences you want to annotate;

```
-r <value> | --reference <value>
```

this argument sets a fasta file with references for IgCAT inner database, and it requires a marking file with information about IG regions bounds;

```
-m <value> | --marking <value>
```

it is a marking file in IgBLAST marking format (example [8]), which contains regions bound data for every IG sequence in the reference fasta file.

[8] ftp.ncbi.nlm.nih.gov/blast/executables/igblast/release/internal_data/human/

For detailed output you can specify the output directory:

```
--outdir <value>
```

If you want to run script on protein sequences data, just set:

```
--amino
```

To change output to default IgBLAST marking data files use:

```
--igblast-like
```

Most of time you need not to use additional output filtration, but you can:

```
--filter
```

enable simple filtration (default: disabled)

You can choose the alignment method for annotation with or without affine gaps penalties. Semiglobal method without affine gaps penalties is set by default. We strongly recommend it for regions detection. But you can choose other method by one of options:

```
--global
```

use global alignment without affine gaps penalties

```
--local
```

use local alignment without affine gaps penalties

```
--semiglobal
```

use semiglobal alignment without affine gaps penalties (default)

```
--affine-global
```

use global alignment with affine gaps penalties

```
--affine-local
```

use local alignment with affine gaps penalties

```
--affine-semiglobal
```

use semiglobal alignment with affine gaps penalties

If you want to customize alignment options, set:

```
--matrix <value>
```

use external alignment matrix [txt]

```
--gap <value>
```

simple gap score (default: -5)

```
--gap-open <value>
```

affine open gap score (default: -10)

```
--gap-ext <value>
```

affine extension gap score (default: -1)

In addition you can run IgCAT alignment in parallel mode and modify input sequence names:

```
--par
```

Use parallel mode

```
--group
```

Add germline group to sequence name

You can always see this help, using:

```
--help
```

4 Scripts for IG annotation comparison

We provide some scripts for IG annotation comparison. If you have annotated one dataset by two different tools or if you have test dataset and want to verify IgCAT you can use our scripts to determine the differences. The first script named compare_marking.py compares etalon annotation you suppose to be true and annotation you want to check. Script will gather statistics of how checking annotation is differ from etalon annotation. You can run the script from $IGCAT_HOME/scripts as follow:

```
python compare_marking.py etalon_marking checking_marking
```

where 'etalon_marking' and 'checking_marking' are your files with annotations in IgBLAST format. The script will generate text with comparison results.

If you have one dataset annotated by two different tools you can compare how these annotations are differ. For that you can use the second script named diff_info.py . To run it just go to $IGCAT_HOME/scripts and use command:

```
python compare_marking.py marking1 marking2
```

where 'marking1' and 'marking2' are your files with annotations in IgBLAST format. The script will generate text with comparison results. Results are formatted as table. Columns description is follow:

Column number	Description
1	Region bound name
2	Count of different predictions
3	Average bound shift in whole dataset
4	Average bound shift in set of sequences, where difference in predictions occurs
5	Rate of different predictions to total number of predictions

5 Testing

To implement the tests highlighted in IgCAT article [3], you can use our python scripts. These scripts generate reference and test data, annotate different datasets using IgCAT, and check the annotation quality. Furthermore, they can compare similarity of annotation predictions created by different tools. Our python scripts can also be useful for investigation how IgCAT container filling influences upon annotation quality.

5.1 Testing IgCAT prediction quality

Testing of prediction quality consists of two part: generation of valid reference data and estimation of the prediction accuracy. We used VBASE[2] database of known human IG sequences to generate valid database. We simply combined V,D,J-genes in all possible ways, so we reconstructed natural principles. This dataset can be simply generated by our script using following instructions. To do so change work directory from IgCAT home to data/generator/vdj_combinator using the command:

```
$ cd data/generator/vdj_combinator
```
then run python script named vdj_combinator.py:
```
$ python vdj_combinator.py
```
This will generate 6 files into working directory: 3 fasta files (vl.fasta, vh.fasta, vk.fasta) with possible human immunoglobulin sequences, and 3 kabat files (vl.kabat, vh.kabat, and vk.kabat) with bounds marking to corresponding fasta files. Copy them into your reference database directory. Alternatively, you can find these files already generated in directories: $IGCAT_HOME/data/germline/human for fasta files, and $IGCAT_HOME/data/nomenclature/human for kabat ones.

Testing the influence of IgCAT container filling to the annotation quality can be implemented by test_random_filling.py script. To do so just go to the $IGCAT_HOME/scripts/test folder, and run following command:
```
$ python test_random_filling.py ../../data/germline/human/vh.fasta
../../data/nomenclature/human/vh.kabat
```
After the run has finished, you can find results in the out_random_filling folder. This folder contains many test-X folders, where X is the number of the test. Every test-X folder contains many train-Y folders, where Y is the number of sequences used for filling IgCAT container. Train-Y folder contains results with statistics.

5.2 Comparing IgCAT with another tools

You can compare the IG annotation by IgCAT with IG annotation by another tool. For that you need to annotate the same dataset by both tools and to compare their predictions using $IGCAT_HOME/scripts/diff_info.py. We have prepared data, which was annotated by IgBLAST and Rosie tools (you can find it in folders test/igbase and test/rosie). There is directory named 'fasta' with sequences already annotated by according tools and directory named 'prediction' with prediction results. So to implement the test you just need to annotate sequences from 'fasta' folder by IgCAT and compare marking. Following commands will help you to compare IgCAT with IgBLAST, just run them from IgCAT home folder:
```
$ mkdir igblast_compare
```

```
$ java -Dlogback.configurationFile=logback.xml -jar ig-regions/target/ig-regions-1.0-SNAPSHOT.jar
-r data/germline/human/vh.fasta -m data/nomenclature/human/vh.kabat -s test/igbase/fasta/heavies.fasta
--amino --igblast-like --outdir igblast_compare
```

```
$ python scripts/diff_info.py igblast_compare/igblast_like.marking test/igbase/igblast/heavies.kabat
--part >> igblast_compare/diff
```
In the file igblast_compare/diff you can see statistics of prediction differences.

To compare IgCAT with Rosie just run following:
```
$ mkdir rosie_compare
```

```
$ java -Dlogback.configurationFile=logback.xml -jar ig-regions/target/ig-regions-1.0-SNAPSHOT.jar
-r data/germline/human/vh.fasta -m data/nomenclature/human/vh.kabat -s test/rosie/fasta/vh.fasta
```

```
--amino --igblast-like --outdir rosie_compare
```

```
    $ python scripts/diff_info.py rosie_compare/igblast_like.marking test/rosie/prediction/vh.kabat
>> rosie_compare/diff
```

In the file rosie_compare/diff you can see statistics of prediction differences. See section 4 for this file formatting description.

6 Citation

If you want to use our for your scientific research, please cite our paper [3].

References

[1] S. Lyskov, F. C. Chou, S. O. Conchuir, B. S. Der, K. Drew, D. Kuroda, J. Xu, B. D. Weitzner, P. D. Renfrew, P. Sripakdeevong, B. Borgo, J. J. Havranek, B. Kuhlman, T. Kortemme, R. Bonneau, J. J. Gray, and R. Das. Serverification of molecular modeling applications: the Rosetta Online Server that Includes Everyone (ROSIE). *PLoS ONE*, 8(5):e63906, 2013.

[2] I. M. Tomlinson and et al. VBASE Sequence Directory MRC. Centre for Protein Engineering Cambridge, UK. Available: http://www2.mrc-lmb.cam.ac.uk/vbase/. Accessed 23 July 2014, 1996.

[3] P. Yakovlev, S. Knyazev, A. Karabelsky, and Y. Porozov. IgCAT: a Prefix Tree-Based Immunoglobulin Annotation System. *Nucleic Acids Res.*, 2014.

[4] J. Ye, N. Ma, T. L. Madden, and J. M. Ostell. IgBLAST: an immunoglobulin variable domain sequence analysis tool. *Nucleic Acids Res.*, 41(Web Server issue):34–40, Jul 2013.