# Scythe Documentation

***Release 0.1***

**Janina Mass**

May 22, 2014

# CONTENTS

Scythe – Selection of Conserved Transcripts by Homology Evaluation

This documentation is under construction!

Contents:

CHAPTER

ONE

# INTRODUCTION

Scythe picks best matching transcripts for one-to-one orthologous genes from two or more species. The goal is to provide the best, i.e. most homologous set of sequences, for a subsequent multiple sequence alignment in order to minimize sources for misalignment in an automated fashion. Scythe will perform pairwise global alignments using the Needleman-Wunsch algorithm [NE1970] as implemented in needleall as part of the EMBOSS package [EMB2000].

Please see the *Tutorial* on how to use Scythe. Important! needleall seems to be broken in EMBOSS 6.6.0.0; this is working with needleall from EMBOSS 6.4.0.0.

## 1.1 Algorithms and Application

Scythe performs pairwise global alignments [NEED1970] to measure the similarity between transcripts. Transcripts are only compared between species.

There are three main strategies implemented in Scythe to deal with the sequences after scoring: A single-linkage approach either starting with reference transcripts as seed (sl_ref) and adding best matching sequences from non-reference species or starting out with the best matching transcript pair for a gene (sl_glob). A *reference species* is defined as a species that has only one transcript (*reference transcript*) for a gene. Every gene is processed individually, a *reference species* is only local to a gene and is automatically derived from the input data. Alternatively, the maximum-sum (mx_sum) approach calculates the score for all transcript pairings between the species and return a maximum-scoring set. Please note that this approach might not be feasable for large data sets.

### 1.1.1 mx_sum

mx_sum (maximum-sum) max sum algorithm returns an optimal solution for the problem of score maximization. There are, however, scenarios where this would not represent the desired result: In cases of single-transcript outliers, their pairwise score to all other sequences is always taken into account and may favor sequences that are less dissimilar to the outlier.

### 1.1.2 sl_ref

sl_ref(single-linkage-reference)... It works well in cases where the user wants to focus on finding similar gene models given one reference species, e. g. when the user has already decided on a splice variant in one species.

### 1.1.3 sl_glob

sl_glob (single-linkage-global) due to the single linkage selection approach, the algorithm starting with the best matching pair should provide a good starting point in absence of a reference gene model and may be able circumvent outliers.

### 1.1.4 bf_msa

!todo brute force all vs all msa comparison using mafft

CHAPTER

# TWO

# TUTORIAL

## 2.1 Installation

### 2.1.1 Requirements

- Python 3

#### python modules

- configparser
- mysql-connector-python
- httplib2

```
pip install --allow-external mysql-connector-python mysql-connector-python
pip install httplib2
pip install configparser
```

In case you want to work using a virtual environment switch to that environment first (see *Install Scythe in a virtual environment*).

#### external software

- needleall http://emboss.sourceforge.net/

!Important: needleall from (the ubuntu package for) EMBOSS 6.6.0.0 causes problems. This program was tested and works with needleall from EMBOSS 6.4.0.0.

Install emboss on debian-based systems:

```
sudo apt-get install emboss
```

### 2.1.2 Install Scythe in a virtual environment

Install virtualenv via pip:

```
pip install virtualenv
```

Create a virtual environment that uses Python 3 and activate:

```
virtualenv -p /usr/bin/python3 python3env
source python3env/bin/activate
```

Finally, install the scythe package:

```
(python3env)you@host:~$ pip install scythe
```

### 2.1.3 Without virtual environment

Via pip:

```
pip install scythe
```

From source:

```
wget TODO
tar xvf TODO
cd TODO/DIR
python setup.py install
```

## 2.2 Using Scythe

There are three ways to use Scythe: GUI, command line parameters, or configuration file.

The GUI allows you to directy obtain data from ENSEMBL, have it converted and processed by Scythe.

If you are using local files, make sure you have directories prepared that contain the *loc* and fasta files, and that you have a *grp* file ready. (See *Converters* on how to convert other formats to *loc* or *grp*.)

Read more about configuration files under *Configuration Files*.

### 2.2.1 Graphical user interface

Call the GUI via

```
scythe-gui.py
```

or with configuration file *conf.py*

```
scythe-gui.py conf.scy
```

Please note that Scythe will write intermediate files to the current working directory. You might want to create a new directory for each run beforehand. Data that was already downloaded from ENSEMBL will not be downloaded again. If you want to reuse this data but try different parameters, just change the output directory (via the GUI) and start Scythe from the same working directory.

### 2.2.2 Command line interface

```
    #######################################
    # scythe.py v0.1                      #
    #######################################
  usage:
     scythe.py -i DIR -g .grpFILE --cleanup
```

```
usage with configuration file:
   scythe.py --config configuration.scy

general options:
  -C, --config                    use configuration file instead of
                                  command line parameters
  -c, --cleanup                   remove temporary files when done
  -h, --help                      prints this
  -i, --in_dir=DIR                folder w/ subfolders "fa" and "loc"
  -r, --sl_ref                    find best matches to reference
  -g, --sl_glob                   best scoring pair as seed
  -m, --mx_sum                    optimize sum of pairwise scores
  -o, --out_dir=DIR               output directory [default:./]
  -v, --verbose                   be wordy

alignment options:
   -O, --gap_open=FLOAT          needleall gap opening cost [default 10]
   -E, --gap_extend=FLOAT        needleall gap extension cost

fasta options:
  -d, --delim=STRING              split fasta headers at STRING
  -a, --asID=INT                  use INTth part of fasta header as transcript-ID
                                  (default:0)
debug options:
  -s, --stop_after=NUM            stop after NUM groups

further help:
  Please see documentation in the 'docs'-directory.
```

Please note that you cannot automatically download sequences from ENSEMBL via the command line interface.

CHAPTER

THREE

# FORMAT

The formats have a very simple, human readable structure allowing users to also manually modify or create them. For locus-gene model identifier relations: *loc format* For grouping of orthologous genes *grp format*

## 3.1 loc format

Each row of a *loc* file represents a set of gene models (GM) for a gene locus (L). Rows start with a the gene locus ID followed by its gene model identifiers. The GM in the second column in the file is treated as the reference form for its gene locus, i.e. in a case of a tie, this gene model is preferred. The columns are tab-separated.

File *Spec0.loc*:

```
Sp0L0       Sp0L0GM0        Sp0L0GM1        . . . Sp0L0GMjLF
Sp0L1       Sp1L0GM0        Sp0L1GM1        . . . Sp0L1GMkLF
.
.
.
Sp0Lm       Sp1LmGM0        Sp0LmGM1        . . . Sp0LmGMlLF
```

## 3.2 grp format

Each row of a *grp* file represents an orthologous group. Rows start with a unique group ID followed by orthologous gene identifiers for the respective species. Rows don't need to have the same number of columns. The gene loci order within a row is irrelevant. The programm will keep the identifiers for its output. The columns are tab-separated.

File *orthogroups.grp* (Sp: species, L: orthologous gene loci):

```
0   SpaLw   SpbLw   SpcLw   . . . SpzLw
1   SpaLx   SpbLx   SpcLx   . . .
.
.
.
N   SpaLz   SpbLz   SpcLz   . . .
```

CHAPTER

# FOUR

# CONVERTERS

Scythe uses simple, human readable formats to store gene-transcript and ortholog information. See also *Format*.

## 4.1 loc

Scripts to convert the following formats to *loc* format are included:

- gff3
- tab-separated (eg ENSEMBL BioMart)

run

```
scythe_loc_gff.py -f GFF
```

or

```
scythe_loc_tsv.py -f FILE.tsv
```

See also: *loc format*.

## 4.2 grp

Please note that the *grp* converters need a concatenated *loc* file in addition to the orthology information. Scripts to convert the following formats to *grp* are included:

- orthomcl
- proteinortho
- tab-separated (eg ENSEMBL BioMart)

run

```
scythe_grp_orthomcl.py
```

```
scythe_grp_proteinortho.py
```

```
scythe_grp_tsv.py
```

See also *grp format*.

**undoc-members**

```
#------------ loc output format ---------------------------#

LOCUS0        TRANSCRIPT0_0        TRANSCRIPT0_1        ...        TRANSCRIPT0_n
LOCUS1        TRANSCRIPT1_0        ...        TRANSCRIPT1_m
.
.
.
LOCUSk        TRANSCRIPTk_0        ...TRANSCRIPTk_l

#----------------------------------------------------------#


#-------- gff version 3 input format ----------------------#
example (tab is shown as \t):

##gff-version 3
L1i\texample\tgene\t1000\t9000\t.\t+\t.\tID=L1;Name=L1;Note=example
L1.a\texample\tmRNA\t1000\t9000\t.\t+\t.\tID=L1.a;Parent=L1;Name=L1.a


Note that this script only relies on the "ID" and "Parent" tags,
"Name" will be ignored. If "longest" is specified (eg phytozome)
and =="1", the transcript will be placed on the first position
for its gene.
#----------------------------------------------------------#



#####################################
#  scythe_loc_gff.py  -f FILE.gff3  #
#####################################

-f, --file=gff3_FILE
-o, --output=FILE        output file [default: gff3_FILE.loc]
-h, --help               prints this
-H, --HELP               show help on format

#------------ loc output format ---------------------------#

LOCUS0        TRANSCRIPT0_0        TRANSCRIPT0_1        ...        TRANSCRIPT0_n
LOCUS1        TRANSCRIPT1_0        ...        TRANSCRIPT1_m
.
.
.
LOCUSk        TRANSCRIPTk_0        ...TRANSCRIPTk_l

#----------------------------------------------------------#



example ensembl query:

http://www.ensembl.org/biomart/
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Query>
<Query  virtualSchemaName = "default" formatter = "TSV"
header = "0" uniqueRows = "0" count = "" datasetConfigVersion = "0.6" >
<Dataset name = "hsapiens_gene_ensembl" interface = "default" >
```

```
<Filter name = "biotype" value = "protein_coding"/>
<Attribute name = "ensembl_gene_id" />
<Attribute name = "ensembl_transcript_id" />
<Attribute name = "ensembl_peptide_id" />
<Attribute name = "cds_length" />
</Dataset>
</Query>


###########################
#  scythe_loc_tsv.py      #
###########################
-f, --file=ENSEMBLBioMart.tsv
                          format: 1st column: gene id, 2nd column:transcript id,
                                  3rd column: peptide id, 4th column: cds length;
                                  gene ids can occur multiple times
-c, --custom=COLx,COLy,COLz,...   COLi in ["gene","transcript", "protein", "length"]
                                  Use this if your file is different from the described biomart c
                                  "cds_length" is optional but recommended, at least one of
                                  ["transcript", "protein"] need to be included
-o, --output=FILE         output file [default: ENSEMBLEBioMart.tsv.loc]
-h, --help                prints this
-H, --HELP                show help on format
#---------------------------------#
```

CHAPTER

FIVE

# CONFIGURATION FILES

The configuration files allow you save your settings/restore your settings. Most entries should be self-explanatory.

## 5.1 Example file

```
[Mode]
use_ensembl=yes
use_local_files=no

[Paths]
fasta_directory=/home/nin/py3env/fa/
loc_directory=/home/nin/py3env/loc/
grp_file=/home/nin/py3env/magohopopafe.shared_by_all.grp
output_directory=/home/nin/py3env

[Cleanup]
clean_up_directories=yes

[Run_options]
max_threads=1
split_input=1

[Penalties]
gap_open_cost=10
gap_extend_cost=0.5
substitution_matrix=EBLOSUM62

[Algorithm]
use_sl_glob=unset
use_sl_ref=unset
use_mx_sum=yes

[Fasta_header]
fasta_header_delimiter=" "
fasta_header_part=0
```

## 5.2 [Fasta_header]

In case the ids in your *loc* file are truncated or your fasta headers carry information in addition to the sequence identifier, you can define where to split the fasta header.

Example:

```
>other:xyz_sequenceID_length:123_species:x
MYQVLQAYDWKYLHDNHDCNFQVGGADQLGNI...
```

and:

```
[Fasta_header]
fasta_header_delimiter = "_"
fasta_header_part = 1
```

Would result in taking only *sequenceID* into account.

## 5.3 [Run_options]

Multi core support is not yet implemented.

## 5.4 [Algorithm]

See *Algorithms and Application*

CHAPTER

# SIX

# MODULES

## 6.1 convert

`scythe2.convert.scythe_loc_gff.`**`checkGff3`**(*infile*)

`scythe2.convert.scythe_loc_gff.`**`formatHelp`**()

`scythe2.convert.scythe_loc_gff.`**`main`**()

`scythe2.convert.scythe_loc_gff.`**`read_gene_mrna`**(*infile*)

`scythe2.convert.scythe_loc_gff.`**`read_gff2loc`**(*infile*, *outfile*)

`scythe2.convert.scythe_loc_gff.`**`read_gff_attrib`**(*infile*, *attrib*)

`scythe2.convert.scythe_loc_gff.`**`usage`**()
    Print usage.

`scythe2.convert.scythe_loc_gff.`**`writeLoc`**(*genesDct*, *outfile*)

**exception** `scythe2.convert.scythe_loc_tsv.`**`WrongNumberOfColumnsException`**

`scythe2.convert.scythe_loc_tsv.`**`formatHelp`**()

`scythe2.convert.scythe_loc_tsv.`**`howTo`**()
    Print an ENSEMBLE example query

`scythe2.convert.scythe_loc_tsv.`**`main`**()

`scythe2.convert.scythe_loc_tsv.`**`readTsv`**(*infile=None*)

`scythe2.convert.scythe_loc_tsv.`**`usage`**()

`scythe2.convert.scythe_loc_tsv.`**`writeLoc`**(*genesDct*, *outfile*)

## 6.2 gui

## 6.3 helpers

`scythe2.helpers.ensembl2grp.`**`main`**()

`scythe2.helpers.ensembl2grp.`**`readTsvFiles`**(*listoftsv*, *outfile*)

`scythe2.helpers.ensembl2grp.`**`usage`**()

scythe2.helpers.ensembl_ortho_mysql.**fetch1to1orthologs**(*method_link_species_set_ids*, *release*)

Return (homology_id, genome_db_id, stable_id)

scythe2.helpers.ensembl_ortho_mysql.**fetchOrthoFromMySQL**(*specieslist=['homo_sapiens', 'pan_troglodytes', 'mus_musculus'], release=74, outfile=None*)

scythe2.helpers.ensembl_ortho_mysql.**getGenome_Db_Ids**(*specnames*, *release*)

scythe2.helpers.ensembl_ortho_mysql.**getHomologyId**(*method_link_species_set_ids*, *release=71*)

scythe2.helpers.ensembl_ortho_mysql.**getHomologyMemberId**(*homology_ids*, *release*)

scythe2.helpers.ensembl_ortho_mysql.**getMethodLinkSpecies_Set_Ids**(*speciesSet_ids*, *release*)

scythe2.helpers.ensembl_ortho_mysql.**getSpecies_Set_Ids**(*genomedb_ids*, *release*)

scythe2.helpers.ensembl_ortho_mysql.**makeTable**(*genomeDB_ids*, *orthoSpec2stableIds*, *orthoIds*, *outfile*)

scythe2.helpers.ensembl_ortho_mysql.**q**(*s*)

**class** scythe2.helpers.mergeSubsets.**Orthogroup**(*list*)

**expand**(*other*)

**toString**()

scythe2.helpers.mergeSubsets.**filterGroups**(*grp*, *outfile*, *numspec=None*, *rename=False*)

scythe2.helpers.mergeSubsets.**main**()

scythe2.helpers.mergeSubsets.**mergeSubsets**(*grp*, *outfile*, *rename=False*)

scythe2.helpers.mergeSubsets.**usage**()

## 6.4 algo

!todo

# CHAPTER SEVEN

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

[NEE1970] Needleman, Saul B.; and Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". Journal of Molecular Biology 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325.

[EMB2000] EMBOSS: The European Molecular Biology Open Software Suite (2000) Rice,P. Longden,I. and Bleasby, A.Trends in Genetics 16, (6) pp276–277

# PYTHON MODULE INDEX