[1]

PAGAN (v. 0.31): brief tutorial

Ari Löytynoja
ari.loytynoja@gmail.com

September 6, 2011

Contents

1 Background **1**

2 Installation of PAGAN **2**

3 Using PAGAN **3**
3.1 Phylogenetic multiple alignment 3
3.2 Phylogenetic placement of sequences into an existing alignment 4
3.3 Additional program options 6
3.4 Inference of ancestral sequences for existing alignments 8
3.5 Configuration files: input, output and format 8
3.6 Checking for PAGAN updates 9

Appendix: installing `boost` from source code **10**

1 Background

PAGAN is a general-purpose method for the alignment of sequence graphs. The main advantage of the graph representation of sequences is the graphs' ability to describe the uncertainty in the presence of characters at certain sequence positions. This is highly useful in phylogenetic progressive alignment and allows for capturing the different properties of insertions and deletions when alignments are iterated for multiple sequences. However, graphs also allow describing the uncertainty in input sequences and modelling e.g. homoploymer errors in Roche 454 reads, or representing inferred ancestral sequences against which other sequences can then be aligned.

PAGAN is still under development and will hopefully evolve to an easy-to-use, general-purpose method for phylogenetic sequence alignment. The graph representation has features that make it especially powerful for phylogenetic placement of sequences into existing alignments. As such a tool has direct applications in the analyses of increasingly abundant RNA-seq data and there is no satisfactory solution available, we have focused on this problem and implemented the necessary functionality first.

This document explains how to install and start using the PAGAN software.

[1]See here for the explanation of the PAGAN logo.

2 Installation of PAGAN

PAGAN is open-source software licensed under the GPL. The C++ source code is provided at http://www.ebi.ac.uk/∼ari/pagan. PAGAN is developed and tested on a Linux system and we can currently only provide instructions for that platform, with a special focus on the popular Ubuntu/Debian-based distributions.

Installation of Boost libraries

PAGAN requires **-dev versions** of two utility libraries from the Boost project that may not be included in standard OS installations. These libraries have to be installed **before** compiling the PAGAN source code. On Ubuntu, they can be installed using commands:

```
apt-cache search libboost-program-options
apt-cache search libboost-regex
```

(See which version number, ending with `-dev`, is provided and edit the command below.)

```
sudo apt-get install libboost-program-options1.40-dev
sudo apt-get install libboost-regex1.40-dev
```

If your distribution does not provide Boost libraries (highly unlikely) or you are not allowed to install software on your system, you can download and install the necessary library from the Boost project repositories directly into your PAGAN source code directory. See Appendix for details.

Download and installation of PAGAN using `git`

The most recent version of the PAGAN source code is available from the git-repository and snapshots of this are downloadable as compressed tar-packages. The `git` software can be found at git-scm.com. On Ubuntu, it can also be installed using command:

```
sudo apt-get install git-core
```

Given that `git` is installed, the PAGAN source code can be downloaded and compiled using commands:

```
git clone http://www.ebi.ac.uk/~ari/pagan
cd pagan/src
make -f Makefile.no_Qt
./pagan
```

Without `git`, the latest (but possibly not up-to-date) tar-packaged source code can be downloaded and compiled using commands:

```
wget http://www.ebi.ac.uk/~ari/pagan/tgz/pagan.src.latest.tgz
tar xzf pagan.src.latest.tgz
cd pagan/src
make -f Makefile.no_Qt
./pagan
```

3 Using PAGAN

PAGAN is a command-line program. It can be used by (a) specifying a list of options (command-line arguments) when executing the program, or (b) creating a configuration file with the options and specifying that when executing the program. The configuration file does not need to be created from scratch as PAGAN can output the options specified for an analysis in a file. This file can then be edited if necessary and specified as the configuration file for another analysis. Alternatively, the file can be considered as a record of a particular analysis with a full description of options and parameters used.

A list of the most important program options is outputted if no arguments are provided:

```
./pagan
```

and a more complete list is given with the option `--help`:

```
./pagan --help
```

In general, the option names start with `--` and the option name and value (if any) are separated by a space. The configuration file makes an exception and can be specified without the option name:

```
./pagan option_file
```

Also this one can be given in the standard format and the following command is equivalent:

```
./pagan --config-file option_file
```

3.1 Phylogenetic multiple alignment

PAGAN is based on a progressive algorithm that aligns sequences according to a guide tree. It (currently) cannot compute a tree by itself and requires the user to provide a **rooted** binary tree relating the sequences. The leaf names in the tree and the sequence names (until the first space) in the sequence file have to match exactly. Alignment is only performed for the parts of the guide phylogeny that have sequences associated; the unnecessary branches and sequences are pruned/dropped out.

The minimal command to perform the alignment is:

```
./pagan --seqfile sequence_file --treefile tree_file
```

The resulting alignment will be written in files `outfile.fas` and `outfile.xml`. If you want to use another file name, you can specify that with option `--outfile`:

```
./pagan --seqfile sequence_file --treefile tree_file --outfile another_name
```

PAGAN will automatically add suffix `.fas` and `.xml`.

The sequence input file has to be in FASTA format and the guide tree in Newick tree format, with branch lengths as substitutions per site. The resulting alignment will be written in FASTA format and in XML-based HSAML format. PAGAN supports the alignment of nucleotide, amino-acid and codon sequences although the last two are still experimental.

3.2 Phylogenetic placement of sequences into an existing alignment

PAGAN can reconstruct ancestral nodes for a given phylogeny based on an existing alignment. Additional sequences can then be aligned and added to this reference alignment without affecting the relative alignment of original sequences. The main application of this is phylogenetic placement of short reads, coming from NGS platforms, into existing alignments but it can, in principle, be used for any sequences, including amino-acid ones.

The minimal command to perform the alignment is:

```
./pagan --ref-seqfile ref_alignment_file --ref-treefile ref_tree_file
 --readsfile reads_file
```

The reference alignment has to be in FASTA format and the reference tree in Newick or Newick eXtended (NHX) format; the reads (or sequences in general) that will be added in the reference alignment can be either in FASTA or FASTQ format. If the reference alignment consists of one sequence only, the reference guide tree is not required. Again, you can define your own output file with option `--outfile`.

Pair-end reads and 454 data

For the alignment of NGS reads, additional options `--pair-end` and `--454` are useful. The first one merges paired reads into one (separated by a spacer) before the alignment and the second models the ambiguous length of mononucleotide runs in data coming from Roche 454 platform. The pairing of pair-end reads assumes that the reads have identical names (until the first space) with the exception that the left read ends with `/1` and the right read with `/2`. In the resulting alignment, the paired sequence will have suffix `/p12`. Option `--454` only works with FASTQ-formatted data.

Overlapping pair-end reads

The length and overall quality of NGS data can be improved by using such a short fragment length that the reads starting from each end of the fragment overlap in the middle. PAGAN allows merging such reads and handling them as one sequence. Merging is done using option `--overlap-pair-end`, the reads successfully merged having suffix `/m12` in the resulting alignment.

The merging requires significant overlap between the two reads in their pairwise alignment (performed without masking). Options `--overlap-minimum` and `--overlap-identity` can be used to change the minimum length and base identity of this overlap. Shorter overlaps that show perfect base identity are also accepted; the minimum length of this can be changed with option `--overlap-identical-minimum`. The merged reads can be outputted in FASTQ format using option `--overlap-merge-file`. PAGAN can also be used just to merge the overlapping reads:

```
./pagan --overlap-merge-only --readsfile reads_file --overlap-merge-file merge_file
```

Assignment of reads to specific nodes

By default, PAGAN will add the sequences in the bottom of the alignment by aligning them against the root node. This can be overridden with options `--test-every-internal-node` and `--test-every-node` that exhaustively search through either internal nodes or all nodes (including leaf nodes) and add the sequence at the node where it matches the best.

If the phylogenetic position of the sequences to be added is known (e.g. the species in question is an outgroup for the clade X), this information can be provided for the placement. Furthermore, one can name multiple alternative positions for the sequences (because the precise position is not known or the reference contains paralogous genes and the copies which the sequences come from are not known) and PAGAN will choose the one where the sequences matches the best. Sequences are assigned to a specific node or a set of nodes using the NHX tree format with an additional tag `TID`. The tag identifier can be any string, in the example below we use number '001'. One can use any number/combination of tags as long as each node and each sequence has at most one tag.

As an example, let's assume

a reference alignment:

```
>A
AGCGATTG
>B
AACGATCG
>C
TGCGGTCC
```

and a read that we want to add in FASTA:

```
>D TID=001
AGCGATCG
```

or in FASTQ format:

```
@read_D_0001@3@15@13524@18140#0/1 TID=001
AGCGATCG
+
CCCCCCCC
```

The placement of the read in the reference alignment depends on the format of the reference tree:

- a phylogeny in plain Newick format adds the read at the root of the tree:

  ```
  ((A:0.1,B:0.1):0.05,C:0.15);
  ```

- a phylogeny in NHX format with one matching label adds the read to that node:

  ```
  ((A:0.1,B:0.1):0.05[&&NHX:TID=001],C:0.15);
  ```

- a phylogeny in NHX format with several matching labels finds the best node and adds the read to that:

  ```
  ((A:0.1,B:0.1):0.05[&&NHX:TID=001],C:0.15):0[&&NHX:TID=001];
  ```

In all cases, options `--test-every-internal-node` and `--test-every-node` override the placement information given in the guide tree and exhaustively search the best placement.

Consensus ancestor reconstruction and phylogenetic contig assembly

PAGAN's graph reconstruction is based on maximum parsimony. In NGS read placement, the overlapping regions of reads placed to a specific target node are expected to be identical and any differences observed are (with the exception heterozygosity that we ignore) some sort of errors. For the placement of DNA sequences, PAGAN offers an alternative approach to reconstruct ancestor graphs based on majority consensus. This is selected with option `--use-consensus`.

Using a similar majority consensus approach, PAGAN can combine reads placed to a specific target into longer contigs. This is done with option `--build-contigs`; the minimum number of reads supporting a site in the contig can be changed with option `--consensus-minimum`. The contigs, with the reads supporting them, are written to a file named `*_contigs.fas`.

By default, the disconnected contigs are bridged with `n`'s, estimating the size of the contig gap from the outgroup ancestral sequence. With option `--show-contig-ancestor` they can be bridged using the actual ancestral sequences, the contig gaps indicated with lower case characters.

Heuristic speed-ups for placement node search

PAGAN can use Guy Slater's Exonerate to speed up the search of the optimal placement node. The default combination of settings is chosen with option `--fast-placement` and typically one would also use option `--test-every-internal-node` or`--test-every-node`, or have a guide tree with tagged nodes for placement. With this option, PAGAN runs Exonerate local alignment against the chosen nodes (all/all internal/all tagged) and keep the best of them; it would then run Exonerate gapped alignment against those nodes and choose the best for the placement.

With other options, one can choose to perform just a local or gapped alignment with Exonerate, and change the number of nodes kept after each round. The number of nodes kept is based either on the score relative to the best score, or a fixed number of best-ranking nodes. If more than one node is returned by Exonerate, PAGAN will perform alignments against those and choose the best scoring one for the placement. With option `--use-exonerate-anchors`, Exonerate hits are used to reduce the search space in the final sequence placement; this option is automatically chosen with `--fast-placement`.

Typical analysis of NGS data

A typical command to perform the placement of NGS reads could be:

```
./pagan --ref-seqfile ref_alignment_file --ref-treefile ref_tree_file
        --readsfile reads_file [--trim-read-ends] [--discard-overlapping-identical-reads]
        [--rank-reads-for-nodes] [--pair-end] [--overlap-pair-end] [--454]
```

where the optional (and some mutually exclusive) options are in square brackets.

Depending on the output, the thresholds for masking and trimming can be adjusted using the relevant options explained below. The effect of trimming is obvious from the output; the bases written in lower case in the output were masked and considered as N's during the alignment. A sequence of N's matches any sequence and gives rubbish alignments; if that happens, you may need to raise the masking threshold or lower the trimming thresholds.

3.3 Additional program options

Some of the options printed by `./pagan --help` relate to unfinished features and may not function properly. Only the main options are documented here.

Generic options

- Option `--silent` minimises the output (doesn't quite make it silent, though).
- Options `--indel-rate` can be used adjust the insertion and deletion rates. Although it would be possible to consider the two processes separately, this has not been implemented yet.
- Options `--gap-extension` and `--end-gap-extension` define the gap extension probability for regular and terminal gaps. For meaningful results, the latter should be greater (and, for pair-end data, equal to `--pair-read-gap-extension`).
- Options `--dna-kappa` and `--dna-rho` affect the DNA substitution scoring matrix; base frequencies are estimated from the data.

- Options `--codons` translates DNA sequences to codons and aligns them using the codon substitution model. (experimental)
- Options `--scale-branches`, `--truncate-branches` and `--fixed-branches` override the branch lengths defined in the guide tree. By default, long branches are truncated to make the scoring matrix more informative; this can be prohibited with `--real-branches`.
- Option `--output-ancestors` writes the parsimony-reconstructed ancestral sequences for the internal nodes of the tree. The tree indicating the nodes is written in `outfile.anctree`.
- Option `--config-file file_name` specifies a config file. If an option is specified both in a config file and as a command-line argument, the latter one overrides the former.
- Option `--config-log-file file_name` specifies a log file where (non-default) options used for the analysis are written. The format is compatible with the option input and the file can be used a config file.

There are many parameters related to "insertion calling", the type and amount of phylogenetic information required to consider insertion-deletion as an insertion and thus prevent the later matching of those sites. These parameters are still experimental (although some of them are used and affect the resulting alignment) and will be described in detail later.

Phylogenetic placement

Some options are only relevant for the placement of sequences into existing alignment and many of those require FASTQ or NHX formatted input.

- Option `--qscore-minimum` sets the Q-score threshold for sites to be masked (replaced with N's for alignment, shown in lower case in the output); `--allow-skip-low-qscore` further allows skipping those sites (usefulness of this is uncertain). Masking is done by default (see `./pagan --help` for the default threshold) and can be disabled with option `--no-fastq` that prohibits all Q-score-based pre-processing done either by default or chosen by the user (e.g. trimming and 454-specific modelling).
- Option `--trim-read-ends` enables the trimming of FASTQ reads; `--trim-mean-qscore`, `--trim-window-width` and `--minimum-trimmed-length` define the minimum Q-score and width for the sliding window and the minimum length for the trimmed read. Trimming progresses inwards from each end until the mean Q-score for the window exceeds the score threshold; if the read is shortened below the length threshold, it is discarded. The length of removed (trimmed) fragments is indicated in the sequence name field in the format `P1ST$l1:P1ET$l2` where `$l1` and `$l2` refer to the start and end of the read. If the sequences consists of a read pair, the trimming of the right-hand side read is similarly indicated using the tag `P2`.
- Option `--rank-reads-for-nodes` ranks the sequences assigned to a node and aligns them in that order. The ranking is based on their score in the placement alignment used to decide between the alternative nodes. If sequences are assigned to one node each, one round of alignments against each node is performed to define the ranking before the final multiple alignment.

- Reads fully embedded in other reads can be discarded: `--discard-overlapping-reads` identifies reads that overlap in the placement alignment and removes ones that are fully embedded in another one; `--discard-overlapping-identical-reads` extends this by checking that the embedded read is identical (on base level) to the longer read before discarding it; `--discard-pairwise-overlapping-reads` makes all the pairwise alignments to identify embedded reads.
- Option `--reads-distance` sets the expected distance between the read and the pseudo-parent node (against which the read is aligned) and thus affects the substitution scoring used in the alignment. Having the distance very short (default), the alignment is stringent and expects high similarity.
- Reads with too few sites aligned against sites of reference sequences are discarded. (The stringency of the alignment is set using the option above.) Options `--min-reads-overlap` and `--min-reads-identity` set the required overlap and base identity for accepting the read.

The rest of the options are either not important for basic use or self-explanatory (or both).

3.4 Inference of ancestral sequences for existing alignments

The features required for phylogenetic placement of reads can be used without any reads, too. Command:

```
./pagan --ref-seqfile alignment_file --ref-treefile tree_file
```

writes the aligned sequences from `alignment` to files `outfile.fas` and `outfile.xml`, thus providing a tool to convert FASTA files to HSAML format. This maybe more interesting with option `--output-ancestors`. Command:

```
./pagan --ref-seqfile alignment_file --ref-treefile tree_file --output-ancestors
```

includes the parsimony-reconstructed internal nodes in the FASTA output, providing a efficient tool to infer gap structure for ancestral sequences based on existing alignments. The tree indicating the ancestral nodes is written in `outfile.anctree`.

3.5 Configuration files: input, output and format

Configuration files contain option names and values separated by `=` sign, one option per row. Rows starting with a hash sign `#` are comments and ignored. Thus, if the content of file `config.cfg` is:

```
# this is an uninformative comment
ref-seqfile = reference_alignment.fas
ref-treefile = reference_tree.nhx
readsfile = illumina_reads.fastq
outfile = read_alignment
trim-read-ends = 1
```

the command:

```
   ./pagan config.cfg                    ( or  ./pagan --config-file config.cfg  )
```

is equivalent to:

```
./pagan --ref-seqfile reference_alignment.fas --ref-treefile reference_tree.nhx
    --readsfile illumina_reads.fastq --outfile read_alignment --trim-read-ends
```

By adding the option `--config-log-file config.cfg` in the command above, PAGAN creates a config file that is equivalent with the one above (with some more comments). Config files can, of course, be written or extended manually using the same format. One should note, however, that also boolean options need a value assigned, such as `trim-read-ends = 1` in the example above. If a boolean option is not wanted, it should not specified in the config file (or it should be commented out with a hash sign) as setting an option e.g. `0` or `false` does not disable it.

3.6 Checking for PAGAN updates

PAGAN runs locally and does not contact or report to any external server. The only exception is the command `./pagan --version` that checks if a newer version of the program is available for download. The actual upgrade has to be performed manually.

Appendix: installing `boost` from source code

If pre-packaged files are not available for your platform or you are not entitled to install packages and cannot persuade your system administrator to do that, the Boost libraries necessary for the compilation of the PAGAN source code can be installed using following commands:

```
# make a temporary directory
mkdir ~/tmp_boost
cd ~/tmp_boost

# get the source code
wget http://sourceforge.net/projects/boost/files/boost/1.44.0/boost_1_44_0.zip/download
unzip boost_1_44_0.zip
cd boost_1_44_0

# fix the permission
chmod +x tools/jam/src/build.sh

# compile and install
sh ./bootstrap.sh --prefix=$PATH_TO_PAGAN_DIR/boost --with-libraries=program_options,regex
./bjam install

# check that it's there and set to library path
ls $PATH_TO_PAGAN_DIR/boost/lib
export LD_LIBRARY_PATH=$PATH_TO_PAGAN_DIR/boost/lib:$LD_LIBRARY_PATH

# clean up
cd ../..
rm -r ./tmp_boost
```

Note that you need to replace `$PATH_TO_PAGAN_DIR` with the appropriate file path. You also need to copy the line `export LD_LIBRARY_PATH=...` in your ~/.bashrc or similar such that it will automatically be set for future sessions.