Tutorial 4 – A pipeline to upload FASTA files from an EST sequencing and trim the polyA tail

Introduction

In this tutorial, we describe the specification of a pipeline that processes FASTA files from a cDNA sequencing project and trims the polyA tails.

The following steps constitute this pipe:

1. Uploading FASTA files;
2. Trimming polyA sequences;
3. Trimming polyT sequences;
4. Saving the trimmed sequences;

We have previously constructed a pipeline for this tutorial using CoEd, EGene's graphical configuration editor. The EGene's configuration file (`polyA_trimming.gen`) and its counterpart text file (`polyA_trimming.cnf`) can be found at the `config_files` directory. In order to run the pipeline, go to the `/examples/fasta_polyA_trimming_pipe` directory. This directory contains a multiFASTA file called `polyA.fasta`, which contains a set of sequences containing polyA sequences.

To run the pipe, you should type the following command:

```
bigou.pl -c ../config_files/polyA_trimming.cnf
```

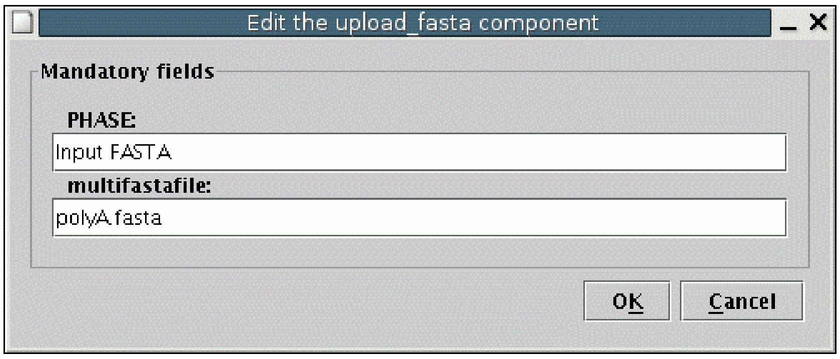
If everything goes well, you should now find the following additional files in this directory:

```
trimmed_sequences.fasta
```

1. Uploading sequences in FASTA format

Configuration parameters in the .cnf file:

```
#=====================================================================
PHASE=Input fasta
program = upload_fasta.pl
#---------------------------------------------------------------------
multifastafile = polyA.fasta
#=====================================================================
```


Edit the upload_fasta component

Mandatory fields

PHASE:
Input FASTA

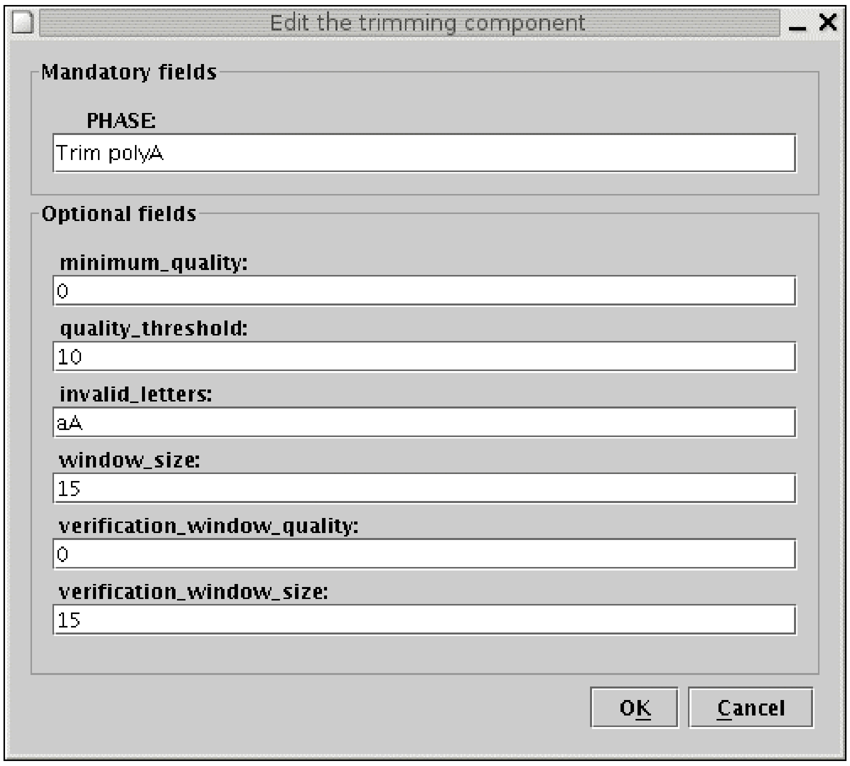
multifastafile:
polyA.fasta

This step uses the component `upload_fasta.pl` to upload a multiFASTA file. This file is composed by multiple concatenated sequences in FASTA format. The only argument to this component is the name of this multiFASTA file (in our case `polyA.fasta`). It is assumed that `bigou.pl` is run while the shell is in the directory that contains the file `polyA.fasta`. Alternatively, the user can specify a complete path for the file (e.g. `/home/test/polyA.fasta`). Note: FASTA files do not contain data about base quality. EGene assumes in this case all bases have a Phred quality equal to 20.

2. Trimming polyA sequences

Configuration parameters in the .cnf file:

```
#======================================================================
PHASE=Trim polyA
program = trimming.pl
#----------------------------------------------------------------------
minimum_quality = 0
quality_threshold = 10
invalid_letters = Aa
window_size = 15
verification_window_quality = 0
verification_window_size = 15
#======================================================================
```


Edit the trimming component

Mandatory fields

PHASE: Trim polyA

Optional fields

minimum_quality: 0

quality_threshold: 10

invalid_letters: aA

window_size: 15

verification_window_quality: 0

verification_window_size: 15

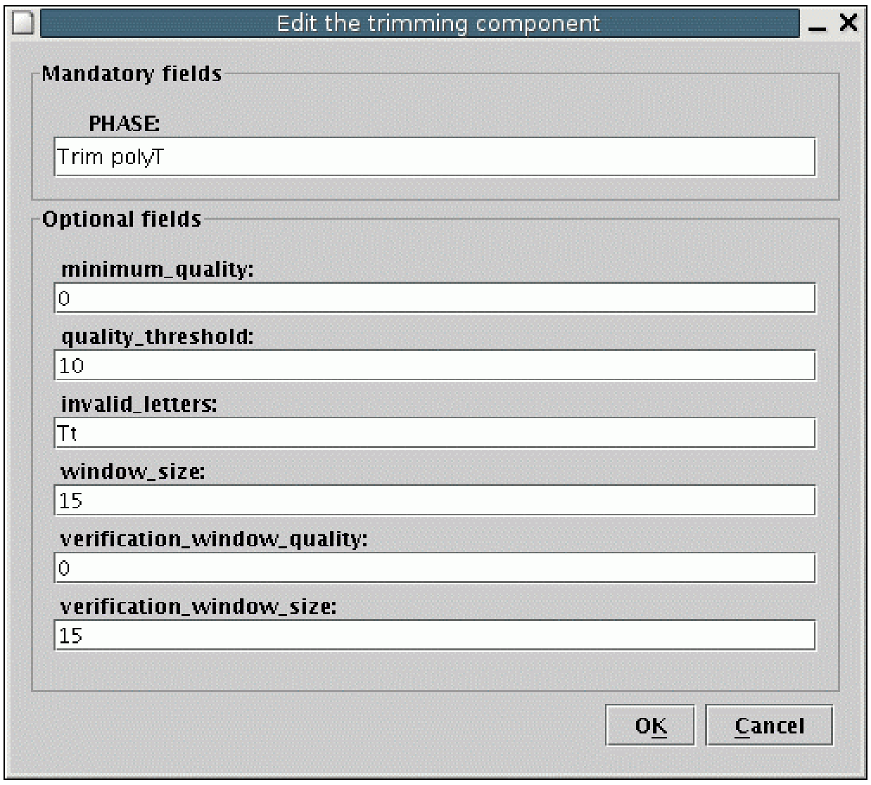
We have not developed a specific component to trim polyA tails from cDNAs. Instead, we use the `trimming.pl` component, already presented in Tutorial 1. This component can trim sequence ends containing low quality bases (measured in terms of Phred evaluation), or when it finds ambiguities (`nN` bases) and masked bases (`xX`). Because the characters to be considered as invalid can be generically specified in the parameter `invalid_letters`, any base can be classified as invalid. Using this feature, in order to trim polyA tails we have to specify the characters “`Aa`” as invalid. The parameters are:

- `minimum_quality`: because quality is not taken into account for FASTA files, this parameter is set to zero.
- `quality_threshold`: indicates the minimum percentage of good bases in a window for it to be considered acceptable. In this case the value `10` (10%) indicates that we consider as polyA a window where at least 90% of the bases are invalid (that is “`a`” or “`A`”).
- `window size`: this is the size of the sliding window at the first step. The program scans the sequence starting, respectively, at the 5' and 3' ends with a window of 15 bp. In either case, the window is moved towards the center while the number of good bases is below the threshold. The program trims all bases between the window and the 5' (or 3') end of the sequence.
- `verification_window_quality`: because quality is not taken into account for FASTA files, this parameter is set to zero.
- `verification_window_size`: size of the sliding window used in the second step. Using the second step enables the program to find polyA tails that are not immediately at the 5’ or 3’ends of the sequence. This is not an uncommon situation in sequencing projects, especially when miscalled bases appear after polyA sequences. It is extremely important for this parameter to be set correctly. The window size should be large enough to preclude trimming when a small stretch of adenines is found in the middle of the sequence (here we consider 15 bases a sufficient size).

3. Trimming polyA sequences

Configuration parameters in the .cnf file:

```
========================================================================
PHASE=Trim polyT
program = trimming.pl
#-----------------------------------------------------------------------
minimum_quality = 0
quality_threshold = 10
invalid_letters = Tt
verification_window_size = 15
verification_window_quality = 0
window_size = 15
#=======================================================================
```



Mandatory fields

PHASE: Trim polyT

Optional fields

minimum_quality:	0
quality_threshold:	10
invalid_letters:	Tt
window_size:	15
verification_window_quality:	0
verification_window_size:	15

We have to specify the characters “Tt” in the `invalid letters` parameter in a second trimming, just to assure that polyA tails present in reverse complementary sequences will be also removed.

Note: Steps 2 and 3 cannot be reduced to a single pass of `trimming.pl` using “`AaTt`” as the invalid letters. This would result in a trimming of AT-rich sequences, not only homopolymeric regions of A or T bases, clearly an undesired effect.

The last two steps are performed by `snoop_filtered.pl` and `outsave.pl` components, both of them already discussed in Tutorial 1.