Tutorial 5 - A pipeline to upload a file containing multiple FASTA sequences and processing it through a set of compositional filters

Introduction

In this tutorial we describe the specification of a pipeline that uploads a multiFASTA file and processes the sequences through a set of compositional filters.

The following steps constitute this pipe:

1. Uploading a file containing multiple FASTA sequences;
2. Using `filter_quality.pl` to filter out all the sequences presenting a G+C content higher than 60%;
3. Creating a multiFASTA file containing all sequences with a G+C content higher than 60%;
4. Use `filter_quality.pl` to filter out all the remaining sequences presenting a G+C content higher than 50%;
5. Creating a multiFASTA file containing all sequences with a G+C content higher than 50%;
6. Using `filter_quality.pl` to filter out all the remaining sequences presenting a G+C content higher than 30%;
7. Creating a multiFASTA file containing all sequences with a G+C content higher than 30%;
8. Creating a multiFASTA file containing the sequences not filtered by previous steps using the `outsave.pl` component.

We have previously constructed a pipeline for this tutorial using CoEd, EGene's graphical configuration editor. The EGene's configuration file (`composition_filter.gen`) and its counterpart text file (`composition_filter.cnf`) can be found at the config_files directory. In order to run the pipeline, go to the `/examples/compositional_filter_pipe` directory. This directory contains the file `sequences.fasta`, which is composed by the following sequences:

- *Toxoplasma gondii* apicoplast genome (21%GC)
- *Plasmodium falciparum* mitochondrial genome (32% GC)
- *Baccillus stearothermophilus* GAPDH gene (55% GC)
- *Streptomyces arenae* gapR gene (69% GC)

Now type the command below:

```
bigou.pl –c ../config_files/composition_filter.cnf
```

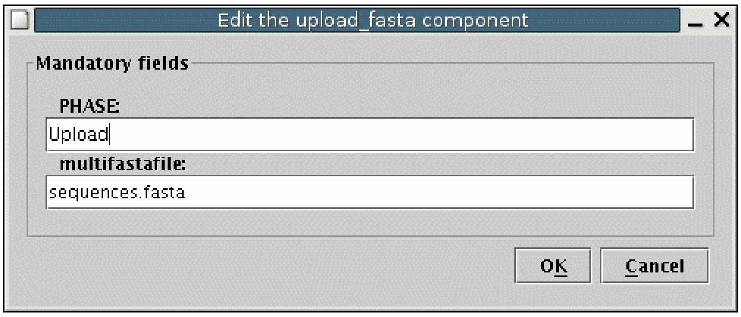
If everything goes well, you should now find the following additional files in this directory:

```
plus_30%GC.fasta
plus_50%GC.fasta
plus_60%GC.fasta
remaining_sequences.fasta
```

Understanding the pipeline and the component parameters

1. Uploading sequences in a multifasta file

```
#==================================================================
PHASE=Upload
program = upload_fasta.pl
#------------------------------------------------------------------
multifastafile = sequences.fasta
#==================================================================
```



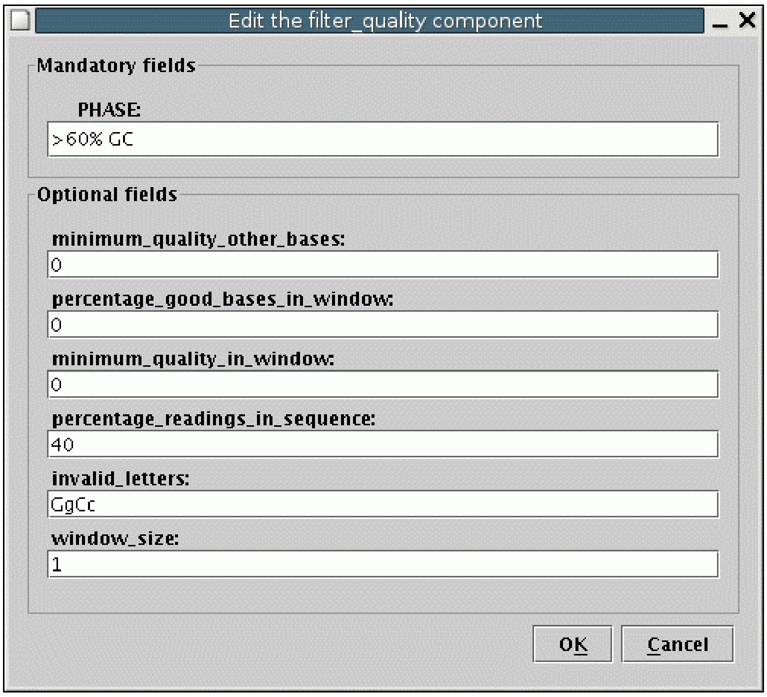
PHASE: Upload

multifastafile: sequences.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 `sequences.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/sequences.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. Using `filter_quality.pl` as a compositional filter

```
#==================================================================
PHASE=>60% GC
program = filter_quality.pl
#------------------------------------------------------------------
minimum_quality_other_bases = 0
percentage_good_bases_in_window = 0
minimum_quality_in_window = 0
percentage_readings_in_sequence = 40
invalid_letters = GgCc
window_size = 1
#==================================================================
```



PHASE: >60% GC

minimum_quality_other_bases: 0

percentage_good_bases_in_window: 0

minimum_quality_in_window: 0

percentage_readings_in_sequence: 40

invalid_letters: GgCc

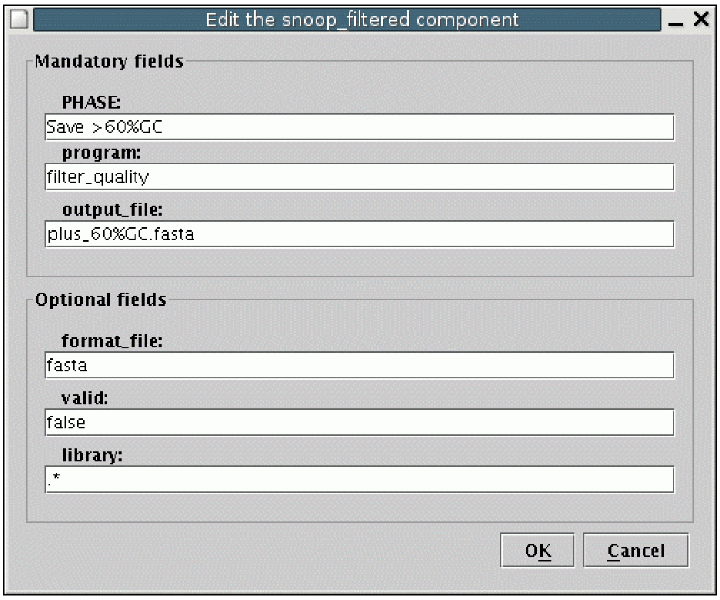
window_size: 1

We describe below the arguments of this component:

- `minimum_quality_other_bases`: we are not interested in evaluating quality, so the parameter is set to zero
- `percentage_good_bases_in_window`: we only want to evaluate the overall composition of the sequence, therefore we set this argument to zero.
- `minimum_quality_in_window`: windows are not going to be used, so this value is set to zero.
- `invalid_letters`: since we intend to filter out sequences based on their G+C content, we have first to declare which are the invalid letters, in both upper and lower cases: "`GcCc`".
- `percentage_readings_in_sequence`: this parameter determines the minimum percentage of valid bases for a read. Thus, if this parameter is set to 40, it means that any read presenting more than 60% G+C bases (invalid letters) content will be tagged as invalid.

3. Using `snoop_filtered.pl` to save filtered sequences

```
#=================================================================
PHASE=Save >60%GC
program = snoop_filtered.pl
#-----------------------------------------------------------------
#-----------------------------------------------------------------
program = filter_quality
output_file = plus_60%GC.fasta
format_file = fasta
valid = false
library = .*
#=================================================================
```



Edit the snoop_filtered component

Mandatory fields

PHASE: Save >60%GC

program: filter_quality

output_file: plus_60%GC.fasta

Optional fields

format_file: fasta

valid: false

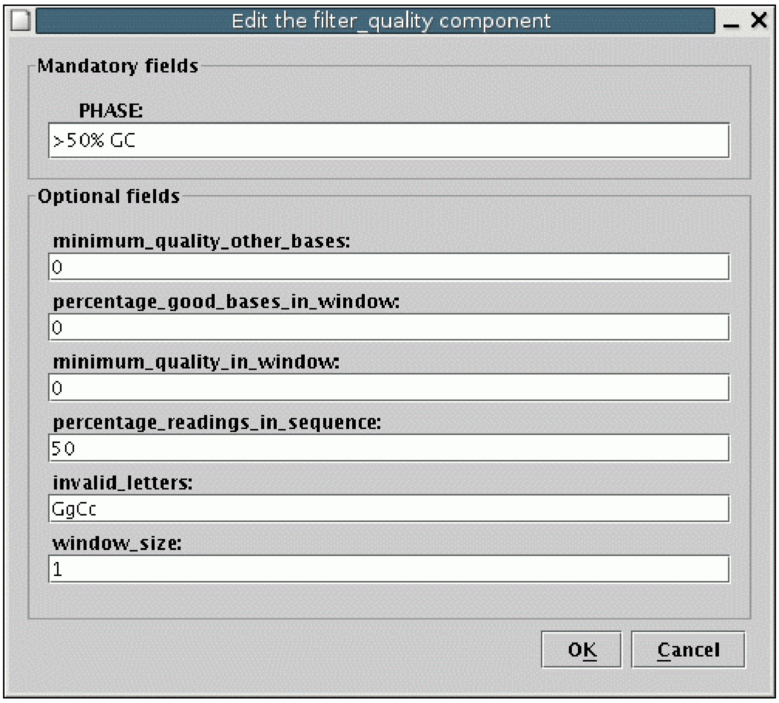
library: .*

The `snoop_filtered.pl` component can be used to save sequences that are tagged either as valid or invalid. In this particular example, we want to save those sequences presenting a G+C content higher than 60%. To do this, we need the following parameter setting:

- `valid`: we want to save the sequences that were discarded for having G+C content to high (therefore tagged as invalid), so we set this argument to false.
- `program` = we want to save sequences invalidated by our compositional filter, which was performed using the program `filter_quality.pl`.
- `output_file`: this is the name of the output file, in our case `plus_60%GC.fasta`.
- `format_file`: this argument sets the format of the description of the sequences in the output file, in our case `fasta`.
- `library` = this argument is used when issuing reports on sequences filtered by similiary against some sequence library. The ".*" value, indicates that there is no restrictions here.

Repeating steps 2 and 3 with different parameters, allows one to create distinct files containing sequences differing in their G+C composition (`plus_30%GC.fasta`, `plus_50%GC.fasta` and `plus_60%GC.fasta`).

Filtering sequences presenting more than 50% G+C bases (invalid letters)



Mandatory fields

PHASE: >50% GC

Optional fields

minimum_quality_other_bases: 0

percentage_good_bases_in_window: 0

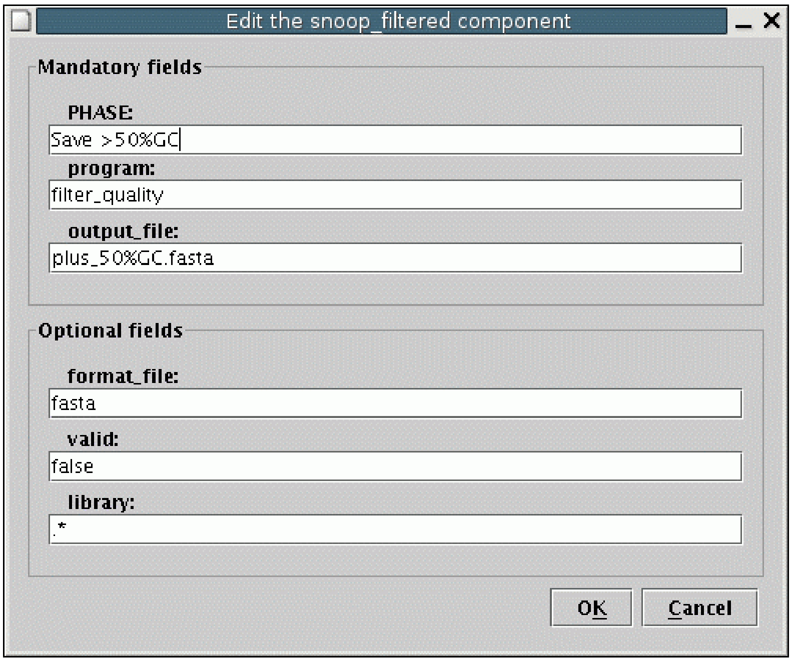
minimum_quality_in_window: 0

percentage_readings_in_sequence: 50

invalid_letters: GgCc

window_size: 1

Saving sequences presenting more than 50% G+C bases (invalid letters)



Mandatory fields

PHASE: Save >50%GC

program: filter_quality

output_file: plus_50%GC.fasta

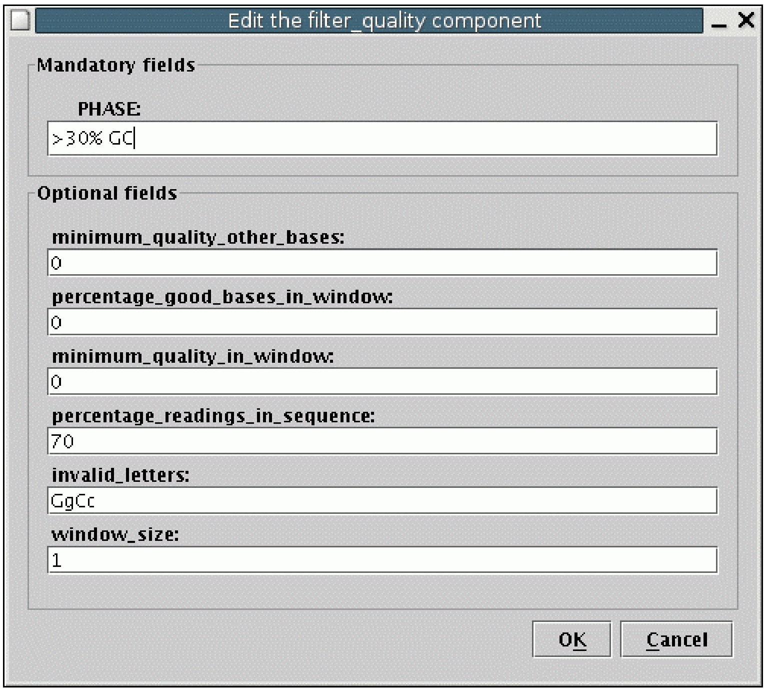
Optional fields

format_file: fasta

valid: false

library: .*

Filtering sequences presenting more than 30% G+C bases (invalid letters)



Mandatory fields

PHASE:
>30% GC

Optional fields

minimum_quality_other_bases:
0

percentage_good_bases_in_window:
0

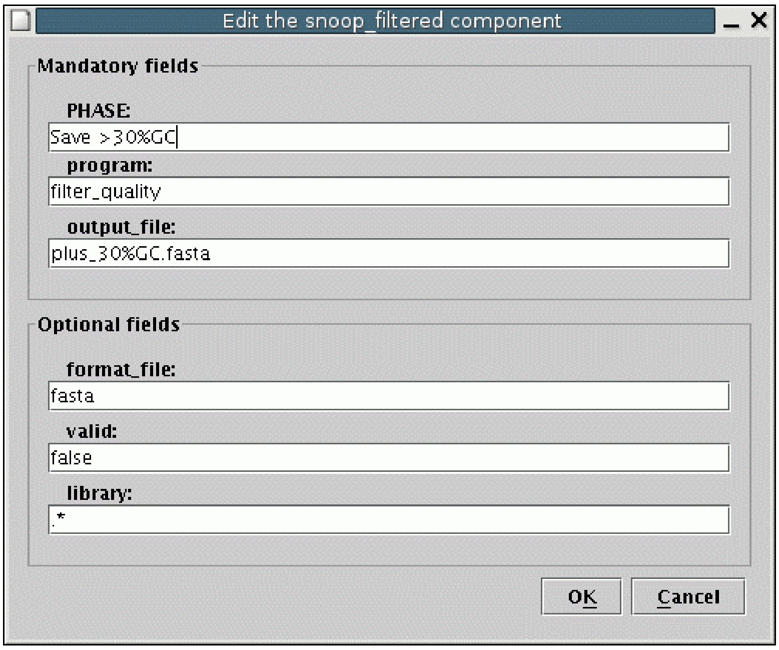
minimum_quality_in_window:
0

percentage_readings_in_sequence:
70

invalid_letters:
GgCc

window_size:
1

Saving sequences presenting more than 30% G+C bases (invalid letters)



Mandatory fields

PHASE:
Save >30%GC

program:
filter_quality

output_file:
plus_30%GC.fasta

Optional fields

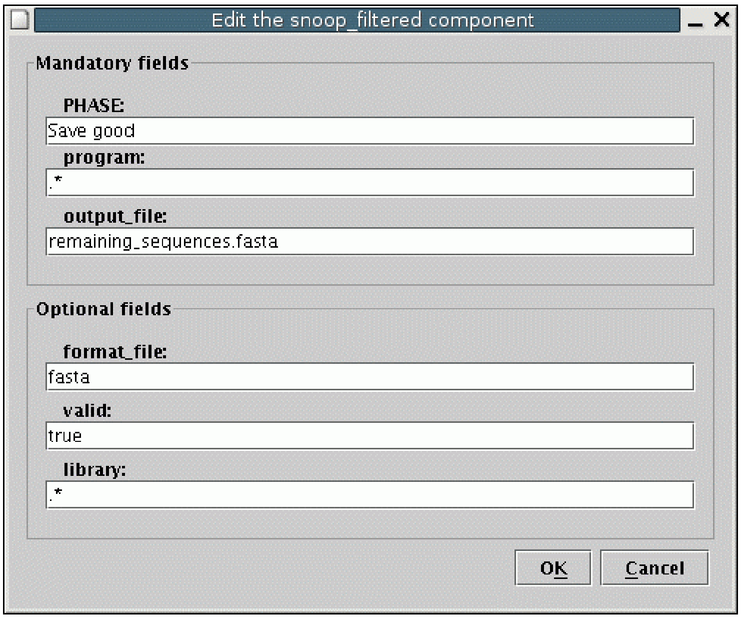
format_file:
fasta

valid:
false

library:
.*

4. Using `snoop_filtered.pl` to save the valid sequences

```
#===============================================================
PHASE=Save good
program = snoop_filtered.pl
#---------------------------------------------------------------
program = .*
output_file = remaining_sequences.fasta
format_file = fasta
valid = true
library = .*
#==============================================================
```



Mandatory fields

PHASE: Save good

program: .*

output_file: remaining_sequences.fasta

Optional fields

format_file: fasta

valid: true

library: .*

The component `snoop_filtered.pl` can be used to create a multifasta file with all the sequences that have not been invalidated at a certain step of the pipeline. In our example, `snoop_filtered.pl` will be the last component of the pipeline and, therefore, will save all the remaining valid sequences at the end of processing. We explain below the parameter settings:

- `program`: since we want the valid sequences, setting the filtering program is not relevant, therefore this parameter should be set to the default ".*" value, indicating any program.
- `output_file`: this argument specifies the name of the file to be generated.
- `format_file`: we want a multifasta file, so this parameter should be set to `fasta.`
- `valid`: this parameter should be set to `true`.
- `library`: specifying a library is not relevant, we should use the default parameter, ".*", which indicates that this argument is irrelevant.

Note that `snoop_filtered.pl` saves ALL the sequences, EVEN those that have been previously filtered out with another pre-determined filter parameters. This is a characteristic of this system. Definition of a range of compositional filtering (e.g. 40 to 60% G+C content) is not currently supported. For this reason, `plus_30%GC.fasta` will contain three sequences, `plus_50%GC.fasta` two sequences and so forth.