Scott Clark
April 23, 2010
How to use Velvetrope

1 Introduction

Why would you want to use velvet rope? THIS IS WHY. (To be filled in later)

2 Installation

Requires (general packages): `python, gcc, swig, python-dev`
Requires (python packages): `numpy, matplotlib, biopython, scipy`

To install (command line, from directory where it was downloaded)

1. Unzip the file.
2. Open the directory created.
3. Open the `src` directory.
4. Run `make`.
5. Go back to the main directory.

```
$ tar xvfz VelvetropeBuildMMDDYY.tar.gz
$ cd VelvetropeBuildMMDDYY
$ cd src
$ make
$ cd ..
```

Velvetrope is now installed!
To run Velvetrope run the following command (from the main directory):

```
$ python Velvetrope.py [options] <input file>
```

For more on the options use the `-h` option or read the user manual:

```
$ python Velvetrope.py -h
```

3 Running the program

3.1 Simplest case

```
$ python Velvetrope.py <input file>
```

Where `<input file>` is a file composed of amino acid sequences in fasta format. The program will compare the first sequence against all of the rest with all of the default parameters.

4 Manually defining options

There are two ways to manually define options (like wanting to run a DNA file, or change the filter thresholds etc). We can do it through the command line or through a parameter file.

4.1 Options via the command line

We can see all of the different command line options by running the following command:

```
$ python Velvetrope.py -h
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #
# Velvetrope - a bitwise algorithm for finding local alignments in sequences          #
# Velvetrope 0.14 (25 Apr 2010)                                                       #
# Copyright (C) 2009,2010 Scott Clark <sc932 at cornell dot edu>                      #
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #

Usage:  python Velvetrope.py [-options] <inputfile>

Where basic options are:
  -h :  show brief help on version and usage

Input options:
  -pf <parameter file>     :  valid parameter file as described in the manual
                              other options override file parameters
  -rd <run depth (rgo)>    :  rgo for read, generate and output;
                              rg for read, generate only; etc [rgo]
  -rs <reference sequence> :  ref seq name (ie 'E coli K12') [defaults to first in file]
  -seq <sequence type>     :  sequence type (currently only AA and DNA supported) [AA]

Output options:
  -o <output file>      :  file in which output is stored [VRout+date]
  -odir <output folder> :  folder in which all output is stored [/VRout+date]
  -log <logging level>  :  logging output level (error,warning,info,debug) [debug]

Plotting options (n>0):
  -plotscreen :  will send plots to screen as well as file
  -allplots   :  make all types of plots
  -CDFplot    :  make CDF plot
  -CDFnum <n> :  number of CDF plots to make
  -GBGplot    :  make Gene By Gene plot
  -Comboplot  :  make Combo (one vs all) plot
  -RegOut     :  don't show regular expressions of locally aligned areas
  -Latex      :  output latex scripts
```

```
Algorithm parameters (x>0.0) (n>0):
  -globSig <x> :  the number of matches in standard deviations away from the expectation
                  an offset needs to have to trigger the global filter [8.0]
  -locSig <x>  :  the number of matches in standard deviations away from the expectation
                  a local vector needs to have compared to a window length previous [8.0]
  -locWin <n>  :  size of the window in which the local filter searches over,
                  also the minimum local alignment length [20]
  -locGap <x>  :  how close local alignments on the same offset need to be to be combined
                  by the multiCDF method (a multiple of window length > 0) [3.5]
  -intron      :  turns the intron (repeat) finder and eliminator on
  -intRat <x>  :  how much one local alignment needs to overlap another in order to be
                  considered a possible intron (repeat) [0.5]
  -intTot <x>  :  how much many local alignments need to overlap another in order to be
                  considered a possible intron (of total) [0.05]
  -regRat <x>  :  ratio of test sequences need to have an amino acid for it to be put in
                  the regular expression [0.5]
  -regGap <n>  :  how large of a gap before the next regular expression is started [5]

Please see the user manual for more information on what each parameter means
```

Each parameter can be defined independently, order does not matter. Every parameter has a default value (shown in the brackets). Variables that are shown as n are positive integers, x are positive real numbers (doubles). Each parameter is explicitly defined in the next section.

4.2 The Parameter File

The other way to redefine options is the parameter file. To invoke a parameter file we use the command line option

```
$ python Velvetrope.py -pf <parameter file> <input file>
```

The parameter file is a text file with three types of lines:

1. Blank lines: do nothing
2. Lines that start with a percentage sign: comments
3. Lines of the form VARIABLE = value

An example parameter file is as follows:

```
% This is a parameter file for Velvetrope
INPUT_FILE = testData/duoBactIntron.faa
SEQ_TYPE = AA
REF_GENE = Escherichia coli str.  K-12 substr.  MG1655
```

When a parameter is defined in the parameter file AND on the command line then the command line definition will take priority.

5 The parameters

Below is a list of all parameters and how the affect the program. They are organized by type and name (in the form `-command line name` | `parameter file name`).

5.1 Basic options

5.1.1 `-h` | n/a

Name: Help
Type: Boolean
What: Prints copyright info and command line help
Default Value: False

5.2 Input Options

5.2.1 `-pf` | n/a

Name: Parameter File
Type: Text
What: Specifies where the parameter file is. All parameters defined in the command line take precedence over those defined in the parameter file.
Default Value: n/a

5.2.2 `-rd` | n/a

Name: Run Depth
Type: r OR rg OR rgo OR g OR go OR o
What: Sets the depth in which the program will run. (r)ead, (g)enerate, (o)utput. If output is not turned on then the program will create an output file with all of the data in it. Otherwise all turned on plots will be generated.
Default Value: rgo

5.2.3 `-rs` | `REF_GENE`

Name:
Type:
What:
Default Value:

5.2.4 `-seq` | `SEQ_TYPE`

Name: Sequence type
Type: AA or DNA
What: Defines the type of sequence. Right now only DNA (6 frame) and AA are supported.
Default Value: AA

5.3 Output Options

5.3.1 `-o` | `OUTFILE`

Name: Output file
Type: Text
What: File in which output is stored (if run depth does not end in o)
Default Value: tmpVRout

5.3.2 `-odir` | n/a

Name: Output directory
Type: Text
What: Folder in which output is stored
Default Value: output/VRout+date (ie output/VRout_m4d21...)

5.3.3 `-log` | n/a

Name: Logging level
Type: debug or info or warning or error
What: Level of logging info displayed to screen
Default Value: info

5.4 Plotting Options

5.4.1 `-plotscreen` | `PLOT_TO_SCREEN`

Name: Screen output switch
Type: Boolean (0 or 1)
What: Whether the plots are shown on the screen or just saved.
Default Value: 0 (False)

5.4.2 `-allplots` | n/a

Name: All plots switch
Type: Boolean (0 or 1)
What: Turns on all of the plot types
Default Value: 0 (False)

5.4.3 `-CDFplot` | `PLOT_CDF`

Name: CDF plot switch
Type: Boolean (0 or 1)
What: Turn on/off the CDF output
Default Value: 0 (False)

5.4.4 `-CDFnum` | `PLOT_CDF_NUM`

Name: CDF plot number
Type: Positive integer ($n > 0$) or 0
What: Number of CDF plots to plot (0 will plot them all)
Default Value: 1

5.4.5 `-GBGplot` | `PLOT_GBG`

Name: Gene-by-gene plot switch
Type: Boolean (0 or 1)
What: Turn on/off the gene-by-gene alignment plot information
Default Value: 0 (False)

5.4.6 `-Comboplot` | `PLOT_COMBO`

Name: Combo plot switch
Type: Boolean (0 or 1)
What: Turn on/off the combined (histogram) output
Default Value: 0 (False)

5.4.7 `-RegOut` | `REG_EXP_OUT`

Name: Regular Expression output switch
Type: Boolean (0 or 1)
What: Show regular expressions of locally aligned areas (including introns)
Default Value: 1 (True)

5.4.8 `-Latex` | `PLOT_LATEX`

Name: Latex output switch
Type: Boolean (0 or 1)
What: Turn on/off the latex output
Default Value: 0 (False)

5.5 Algorithm Parameters

5.5.1 `-globSig` | `GLOBAL_SIG_LEVEL`

Name: Global sigma level
Type: Costive real number ($x > 0$).
What: Sets the number of matches in standard deviations away from the expectation an offset needs to have to trigger the global filter.
Default Value: 4.5

5.5.2 `-locSig` | `LOCAL_SIG_LEVEL`

Name: Local sigma level
Type: Costive real number ($x > 0$).
What: Sets the number of matches in standard deviations away from the expectation a local vector needs to have compared to the value a window length previous.
Default Value: 3.0

5.5.3 `-locWin` | `LOCAL_WINDOW`

Name: Local Window
Type: Positive integer ($n > 0$).
What: The size of the window in which the local filter searches over, also the minimum local alignment length.
Default Value: 20

5.5.4 `-locGap` | `LOCAL_BRIDGE_WIDTH`

Name: Local Bridge Width
Type: Costive real number ($x > 0$).
What: How close local alignments on the same offset need to be to be combined by the multiCDF method (a multiple of window length, ie 3.5)
Default Value: 3.5

5.5.5 `-intron` | `INTRON_FINDER`

Name: Intron Finder Switch
Type: Boolean
What: Turns the intron (repeat) finder on or off
Default Value: False

5.5.6 `-intRat` | `INTRON_TRUTH_RATIO`

Name: Intron Truth Ratio
Type: Costive real number ($x > 0$).
What: The percentage of local alignments that contain a single position for that position to be considered an intron area (ie more than 0.05 of all alignments contain a position)
Default Value: 0.5

5.5.7 `-intTot` | `INTRON_OVERLAP`

Name: Intron Overlap
Type: Costive real number ($x > 0$).
What: How much of a local alignment needs to overlap another (in the sequence of interest and test sequence) in order to be considered a possible intron
Default Value: 0.05

5.5.8 `-regRat` | `REG_EXP_RATIO`

Name: Regular Expression ratio
Type: Costive real number ($x > 0$).
What: What ratio of test sequences need to have an amino acid for it to be put in the regular expression
Default Value: 0.5

5.5.9 `-regGap` | `REG_EXP_GAP`

Name: Regular Expression Gap
Type: Positive integer ($n > 0$).
What: How many wildcards before the next regular expression is started
Default Value: 5

6 Benchmarking

Requires: `BLAST` and `HMMer` to be installed and `PATH` defined such that running `blastp` and `phmmer` in the shell call their respective binaries.

6.1 Sensitivity and Specificity

Sensitivity, or Type I error, in our case is the percentage of correctly identified areas of local alignment. Specificity, or Type II error, is the percentage of incorrectly identified areas of local alignment (what percentage of areas not in a local alignment were falsely classified as being in one).

6.2 The test

There are three parameters to the benchmark, the length of the test sequence L, the average length of the subsequence S and the number of sequences per level of identity to compute N.

To fairly test the algorithms we generate a set of sequences to test so as to avoid any bias of annotated sequences that were discovered using either of the methods or any similar method.

First we generate a sequence of interest of length L.

Next, we generate $N_{sub} = N$ test sequences of length $L_i \propto N(L, \sigma_L) > L_{min}$ ($\sigma_L = \sqrt{(}L)$) where these are randomly generated normally distributed lengths about L with standard deviation σ_L and a hard minimum of $L_{min} = L/2$. If the random number generator generates a length less than L_{min} another number is chosen.

For each of these N_{sub} test sequences we randomly choose a second length D_i similarly with $D_i \propto L_i > N(S, \sigma_S) > S_{min} = S/2$ ($\sigma_S = \sqrt{(}S)$). This is the length of the local alignment between the test sequence and the sequence of interest.

Next we uniformly and independently choose where the local alignments will match up on the two sequences (beginning of SOI matching ending of TS etc).

Then we generate the test sequence as follows: If not in a local alignment, choose a random amino acid. If in a local alignment, choose amino acid corresponding to the particular position on the sequence of interest that aligns with that position with probability p, if not then randomly choose a different amino acid.

This is done for $p = 1.0, 0.9, \ldots, 0.1$.

6.3 Running the test

Run the following command from the main directory:

```
$ ./benchmarker.sh L S N
```

Example:

```
$ ./benchmarker.sh 150 50 500
```

7 Using the test data

We can run the command to see how Velvetrope performs on 19 equidistant bacterial RNA polymerase beta subunit sequences.

```
$ python Velvetrope.py -seq DNA -allplots -plotscreen -CDFplotNum 0 testData/bact23dna.fna
```

The default values allow for a lot of overlap, they are tuned to be highly sensitive. To mitigate this effect we can up the thresholds.

```
$ python Velvetrope.py -seq DNA -allplots -plotscreen -CDFnum 19 -locSig 4.5 testData/bact2
```

There is still some overlap. This comes from the fact that if there is a very long sequence of identity and in the sequence there is an insertion, followed some time later by a deletion then a section in the middle of the the sequence will be exactly one offset away from the rest. This will cause two overlapping local alignments with the current code. To fix this I am going to implement the multi-align algorithm from python (previous writeup) in C.