Scallop User Reference

Mingfu Shao[1] and Carl Kingsford[1]

[1]Computational Biology Department, Carnegie Mellon University
{mingfu.shao, carlk}@cs.cmu.edu

1 Installation

To install Scallop, you need to first download/compile a few software packages (Samtools, Boost, and GUROBI), setup the corresponding environmental variables, and then compile the source code of Scallop.

1.1 Install Samtools

Download Samtools from `http://www.htslib.org/` with version 1.2 or higher. Compile it to generate the htslib file `libhts.a`. Set environment variable `HTSLIB` to indicate the directory of `libhts.a`. For example, for Unix platforms, add the following statement to the file `~/.bash_profile`:

```
export HTSLIB="/directory/to/your/htslib/htslib-1.2.1"
```

1.2 Install Boost

Download Boost from `http://www.boost.org`. Uncompress it somewhere (compiling and installing are not necessary). Set environment variable `BOOST_HOME` to indicate the directory of Boost. For example, for Unix platforms, add the following statement to the file `~/.bash_profile`:

```
export BOOST_HOME="/directory/to/your/boost/boost_1_60_0"
```

1.3 Install GUROBI

Download GUROBI from `http://www.gurobi.com/` and uncompress the package somewhere (compiling and installing are not required). You need to apply an academic license to use the full features of GUROBI (Please refer to the GUROBI documentation for more information.) After that, set two environment variables, `GUROBI_HOME` and `GRB_LICENSE_FILE`, which indicates the directory of GUROBI, and the location of your license file, respectively. For example, for Unix platforms, add the following two statements to the file `~/.bash_profile`:

```
export GUROBI_HOME="/directory/to/your/gurobi/linux64"
export GRB_LICENSE_FILE="/location/of/your/license/gurobi.lic"
```

1.4 Compile Scallop

Get the source code of Scallop through `git`:

```
$git clone git@github.com:shaomingfu/scallop.git .
```

Execute the following commands to generate `Makefile` and compile:

```
$cd src
$aclocal
$autoconf
$autoheader
$automake -a
$./configure
$make
```

The executable file `scallop` will be present at `src/src`. You might want to link it into `bin` through

```
$cd bin
$ln -sf ../src/src/scallop .
```

2 Command line

The usage of Scallop is as follows:

```
$./scallop -c config -i input.gtf -a algo -o output.gtf
```

Parameter `config` configures the behavior of the algorithm. There is such an example configure file at `bin/example.config`.

Currently we work on perfectly estimated splice graph, represented in a `gtf` file with augmented expressions. One such example can be found at `bin/example.expression.gtf`. With this file Scallop will first build the splice graph, and then try to decompose the graph to recover the transcripts as well as their corresponding abundances.

There are three options for `algo` parameter: `scallop1`, `scallop2`, and `greedy`. With option of `scallop1`, the program will only run the core algorithm to partly decompose the given splice graph, which will predict fewer transcripts but with higher accuracy. With option of `scallop2`, the program will completely decompose the given splice graph, using greedy algorithm following the core part of the algorithm. With option of `greedy`, the program will only use greedy algorithm to fully decompose the given splice graph.

The predicted transcripts will be written in parameter `output.gtf`.

3 Simulation and Evaluation

We use Flux Simulator to simulate transcript expression. Before simulating, an annotation file (a `gtf` file) of a particular genome is required. Sometimes Flux Simulator crashes with some `gtf` files for some format issue. To avoid this, you might want to fix the format of the raw `gtf` file through using the script at `bin/fix.gtf.sh`:

```
$./fix.gtf.sh raw.gtf > new.gtf
```

We also need to prepare a parameter file for Flux Simulator. There is such an example parameter file at `bin/flux.exp.params`. Make sure that in this parameter file REF_FILE_NAME is specified as the (fixed) `gtf` file we mentioned above. Now we can run the Flux Simulator:

```
$flux-simulator -p param-file -x
```

An expression file `profile` (specified in the `param-file`) will be generated. Now we need to merge the original `gtf` file with this expression file to create the input `gtf` file for Scallop. We can do this by using `bin/merge.exp.pl`:

```
$./merge.exp.pl new.gtf profile > input.gtf
```

To evaluate the preformance of predicted transcripts, we provided a tool located at `gtfcompare/` (you need to compile the source code):

```
$./gtfcompare output.gtf input.gtf > summary
```

In the `summary` file, for each gene, it gives the number of transcripts in `output.gtf` and `input.gtf` respectively, and the number of them that appears in both file. If this common number is equal to the number for `output.gtf`, a `TRUE` is followed; otherwise, a `FALSE` is followed. A summary line is given at the bottom line of the `summary` file.