Zerone tutorial

Pol Cuscó and Guillaume Filion

October 19, 2015

1 Building instructions

Zerone is available as a Linux command line application and as an R package.

1.1 Downloading

We recommend that you use git to keep Zerone updated. You can clone the repository from Github with the following command on a standard terminal.

```
git clone git@github.com:gui11aume/zerone
```

Note that this requires that you already have a Github account and that the computer you are working on has an SSH key registered on GitHub. If this is not the case, follow the instructions from `https://help.github.com/articles/generating-ssh-keys/`.

Alternatively, if you prefer not to use git, you can download the source code from `https://github.com/gui11aume/zerone/archive/master.zip` with the following commands.

```
wget https://github.com/gui11aume/zerone/archive/master.zip
unzip zerone-master.zip
mv zerone-master zerone
```

This should create a directory named `zerone`.

1.2 Compiling

To build Zerone, execute the following from the `zerone` directory.

```
cd zerone
make
```

This should succeed on most Linux systems because `make` is available by default. If this is not the case, you can obtain it by typing `sudo apt-get install make` on the Ubuntu terminal.

Calling `make` should create an executable called `zerone`.

1.3 Testing

To check that the building was successful, test Zerone with the following commands.

```
cd src/test
make test
cd ../..
```

If it passes the tests without any error message, then everything went fine and you are done with the build. If not, something went wrong. In this case, you can explain how to reproduce the problem on `https://github.com/gui11aume/zerone/issues`.

Installing Rzerone

To install the Rzerone R package, simply run this command from the `zerone` directory.

```
R CMD INSTALL Rzerone
```

Note that you need to have R installed on your computer. If this is not the case, run the command `sudo apt-get install r-base` on Ubuntu.

2 Zerone basics

To run Zerone, you have to specify the files that contain the mapped reads of the ChIP-seq experiment you want to discretize. These can be in BED, SAM/BAM and GEM (.map) formats. You can include as many negative control files and as many experimental replicates as you need. Just enter controls after the `-0` or `--mock` option, and targets after the `-1` or `--chip` option.

For example, you can type in the following commands from the `zerone` directory.

```
./zerone --mock file1.bam,file2.bam --chip file3.bam,file4.bam
```

Where file1.bam and file2.bam are negative controls done without antibody, and file2.bam and file3.bam are two experimental replicates.

This should produce an output like the following.

```
chrX    1       300     0   0   0   0
chrX    301     600     0   0   0   0
chrX    601     900     0   0   0   0
chrX    901     1200    0   0   0   0
chrX    1201    1500    0   0   0   0
chrX    1501    1800    0   0   0   0
chrX    1801    2100    0   0   0   0
chrX    2101    2400    0   0   0   0
```

Note that path expansion will not work with comma separated file names, so if ou want to use path names starting with $\sim$, you can simply specify the mock and chip options as many times as needed, as shown below.

```
./zerone -l -0 file1.map -1 file2.map -1 file3.map
```

With the `-l` or `--list-output` option, Zerone produces an alternative, BED-like output.

```
chrX    1       600
chrX    601     1200
chrX    1201    1800
chrX    1801    2400
chrX    2401    3000
chrX    3001    3600
chrX    3601    4200
chrX    4201    4800
```

Enter the option `-h` or `--help` for usage instructions and `--version` to print the version number.

3 Troubleshooting

In case Zerone crashes, start by recompiling it in debug mode. To do so, run the following commands from the `zerone` directory.

```
make clean
make debug
```

Then repeat the actions that triggered the crash and contact guillaume.filion@gmail.com attaching the debug information.