Normalisation-related programs

Hin-Tak Leung

January 9, 2007

1 Introduction

This document details the normalisation-related programs I wrote for precessing the Affymetrix 500k cel files for Chiamo++ calling. The latest version of the programs, including this document, are kept in my CVS repository.

The basic work flow is a 3-stage process:

1. Read the cel files (NSP and STY separately) and generates a reference intensity file.
2. Use the reference intensity file to perform quantile normalisation of each cel file.
3. concatenate the result of the NSP run, the STY run, and some annotation information together. (and gzip'ing on the way).

The typical timing on a typical machine (2.2GHz opteron, 8GB RAM) in our computing grid for the 1958 cohort (1504 samples), from the last run, is about 11 hours in total, and it is scripted to take up one machine per cohort for that 11 hours:

	NSP	STY
Stage 1	2:14:12	2:08:07
Stage 2	3:07:07	2:59:51
Stage 3	38:10	

Stage 2 is very memory-intensive, and it just happens 8GB RAM[1] can cope with 2000 samples exactly. So for two cohorts with more than 2000 samples, there is an intermediate stage where one splits cohorts, or join reference intensities:

1 Read the cel files (NSP and STY separately) and generates a reference intensity file.

***** Doing a weighted sum of the reference intensities of split cohorts if the cohort size exceeds machine limitation.

2 Use the reference intensity file to perform quantile normalisation of each cel file.

3 concatenate the result of the NSP run, the STY run, and some annotation information together. (and gzip'ing on the way).

The weighted sum is almost instantaneous. For another typical run (IBD, 2005 samples, split into Stage 2a and Stage 2b of 814 +1191 samples – by plate numbers, roughly into two halves):

[1] 8GB on 64-bit architecture. On 32-bit architecture, any single process on linux cannot addresss more than 3GB RAM, even if there is more available.

	NSP	STY
Stage 1	2:59:36	2:53:29
Stage 2a	1:50:03	1:44:44
Stage 2b	2:54:46	2:44:35
Stage 3	53:34	

Note that Stage 2a and 2b can run on different machines, so the total running time is still about 13 hours, rather than 17 hours. The 13 hours depends on manually synchronzing between the stages, so one don't want to split more often than necessary.

The typical memory requirement of Stage 1 is 130MB, I believe, and is not sensitive to cohort size. The memory requirement of Stage 3 is also quite modest and similiar, as the perl script was written with memory restriction in mind, and as a result, it is quite complex with an unusual style.

2 Preparation

The source code should build on most linux box by typing `make` . Part of it is completely newly written, some part is modification/enhancement of Affymetrix's file toolkit, and another part is a trimmed-down boost library for reading gz'ed or bz'ed files.

I usually arrange the directory layout as follows, where each directory contains symbolic links to the real locations of the cel files — there is no need to decompress cel files, as our software transparently decompress gz'ed or bz2'ed files:

```
./58C/NSP
./58C/STY
./IBD2/NSP
./IBD2/STY
./IBD/NSP
./IBD/STY
```

```
$ ls -l 58C/NSP/WTCCC01-9414H9_NSP.CEL.gz
lrwxrwxrwx 1 50139 fuse 81 Jan  6 19:48 58C/NSP/WTCCC01-9414H9_NSP.CEL.gz
   -> ../../../WTCCC-data-20060720/9414/CELs/WTCCC01-9414H9_NSP.CEL.gz
```

2.1 Auxiliary files needed

Besides the source code bundles and the input cel files, one also need two other sets of files:

- The CDF files from Affymetrix (not included, as they are too large and can be obtained elsewhere)
- Some "digest" annotation files which are derived from Affymetrix's; some are included in the `axil` sub-directory of the source bundle.

3 A typical job script

This is a typical grid job script, with some irrelevant details removed, suitable when the whole cohort is within memory limitation. I'll examine each of the lines in details in the later sections.

```
#!/bin/sh
#$ -S /bin/sh

# Stage 1
cd $1/NSP && cel_qnorm_pass1 -o nspintfile *CEL.gz
cd $1/STY && cel_qnorm_pass1 -o styintfile *CEL.gz

# Stage 2
cd $1/NSP && gtype_cel_to_pq -refintensity nspintfile -cdf Mapping250K_Nsp.cdf \
  -split Mapping250K_Nsp_annot.split \
  -subset Mapping250K_Nsp_annotAB.padded -log-average *_NSP.CEL*
cd $1/STY && gtype_cel_to_pq -refintensity styintfile -cdf Mapping250K_Sty.cdf \
  -split Mapping250K_Sty_annot.split \
  -subset Mapping250K_Sty_annotAB.padded -log-average *_STY.CEL*

# Stage 3
nspsty-stitch.pl -o $1 -i annotAB.padded+sorted $1/NSP $1/STY
```

4 Stage 1

All `cel_qnorm_pass1` does is to read every cel file specified on the command line, and output a reference intensity file specified by the `-o` option:

```
# Stage 1
cd $1/NSP && cel_qnorm_pass1 -o nspintfile *CEL.gz
cd $1/STY && cel_qnorm_pass1 -o styintfile *CEL.gz
```

The * shell expansion is subjected to limitation of shell expansion (hence the symbolic-link layout). On most linux boxes, it is 65536 which is sufficient for 2000 samples of file names 30 characters long.

5 Weighted sum

As explained earlier, due to memory limitations, sometimes one need to split cohorts. So for example, for the IBD situation above, one would calculate a weighted sum of two sub-cohorts, and replaces the two sub-cohort reference intensities by their weighted sum as follows:

```
cel_p1merge -o nspoutfile -i nspdesfile
cel_p1merge -o styoutfile -i stydesfile
cp nspoutfile IBD/NSP/intfile
cp nspoutfile IBD2/NSP/intfile
cp styoutfile IBD/NSP/intfile
cp styoutfile IBD2/NSP/intfile
```

The utility `cel_p1merge` takes an input description file `nspdesfile` as follows (`filename` and `weight` on each line):

```
IBD/NSP/intfile     814
IBD2/NSP/intfile   1191
```

and similiarly for `stydesfile` .

6 Stage 2

Stage two is a performed by a substantially enhanced version of a utility from Affymetrix of the same name. Besides the ability to read gz'ed/bz2'ed cel files, here is an excerpt from the updated help message — the normalization only uses about half of the enhancements (the others were added for different purposes):

```
./gtype_cel_to_pq -h
  -single <file>        output files will be written to one single file
                            (must be used together with -log-average and -subset)
  -split <file>         files will be written splitted as directed
                            (must be used together with -log-average and -subset)
  -subset  <file>       text file with list of probeset IDs to extract
  -with-stdv            with stdv values next to the intensity values
  -log-average          output log average values instead of all the intensity
                             of all the probe quartets
  -refintensity <file>  reference intensity file generated by cel_qnorm_pass1
```

The `subset` option is different from Affy's in that the output is ordered as in the subset list (rather than in Affy's internal order as appeared in the cdf files).

```
# Stage 2
cd $1/NSP && gtype_cel_to_pq -refintensity nspintfile -cdf Mapping250K_Nsp.cdf \
  -split Mapping250K_Nsp_annot.split \
  -subset Mapping250K_Nsp_annotAB.padded -log-average *_NSP.CEL*
cd $1/STY && gtype_cel_to_pq -refintensity styintfile -cdf Mapping250K_Sty.cdf \
  -split Mapping250K_Sty_annot.split \
  -subset Mapping250K_Sty_annotAB.padded -log-average *_STY.CEL*
```

The output from Stage 2 are two sets of files, `NSP_chromosome_01.txt`, `NSP_chromosome_02.txt`, etc and `STY_chromosome_01.txt`, `STY_chromosome_02.txt`, etc. The file name and splitting is controlled by the `split` option. Only the information about listed order is used from the `subset` option.

At the end of Stage 2, the signal information is one file per chromosome listed in position order.

7 Stage 3

Stage 3 combines the result from the NSP run and the STY run, across split sub-cohorts, and adding some annotation information back for the final signal files, simultaneously gzipping the result.

```
# Stage 3
nspsty-stitch.pl -o $1 -i annotAB.padded+sorted $1/NSP $1/STY
```

The utility can take multiple NSP/STY directory pairs like this:

```
./nspsty-stitch.pl -h
Usage:
    nspsty-stitch.pl -o stem -i annotAB.padded+sorted NSPdir1 STYdir1 \
                          NSPdir2 STYdir2 NSPdir3 STYdir3 ...
```

```
Options:
    -o <stem>       output files are written as stem_01.txt.gz, stem_02.txt.gz, etc
    -i <annot>      annotation file extract, listing the merged order,
                        plus extra info of the snps.
    -m <manifest>   use manifest to map the plate/well to sample id.
                        (not recommended for duplicate sample id's)
    -h              displaying this help message
```