Agilp Manual Vignette

Benny Chain

Introduction

This package was developed to provide a pipeline for the low-level analysis of gene expression microarray data, primarily focused on the Agilent platform, but which also provides utilities which may be useful for other platforms. Agilp v2.0 is made up of five processing functions, which are described briefly below and in more detail in the individual help files for each function. The package is designed for people with little previous experience of R, or of microarray analysis. The package was developed for use in the Innate2Adaptive Immunity Laboratory, Division of Infection and Immunity, UCL, London run by Benny Chain and Maddy Noursadeghi. A number of references describing the microarray work from the laboratory are given at the end of this document.

A note on default paths in agilp functions

The default folders for all the functions in agilp use folders input, input1 and output within the agilp folder. The user files to be processed (e.g. array scanner files or the output of agilp functions themselves) should be placed within the input or input1 folders as instructed in the help files. Note that after installing the package these folders will initially contain a dummy file called toberemoved.txt in order to avoid the installation process removing empty folders.These dummy files should be deleted before use. To locate the path of the agilp folder on your computer type

```
> file.path(system.file(package="agilp"))

[1] "/private/tmp/RtmpHg7mko/Rinstf361ff04e83/agilp"
```

Alternatively, the user can create their own input and output directories, and alter the path description in the function call as appropriate.

Workflow

A typical workflow for using the package is described below. The examples are designed to be implemented sequentially.

- **AAProcess. Extract and save the red and green median expression data from Agilent scanner arrays, averaging replicate probes**. The raw (unprocessed) expression data is first extracted from the scanner files which are in tab delimited .txt format. This step is platform specific, and uses the column *gMedianSignal* or *rMedianSignal* to identify the data to use. The expression values for replicate probes are averaged. The function can use single or two colour outputs, and treats each colour independently. Output is one spreadsheet per microarray per colour, saved as tab delimited text files to a directory of choice. Each file contains all probe names (first column) and raw (untransformed, unnormalized) expression levels (second column).

  Examples. This example extracts expression data from two sample array scanner files which are found in the folder "scanner" within the agilp package extdata folder and places them in the folder output within the agilp package folder.

  ```
  > library(agilp)
  > inputdir<-file.path(system.file(package="agilp"),"extdata","scanner","", fsep = .Platform$file.sep)
  > outputdir<-file.path(system.file(package="agilp"),"output", "", fsep = .Platform$file.sep)
  > AAProcess(input = inputdir, output = outputdir, s = 9)
  ```

  To remove these files again and empty the output directory use :
  unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

  The number of lines header to be skipped from scanner array files can be changed from the default value of 9 by setting variable s. AAProcess(input = inputdir, output = outputdir, s = 12)

- **filenamex. Copy extracted raw file names to template**. Typically, the array file names are stored in a database We use a simple spreadsheet for this (e.g. Excel or other) which contains details such as the date the experiments were run, the types of samples, the experimenter , and any other important experimental detail (see *sample_template.txt* in the agilp extdata folder for an example). However, a major source of documentation error occurs while entering the array files names into the template file (array file names are often very long arbitrary alphanumeric sequences). For this reason we use *filenamex* which outputs a text file containing the names of all the files in a directory; this list can be easily copied and pasted directly into a template file.

  Examples.This example makes a list of the names of all the files in the folder agilp/extdata/raw and saves it in a file called names.txt (tab delimited) in the folder agilp/output.

  ```
  > library(agilp)
  > inputdir<-file.path(system.file(package="agilp"),"extdata","raw","", fsep = .Platform$file.sep)
  > outputdir<-file.path(system.file(package="agilp"),"output", "", fsep = .Platform$file.sep)
  > filenamex(input=inputdir,output=outputdir)
  ```

  To remove these files again and empty the output directory use :
  unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

- **Baseline. Generate a normalisation baseline file.**
  The *AALoess* normalisation function (see below) requires a 'baseline' file which contains the mean value for each probe from a number of different arrays against which to normalise each new array. *Baseline* generates such a baseline from a set of arrays. The set can be either all arrays in a chosen directory, or selected from a template file (see help file for more details). By default, all array data is base 2 log transformed before calculating the mean. If the files to be analysed contain different numbers of identifiers, the output will contain only the data for the IDs common to the whole set.

  Examples. Takes all the files of unprocessed data in folder agilp/extdata/raw) , calculates the mean expression value for each probe, and saves as agilp/output/testbase.txt.

  ```
  > library(agilp)
  > inputdir<-file.path(system.file(package="agilp"),"extdata","raw","", fsep = .Platform$file.sep)
  > outputbase<-file.path(system.file(package="agilp"),"output", "testbase.txt", fsep = .Platform$file.sep)
  > template<-file.path(system.file(package="agilp"),"extdata","sample_template.txt", fsep = .Platform$file.
  > Baseline(NORM="LOG",allfiles="TRUE",r=2,A=2,B=3,input=inputdir, baseout=outputbase, t = template)
  ```

  Alternatively the following example uses only those data files defined in column 2, rows 2-5 of the template file. In additon, this example does not log transform the data.

  ```
  > Baseline(NORM="FALSE",allfiles="FALSE",r=2,A=2,B=5,input=inputdir, baseout=outputbase, t = template)
  ```

  To remove these files again and empty the output directory use :
  unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

- **Loader. Select set of processed array files from directory according to criteria from a template file**. It is often necessary to select a series of expression array data files (for example generated by *AAProcess*) from a directory containing many such files (our repository has around 1000 array data sets so far) according to some criteria in the template file. For example, one might want to analyze all samples from particular dates, or tissue type etc. *Loader* inputs a template file in which each row contains the details for a separate array, and which has previously been sorted according to some predefined criteria (see *sample_template.txt* in the agilp/extdata for an example). It uses the array file names for a set of rows of the template (first and last row chosen by user, but must be contiguous), and either (if f ="TRUE") transfers the chosen files to a different directory, or outputs a single file (by default saved as alldata.txt) in which each column contains one array set of data.

  Examples. This will copy the files with names given in column 2 , rows 2:3 of the sample template file from the input folder to output folder.

  ```
  > inputdir<-file.path(system.file(package="agilp"),"extdata","raw","", fsep = .Platform$file.sep)
  > outputdir<-file.path(system.file(package="agilp"),"output", "", fsep = .Platform$file.sep)
  > template<-file.path(system.file(package="agilp"),"extdata","sample_template.txt", fsep = .Platform$file.
  > Loader(input=inputdir,output=outputdir,t=template,f="TRUE",r=2,A=2,B=5)
  ```

  Alternatively this will output a single file all_data.txt with the same data in a merged file. The file is also output back to R as object dataout

  ```
  > Loader(input=inputdir,output=outputdir,t=template,f="FALSE",r=2,A=2,B=5)
  > dim(dataout)
  ```

```
[1] 999   4
```

To remove these files again and empty the output directory use :
unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

For further detail on function variables see individual help files.

- **AALoess. Normalises a set of gene expression data files using LOESS against a reference data set.** An important aspect of analysing sets of microarray expression data is to be able to compare different experiments carried out at different times. This requires being able to normalize data sets as far as possible correcting for global differences between experiments. *AALoess* function normalises data sets which share a common set of identifiers (typically probe names if all the experiments are carried out on a single platform) by comparison to a baseline file which typically contains an average of many datasets (for example as produced by *Baseline*). If the files to be analysed contain different numbers of identifiers, the output will contain only the data for the IDs common to the whole set. The underlying assumption is that the overall expression distribution over all the probes is the same in all samples, and that most probes are unchanged when comparing any two samples. The normalisation rationale is discussed in more detail in Chain et al. (see Bibliography) . The function also outputs a frequency histogram of the distances between each sample an the baseline which has proved useful in identifying technical outliers (e.g. if there is a problem in hybridization, for example).

  Examples. This example takes the files in agilp/extdata/raw, LOess normalises them and saves them in agilp/output folder.

```
> inputdir<-file.path(system.file(package="agilp"),"extdata","raw","", fsep = .Platform$file.sep)
> outputdir<-file.path(system.file(package="agilp"),"output", "", fsep = .Platform$file.sep)
> baselinedir<-file.path(system.file(package="agilp"),"extdata","testbase.txt", fsep = .Platform$file.sep)
> AALoess(input=inputdir, output=outputdir, baseline = baselinedir, LOG="TRUE")
```

  After running this function review the SSE plot and file to identify outliers - note these in template file.

  Alternatively the following example does not log transform the data.

```
> AALoess(input=inputdir, output=outputdir, baseline = baselinedir, LOG="FALSE")
```

  To remove these files again and empty the output directory use :
  unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

- **IDswop. Maps expression data across different bioinformatic identifiers.** *IDswop* replaces one set of identifiers associated with an expression data set (for example platform specific probe IDs) by another set (for example ENSEMBL transcript IDs). This allows expression data sets from different platforms to be combined for analysis . Any appropriate mapping spreadsheet file which contains the source identifiers in one column and the equivalent target IDs in another column can be used. We have typically used files produced by biomaRt, but suitable annotation files can also be made using the **select** function, in the AnnotationDbi package. If multiple source IDs(e.g. probe IDs) map to the same target ID the function returns the mean or the maximum value within the set.

  Examples. This example maps expression data from Agilent ProbeIDs (found in column one of input files) to corresponding EnsemblID. Each output file has the name of the original file with the suffix "_EnsemblID". If multiple probes map to the same ID and vary by more than 20% they are discarded.

```
> inputdir<-file.path(system.file(package="agilp"),"extdata","raw","", fsep = .Platform$file.sep)
> outputdir<-file.path(system.file(package="agilp"),"output", "", fsep = .Platform$file.sep)
> annotation<-file.path(system.file(package="agilp"),"extdata","annotations_sample.txt", fsep = .Platform$
> IDswop(input=inputdir,output=outputdir,annotation=annotation,source_ID="ProbeID",target_ID="EnsemblID",
```

  Alternatively the following keeps all output IDs, even when multiple probes map to a single annotation.

```
> IDswop(input=inputdir,output=outputdir,annotation=annotation,source_ID="ProbeID",target_ID="EnsemblID",
```

  More details of how to use this function are given in the *IDSwop* individual help file. *IDSwop* and *Equaliser* (see below) should be used BEFORE *Baseline* and *AALoess*.

  To remove these files again and empty the output directory use :
  unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

- **Equaliser. Trims a set of gene expression data files to include only the set of identifiers common to all files.** Different platforms will typically generate files with different numbers of distinct identifiers (for example compare the Agilent 44K and Agilent 60K whole genome expression arrays. Different numbers of identifiers will also often result

from use of *IDSwop*. *Equaliser* is a utility which takes a set of gene expression data files and trims each file so as to include only the set of IDs which are common to all files in the set. More details of how to use this function are given in the *IDSwop* individual help file. *Equaliser* and *IDSwop* (see above) should be used BEFORE *Baseline* and *AALoess*.

Examples Takes four files in folder agilp/dataset/raw, selects only those entries common to all four, and outputs the shortened files to the folder agilp/output.

```
> inputdir<-file.path(system.file(package="agilp"),"extdata","raw","", fsep = .Platform$file.sep)
> outputdir<-file.path(system.file(package="agilp"),"output", "", fsep = .Platform$file.sep)
> Equaliser(input = inputdir, output = outputdir)
```

To remove these files again and empty the output directory use :
unlink(paste(file.path(system.file(package="agilp"),"output",""),"*.*",sep=""), recursive=FALSE)

Examples

All the examples given above and in the individual help files use data files contained within the /extdata folder within the installed agilp folder. The output is sent to the agilp/output folder. To locate the path of the agilp folder on your computer type

```
> file.path(system.file(package="agilp"))

[1] "/private/tmp/RtmpHg7mko/Rinstf361ff04e83/agilp"
```

Bibliography

1. Tomlinson GS, Booth H, Petit SJ, Potton E, Towers GJ, Miller RF, Chain BM, Noursadeghi M. Adherent human alveolar macrophages exhibit a transient pro-inflammatory profile that confounds responses to innate immune stimulation. PLoS One. 2012;7(6):e40348. Epub 2012 Jun 29. PubMed PMID: 22768282; PubMed Central PMCID: PMC3386998.

2. Tomlinson GS, Cashmore TJ, Elkington PT, Yates J, Lehloenya RJ, Tsang J, Brown M, Miller RF, Dheda K, Katz DR, Chain BM, Noursadeghi M. Transcriptional profiling of innate and adaptive human immune responses to mycobacteria in the tuberculin skin test. Eur J Immunol. 2011 Nov;41(11):3253-60. doi: 10.1002/eji.201141841. Epub 2011 Aug 30. PubMed PMID: 21805471; PubMed Central PMCID: PMC3258543.

3. Le Bert N, Chain BM, Rook G, Noursadeghi M. DC priming by M. vaccae inhibits Th2 responses in contrast to specific TLR2 priming and is associated with selective activation of the CREB pathway. PLoS One. 2011 Apr 1;6(4):e18346. PubMed PMID: 21483768; PubMed Central PMCID: PMC3069967.

4. Chain B, Bowen H, Hammond J, Posch W, Rasaiyaah J, Tsang J, Noursadeghi M. Error, reproducibility and sensitivity: a pipeline for data processing of Agilent oligonucleotide expression arrays. BMC Bioinformatics. 2010 Jun 24;11:344. PubMed PMID: 20576120; PubMed Central PMCID: PMC2909218.

5. Tsang J, Chain BM, Miller RF, Webb BL, Barclay W, Towers GJ, Katz DR, Noursadeghi M. HIV-1 infection of macrophages is dependent on evasion of innate immune cellular activation. AIDS. 2009 Nov 13;23(17):2255-63. PubMed PMID: 19741482; PubMed Central PMCID: PMC2873676.

6. Rasaiyaah J, Noursadeghi M, Kellam P, Chain B. Transcriptional and functional defects of dendritic cells derived from the MUTZ-3 leukaemia line. Immunology. 2009 Jul;127(3):429-41. PubMed PMID: 19538250; PubMed Central PMCID: PMC2712111.