An Introduction to *ArrayExpressHTS*: RNA-Seq Pipeline for transcription profiling experiments

Andrew Tikhonov, Angela Goncalves

Modified: 15 March, 2011. Compiled: October 31, 2011

1 ArrayExpressHTS Package

ArrayExpressHTS is an R based pipeline for pre-processing, expression estimation and data quality assessment of high throughput sequencing transcriptional profiling (RNA-seq) datasets. The pipeline starts from raw sequence files and produces standard Bioconductor R objects containing transcript measurements for downstream analysis along with web reports for data quality assessment. It may be run locally on a user's own computer or remotely on a distributed R-cloud farm at the European Bioinformatics Institute. It can be used to analyse user's own datasets or public RNA-seq datasets from the ArrayExpress Archive.

2 General Overview

The pipeline starts from raw sequence files and produces standard Bioconductor R objects of class *ExpressionSet* containing expression levels over features for downstream analysis, along with HTML reports for data quality assessment. Further informative intermediate data, such as alignment and annotation files, are also available. The steps of the pipeline are:

- Prepare the data and experimental metadata
- Assess raw data quality and produce quality report
- Align sequencing data files to a reference
- Filter reads
- Assess alignment quality and produce quality report
- Estimate expression
- Generate a compared report

3 Running AEHTS on the EBI R-Cloud

3.1 Find a dataset to analyse or upload your own data

ArrayExpressHTS can be used on the EBI R-Cloud to analyse public and private datasets available through ArrayExpress. You can use the web interface[1] to search for datasets in ArrayExpress. The procedure of getting one's data available for analysis on the cloud is to submit it to the AE Archive, where the data will remain password protected. Simple `MAGE-TAB` templates for submission can be obtained online[2] and curators will assist in file preparation and validation.

[1]http://www.ebi.ac.uk/arrayexpress/

[2]http://www.ebi.ac.uk/fg/submissions_overview.html

3.2 Launch the R-Cloud Workbench, register and create a new project

The `EBI R-Cloud` is a new service at the EBI which allows R users to log in and run distributed computational jobs remotely on a powerful 64-bit linux cluster. The `R-Cloud` is available through a GUI client, the `R-Cloud Workbench`. Follow the instructions on the page[3] to download or launch the Workbench. When connecting for the first time you will be requested to register. You will need to provide a username, password and e-mail address to allow you to retrieve your long running projects the next time you log in. Once registered you can log in and create a new project. To find your way around the workbench visit the documentation online[4].

3.3 Run the whole pipeline with default options

Running *ArrayExpressHTS* within R with default options is straight forward with a call to function `ArrayExpressHTS`. You must provide an ArrayExpress accession number for the dataset you want to analyse. For example, the publicly available dataset `E-GEOD-16190` comes from a study by Chepelev et al. on detecting single nucleotide variations in expressed exons of the human genome. To re-analyse this data with default options run the following commands on t he Workbench console:

```
> library("ArrayExpressHTS")
> aehts <- ArrayExpressHTS("E-GEOD-16190")
```

When the pipeline finishes, `e` will contain an *ExpressionSet* object that can then be used for further analysis. Running the whole pipeline will take time depending on the size of the dataset and the amount of computing nodes allocated for parallel computation. Reports and intermediate files are available as soon as they are ready and can be visualised through the File Browsing tab. A directory will be automatically created on your working space for each of the samples in the dataset. Any file can be copied by dragging and dropping from the File Browser into, for example, your Desktop. Beware of the file size, which is generally the reason why downloading takes long. To run the pipeline with options other than the default ones check the 'Configuration options' section.

4 ArrayExpressHTS Local Configuration

4.1 Pre-requisites

In order to run the pipeline locally you will need do have at least:

- Unix based OS (linux, MacOS).
- R environment for statistical computing[5]
- Bioconductor[6] The following packges are required: *Rsamtools*, *IRanges*, *Biostrings*, *ShortRead*, *Hmisc*, *R2HTML*, *XML*, *Biobase*, *svMisc*, *sampling*, *xtable*, *DESeq*, *ArrayExpress*, *RColorBrewer*, *biomaRt*
- SAMtools installed[7]
- At least one aligner installed: BWA[8] and/or Bowtie[9] and/or TopHat[10]

[3]http://www.ebi.ac.uk/tools/rcloud
[4]http://www.ebi.ac.uk/Tools/rcloud/quick_start.html
[5]http://www.r-project.org/
[6]http://www.bioconductor.org/install/index.html
[7]http://samtools.sourceforge.net/
[8]http://bio-bwa.sourceforge.net/
[9]http://bowtie-bio.sourceforge.net/tutorial.shtml
[10]http://tophat.cbcb.umd.edu/

- Cufflinks[11] and/or MMSEQ[12]
- *ArrayExpressHTS* package installed.

Check out 'Running AEHTS on the EBI R-Cloud' section to try out the pipeline on publicly available datasets without having to install any of these.

4.2 Process public ArrayExpress dataset

It is possible to run *ArrayExpressHTS* on publicly available ArrayExpress data on your computer. It will take longer as the data will be processed sequentially as opposed to R-Cloud where it's processed in parallel. However this possibillity provides a great benefit and can of course be of use.

You need to provide an ArrayExpress accession number for the dataset you want to analyse. For example, the publicly available dataset `E-GEOD-16190` that was mentioned above by Chepelev et al. on detecting single nucleotide variations in expressed exons of the human genome. To run the pipeline, execute the following R command:

```
> library("ArrayExpressHTS")
> aehts <- ArrayExpressHTS("E-GEOD-16190", usercloud = FALSE)
```

The parameter 'usercloud' signals the pipeline that it is running in a non-parallel environment rather than R-Cloud. When the pipeline finishes, `e` will contain an *ExpressionSet* object that can then be used for further analysis.

4.3 Prepare your own data

Users can process their owne datasets, that are non public, experimental or other. Data analysis starts with obtaining the input raw read files and creation of corresponding experimental metadata. Create a directory with the name you want to give your project and a subdirectory named 'data' on that project directory to hold your raw read and metadata files.

```
> dir.create("testExperiment")
> dir.create("testExperiment/data")
```

If your data is paired the pipeline is expecting the mates to be separated in two files with the names ending with _1 and _2, e.g.: 1974_1.fastq and 1974_2.fastq. Place your `fastq` files into the 'data' directory.

The experimental metadata serves to create a set of options used to configure the analysis and includes:

- essential sample annotation, including the experimental factors and their values (e.g. sex of the sample, time in a time series experiment);
- essential information about the biological system from which samples were taken, for instance, the organism and organism part (if known)
- experimental protocol information, such as the retaining of strand information and the insert size in paired-end reads;
- experiment design information including the links between files and samples and their experimental factors;
- machine related information, such as the instrument used and the scale of the quality information.

[11] http://cufflinks.cbcb.umd.edu/tutorial.html
[12] http://www.bgx.org.uk/software/mmseq.html

This information can be easily conveyed using the MAGE-TAB format[13], in particular using the IDF (Investigation Description Format) and the SDRF (Sample and Data Relationship Format) files. You can download and adapt the examples online[14] Name these files somename.idf.txt and somename.sdrf.txt. The IDF and SDRF should be placed into the 'data' folder.

4.4 Preparing the References and Annotation data

References are the organism's genome or transcriptome your reads will be aligned to. For the alignment the references also need to be indexed. References can be stored anywhere on the filesystem as later it can be specified in ArrayExpressHTS where they are located. If you would like to use a custom reference not available in Ensembl do prepare custom references. To get a reference genome from Ensembl you can use the prepareReference command with the organism, version and type.

```
>     # create directory
>     #
>     # Please note, tempdir() is used for automamtic test
>     # execution. Select directory more appropriate and
>     # suitable for keeping reference data.
>     #
>     referencefolder = paste(tempdir(), "/reference", sep = "")
>     dir.create(referencefolder)
>     # download and prepare reference
>     prepareReference("Homo_sapiens", version = "GRCh37.61", type = "genome", aligner = "bowtie"
>     prepareReference("Homo_sapiens", version = "GRCh37.61", type = "transcriptome", aligner = "
>     prepareReference("Mus_musculus", version = "current", type = "genome", location = reference
>     prepareReference("Mus_musculus", version = "current", type = "transcriptome", location = re
>
```

Annotation is used throughout all steps of the pipeline and therefore needs to be prepared as well:

```
>     # download and prepare annotation
>     prepareAnnotation("Homo_sapiens", "current", location = referencefolder )
>     prepareAnnotation("Mus_musculus", "NCBIM37.61", location = referencefolder )
```

4.5 Running the pipeline

After preparing your own data and references you are ready to launch the pipeline. Start R, load the ArrayExpressHTS package and call the ArrayExpressHTSFastQ function. Specify the location of the reference data in 'refdir' parameter. Below is the example experiments that comes with the pipeline, which is a very shortened version of E-GEOD-16190 experiment [15].

```
>     # In ArrayExpressHTS/expdata there is testExperiment, which is
>     # a very short version of E-GEOD-16190 experiment, placed there
>     # for testing reasons.
>     #
>     # Experiment in ArrayExpress:
>     # http://www.ebi.ac.uk/arrayexpress/experiments/E-GEOD-16190
>     #
>     # the following piece of code will take ~1.5 hours to compute
>     # on local PC and ~10 minutes on R-Cloud
```

[13]http://tab2mage.sourceforge.net/docs/magetab_docs.html
[14]http://www.ebi.ac.uk/Tools/rwiki/Wiki.jsp?page=ArrayExpressHTS%20Data%20Preparation.
[15]http://www.ebi.ac.uk/arrayexpress/experiments/E-GEOD-16190

```
>     #
>
>     # if executed on a local PC, make sure tools are available
>     # to the pipeline. Check initDefaultEnvironment help page
>     # for instructions.
>     #
>
>     # create a temporary folder where experiment will be copied
>     # computing experiment in the package folder may cause issues
>     # with file permissions and therefore failures.
>     #
>     #
>     srcfolder <- system.file("expdata", "testExperiment", package="ArrayExpressHTS");
>     dstfolder <- tempdir();
>     file.copy(srcfolder, dstfolder, recursive = TRUE);
>     # run the pipeline
>     #
>     # set usercloud = FALSE if executing on local PC,
>     # therefore parallel computation will be disabled
>     #
>     aehts = ArrayExpressHTSFastQ(accession = "testExperiment", organism = "Homo_sappiens", dir
>     # load the expression set object
>     loadednames = load(paste(dstfolder, "/testExperiment/eset_notstd_rpkm.RData", sep=""));
>     loadednames;
>     get('library')(Biobase);
>     # print out the expression values
>     #
>     head(assayData(eset)$exprs);
>     # print out the experiment meta data
>     experimentData(eset);
>     pData(eset);
>
```

5 Configuration options

You can override the default options used by `ArrayExpressHTS` by setting the function's `options` parameter to a `list` specifying values for each option. The following options are available:

- *stranded*: set to TRUE if a strand specific protocol was used
- *insize*: an integer, which will be automatically determined if set to NULL
- *insizedev*: an integer, which will be automatically determined if set to NULL
- *reference*: The reference should be set to either 'genome' or 'transcriptome'.
- *aligner*: 'tophat', 'bowtie', 'bwa' or 'custom'
- *aligner_options*: string of options to be directly passed to the aligners according to their own manual pages
- *count_feature*: count over 'genes' or 'transcripts'
- *count_method*: 'cufflinks', 'mmseq' or 'count' *count* can only be used with the reference set to transcriptome, though it will estimate gene level counts if count_feature is set to transcript and the sequence names in the reference include both transcript and gene names (e.g. see

fasta files from Ensembl). It involves counting reads overlapping known transcripts. Reads are discarded if they overlap more than one isoform of the same gene or there is some ambiguity as from which gene they originated from. Count values are thus not very useful by themselves but can be used for comparison of expression between conditions.

Discarding multi-mapping reads leads to information loss and systematic underestimation of expression. The mmseq and cufflinks statistical methods can be used estimate gene and transcript level expression taking into account all reads. *mmseq* can only used with `SAM/BAM` files generated by the TopHat or Bowtie aligners. See also *standardise* for a discussion on the types of values returned by these methods.

- *standardise*: 'TRUE' or 'FALSE' The three supported count methods 'cufflinks', 'mmseq' and 'count' produce different types of values by default: for the first two the expression estimates are in FPKM (Fragments Per Kilobase of exon per Million fragments mapped), and for count the values produced are in number of aligned reads.

  The type of values returned by the pipeline can be controlled by setting the standardise parameter to 'TRUE' or 'FALSE', regardless of the counting method. They return respectively per feature (gene or transcript) counts/estimates and counts/estimates standardised by feature length and scaled to the number of aligned reads in the sample (FPKM).

- *normalisation*: 'node' or 'tmm' Normalisation is generally required to remove systematic effects that occur in the data. *normalisation* can be set to either *none* or *tmm*, where *tmm* uses the trimmed mean of M-values for normalisation as implemented in the edgeR[16][17].

  Note: when using 'cufflinks' or 'mmseq' with 'none' or 'tmm' the expression estimates do not correspond one-to-one to read counts. This because, unlike the count method which only uses uniquely mapping reads, both these methods try to estimate transcript abundance from all reads including multi-mapping ones (reads that map to more than one transcript or location).

6 Downstream analysis

6.1 Differential Expression

Differential expression for count data can be determined, downstream of the pipeline, with count based DE packages such as edgeR[18] and DEseq[19]. We do not recommend however using the output of mmseq and cufflinks with these methods. Users should instead provide their own test, of which t-tests or likelihood ratio tests would be suitable examples.

[16]http://www.bioconductor.org/packages/release/bioc/html/edgeR.html
[17]http://genomebiology.com/2010/11/3/R25)
[18]http://www.bioconductor.org/packages/release/bioc/html/edgeR.html
[19]http://www.bioconductor.org/packages/release/bioc/html/DESeq.html