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

Andrew Tikhonov, Angela Goncalves

Modified: 27 February, 2012. Compiled: April 26, 2022

1 ArrayExpressHTS Package

ArrayExpressHTS is an R based pipeline for pre-processing, expression estimation and data quality assessment of high throughput sequencing transcription profiling (RNA-seq) datasets. The pipeline starts from raw sequence files and produces standard Bioconductor R objects containing transcript expression measurements for downstream analysis along with web reports for data quality assessment. It can 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 accepts raw sequence files and produces Bioconductor R objects of class *ExpressionSet* containing expression levels over features and quality assessment reports in HTML format. Informative intermediate data, such as alignment and annotation files, are also available.

The steps the *ArrayExpressHTS* pipeline performs to process data are:

- Download the raw data and experimental metadata if necessary, prepare for processing
- 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

The steps a user needs to perform to run the pipeline:

- *ArrayExpressHTS* package uses a number of external tools to process the sequencing data. On the R Cloud the package is pre-configured and you don't need to setup and configure the tools. However if you're running the pipeline on you desktop, the pipeline needs to be configured.
- Install external tools and configure the *ArrayExpressHTS* package. On the R Cloud everything is installed and pre-configured, so you don't need to do anything for this step. However if you're running the pipeline on you desktop, the tools needs to be installed and the pipeline needs to be configured.

- Prepare organism reference and annotation data. R Cloud provides preprocessed references and annotations for most common organisms. References are taken from `Ensembl` and are updated regularly. If you are running the pipeline locally, or you don't find the organism in the prepared organisms list, you will need to prepare reference and annotation data.
- Launch the pipeline.
- Find and analyse reports and logs

2.1 What data can be analysed

- Public and private datasets available in *ArrayExpress*. You can use the web interface[1] to search for datasets in *ArrayExpress*. The procedure of getting one's data available for analysis from the *ArrayExpress* 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. This is available to both remote R Cloud and local configuration.
- Custom datasets uploaded to the R Cloud or custom datasets available locally on you desktop or a local host. Uploading of data files can be performed using the means of the R Cloud Workbench, the R Cloud GUI client[3].

2.2 Supported Configurations

ArrayExpressHTS can run on:

- R Cloud, where, first – pretty much everything related to configuration of tools and preparation of reference and annotation data is taken care of. Second – the EBI computation cluster, which allows massive parallelization. *ArrayExpressHTS* supports parallelization and, by using it, it is capable of producing results tens and hundreds of times faster than a non-parallel configuration.
- Local configuration or a local host. The user needs to setup external tools and configure the package to use them, prepare reference data and launch the pipeline the same way as in the R Cloud.

3 Running AEHTS on the R Cloud

The pipeline uses external tools and prepared (indexed) references and annotations. Please note that when the *ArrayExpressHTS* is used on the R Cloud, tool configuration and preparation of references is taken care of behind the scenes.

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

The R Cloud is a service at the EBI which allows R users to log in and run distributed computational tasks remotely on a EBI 64-bit linux cluster. The R Cloud is available through the R Cloud Workbench or, alternatively, `R Cloud API` for programmatic access. Follow the instructions on the page[4] to download or launch the R Cloud Workbench.

The R Cloud requires registration. Open the registration panel in the R Cloud Workbench and provide username, password and email address to register. After registration is complete, you can log in and create a new project. Please note that R Cloud Workbench can disconnect from and connect to a project without interrupting the running task. You can also shutdown the running server and start the project on a new fresh server if the need arises.

To find your way around the workbench visit the documentation online[5].

[1] http://www.ebi.ac.uk/arrayexpress/
[2] http://www.ebi.ac.uk/fg/submissions_overview.html
[3] http://www.ebi.ac.uk/tools/rcloud
[4] http://www.ebi.ac.uk/tools/rcloud
[5] http://www.ebi.ac.uk/Tools/rcloud/quick_start.html

3.2 References and annotations

References are the organism's genome or transcriptome the reads will be aligned to. For the alignment to function properly the references need to be indexed (prepared). References and indexes need to be stored on the files system along with the experiment data.

EBI provides up-to-date `Ensembl` references with prepared indexes for most commonly used organisms. The pipeline automatically takes EBI references, unless it is explicitly specified to use a custom reference.

The pipeline can be specified to use a custom reference by setting the parameter `refdir` in functions `ArrayExpressHTS` and `ArrayExpressHTSFastQ` to the folder where custom reference is located.

If you don't find required reference in the EBI reference storage, and you know that it's available in `Ensembl`, you can use commands `prepareReference` and `prepareAnnotation` to automatically download and prepare references and annotations from `Ensembl`. Please note that annotation is used throughout all steps of the pipeline and therefore needs to be prepared as well.

To get a reference genome from `Ensembl`, use `prepareReference`. The usage is as follows:

```
> # create directory
> #
> # Please note, tempdir() is used for automatic 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", location = referencefolder )
> prepareReference("Homo_sapiens", version = "GRCh37.61",
+          type = "transcriptome", aligner = "bowtie", location = referencefolder )
> prepareReference("Mus_musculus", version = "current",
+          type = "genome", location = referencefolder )
> prepareReference("Mus_musculus", version = "current",
+          type = "transcriptome", location = referencefolder )
>
```

For the annotation:

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

3.3 Custom references and annotations

There are currently no automatic means to prepare custom, non `Ensembl` references.

3.4 Process public ArrayExpress dataset

Running *ArrayExpressHTS* for *ArrayExpress* experiments is straight forward. Having started a new project, load the package using `library(ArrayExpressHTS)` and start the pipeline by `ArrayExpressHTS` command with an *ArrayExpress* accession number as a parameter. 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 start the pipeline, run the following commands in the Workbench console:

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

A few notes:

- Running the entire pipeline sequence will take time. Normally, from half an hour to several hours. The time depends on the size of the dataset, sizes of individual data files and the amount of parallel nodes in the computational cluster.
- Experiment directory, e.g. `E-GEOD-16190`, will be automatically created in the working directory. This directory will contain all computation results as well as intermediate files and quality reports.
- Quality reports are available as soon as they are produced and can be viewed using the Workbench File Browser. If needed the report files can be drag'n'dropped from the Workbench File Browser to your local computer.
- Cluster log files are available in `cluster-log` folder. Logs are being collected throughout the processing, however the output does not always appear immediately and can sometimes be slightly delayed. If needed the log files can be drag'n'dropped from the Workbench File Browser to your desktop computer. Please mind the file sizes.

Results of the pipeline:

- When the pipeline finishes, `aehts` will contain an *ExpressionSet* object that encapsulates expression levels and experiment metadata. The object is also saved in the `E-GEOD-16190` folder as `eset_notstd_rpkm.RData` and can be loaded as a usual R object via `load` command.
- Compared reports are available in the `compare_report` folder in the main experiment folder.
- Reports related to single data files are located in the corresponding folders, e.g. `SRR060747/report` and `SRR060747/tophat_out/report`.

To run the pipeline with options other than the default ones check the 'Configuration options' section.

3.5 Prepare a custom dataset

The *ArrayExpressHTS*package contains a packaged test experiment, which includes test `fastq` files and metadata that can be used as a sample custom dataset. Refer to the `ArrayExpressHTSFastQ` help page for commands and instructions on how to extract the data from the package and create a custom experiment.

Create an experiment directory with an arbitrary name and a subdirectory named `data` to contain your raw read and metadata files.

The structure should look like this:

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

Upload `fastq` data files into the `data` folder. If your data is paired, the pipeline will expect the mates to be separated in two files with same names ending with _1 and _2, e.g.: 1974_1.fastq and 1974_2.fastq.

Create experiment metadata SDRF (Sample and Data Relationship Format) and IDF (Investigation Description Format) files according to `MAGE-TAB`[6] - a TAB delimited format. The metadata serves to create a set of options to configure the analysis and build the result *ExpressionSet* object. The metadata includes:

- sample annotation, including the experimental factors and their values (e.g. sex of the sample, time in a time series experiment);
- biological system from which samples were taken, 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;

[6] http://tab2mage.sourceforge.net/docs/magetab_docs.html

- 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.

For example you have 3 "Paired End" sequencing runs of "Homo sapiens", each run consists of a pair of mate files ending with _1 and _2. The files are:

- *sampleHomo001_1.fastq*
- *sampleHomo001_2.fastq*
- *sampleHomo002_1.fastq*
- *sampleHomo002_2.fastq*
- *sampleHomo003_1.fastq*
- *sampleHomo003_2.fastq*

And 2 "Single Read" runs of "Mus musculus". The files are:

- *sampleMus001.fastq*
- *sampleMus002.fastq*

In total 8 `fastq` files in the `data` folder.
Here are a few commands to construct an SDRF for this experiment:

```
> # "Sample"            "Organism"      "Base.Length"
> #  sampleHomo001      Homo sapiens      260
> #  sampleHomo002      Homo sapiens      260
> #  sampleHomo003      Homo sapiens      260
> #  sampleMus001       Mus musculus      0
> #  sampleMus002       Mus musculus      0
>
>
> dir.create("testExperiment")
> dir.create("testExperiment/data")
> mysdrf = data.frame(
+         "Sample" = c(
+                 "sampleHomo001",
+                 "sampleHomo002",
+                 "sampleHomo003",
+                 "sampleMus001",
+                 "sampleMus002"),
+         "Organism" = c(
+                 "Homo sapiens",
+                 "Homo sapiens",
+                 "Homo sapiens",
+                 "Mus musculus",
+                 "Mus musculus"),
+         "Base.Length" = c(
+                 180, 180, 180,
+                 260, 260))
> write.table(mysdrf,
+         file = "testExperiment/data/experiment.sdrf.txt",
+         sep="\t", quote = FALSE,
+         row.names = FALSE);
```

Name metadata files as `<name of experiment>.idf.txt` and `<name of experiment>.sdrf.txt` and place them into the `data` folder.

3.6 Run the pipeline on a custom dataset

To launch the pipeline on a custom dataset use `ArrayExpressHTSFastQ` function. The function accepts `accession` parameter, which is the experiment folder name containing `data` folder. If you are using custom reference, use `refdir` parameter to specify the location of the reference.

Below is an example experiment packaged in the *ArrayExpressHTS*. The experiment is a very shortened version of "E-GEOD-16190" experiment[7].

```
> # In ArrayExpressHTS/expdata there is testExperiment, which is
> # a very short version of E-GEOD-16190 experiment, placed there
> # for testing.
> #
> # 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
> #
> # Create a temporary folder where experiment will be copied.
> # If experiment is computed in the package folder it may cause
> # issues with file permissions and unwanted failures.
> #
> #
> srcfolder <- system.file("expdata", "testExperiment",
+         package="ArrayExpressHTS");
> dstfolder <- tempdir();
> file.copy(srcfolder, dstfolder, recursive = TRUE);
> # run the pipeline
> #
> aehts = ArrayExpressHTSFastQ(accession = "testExperiment",
+         organism = "Homo_sapiens", dir = dstfolder);
> # 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);
> # figure out if there's valuable data
> all(exprs(eset) == 0)
> # locate it
> head(which(exprs(eset) != 0))
> # print it
> exprs(eset)[ head(which(exprs(eset) != 0)) ]
>
```

[7] http://www.ebi.ac.uk/arrayexpress/experiments/E-GEOD-16190

4 R Cloud configuration options

You can configure usage of the R Cloud in order to better control resources and execution flow. The options can be defined by setting the `rcloudoptions` parameter in `ArrayExpressHTS` and `ArrayExpressHTSFastQ`.

Example:

```
> library("ArrayExpressHTS")
> aehts <- ArrayExpressHTS("E-GEOD-16190",
+          rcloudoptions = list(
+                  "nnodes"   = "automatic",
+                  "pool"     = "16G",
+                  "nretries" = 10))
```

The options are:

- *nnodes*: 'automatic' or a numeric value. e.g. 10, controls the number of nodes in the computational cluster. If a numeric values is defined rather than "automatic", the package will allocate the specified amount, otherwise the amount will be calculated automatically.
- *pool*: '4G', '8G', '16G', '32G', '64G', defines the server pool from which the nodes will be allocated. Each pool is limited memory-wise, which is signified by the values.
- *nretries*: numeric value from 0 to 10, controls the number of retries if there is not enough resources to immediately create a cluster.

5 ArrayExpressHTS Local Configuration

5.1 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[8]
- Bioconductor[9] packages: *Rsamtools*, *IRanges*, *Biostrings*, *ShortRead*, *Hmisc*, *R2HTML*, *XML*, *Biobase*, *svMisc*, *sampling*, *xtable*, *DESeq*, *RColorBrewer*, *biomaRt*
- SAMtools installed[10]
- At least one aligner installed: BWA[11] and/or Bowtie[12] and/or TopHat[13]
- Cufflinks[14] and/or MMSEQ[15]
- *ArrayExpressHTS* package installed.

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

[8] http://www.r-project.org/
[9] http://www.bioconductor.org/install/index.html
[10] http://samtools.sourceforge.net/
[11] http://bio-bwa.sourceforge.net/
[12] http://bowtie-bio.sourceforge.net/tutorial.shtml
[13] http://tophat.cbcb.umd.edu/
[14] http://cufflinks.cbcb.umd.edu/tutorial.html
[15] http://www.bgx.org.uk/software/mmseq.html

5.2 Configure ArrayExpressHTS to use external tools

ArrayExpressHTS uses options e.g. `ArrayExpressHTS.cufflinks` to configure the paths to the external tools it uses.

You can set them manually before you start the pipeline, or, alternatively, you can have these options set in an .Rprofile file, which is loaded when you launch your local R. The values will be imported by the pipeline when the *ArrayExpressHTS* package is loaded.

```
> # setup tools
> setPipelineOptions("ArrayExpressHTS.bowtie"
+          = "/path/to/tools/bowtie-0.12.7")
> setPipelineOptions("ArrayExpressHTS.cufflinks"
+          = "/path/to/tools/cufflinks-1.1.0.Linux_x86_64")
> setPipelineOptions("ArrayExpressHTS.tophat"
+          = "/path/to/tools/tophat-1.3.2.Linux_x86_64")
> setPipelineOptions("ArrayExpressHTS.samtools"
+          = "/path/to/tools/samtools-0.1.18")
>
```

Please note the versions of the tools – these versions are supported. Normally bioinformatics tools tend to change their output format from time to time when a new version comes out. Usage of not supported versions is possible at your own risk.

Also please note that normally, only a few of the tools are used. For the default setup - tophat, bowtie, cufflinks and samtools. The other ones – bwa, mmseq, bam2hits are alternatives and can be left default if not used.

5.3 References and annotations for local processing

For local configuration you would also need to prepare references. Please refer to 'References and annotations' and 'Custom references and annotations' sections above for instructions.

5.4 Process public ArrayExpress dataset on your local computer

It is possible to process *ArrayExpress* datasets on your computer. Please note, the local processing will be sequential and therefore will take longer than on R Cloud where it's parallel.

The process of launching of the pipeline will be identical to the one described above in section 'Process public ArrayExpress dataset'. Provide an *ArrayExpress* accession number, e.g. `E-GEOD-16190`, publicly available dataset by Chepelev et al. mentioned above. To run the pipeline, execute the following R commands:

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

The parameter `usercloud` sets the pipeline to a non R Cloud mode. When the pipeline finishes, `aehts` will contain an *ExpressionSet* object that can then be used for further analysis.

5.5 Prepare custom datasets on your local computer

The procedure is identical to the one described in section 'Prepare a custom dataset'.

5.6 Prepare references and annotations

The pipeline requires reference and annotation data to be located locally where the processing is performed. Therefore, if you're going to process data locally on your desktop or a local host you need to prepare your own references. Please refer to sections 'References and annotations' and 'Custom references and annotations' for more information on how to prepare references.

5.7 Process custom datasets on your local computer

To launch the pipeline on a custom dataset, use `ArrayExpressHTSFastQ` function. The function accepts `accession` parameter, which is the experiment folder name containing 'data' folder. If you are using custom reference, use `refdir` parameter to specify the location. Since the configuration is local, set `usercloud` to FALSE.

```
> aehts = ArrayExpressHTSFastQ(accession = "testExperiment",
+         organism = "Homo_sapiens", dir = dstfolder, usercloud = FALSE);
```

6 Configuration options

You can override the default options used by `ArrayExpressHTS` and `ArrayExpressHTSFastQ` by setting the `options` parameter to a `list` specifying values for each option. Example:

```
> library("ArrayExpressHTS")
> aehts <- ArrayExpressHTS("E-GEOD-16190",
+         options = list(
+                 "insize" = 200,
+                 "count_method" = "mmseq",
+                 "aligner" = "bwa",
+                 "aligner_options" = "-t 16 -M 10"))
```

The options are:

- *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 is 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).

7 Downstream analysis

7.1 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