HAL
open science

De Novo Transcriptomic Approach to Study Thyroid Hormone Receptor Action in Non-mammalian Models

Nicolas Buisine, Gwenneg Kerdivel, Laurent Sachs

To cite this version:

Nicolas Buisine, Gwenneg Kerdivel, Laurent Sachs. De Novo Transcriptomic Approach to Study Thyroid Hormone Receptor Action in Non-mammalian Models. Thyroid Hormone Nuclear Receptor. Methods and Protocols, 1801, 2018, 10.1007/978-1-4939-7902-8_21 . hal-02864269

HAL Id: hal-02864269

https://hal.science/hal-02864269v1

Submitted on 7 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

de novo Transcriptomic approach to study thyroid hormone receptor action in non-mammalian models

Nicolas Buisine[1], Gwenneg Kerdivel [1], Laurent M. Sachs[1,2]

[1] *Function and Mechanism of Action of Thyroid Hormone Receptor group, UMR 7221 CNRS and Muséum National d'Histoire Naturelle, Sorbonne Universités, CP 32, 57 rue Cuvier, 75005 Paris, France.*

[2] Correspondence : sachs@mnhn.fr

Abstract

Thyroid hormones are pleiotropic hormones involved in chordates physiology. Understanding their functions and mechanisms is also instrumental to diagnose dys-regulations and get a predictive power that can applied to medecine, ecology etc. Today, high-throughput sequencing technologies offer the opportunity to address this issue not only in model organisms but also in non-model organisms. Here, we describe a method that make use of RNA-seq to address differential expression analysis in non-model organism.

Key words

Thyroid hormones, transcriptome assembly, transcriptome annotation, differential expression analysis

Running Head

Transcriptomics in non conventional models

1. Introduction

Thyroid hormones (THs) are phylogenetically conserved molecules that affect many aspects of development, growth and metabolism of chordates [1]. THs were originally known to promote metamorphosis, the maturation of juvenile animals and maintain the basal metabolic rate. In fact,

the roles of THs are more diverse, with an action during developmental and adult neuro-plasticity [2,3], animal dispersal [4], hibernation [5], sex differentiation [6,7], reproduction [8], lactation [5], stress and inflammatory responses [9]. Identifying the molecular and cellular mechanisms that govern the diversity of these effects is critical for understanding physiology because perturbations of TH signaling can have very diverse outcome affecting health, life quality, biodiversity and ecosystems. Unfortunately, these effects are rarely probed in non-model organisms, thus restricting our knowledge of the key genes involved. In addition, even in model organisms, the molecular dissection of physiological processes has been mostly driven by candidate gene approaches and is thus based on an *a priori*. Today, the availability of next-generation sequencing (NGS) offers the opportunity to access naively the complete spectrum of genes involved in a particular biological process, and focus on species and environment-specific regulations of gene expression [10]. However, scientific communities working on non-model organisms frequently suffer from limited budget and it is not always easy to choose the best approach given a specific biological question, a limited budget, and challenging sample material.

In this article, we describe a cost effective and reasonably fast method to measure gene expression by RNA-Seq, for species where no genome sequence is available. This is a two steps process: assemble *de novo* a set of coding sequences from paired-end RNA-Seq that will be used as a proxy for genome sequences, and measure gene expression by conventional (single-end) RNA-Seq. Although less resolutive, this nonetheless bypass the need for whole genome sequencing, assembly and annotation, which are notoriously difficult, time consuming, costly and often stay beyond reach of the small scientific communities centered around non-model organisms.

2. Materials

2.1 Computational requirements

1. A 64-bit computer or server running any flavor of Unix or the GNU Linux operating system, for routine work.

2. Access to a server with multiple cores and large amount (> 100GB) of RAM for assembly. The TRINITY assembler typically requires around 1GB of RAM per million of paired-end reads. Other assemblers may require more RAM. The amount of RAM is dependent on the total number of reads to assemble.

3. Access to a farm of computing cores, for annotation. For sequence alignment of assembled sequences against databases, such as NCBI's nr, the emphasis should be put on the number of computing cores rather than the total amount of RAM available, since sequence comparison can be easily parallelized on multiple CPUs and is not memory intense. Although not essential *per se*, GPU servers (with up to 4096 computing units) scales up computing time almost linearly with the number of computing cores, which makes them attractive solutions.

2.2 Data

1. Protein sequence data bases: NCBI's non redundant (nr) protein sequence database and any protein sequences database from a reference species closely related to the studied species.

2. Assembly: RNA should be sequenced with paired-end protocols, in FASTQ format (or FASTA format, together with QUAL files). At least 100-150 million raw reads are needed (>50 million high quality reads).

3. Measure of gene expression: conventional, single end RNA sequencing with biological replicates, with around 50-60 million raw reads per sample.

2.3 Softwares

1. Quality control and read pre-processing: reads quality can be assessed with the FASTQC toolkit (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/). The pre-processing of reads can be performed using any software designed to remove low-quality bases and trim residual adapter sequences (see Note 1).

2. Assembly: many de novo assembler may be used depending on the available computational

resources. In our lab, we use the Trinity assembler [11,12] but other assembler, such as SOAPdenovo [13], AbySS [14], Velvet [15] or Oases [16], may also be used.

3. Blast2 [17] (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast2/) or Blast+ [18] (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/).

4. Bowtie [19] or Bowtie2 [20] (http://bowtie-bio.sourceforge.net).

5. DESeq [21] or DESeq2 [22] (http://bioconductor.org/packages/release/bioc/html/DESeq.html).

6. Bedtools [23] (https://code.google.com/archive/p/bedtools/).

7. The R environment (http://www.r-project.org).

8. Assembly statistics: Assemblathon tools [24].

3. Methods

3.1 General workflow

This review section focuses on the bioinformatic part of the process. Although not described here, animal treatments, tissue/RNA extraction and high throughput sequencing should be produced with the maximum quality possible. Importantly, transcriptome assembly and measure of gene expression each require a specific protocol and specific type of NGS data.

The conceptual steps for the detection of differential expression after *de novo* transcriptome assembly are the following:

- Clean up the reads to remove low quality reads or trim the 5' and/or 3' end (see Note 1).
- Assemble the transcriptome (see Note 2).
- Clusterize contigs, so that multiple contigs are coalesced into single mRNA species.
- Annotate clustered contigs, by sequence comparison against databases of known sequences (see Note 3).
- Identify differentially expressed genes, after mapping of conventional RNA-Seq reads onto reference sequences.

3.2 A word of warning

Bioinformatic processing of large datasets typically produce hundreds (if not thousands) of files, which may be hard to store and keep track of. The success of the project thus relies on rigorous good laboratory practices that minimize human errors (see Notes 4 and 5). Furthermore, we should emphasize the power of the versatile toolkits readily available in the Unix/Linux environments and we strongly suggest to make heavy use of them (see Notes 6, 7, 8 and 9). In term of project management, three points are important: a well planned project with a realistic budget, producing large sets of high quality raw sequences, set regular checkpoints during the project and impose stringent quality controls (see Notes 10, 11 and 12).

3.3 Quality control and read pre-processing

1. Assess the quality of paired-end-reads using FASTQC toolkit. For each file, run the following command from a bash terminal:

```
fastqc <read file>
```

2. If necessary (average per base sequence quality lower than 28, over-representation of adaptator sequences), use any cleaning software to correct for any biases revealed by the quality control (see Notes 1 and 13). AlienTrimmer can be run with the following command:

```
AlienTrimmer -if <forward read file> -ir <reverse read file> -
c alien.txt
```

Example:

```
AlienTrimmer -if 500892_0035_AHVL_1.fastq  -ir
500892_0035_AHVL_2.fastq -c alien.txt
```

3. This creates three output files, containing trimmed forwards and reverse reads files, plus an additional file containing singletons, respectively:

`500892_0035_AHVL_1.fastq.at.fq`, `500892_0035_AHVL_2.fastq.at.fq` and `500892_0035_AHVL_1.fastq.at.sgl.fq`

The file `alien.txt` is a text file containing the sequences of adaptors and of all putative contaminating sequences (one per line). Other options may be considered, such as `-q` that allows to set a threshold for quality score and `-l` that define the minimum length of trimmed reads that will be kept.

4. Re-assess the quality of paired-end-reads using FASTQC toolkit and verify that low quality reads have been filtered out and reads trimmed appropriately. It is not uncommon to observe an inefficient biases correction. In this case, consider using other read cleaning software (see Note 1).

3.4 *de novo* transcriptome assembly and evaluation of the assembly procedure

1. Merge the FASTQ files corresponding to the forward and reverse reads file, separately. Merge datasets from different tissues/conditions to increase the total number of reads if needed (see Note 14). On bash-like environment, this is easily carried out with the `cat` command (that outputs the content of the files pointed to) and redirection (with the > sign) into a new file:

```
cat <pair1_1.fastq> <pair1_2.fastq> …  > pair1_all.fastq
cat <pair2_1.fastq> <pair2_2.fastq> …  > pair2_all.fastq
```

Example:

```
cat 500892_0035_AHVL_1.trim.fastq
500882_0033_AHVL_1.trim.fastq > 5008-92-82_00-35-
33_AHVL_1.trim.fastq
cat 500892_0035_AHVL_2.trim.fastq
500882_0033_AHVL_2.trim.fastq > 5008-92-82_00-35-
33_AHVL_2.trim.fastq
```

2. If needed, copy files to the server running TRINITY with the bash command `scp`, iterating over a loop. The following command copies all the FASTQfiles in the current folder (`*.fastq`) and copies them in the `/datasets/` folder of the `my_server.my_institution` server, under the `me` account:

```
for CURRENT_FILE in *.fastq; do
scp $CURRENT_FILE me@my_server.my_institution:/datasets/ ;
done
```

3. *de novo* assembly can be performed with numerous tools, many of which are freely available (see Materials section). In this method, we will focus on the TRINITY assembler. According to TRINITY's manual, it is recommended to use a computer with around 1 GB of RAM per million of paired-end reads used for the assembly. Several options may be used to perform *de novo* assembly with TRINITY (see Note 15).

Calling TRINITY by the command line has the following synopsis:

```
Trinity.pl [options] --left <left FASTQ paired file> --right
<right FASTQ paired file>
```

Example:

```
Trinity.pl --seqType fq --max_memory 50G --left
5008-92-82_00-35-33_AHVL_1.trim.fastq --right
5008-92-82_00-35-33_AHVL_2.trim.fastq --CPU 16
--PasaFly   2> 20180707.trinity.error.log  >
20180707.trinity.log
```

The `--PasaFly` option specifies the use of the alternative PASA-like algorithm, which is more conservative and thus detects fewer isoforms. The number of CPU and the maximum amount of RAM to use are set by the `--CPU` and `--max_memory` options. These two options should match the maximum resources allocated on the server. As discussed above, they condition the total number of reads to use for assembly. Error messages (after `2>`) are stores in the `20180707.trinity.error.log` file and standard output (after `>`) in `20180707.trinity.log.` TRINITY output is located in the `trinity_out_dir` folder. The file `Trinity.fasta` contains the contig sequences, in FASTA format:

```
>TR1|c0_g1_i1 len=285 path=[263:0-284] [-1, 263, -2]
AGGTCCAATAGTGCGCTTTCTTGCGTATTTGTGCTTTTATGGTGGTGTGCCACTGCACGA
```

```
CCATTGGAGGTACCAATTTTCCTAACCCTTTATTCAGTCTGCTTTTATTTTTGGGTCCCT
CTGCACTATGGAGCGCCTTTCTGTTTAACTTTTGCAATCGCTTGAATGATACTGAGTTGC
TCTCAGGCTATACATGAGATCACAGGCCTGAGAATAGACCACAGTGCGCAATAGAGCTCC
GAAGCCTGTGCAGTCAGCTGATTGGCAGGGATCCCAAGTTTTGTG
>TR2|c0_g1_i1 len=317 path=[397:0-25 398:26-26 399:27-316] [-
1, 397, 398, 399, -2]
AATGCATGGGTGAAATGCATGTTGCGGGTGCTGAATGGTCTGGATGCCGCCTATTACCAG
ATTACTATGCCACTGGCTTATGATGCTCTTCTGGTTTATATATATATGCATCCCTTGGAT
ACTCTGCACTTTAGTGCTATATGTTCTACATATCTGTCCTATATACCTTGATCTCTTGTG
```

Notice that contig names provide a number of informations, including their length. It is good practice to rename of output file:

```
mv  trinity_out_dir/Trinity.fasta

./20180707.Trinity_jaccPasa.fas
```

4. Collect statistics describing the total number of contigs. To this end, the name of each contig contains its size. This is easily collected with a combination of `awk` and `sed` commands, under Unix/Linux environment:

```
awk '{print $2;}'20180707.Trinity_jaccPasa.fas | sed

's/len=//' > 20180707.Trinity_jaccPasa.fas.contig_size
```

5. The contig length cumulative density distribution is then plotted from `R` or any spreadsheet program (Figure 1A, B, C, D).

6. Transcript assembly generally generates a large number of short sized contigs, corresponding mostly to assembly artifacts and/or very partially assembled transcripts (typically, a pair of overlapping forward and reverse reads). It is good practice to discard all contigs shorter than some threshold, established from the statistics above (e.g. 200 bp). This can be carried out with a moderately advanced `awk` script:

```
BEGIN {RS = ">" ; ORS = "\n"; FS="\n"; THRESHOLD=MIN_SIZE;}

{sq="";

for (i = 2; i <= NF; i++){sq=sq "" $i;}

 if (length(sq)>=THRESHOLD){print ">"$1;for(j=0;j<=length(sq)/

60;j+=1){print substr(sq,1+(j*60),60);}}
```

Save the script into a text file (called `filter_fasta.awk`, for example) and call it from the command line:

```
awk -f  filter_fasta.awk -v MIN_SIZE=1000
20180707.Trinity_jaccPasa.fas >
20180707.Trinity_jaccPasa.fas.length_mt_2000.fas
```

The option `-v MIN_SIZE=1000` sets the threshold of the minimum contig length allowed (1000 bp in this case). Notice the name of the output file.

3.5 Clustering of contigs and annotation of *de novo* assembled transcriptomes

1. Clustering of contigs into a set of non redundant reference sequences. This may be carried out with CD-HIT-EST [5], which was initially designed to clusterize ESTs. Option `-T` specifies the number of CPUs to be used. If set to 0, all CPUs will be used. Option `-n` specifies word length (between 8 and 10 for similarity threshold between 90 and 100%), `-r 1` sets to compare both strands, and `-o` specifies the output file.

```
cd-hit-est -r <strand> -T <CPUs> -i <Fasta file of redundant
sequences>  -o <output file> -c <similarity threshold> -n
<word length>
```

Example:

```
cd-hit-est -r 1 -T 0 -i
20180707.Trinity_jaccPasa.fas.length_mt_2000.fas  -o
20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.fas -c
0.95 -n 8
```

The clustering details are stored in an additional `clstr` file (`20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.fas.clstr`).

2. Annotation of the references sequences. This corresponds to defining orthologous relationships with known protein sequences. This is typically performed with the non redundant (nr) protein

sequence database (NCBI) and/or, if available, to the RefSeq protein sequences of a closely related species. This can be carried out with the BLASTX (see Note 16), although this typically requires the use of custom made python or perl script to parse and filter out the results. Alternatively, these services are provided by CRB-BLAST [25], which runs BLAST, detects the Best-Reciprocal-Blast-Hits and sets orthologous relationships. CAUTION: this step can be computationally intensive, and the use of dedicated hardware with multiple cores can significantly accelerate the process (see above). CRB-BLAST is simple to use:

```
crb-blast --query <reference sequences> --target <known
proteins> --threads <CPUs> --output <Output file>
```

Example:

```
crb-blast --query
20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.fas --
target nr.fa --threads 256 --output
20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.annota
tion.tsv
```

The output is a tabulated file where the first two fields correspond to the query name (name of the unannotated sequence) and the target name (name of the putative ortholog). Other informations (% similarity, alignment length, blast e_value and bit_score...) are provided, and it may be worth exploring them to adjust the stringency further if needed. This can easily be carried out with a simple `awk` command or script, that would allow to set various thresholds based on alignment length, bit_score... In case known transcripts are available, they can be used to tweak the stringency and better control the specificity and sensitivity of the search.

A simple `python` script can update the name of references sequences. In principle, a hash table (*a.k.a.* dictionary) is build from the annotation.tsv file and is used to update individual sequence names. The following script (`rename.py`) is a straightforward example:

```
#! /usr/bin/env python
```

```
import sys
ANNOTATIONFN=sys.argv[1]
SEQUENCE_FILE_NAME=sys.argv[2]
# Build dictionary
annotation={}
inputfile=file(ANNOTATIONFN,'r')
for records in inputfile:
    old,new=records.strip().split()[:2]
    annotation[old]=new
inputfile.close()
# Parse sequence file
sequence_file=file(SEQUENCE_FILE_NAME,'r')
for cl in sequence_file:
 # Change sequence name if needed
    if cl[0]=='>' and cl.split()[0] in annotation.keys():
        print ">%s"%annotation[cl.split()[0]]
    else:
        print cl,
```

Annotated sequences are printed to the standard output. Invoke the `rename.py` script with the following command:

```
python ./rename.py <annotation file> <sequence file>  >
<output file>
```

where the annotation file is:

```
20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.annota
tion.tsv
```

and the sequence file:

```
20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.fas.
```

In order to be consistent, the output file name could be named:

`20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.annotation.fas`.

Notice the redirection symbol ('>') at the end of the command line, so that the output is properly redirected into the file.

3.6 Differential analysis of gene expression

Lets consider a typical RNA-Seq experiment with two datasets (control and treated with THs), composed of three biological replicated where RNA have been sequenced with the simple Illumina 50bp single-end protocol (`RFTK.00356.28`, `RFTL.00356.29` and `RFTM.00356.30` for control, and `XTD.024.72`, `XTD.025.73` and `XTD.026.74` after TH treatment). The point of this section is to detect the genes (more specifically clustered contigs, used as proxies for genes) which are differentially expressed between the two experimental conditions. The workflow is straightforward: for each RNA-Seq library, reads are mapped to the references sequences produced above and a count table is derived (i.e. the number of reads per contig). Differential analysis is run with the `R` package DESeq. Although not detailed here, quality controls (see above), clipping and adaptors removal also apply here as a pre-requisite.

1. Reads mapping. For each conditions, align RNA-Seq reads to the set of reference sequences using a read such as BOWTIE. BOWTIE2 is more appropriate for mapping reads longer than 50bp. Mapping parameters are largely dependent on reads length and quality, and although a large palette of parameters may be suitable, it is important to highlight that mapping specificity is function of the seed length (`-l` parameter), the number of mismatches allowed in the seed (`-n`) and the maximum number of alignments tolerated (`-m`). For best results, one should consider using stringent parameters (large `-m`, low `-n`) and keeping only uniquely mapped reads (`-m 1`). Also, it is important to remove the PCR amplification biases that may be generated during the sequencing library construction by collapsing all the reads mapping at the same location. This is carried out

with the Unix/Linux tools `sort` and `uniq`. At the same time, BOWTIE output is transformed into a standard format (BED).

Preparing the reference transcriptome for mapping and building the bowtie has the form:

```
bowtie-build <reference sequences> <name of bowtie index>
```

Example:

```
bowtie-build
20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.annota
tion.fas species.transcriptome
```

For each RNA-Seq library, mapping single end reads requires a single call to BOWTIE piped to `awk` and `uniq`:

```
bowtie -l <seed length> -n <mismatches> -m <max alignments>
<index> <reads file>
```

Example:

```
bowtie -l 45 -n 1 -m 1 species.transcriptome
RFTK.00356.28.fastq | awk 'BEGIN{OFS="\t"}{print
$4,$5,$5+length($6),".",0,$3;}' | sort -k 1,1 -k 2,2n -k 6,6 |
uniq >  RFTK.00356.28.fastq.bow.sorted.uniq.bed
```

2. Derive an annotation of the features to detect. In our case, there is a single feature per reference, starting with the first base pair (counted as position `0`) and as long as the reference. This can be produced with the help of the following `awk` script (`contigs2bed.awk`) acting on the reference sequences file:

```
BEGIN {RS = ">" ; ORS = "\n"; FS="\n";OFS="\t"; }
{sq="";
 for (i = 2; i <= NF; i++){sq=sq "" $i;}
 {print $1,0,length(sq);}}
```

This script is called:

```
awk -f contigs2bed.awk <reference sequences> > <annotated features>
```

Example:

```
awk -f contigs2bed.awk annotation.fas >  annotation.bed
```

For the sake of keeping commands onto a single line, `annotation.fas` stands for `20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.annotation.fas" and "annotation.bed for 20180707.Trinity_jaccPasa.fas.length_mt_2000.clusterc95.annotation.bed.`

3. Derive individual read count tables. INTERSECTBED, from the BEDTOOLS package, is a very popular tool to count bed file entries (reads) overlapping with annotated features (see above). This should be applied onto all the `.bed` files. The call to INTERSECTBED is straightforward:

```
intersectBed -c -a <annotated references> -b <reads> > <individual read count>
```

Example:

```
intersectBed -c -a annotation.bed -b RFTK.00356.28.fastq.bow.sorted.uniq.bed > RFTK.00356.28.fastq.bow.sorted.uniq.bed.cnt
```

In fact, this process can be automated with a simple loop:

```
for CURRENT_FILE in *.bed ; do intersectBed -c -a annotation.bed -b $CURRENT_FILE  > $CURRENT_FILE.cnt ; done
```

4. Combine reads count onto a single table. This requires two steps. First, the number of reads per reference sequence corresponds to the 4th column of the individual `.cnt` files. Thus, for each `.cnt` file, this column will be extracted and stored in a temporary file with the `cut` command. All these columns will then

be added one after the other, *column wise*, with the `paste` command. The resulting table can be called `annotation.treatments.cnts`. The first line creates a header in the output file, and the last line redirects the output of the `paste` command after the header (notice the >>, instead of the > symbol):

```
echo -e "gene\tRFTK.00356.28\tRFTL.00356.29\tRFTM.00356.30\
tXTD.024.72\tXTD.025.73\tXTD.026.74" >
annotation.treatments.cnts

cut -f 4 RFTK.00356.28.fastq.bow.sorted.uniq.bed.cnt > 28.col4
cut -f 4 RFTL.00356.29.fastq.bow.sorted.uniq.bed.cnt > 29.col4
cut -f 4 RFTM.00356.30.fastq.bow.sorted.uniq.bed.cnt > 30.col4
cut -f 4 XTD.024.72.fastq.bow.sorted.uniq.bed.cnt > 72.col4
cut -f 4 XTD.025.73.fastq.bow.sorted.uniq.bed.cnt > 73.col4
cut -f 4 XTD.026.74.fastq.bow.sorted.uniq.bed.cnt > 74.col4
paste annotation.bed 28.col4 29.col4 30.col4 72.col4 73.col4
74.col4 >> annotation.treatments.cnts
```

5. Filter out reference sequences with low reads counts. Low reads count can strongly affect differential analysis, since stochastic variations of a few reads may artefactually overestimate the biological response. It is therefore good practice to filter them out. Although this threshold is set empirically, references sequences associated with a total number of reads (across all conditions) less than 50 will be discarded. One might want to adjust this threshold according to the sequencing depth or other variable. Again, this is carried out with a simple `awk` script:

```
awk 'NR==1{print $0;}NR>1{if ($2+$3+$4+$5+$6+$7>=50){print
$0;}}'  annotation.treatments.cnts >
annotation.treatments.filtered.cnts
```

The resulting file, `annotation.treatments.filtered.cnts`, will be used for differential

analysis.

6. Differential analysis. DESeq [21] models variance with a generalization of the Poisson distribution (the negative binomial distribution). The following `R` script, based on the DESeq abundant documentation, reads the `annotation.treatments.filtered.cnts` file, creates a DESeq object, normalizes sequencing depth, estimates dispersion and performs the statistical test. It also generate a number of diagnostic graphs (MAplot, PCA...). Importantly, the `nbinomTest` parameters should match those indicated in the condition vector (`condition <- c( "cntrl", "cntrl", "cntrl","treat","treat","treat")`). Results are saved in the `.csv` file. The following `R` commands can be typed in directly from within the `R` environment:

```
library(DESeq)
input_file_name<-'annotation.treatments.filtered.cnts'
d<-read.table(input_file_name,sep="\
t",row.names=1,header=TRUE)

# The '#' symbol marks a comment ignored by the R interpretor.
#  This is useful to annotate a script

# prepare labels/designs
condition <- c(  "cntrl", "cntrl",
"cntrl","treat","treat","treat" )

# Simple comparison
## Let's make an object DEseq can manipulate
cds<-newCountDataSet(d,condition)
```

```
## Normalization
cds<-estimateSizeFactors(cds)

## Estimate dispersion
cds<-estimateDispersions(cds,method="pooled",sharingMode="fit-
only",fitType="local")

# PCA on stabilized variance
vsdFull = varianceStabilizingTransformation(cds)

png(paste(outfileBaseName,"PCA","png",sep="."))
plotPCA(vsdFull)
dev.off()

# binomial test
res = nbinomTest(cds,"cntrl","treat")
outfileBaseName=paste(input_file_name,"DE",sep='.')

# output MAplot
png(paste(outfileBaseName,"MAplot","png",sep="."))
plotMA(res)
dev.off()

# output histogram pValues
png(paste(outfileBaseName,"histPval","png",sep="."))
hist(res$pval,breaks=100, col="skyblue",border="slateblue",
main="")
```

```
dev.off()

# output results
out=paste(outfileBaseName,'csv',sep=".")
write.table(res,file=out,sep="\t",quote=FALSE,row.names=FALSE)
```

Alternatively, it would be more efficient to save this set of commands in a text file (with an explicit name like `DiffAnalysis.R`) and run through the `R` interpretor from the `bash` command line (see `R` documentation for details):

```
Rscript ./DiffAnalysis.R
```

4. Notes

1. This list of reads pre-processing tools is not exhaustive : Trimmomatic (http://www.usadellab.org/cms/?page=trimmomatic), AlienTrimmer (ftp://ftp.pasteur.fr/pub/gensoft/projects/AlienTrimmer/), AdapterRemoval (https://github.com/MikkelSchubert/adapterremoval), fastx_clipper (http://hannonlab.cshl.edu/fastx_toolkit/), Btrim (http://graphics.med.yale.edu/trim/).
2. For assembly, the stress is on the computing resources available. If not available on site, computing time can be paid for at cloud service providers. The cost should be planned from the start of the project.
3. For transcripts annotation, the emphasis of computing power should be put on the number of computing cores rather than the total amount of RAM available, since sequence comparison can be easily parallelized on multiple CPUs and is not memory intense. Computing time scales up almost linearly with the number of computing cores, which makes GPU servers (with up to 4096 computing units) attractive solutions.
4. There are two main constrains: 1) storage space is limited. Keeping all files at intermediate processing levels will consume storage space quickly. On the other hand, computing

resources are limited and re-running processing steps multiple times may not be practical. It is just a matter of finding the good balance that optimizes both storage and computing time.

2) it is often necessary to explore a number of alternative softwares and parameter sets, which increases dramatically the number of files produced in the course of the project. Although it is not scientific issue *per se*, the large number of files is a clear source of human error, amplified by any lack of clear and rigorous file naming convention. We therefore strongly advise to establish a strict file naming convention, in addition to regular laboratory good practices.

5. A possible naming convention would rely on the fact that file names suffer from little length limitations (in fact, <257 characters on most modern file systems) and accepts any number of dot under Unix/Linux. In this case, a processed file would inherit the name of the parent input files together with a suffix related to the processing step, suffixes being separated by a dot. The whole point is to make the file name explicit relative to its content. Suppose a folder contains the following files:

```
1 - 20180707.Trinity_jacc.fas
2 - 20150707.Trinity_jaccPasa.fas
3 - 20180707.Trinity_jaccPasa.fas.stats
4 - 20180707.Trinity_jaccPasa.fas
5 - 20180707.Trinity_jaccPasa.cd-hit-est_clusterc95.fas.gz
6 - 20180707.Trinity_jaccPasa.cd-hit-
est_clusterc95.annotated.fas.gz
7 - 20180707.Trinity_jaccPasa.fas
8 - 20180707.Trinity_jaccPasa.length_mt_2000.cd-hit-
est_clusterc95.fas.gz
9 - 20180707.Trinity_jaccPasa.length_mt_2000.cd-hit-
est_clusterc95.annotated.fas.gz
10 - 20180707.Trinity_jacc.fas
11 - 20180707.Trinity_jacc.cd-hit-est_clusterc95.fas.gz
12 - 20180707.Trinity_jacc.cd-hit-est_clusterc95.annotated.fas.gz
13 - 20180707.Trinity_jacc.fas
14 - 20180707.Trinity_jacc.length_mt_2000.cd-hit-
est_clusterc95.fas.gz
15 – 20180707.Trinity_jacc.length_mt_2000.cd-hit-
est_clusterc95.annotated.fas.gz
```

The name of file 1 can be decomposed into `20150707, Trinity_jacc and fas`, which corresponds to the raw sequences file `20180707` subjected to assembly with `Trinity` and `jacc` option. The file is a multiple FASTA file (`fas`). For file 8, `Trinity`

was run with `jacc` and `Pasa` options, where only contigs longer than 2000 bp were kept (`length_mt_2000`), before clustering with the cd-hit-est software and `clusterc95` option. This file contains the sequence of clustered contigs (`fas`) and is compressed (`gz`). File 9 is similar except that contigs have been annotated. Looking at the name of a file is enough to know what is corresponds to.

6. Although the command line interface looks puzzling at first, the galaxy of GNU tools (available on virtually all Unix systems and Linux) offer a large tool set of every day use in bioinformatics. These include softwares that can be used to combine files into a single file (`cat`), sort records by chromosome position (`sort`), remove identical entries (`uniq`)... Also, most bioinformatic packages are available from public depots, which make software installation simple and fast.
7. At some point of the project, one will naturally want to develop his/her own tools or deploy some file processing facility (*e.g*. custom filters). The `awk` program provides a simple (but powerful) scripting interface for file processing services. Although remarkably efficient on tabulated text files, it proves also valuable on other file types as well (such as FASTA files). The syntax is simple and numerous tutorials (web pages and videos) are available on the internet. In a nutshell, `awk` processes each line of a file, one after the other. For each line, columns are identified by a dollar sign and a number (the column number), so that the first column is `$1`, the second is `$2`... Printing a line (`$0`, the entire line) depending on whether the value of one column (e.g. `$3`) is larger than or equal to a threshold is illustrated by the following (simple) statement:

```
if ($3>=THRESHOLD) {print $0;}.
```

Arithmetic (and a wealth of other) operations are also possible. See `awk` documentation for details. The initial investment will quickly become worthwhile as the learning curve is very steep.

8. An other scripting language worth to consider is `python`. This language provides several programing paradigms (procedural, object oriented) and may be more appropriate for

complex tasks. Nonetheless, `python` is very versatile, easy to learn, its syntax is simple and cookbook approaches work well. There are also numerous packages and tutorials available on the internet, including in bioinformatics. Historically, the `perl` language was used but its syntax is less rigorous and it may be more difficult to analyze and reuse someone else's script. `Perl` interpreter is also slower than `python's`. For these reasons, nowadays, `perl` attracts less attention from researchers.

9. Finally, the power of Unix/Linux also comes with the command interpreter. Many exists but BASH and CSH/TCSH are found most of the time. In addition to interpret commands, they are also a rich scripting language with loops, control structures and moderately complex datatypes (such as arrays). It is good practice to copy all commands in a text file (and thus make a bash script) for two reasons: 1) keep a trace of what has been done (good laboratory practice), and 2) if needed, it is possible to re-run all the commands in it by simply executing the script.
10. Sequencing is not the (only) limiting step, and it is useless to sequence without proper manpower and know-how to process and analyze the datasets. Given the large time stamp of the bioinformatic component of the project, manpower is much more expensive than a few sequencing runs. Other non negligible sources of cost also include softwares, computing hardware, hosting facilities... In some cases, it might be worth considering to rent computation resources from a cloud. All this should be budgeted as early as at the design phase of the project. Too often, emphasis is restricted just on sequencing and projects later suffers from limited bioinformatics resources.
11. The total number of (high quality) reads directly impacts the assembly quality and resolution. Trying to assemble a transcriptome with single end reads, too little reads or poor quality reads will almost certainly fail, and efforts should focus on the generation of high quality and abundant raw sequences. We found that from >50 million (high quality) reads (meaning > 100-150 M raw reads), adding new reads does not significantly help detecting

novel transcripts (although contigs length still increases). Sequencing multiple libraries originating from different tissues would also capture more tissue specific transcripts, increase the diversity of the sequences assembled *de novo* and help describe more fully the repertoire of expressed sequences. Two critical aspects of transcriptome assembly are sequencing depth and reads quality.

Depending on the FASTQC quality controls, it may be worth clipping the low quality 5' and/or 3' end of reads. At the same time, reads length is an important parameter of sequence assembly and over-clipping might have a strong negative effect on assembly efficiency. It is all a matter of finding the good balance between reads quality and reads length and, although necessary, this puts stress on computing time and storage space. To this regard, a set of known transcripts (in databases or characterized by molecular biology and annotated "by hand") may prove to be a very valuable resources to limit the parameter space search and quickly channel toward a set of "biologically sound" parameters, where the pipeline independently reconstructs known transcripts. In fact, creating a high quality resource (a gold standard) is an integral part of the project.

12. During the course of the project, it is important to derive a number of quality controls at each milestone, in order to help decide whether or not to continue to the next step. There are numerous pitfalls: poorly curated raw sequence reads, the use of single end instead of paired-end reads, using non-optimal bioinformatics parameter sets... In addition to experimental and bioinformatic reconstruction biases, biological diversity adds an extra level of complexity (shared exons between transcripts, transcripts originating from duplicated genes, repeated/transposable elements in coding sequences...). As a result, chimerism is relatively common, UTRs are often missing... and stringent quality controls, together with experimental validation of a selected number of transcripts, are essential. One such quality control is shown in Figure 1E. This represents the dot-plot of a randomly picked-up reference sequence to its cognate ortholog, showing high co-linearity between them, and thus high quality assembly. The averaged density of mapped reads between the

two experimental conditions is shown at the bottom, nicely showing that there is no chimerism and that the contrast of reads density can be attributed to the expected transcript.

13. For optimal representation of transcripts description in a specie, consider merging reads originating from different tissues and/or experimental conditions. The larger the number of reads, the higher sensitivity and specificity. Ultimately, the reconstruction of lowly expressed transcripts and the capture of a large diversity of transcripts is dependent on the total number of reads available and the processive power (number of CPUs, RAM) available on the server.
14. Among others, two options may be interesting to consider when using TRINITY. The first option, `--jaccard_clip`, is typically used with paired end reads to limit the artifactual assembly of fusion transcript [11, 12]. The `--PasaFly` option is typically used to provide a good balance between transcript fragmentation (optimized total count of assembled contigs) and the reconstruction of full length transcripts. The choice between these parameters is not trivial and depends, in part, on the genome and genes structure and may be difficult to set *a priori*.
15. Transcriptome assembly generally generates hundreds thousands of sequence that can contain artifacts and chimeric transcripts as well as various alternative transcripts for each genes. This can also include new genes/transcripts specific to the species under study. If the aim of the work is to compare different species, it may be useful not to consider these specific transcripts and to perform the analysis at the gene-level after clustering of contigs.
16. When using BLAST in search for orthologs, two main parameters have to be looked at: word length (`-W`) and the substitution matrix. BLAST recodes sequences (query and subject) in an alphabet of `W` characters and then looks for identical sub-sequences. Thus, increasing `W` reduces computing speed but at the cost of sensitivity. Inversely, lowering `W` increases computing time and sensitivity, but may also increase significantly background noise. Again, using a gold standard may be instrumental in setting appropriate parameters. BLOSUM

matrices are popular matrices and the rule of thumb is that BLOSUM45 should be used when the expected similarity level is about 45%, BLOSUM60 for similarity ~ 60%...

Acknowledgment

This work was supported by the " Centre National de la Recherche Scientifque " (PEPS ExoMod " Triton " to L.M.S.) and the " Muséum National d'Histoire Naturelle " (Action Transversale du Muséum " Formes possibles, Formes réalisées " and Action Transversale du Muséum " Cycles biologiques " to L.M.S., and the Scientifc council post-doctoral position to G.K.). We thank J. Pedraza and the PCIA services for providing intensive computation infrastructure.

References

1. Laudet V (2011) The origins and evolution of vertebrate metamorphosis. Curr Biol 21(18):R726-737

2. Raymaekers SR, Darras VM (2017) Thyroid hormones and learning-associated neuroplasticity. Gene Comp Endocrinol 247:26-33

3. Rovet JF (2014) The rôle of thyroid hormones for brain development and cognitive function. Endocr Dev 26:26-43

4. Edeline E, Bardonnet A, Bolliet V et al (2005) Endocrine control of Anguilla anguilla glass eel dispersal: effect of thyroid hormones on locomotor activity and rheotactic behavior. Horm Behav 48(1):53-63

5. Wilsterman K, Buck CL, Barnes BM, Williams CT (2015) Energy regulation in context: Free-living female artic ground squirrels modulate the relatioship between thyroid hormones and activity among life history stages. Horm Behav 75:111-119

6. Sharma P, Tang S, Mayer GD, Patiño R (2016) Effects of thyroid endocrine manipulation on sex-related gene expression and population sex ratios in Zebrafish. Gen Comp Endocrinol 235:38-47

7. Sun BJ, Li T, Mu Y et al (2016) Thyroid hormone modulates offspring sex ratio in a turtle with temperature-dependent sex determination. Proc Biol Sci 283(1841).pii:20161206

8. Nishiwaki-Ohkawa T, Yoshimura T (2016) Molecular basis for regulating seasonal reproduction in vertebrates. J Endocrinol 229(3):R117-127

9. Geven EJ, Klaren PH (2017) The teleost head kidney: Integrating thyroid and immune signalling. Dev Comp Immunol 66:73-83

10. da Fonseca RR, Albrechtsen A, Themudo GE et al (2016) Next-generation biology: Sequencing and data analysis approaches for non-model organisms. Mar Genomics 30:3-13

11. Grabherr MG, Haas BJ, Yassour M et al (2011) Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nat Biotechnol 29(7):644–652

12. Haas BJ, Papanicolaou A, Yassour M et al (2013) De novo transcript sequence reconstruction from RNA-Seq: reference generation and analysis with Trinity. Nat Protoc 8(8):1494-1512

13. Luo R, Liu B, Xie Y et al (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience 1(1):18

14. Birol I, Jackman SD, Nielsen CB et al (2009) De novo transcriptome assembly with ABySS. Bioinforma Oxf Engl 25(21):2872–2877

15. Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 18(5):821–829

16. Schulz MH, Zerbino DR, Vingron M, Birney E (2012) Oases: robust de novo RNA-seq

assembly across the dynamic range of expression levels. Bioinforma Oxf Engl 28(8):1086–1092

17. Altschul SF, Madden TL, Schaffer AA et al (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25:3389-3402

18. Camacho C, Coulouris G, Avagyan V et al (2009) BLAST+: architecture and applications. BMC Bioinformatics 10:421

19. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10(3):R25

20. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9(4):357-359

21. Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biol 11(10):R106

22. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15(12):550

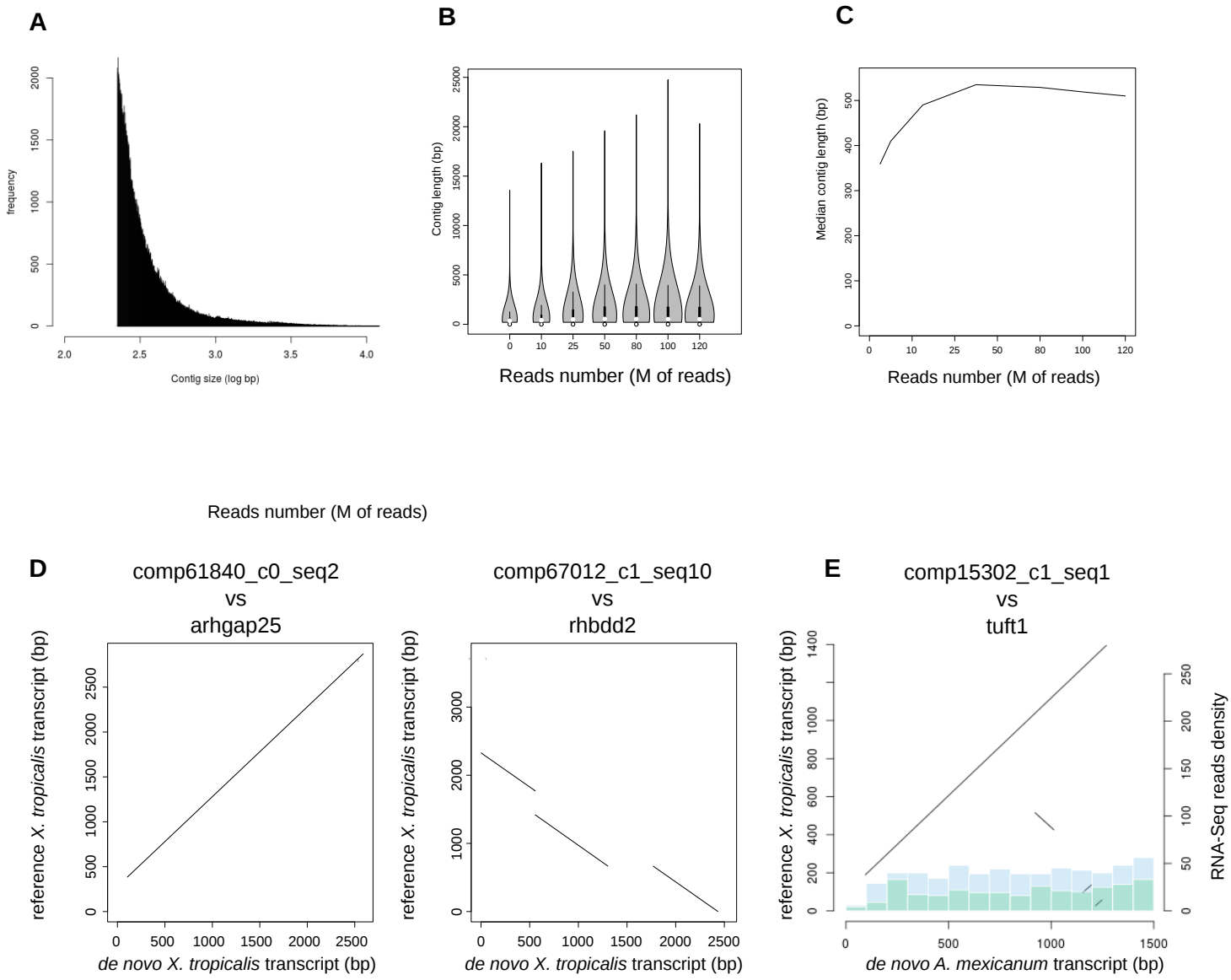
23. Quinlan AR, Hall AM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics 26(6):841-842

24. Bradnam KR, Fass JN, Alexandrov A et al (2013) Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. Gigascience 2(1):10

25. Aubry S, Kelly S, Kümpers BMC et al (2016) Deep Evolutionary Comparison of Gene Expression Identifies Parallel Recruitment of *Trans*-Factors in Two Independent Origins of C_4 Photosynthesis. PLoS Genet 12(5):e1006087

Figures caption

Figure 1: A. Size distribution of the contigs produced by TRINITY, before filtering. Example from real dataset. Notice that the vast majority of contigs are very short sized. B, C. Violin plot and median value of the contigs length produced, as a function of the total number of reads used for assembly. D. Dot-plot between assembled transcripts and expected sequence, from *Xenopus tropicalis*. E. Dot-plot between the assembled tuft1 transcript in *Ambystoma mexicanum* and the orthologous sequence from *Xenopus tropicalis*. Histograms represent RNA-Seq reads density, in control experiment and after hormone treatment. Example from real dataset.


A
frequency
Contig size (log bp)
B
Contig length (bp)
Reads number (M of reads)
C
Median contig length (bp)
Reads number (M of reads)
Reads number (M of reads)
D
comp61840_c0_seq2
vs
arhgap25
reference X. tropicalis transcript (bp)
de novo X. tropicalis transcript (bp)
comp67012_c1_seq10
vs
rhbdd2
reference X. tropicalis transcript (bp)
de novo X. tropicalis transcript (bp)
E
comp15302_c1_seq1
vs
tuft1
reference X. tropicalis transcript (bp)
RNA-Seq reads density
de novo A. mexicanum transcript (bp)