Doublesex Project: Step by step methods using *Acromyrmex echinatior* data

Description of the pipeline used for the Doublesex project.

1. Getting the data

1. Choosing the data we want to download

Data used in the project was downloaded from the SRA NCBI database (public access).
We need to choose the data we want to download in advance and get a list of the SRA accession numbers. Getting also the RunInfo table, which includes all the sample details, is also very helpful. For convention I have used this notation:

Accession list:
Xyy_[SRA Study Number]_SRR_Acc_List.txt
E.g. Aec_SRP031846_SraRunTable.txt

RunInfo Table
Xyy_[SRA Study Number]_SraRunTable.txt
E.g. Aec_SRP031846_SRR_Acc_List.txt

X is the inicial of the genus name (e.g A for Acromyrmex).
yy are the two fist letters of the species name (e.g. ec for echinatior)

Data used in the example is found in: Achinatior SRA data.
Information about the samples is downloaded from the site. Two files conatining all info and samples SRA ID's:

1. **Aec_SRP031846_SraRunTable.txt**

```
Assay_Type  BioSample   Experiment  LibrarySelection    LibrarySource   LoadDate
RNA-Seq SAMN02380854    SRX366954   cDNA    TRANSCRIPTOMIC  2015-12-15  6032    4165
RNA-Seq SAMN02380854    SRX366954   cDNA    TRANSCRIPTOMIC  2015-12-15  2764    1959
RNA-Seq SAMN02380854    SRX366954   cDNA    TRANSCRIPTOMIC  2015-12-15  3008    2105
RNA-Seq SAMN02380857    SRX366956   cDNA    TRANSCRIPTOMIC  2015-12-15  3793    2622
RNA-Seq SAMN02380857    SRX366956   cDNA    TRANSCRIPTOMIC  2015-12-15  3292    2321
RNA-Seq SAMN02380857    SRX366956   cDNA    TRANSCRIPTOMIC  2015-12-15  3341    2344
RNA-Seq SAMN02380858    SRX366958   cDNA    TRANSCRIPTOMIC  2015-12-14  3381    2368
RNA-Seq SAMN02380858    SRX366958   cDNA    TRANSCRIPTOMIC  2015-12-15  3171    2251
RNA-Seq SAMN02380858    SRX366958   cDNA    TRANSCRIPTOMIC  2015-12-15  3225    2295
RNA-Seq SAMN02380869    SRX366960   cDNA    TRANSCRIPTOMIC  2015-12-14  4291    2993
RNA-Seq SAMN02380869    SRX366960   cDNA    TRANSCRIPTOMIC  2015-12-15  2844    2027
RNA-Seq SAMN02380869    SRX366960   cDNA    TRANSCRIPTOMIC  2015-12-15  3093    2187
RNA-Seq SAMN02380868    SRX366962   cDNA    TRANSCRIPTOMIC  2015-12-15  3556    2475
RNA-Seq SAMN02380868    SRX366962   cDNA    TRANSCRIPTOMIC  2015-12-15  3197    2252
RNA-Seq SAMN02380868    SRX366962   cDNA    TRANSCRIPTOMIC  2015-12-15  3251    2295
RNA-Seq SAMN02380859    SRX366964   cDNA    TRANSCRIPTOMIC  2015-12-15  3597    2504
RNA-Seq SAMN02380859    SRX366964   cDNA    TRANSCRIPTOMIC  2015-12-15  3060    2160
RNA-Seq SAMN02380859    SRX366964   cDNA    TRANSCRIPTOMIC  2015-12-15  3109    2198
RNA-Seq SAMN02380867    SRX366965   cDNA    TRANSCRIPTOMIC  2015-12-14  4179    2899
RNA-Seq SAMN02380867    SRX366965   cDNA    TRANSCRIPTOMIC  2015-12-15  3667    2587
RNA-Seq SAMN02380867    SRX366965   cDNA    TRANSCRIPTOMIC  2015-12-15  3724    2624
RNA-Seq SAMN02380864    SRX366967   cDNA    TRANSCRIPTOMIC  2015-12-14  3981    2762
RNA-Seq SAMN02380864    SRX366967   cDNA    TRANSCRIPTOMIC  2015-12-15  2973    2124
RNA-Seq SAMN02380864    SRX366967   cDNA    TRANSCRIPTOMIC  2015-12-15  3563    2506
RNA-Seq SAMN02380865    SRX366969   cDNA    TRANSCRIPTOMIC  2015-12-15  3850    2669
RNA-Seq SAMN02380865    SRX366969   cDNA    TRANSCRIPTOMIC  2015-12-15  2942    2102
RNA-Seq SAMN02380865    SRX366969   cDNA    TRANSCRIPTOMIC  2015-12-15  3526    2485
```

1. **Aec_SRP031846_SRR_Acc_List.txt**

```
SRR1015499
SRR1015500
SRR1015501
SRR1015503
SRR1015504
SRR1015505
SRR1015507
SRR1015508
SRR1015509
SRR1015511
SRR1015512
SRR1015513
SRR1015515
SRR1015516
SRR1015517
SRR1015519
SRR1015520
SRR1015521
SRR1015523
SRR1015524
SRR1015525
SRR1015527
SRR1015528
SRR1015529
SRR1015531
SRR1015532
SRR1015533
```

2. Downloading the SRA files

Aspera download was used to download the data files locally.

Script: aspera_downloads.sh

```
#!/bin/bash
# Download a bunch of *.sra files from the NCBI SRA, using the aspera client


max_bandwidth_mbps=5000

# SRA files written line by line in a STDIN file

while read file
do
 /Users/afarre/Applications/Aspera\ Connect.app/Contents/Resources/ascp \
 -i /Users/afarre/Applications/Aspera\ Connect.app/Contents/Resources/asperaweb_id_d
 -k1 -QTr -l${max_bandwidth_mbps}m \
 anonftp@ftp-trace.ncbi.nlm.nih.gov:/sra/sra-instant/reads/ByRun/sra/${file:0:3}/${f
done <"$1"
```

With this script we get the SRA files to out local machine in a compressed format (file.sra)

3. Transfer to the server

Transfer the SRA files ti the SPARTAN server using `rsync`.

```
rsync ~/SRA/* afarre@spartan.hpc.unimelb.edu.au:/data/projects/punim0356/SRA/.
afarre@spartan.hpc.unimelb.edu.au's
```

4. Conversion to fastq format

In order to be able to use the data we have to convert it to fastq format. We use the `fastq-dump` from the SRA-toolkit for that.

Pair end reads (PE) and single end (SE) reads requiere different commands (-r option). We can find out if a sample is PE or SE in the RunInfo Table.

Script: dumpthemall.sh

```
#!/bin/bash
# ------------------------------------------------------------------
# [Author]  Anna Farre Orteu
#           afarre@student.unimelb.edu.au
#           Fastq-dump for paired and single end sra files
# ------------------------------------------------------------------

bold=$(tput bold)
normal=$(tput sgr0)
```

```
subject=fatsq-dump
version=0.1.0
usage="
${bold}DESCRIPTION:
${normal}Create scripts that will fastq-dump sra files for paired and single end lib
Paired end and single end have to be run ${bold}separately

${bold}USAGE:
${normal}dumpthemall [OPTION] [list of SRA files to fastq-dump]

where:
    -h  show this help text
    -p  path to SRA files
    -r  library type: ${bold}p ${normal}(paired end) and ${bold}s ${normal}(single e

"
# --- Option processing --------------------------------------------
if [ $# == 0 ] ; then
    echo "$usage"
    exit 1;
fi

while getopts ":p:r:h" optname
  do
    case "$optname" in
      "p")
        if [ ${OPTARG: -1} == "/" ] ; then
            path="$OPTARG"
        else
            path="$OPTARG/"
        fi
        ;;
      "r")
        if [ $OPTARG == "p" ] ; then
            options="--split-3 --readids --skip-technical --clip --read-filter pass
            echo $options
        elif [ $OPTARG == "s" ] ; then
            options="--readids --skip-technical --clip --read-filter pass --dumpbase
            echo $options
        else
            echo "Invalid argument. Specify p (paired) or s (single) end library"
        fi
        ;;
      "h")
        echo "$usage"
        exit 0;
        ;;
```

```
      "?")
        echo "Unknown option $OPTARG"
        exit 0;
        ;;
      ":")
        echo "No argument value for option $OPTARG"
        exit 0;
        ;;
      *)
        echo "Unknown error while processing options"
        exit 0;
        ;;
    esac
  done

if [ $OPTIND -eq 1 ]; then
    printf "\nNo options were passed
    $usage"
    exit 1;
fi

shift $(($OPTIND - 1))

param1=$1

# ---------------------------------------------------------------------
#  SCRIPT LOGIC GOES HERE
# ---------------------------------------------------------------------
while read line
do
    pathScript=${path}${line}_fastq-dump.sh
    cat <<-EOF > ${pathScript}
    #!/bin/bash"
    #SBATCH -p physical"
    #SBATCH --time=03:00:00"
    #SBATCH --nodes=1"
    #SBATCH --ntasks=1"
    #SBATCH --cpus-per-task=1"
    #SBATCH --job-name=fastq-dump"
    #SBATCH --mem-per-cpu=50000"
    #SBATCH --mail-type=ALL"
    #SBATCH --mail-user=afarre@student.unimelb.edu.au"
    #SBATCH --out=slurm_%j.out"
    #SBATCH --err=slurm_%j.err"

    fastq-dump $options ${path}${line}.sra
    EOF
```

```
done <"$1"
```

Commands:

```
[afarre@spartan SRA]$ pwd
/data/projects/punim0356/SRA
[afarre@spartan SRA]$ bash ~/scripts/dumpthemall.sh -p . -r p ../data_info/Aec_SRP03
```

5. Compress files

In order to save space in the server, we compress the `.fastq` files to `.fastq.gz.`

Script: gzip.sh

```
#!/bin/bash

# Gunzip all files listed
# Input through STDIN
# Use wildcards to zip multiple files
# Creates a script per file

for filename in `ls $@`
do

pathScript=$(pwd)/gzip_${filename}.sh

cat <<-EOF > ${pathScript}
#!/bin/bash

#SBATCH -p physical
#SBATCH --time=9-24
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --job-name=gzip_${filename}
#SBATCH --mem=25GB
#SBATCH --mail-type=ALL
#SBATCH --mail-user=afarre@student.unimelb.edu.au
#SBATCH --out=slurm_%j.out
#SBATCH --err=slurm_%j.err

gzip ${filename}

EOF
done
```

```
[afarre@spartan SRA]$ pwd
/data/projects/punim0356/SRA
[afarre@spartan SRA]$
bash ~/scripts/gzip.sh *fastq
```

2. Pre-processing

Once the data is in fastq format we need some steps of pre-procession.

1. Trimming the reads

We use TrimGalore to trim adapters and low quality parts of the reads.

Pair end reads (PE) and single end (SE) reads requiere different commands/scripts. We can find out if a sample is PE or SE in the RunInfo Table.

The script can run on compressed (`.gz`) data.

Script for PE: trimGalore.sh

```
#!/bin/bash
# ------------------------------------------------------------------
# [Author]  Anna Farre Orteu, 2017
#           afarre@student.unimelb.edu.au
#           Create scripts to run TrimGalore!
# ------------------------------------------------------------------

bold=$(tput bold)
normal=$(tput sgr0)

subject=TrimGalore!
usage="
${bold}DESCRIPTION:
${normal}Create scripts that will run TrimGalore!
Trims adapters, low quality read ends and runs FastQC
Paired end data ONLY
gz compressed data
extention must be:
    *_pass_1.fastq.gz
    *_pass_2.fastq.gz

trimGalore.sh [options] list_SRA.txt

where:
    -o output path
```

```
    -i input path

${bold}REQUIEREMENTS
${normal}cutadapt
fastqc
"

# --- Option processing --------------------------------------------
if [ $# == 0 ] ; then
    echo "$usage"
    exit 1;
fi
while getopts ":o:i:h" optname
  do
    case "$optname" in
      "i")
        if [[ $OPTARG == /* ]] ; then
            if [ ${OPTARG: -1} == "/" ] ; then
                inputPath="$OPTARG"
            else
                inputPath="$OPTARG/"
            fi

        else
            if [ ${OPTARG: -1} == "/" ] ; then
                inputPath="$(pwd)/$OPTARG"
            else
                inputPath="$(pwd)/$OPTARG/"
            fi
        fi
        ;;
      "o")
        mkdir -p $OPTARG

        if [[ $OPTARG == /* ]] ; then
            if [ ${OPTARG: -1} == "/" ] ; then
                outputPath="$OPTARG"
            else
                outputPath="$OPTARG/"
            fi

        else
            if [ ${OPTARG: -1} == "/" ] ; then
                outputPath="$(pwd)/$OPTARG"
            else
                outputPath="$(pwd)/$OPTARG/"
            fi
        fi
```

```
        ;;
      "h")
        echo "$usage"
        exit 0;
        ;;
      "?")
        echo "Unknown option $OPTARG"
        exit 0;
        ;;
      ":")
        echo "No argument value for option $OPTARG"
        exit 0;
        ;;
      *)
        echo "Unknown error while processing options"
        exit 0;
        ;;
    esac
  done

if [ $OPTIND -eq 1 ]; then
    printf "\nNo options were passed
    $usage"
    exit 1;
fi

shift $(($OPTIND - 1))

# ---------------------------------------------------------------------
#  SCRIPT LOGIC GOES HERE
# ---------------------------------------------------------------------


list=$@

while read filename
do

pathScript=${outputPath}trimGalore_${filename}.sh

cat <<-EOF > ${pathScript}
#!/bin/bash

#run TrimGalore on raw data

#SBATCH -p physical
#SBATCH --time=03:00:00
#SBATCH --job-name=trimGalore_${filename}
```

```
#SBATCH --mem=25GB
#SBATCH --mail-type=ALL
#SBATCH --mail-user=afarre@student.unimelb.edu.au
#SBATCH --out=slurm_trimGalore_${filename}_%j.out
#SBATCH --err=slurm_trimGalore_${filename}_%j.err


#path to list of files to use

module load Python
module load fastqc

/home/afarre/TrimGalore-0.4.5/trim_galore --fastqc --gzip --output_dir ${outputPath}
EOF
done <"${list}"
```

2. Quality Control: Running FASTQC

To check the quality of the reads we use FASTQC. We can run it on the raw reads or in the trimmed ones.

Fatqc creates multible html files with results of the quality contol tests.

Script: fastqcthemall.sh

```
#!/bin/bash
# ------------------------------------------------------------------
# [Author]  Anna Farre Orteu
#           afarre@student.unimelb.edu.au
#           Fastqc for paired and single end fastq files
# ------------------------------------------------------------------

bold=$(tput bold)
normal=$(tput sgr0)

subject=fatsq-dump
version=0.1.0
usage="
${bold}DESCRIPTION:
${normal}Run fastqc in paired and single end fastq files

${bold}USAGE:
${normal}fastqc.sh [OPTION] [list of fastq files]

where:
    -h  show this help text
    -o  path where output has to be saved
```

```
"
# --- Option processing --------------------------------------------
if [ $# == 0 ] ; then
    echo "$usage"
    exit 1;
fi

while getopts ":o:h" optname
  do
    case "$optname" in
      "o")
        mkdir -p $OPTARG

        if [[ $OPTARG == /* ]] ; then
            if [ ${OPTARG: -1} == "/" ] ; then
                outputPath="$OPTARG"
            else
                outputPath="$OPTARG/"
            fi

        else
            if [ ${OPTARG: -1} == "/" ] ; then
                outputPath="$(pwd)/$OPTARG"
            else
                outputPath="$(pwd)/$OPTARG/"
            fi
        fi
        ;;
      "h")
        echo "$usage"
        exit 0;
        ;;
      "?")
        echo "Unknown option $OPTARG"
        exit 0;
        ;;
      ":")
        echo "No argument value for option $OPTARG"
        exit 0;
        ;;
      *)
        echo "Unknown error while processing options"
        exit 0;
        ;;
    esac
  done

if [ $OPTIND -eq 1 ]; then
```

```
    printf "\nNo options were passed
    $usage"
    exit 1;
fi

shift $(($OPTIND - 1))

param1=$@

# -----------------------------------------------------------------
#  SCRIPT LOGIC GOES HERE
# -----------------------------------------------------------------

for filename in `ls ${param1}`
do

pathScript=fastqc_${filename}.sh
cat <<-EOF > ${pathScript}
#!/bin/bash

#SBATCH -p cloud
#SBATCH --time=09:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --job-name=fastqc_${filename}
#SBATCH --mem-per-cpu=4000
#SBATCH --mail-type=ALL
#SBATCH --mail-user=afarre@student.unimelb.edu.au
#SBATCH --out=slurm_%j.out
#SBATCH --err=slurm_%j.err

module load fastqc

fastqc -o ${outputPath} -t 8 ${filename}
EOF
done
```

3. Necklace

Necklace is a pipeline for RNA-seq analysis developed by the Oschlack Lab. It was made for RNA-Seq analyses involving species with incomplete genome or annotations. ie. most organisms other than human, mouse, drosophila etc.

1. Input for Necklace

Necklace takes as input RNA-seq files. However read names have to follow a very specific pattern:

DRRXXX_YYYY/Z

DRRXXX SRA sample name/number
YYYY read number
Z read pair number - 1 or 2

Script: readRename.sh

```
#!/bin/bash

#Rename reads in fastq.gz files coming from SRA
#to be used in Necklace

#Example:
    #Old name: DRRXXX.YYYY.Z
    #New name: DRRXXX_YYYY/Z

    #DRRXXX SRA sample name/number
    #YYYY read number
    #Z read pair number - 1 or 2


for filename in `ls $@`
do
cat <<-EOF > ${filename}_readRename.sh
#!/bin/bash

#SBATCH -p physical
#SBATCH --time=01:00:00
#SBATCH --job-name=readconvert
#SBATCH --mem=10GB
#SBATCH --out=slurm_%j.out
#SBATCH --err=slurm_%j.err
#SBATCH --mail-user=afarre@student.unimelb.edu.au
#SBATCH --mail-type=ALL


gunzip -c ${filename} | sed 's/\(^@.\)*\.\(.*\)\.\([1-2]\)/\1_\2\/\3/g' | gzip > nec
EOF
done
```

2. Running Necklace

Necklace gets the information about the input from a `.config` file.

Script: Vem_necklace.config

```
// sequencing data
reads_R1="/data/projects/punim0304/Vemeryi/SRA/trimGalore/necklaceInput/DRR030152_pa
reads_R2="/data/projects/punim0304/Vemeryi/SRA/trimGalore/necklaceInput/DRR030152_pa

//The reference genome and its annotation
annotation="/data/projects/punim0304/Vemeryi/genomes/GCF_000949405.1_V.emery_V1.0_ge
genome="/data/projects/punim0304/Vemeryi/genomes/GCF_000949405.1_V.emery_V1.0_genomi

//The genome and annotation of a related species
annotation_related_species="/home/afarre/original_genomes/Drosophila_melanogaster.BD
genome_related_species="/home/afarre/original_genomes/Drosophila_melanogaster.BDGP6.
```

It can be created with the following **commands**:

```
#IMPORTANT: no spaces between samples. Coma separated without spaces.
#ls -m -> will produce spaces. Remove them before running

echo -e //Sequencing data'\n'reads_R1=\"$(ls -m /data/projects/punim0304/Vemeryi/SRA
echo -e '\n'//The reference genome and its annotation'\n'annotation=\"$(ls /data/pro
echo -e genome=\"$(ls /data/projects/punim0304/Vemeryi/genome/*fna)\" >>Vem_necklace

echo -e  '\n'//The genome and annotation of a related species >>Vem_necklace.config
echo -e  annotation_related_species="/home/afarre/original_genomes/Drosophila_melanc
echo -e  genome_related_species="/home/afarre/original_genomes/Drosophila_melanogast

echo -e  '\n'//The genome and annotation of a related species >>Vem_necklace.config
echo -e  annotation_related_species="/home/afarre/original_genomes/Drosophila_melanc
echo -e  genome_related_species="/home/afarre/original_genomes/Drosophila_melanogast
```

Then Necklace can be run with **run_necklace.sh**:

```
#!/bin/bash

#SBATCH -p physical
#SBATCH --time=3-12
#SBATCH --job-name=necklace
#SBATCH --cpus-per-task=10
#SBATCH --mem=100GB
#SBATCH --mail-type=ALL
#SBATCH --mail-user=afarre@student.unimelb.edu.au
#SBATCH --out=slurm_%j.out
#SBATCH --err=slurm_%j.err

module load Java

cd /home/afarre/Vem_necklace/

MAX_JAVA_MEM=2g /home/afarre/.local/necklace-necklace_v0.9/tools/bin/bpipe run /home
```

Kallisto

1. Transcript-only fasta file

To run Kallisto first we have to create an index file that the program will use to do the pseudo alignment. This index file conatins **only transcripts**. Then the fist step is to create a fasta file with the only the transcript sequences.

Commands

```
[afarre@spartan Vem_necklace]$ pwd
/home/afarre/Vem_necklace/
[afarre@spartan Vem_necklace]$ module load Cufflinks
[afarre@spartan Vem_necklace]$ gffread -w GCF_000949405.1_V.emery_V1.0_genomic_kalli
```

2. Kallisto index file

Using the fasta file conataining only transcript sequences, create an index file.

Script: kallisto_index.sh

```
#!/bin/bash

#create an index for kallisto
#input: $1 index name $2 fasta file (transcripts)


#SBATCH -p physical
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --job-name=kallisto_idx
#SBATCH --mem=20GB
#SBATCH --mail-type=ALL
#SBATCH --out=slurm_%j.out
#SBATCH --err=slurm_%j.err
#SBATCH --mail-user=afarre@student.unimelb.edu.au


kallisto index --index=$1 $2
```

3. Run Kallisto

Paired-end and single-end samples require different scripts/commands.

Script: kallisto_pe.sh

```
#!/bin/bash
# ------------------------------------------------------------------
# [Author]  Anna Farre Orteu, 2017
#           afarre@student.unimelb.edu.au
#           Create scripts to run kallisto
# ------------------------------------------------------------------

bold=$(tput bold)
normal=$(tput sgr0)

subject=kallisto
usage="
${bold}DESCRIPTION:
${normal}Run kallisto for paired end data
use v0.43.0 when multithreading, v0.43.1 crashes

Only for ${bold}paired-end ${normal}samples

kallisto.sh [options] list.txt
```

```
where:
    -o output path
    -i kallisto index file
    -f input path (where fastq files are)
    -r extension for R1 (read 1) e.g _pass_1_val_1.fq.gz
    -l extension for R2 (read 2) e.g _pass_1_val_1.fq.gz

    list.txt contains all SRA sample IDs

"

# --- Option processing --------------------------------------------
if [ $# == 0 ] ; then
    echo "$usage"
    exit 1;
fi
while getopts ":o:i:f:r:l:h" optname
  do
    case "$optname" in
      "i")
        kallisto_index="$(pwd)/$OPTARG"
        ;;
      "o")
        mkdir -p $OPTARG

        if [[ $OPTARG == /* ]] ; then
            if [ ${OPTARG: -1} == "/" ] ; then
                outputPath="$OPTARG"
            else
                outputPath="$OPTARG/"
            fi

        else
            if [ ${OPTARG: -1} == "/" ] ; then
                outputPath="$(pwd)/$OPTARG"
            else
                outputPath="$(pwd)/$OPTARG/"
            fi
        fi
        ;;
      "f")
        if [[ $OPTARG == /* ]] ; then
            if [ ${OPTARG: -1} == "/" ] ; then
                inputPath="$OPTARG"
            else
                inputPath="$OPTARG/"
            fi
```

```
        else
            if [ ${OPTARG: -1} == "/" ] ; then
                inputPath="$(pwd)/$OPTARG"
            else
                inputPath="$(pwd)/$OPTARG/"
            fi
        fi
        ;;
      "r")
        read1Name="$OPTARG"
        ;;
      "l")
        read2Name="$OPTARG"
        ;;
      "h")
        echo "$usage"
        exit 0;
        ;;
      "?")
        echo "Unknown option $OPTARG"
        exit 0;
        ;;
      ":")
        echo "No argument value for option $OPTARG"
        exit 0;
        ;;
      *)
        echo "Unknown error while processing options"
        exit 0;
        ;;
    esac
  done

if [ $OPTIND -eq 1 ]; then
    printf "\nNo options were passed
    $usage"
    exit 1;
fi

shift $(($OPTIND - 1))

# ---------------------------------------------------------------
#  SCRIPT LOGIC GOES HERE
# ---------------------------------------------------------------


list="$@"
```

```
while read filename
do

mkdir -p ${outputPath}${subject}_${filename}/
pathScript=${outputPath}${subject}_${filename}/${subject}_${filename}.sh

cat <<-EOF > ${pathScript}
#!/bin/bash


#SBATCH -p physical
#SBATCH --time=03:00:00
#SBATCH --job-name=${subject}_${filename}
#SBATCH --cpus-per-task=8
#SBATCH --mem=40GB
#SBATCH --mail-type=ALL
#SBATCH --mail-user=afarre@student.unimelb.edu.au
#SBATCH --out=slurm_${subject}_${filename}_%j.out
#SBATCH --err=slurm_${subject}_${filename}_%j.err


/home/afarre/.local/kallisto_linux-v0.43.0/kallisto quant --bootstrap-samples 100 --

EOF
done<"${list}"
```

4. Sleuth: Analysis of Kallisto's results