Supplementary data SeqBox: RNAseq/ChIPseq reproducible analysis on a consumer game computer

Marco Beccuti, Francesca Cordero, Maddalena Arigoni, Riccardo Panero, Susanna Donatelli and Raffaele A Calogero

7/14/2017

Contents

Introduction

The docker4seq package was developed to facilitate the use of computing demanding applications in the field of NGS data analysis.

The docker4seq package uses docker containers that embed demanding computing tasks, e.g. short reads mapping.

This approach provides multiple advantages:

- user does not need to install all the software on its local server
- results generated by different containers can be organized in pipelines
- reproducible research is guarantee by the possibility of sharing the docker containers used for the analysis

Requirements

The minimal hardware requirements are a 4 core 64 bits linux computer, 32 Gb RAM, one SSD 250GB, with a folder with read/write permission for any users (chmod 777), and docker installed.

docker4seq and its graphical interface **4SeqGUI** can fit ideally in the NUC6I7KYK, Intel mini-computer equipped with Kingston Technology HyperX Impact 32GB Kit (2x16GB), 2133MHz DDR4 CL13 260-Pin SODIMM and Samsung 850 EVO - 250GB - M.2 SATA III Internal SSD.

MANDATORY: The first time *docker4seq* is installed the **downloadContainers** function needs to be executed to download in the local repository the docker containers that are needed by *docker4seq*.

```
library(docker4seq)
downloadContainers(group="docker")
```

Dockers containers

At the present time all functions requiring some sort of calculation are embedded in the following docker containers:

- docker.io/rcaloger/annotate.2017.01 used by rnaseqCounts, rsemanno
- docker.io/rcaloger/bwa.2017.01 used by bwaIndexUcsc, bwa
- docker.io/rcaloger/chipseq.2017.01 used by chipseqCounts, chipseq
- docker.io/rcaloger/r332.2017.01 used by experimentPower, sampleSize, wrapperDeseq2
- docker.io/rcaloger/mirnaseq.2017.01 used by mirnaCounts
- docker.io/rcaloger/rsemstar.2017.01 used by rnaseqCounts, rsemstarIndex, rsemstarUscsIndex
- docker.io/rcaloger/skewer.2017.01 used by skewer

docker container nomenclature

In case of updates required to solve bugs, which do not affect the calculation docker.io/rcaloger/XXXXX.YYYY.ZZ the fiels ZZ will be updated.

In case of updates which affect the calculation, e.g. new release of Bioconductor libraries, the field YYYY will be updated. Previous versions will be maintained to allow reproducibility.

Reproducibility

Within any folder generated with docker4seq functions it is saved the file **containers.txt**, which indicates the containers available in the local release of docker4seq.

In case, user would like to download a set of dockers containers different from those provided as part of the package those needs to be described in a file with the following format, **docker.repository/user/docker.name** which is passed to downloadContainers:

```
downloadContainers(group="docker", containers.file="my_containers.txt")
#an example of the my_containers.txt file content
docker.io/rcaloger/bwa.2017.01
docker.io/rcaloger/chipseq.2017.01
docker.io/rcaloger/r340.2017.01
```

Available workflows

At the present time are available the following workflows:

- **mRNAseq**, which allows:
  - adapter trimming with skewer
  - mapping with STAR
  - counting genes and isoforms with RSEM
  - ENSEMBL gene annotation.
  - organizing the output of RSEM in tables to be used for differential expression analysis
  - visualizing experiment data with PCA
  - evaluating experiment power and sample size
  - detecting differentially expressed genes/isoforms
- **miRNAseq**, which executes the workflow described in Cordero et al. PLoS One. 2012;7(2):e31630, embedding the following steps:
  - trimming adapters with cutadapt
  - miRNAs mapping on mirbase hairpins using SHRiMP
  - quantification of mature miRNAs.
  - visualizing experiment data with PCA
  - evaluating experiment power and sample size
  - detecting differentially expressed miRNAs
- **ChIPseq**, which allows:
  - adapter trimming with skewer
  - mapping with BWA
  - peak calling using either MACS v 1.4 or SICER v 1.1
  - associating peaks to the nearest gene, UCSC annotation
  - full annotation of the nearest gene

The most computing expensive steps of the analyses are embedded in the following docker4seq functions: **rnaseqCounts**, **mirnaCounts**, **chipseqCounts**. These functions are also the only one that have RAM and computing power requirements not usually available in consumer computers. Below its shown the time required to run the above three functions increasing the number of sequenced reads.

rnaseqCounts performances on different hardware configurations

The conversion between fastq files to counts is the most time consuming step in RNAseq data analysis and it normally requires high-end server. We tested **rnaseqCounts** on different hardware:

```
+ SeqBox: NUC6I7KYK CPU i7-6770HQ 3.5 GHz (1 core, 8 threads), 32 Gb RAM, HD 250 Gb SSD
+ SGI UV200 server: CPU E5-4650 v2 2.40GHz (8 cores, 160 threads), 1 Tb RAM, RAID 6, 100 Tb SATA
```

We run respctively 26, 52, 78, and 105 million reads increasing the number of threads till reaching the limit of the hardware or up to 64 threads, i.e. vaules shown in parenthesis in next figure. It is notable that SeqBox, mapping in 5 hours more than 100 milion reads, is able to handle can handle, in 20 hours, the throughput of the Illumina benchtop sequencer NextSeq 500, which produces up to 400 milion reads in a 30 hours run.

mirnaCounts performances on different hardware configurations

We run resepctively 3, 6, 12, and 24 miRNA samples in parallel using **mirnaCounts**, increasing the number of threads till reaching the limit of the hardware or 24 threads, i.e. vaules shown in parenthesis in next figure.

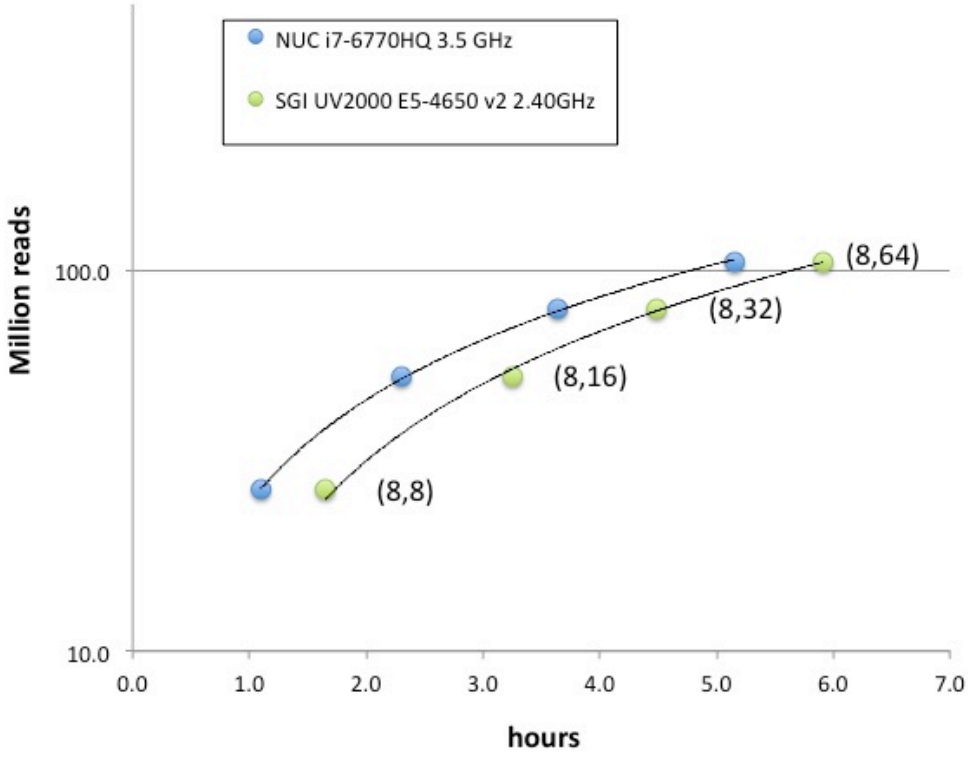

Figure 1: rnaseqCounts overall performance

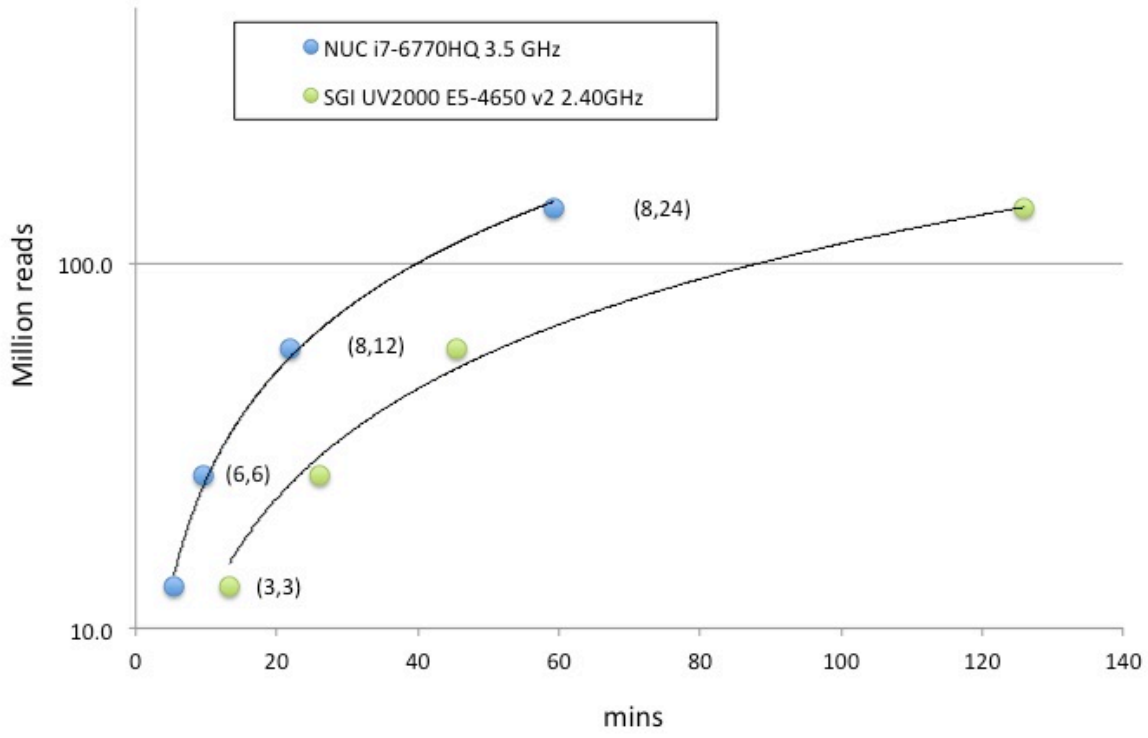

Figure 2: mirnaCounts overall performance

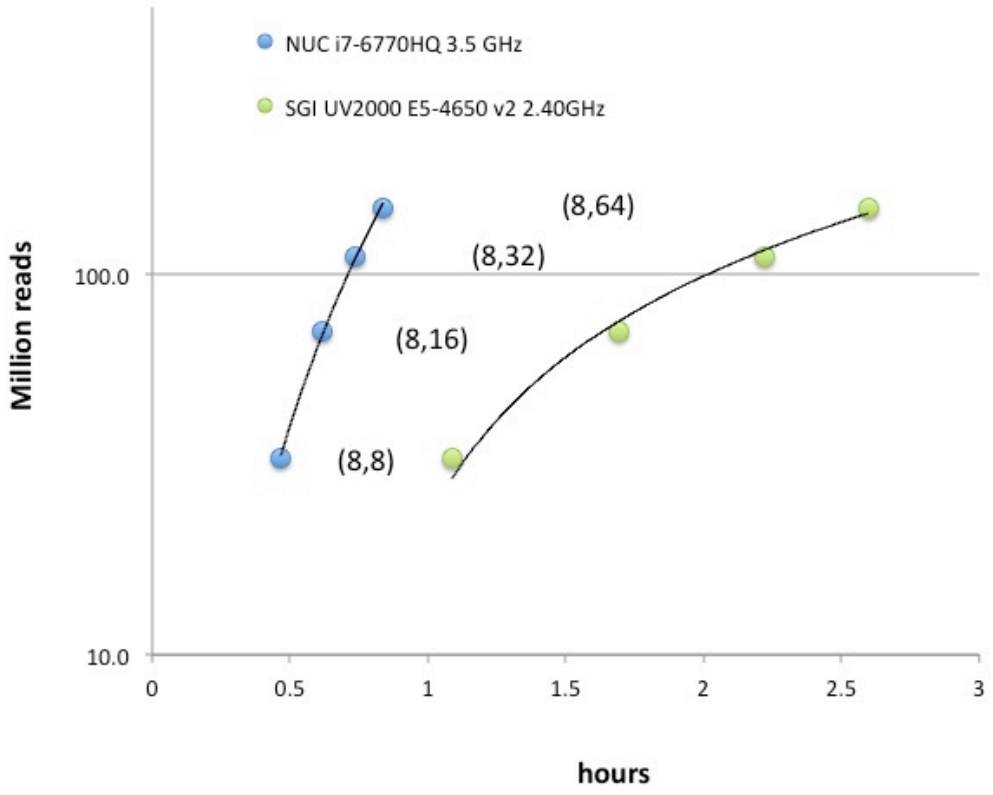

Figure 3: chipseqCounts overall performance

chipseqCounts performances on different hardware configurations

We run resepctively 37, 70, 111, and 149 million reads increasing the number of threads till reaching the limit of the hardware or up to 64 threads, i.e. vaules shown in parenthesis in next figure.

From the point of view of parallelization the **rnaseqCounts** is the one that embeds the most parallelized tools: i) mapping with STAR and ii) quantifying transcripts with RSEM. Both these tools have a massive I/O requirement. On the basis of the results shown above parallelization does not improve very much the overall performaces, but compensate the poor I/O of the RAID based on SATA disk array. On the other side the presence of an SSD with very high I/O compensate in the limited amount of cores of SeqBox.

In the case of **mirnaCounts** and **chipseqCounts** the parallelization is very little and it is only available for the reads mapping procedure. On the otherside both functions have a massive I/O. The reduced parallelization of these two analyses combined with the higher I/O of the SSD with respect to the SATA array makes SeqBox extremely effective even with very high number of reads to be processed.

RNAseq workflow: Howto

The mRNAseq workflow can be run using **4SeqGUI** graphical interface (linux/MAC):

Sample quantification is made of these steps:

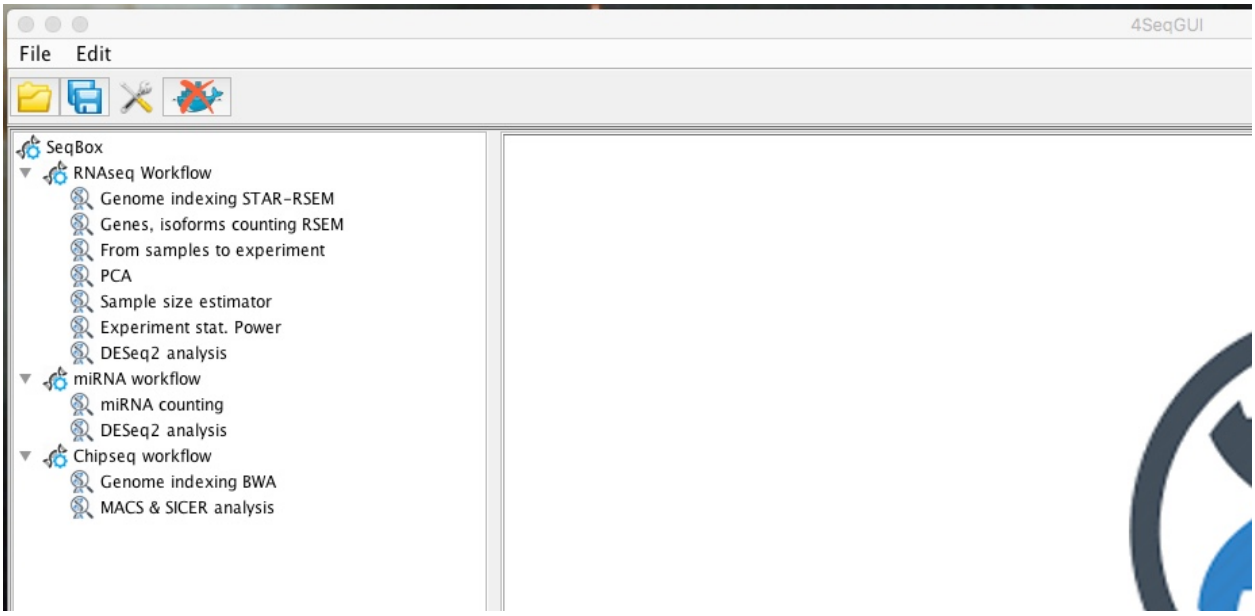

Figure 4: mRNAseq workflow

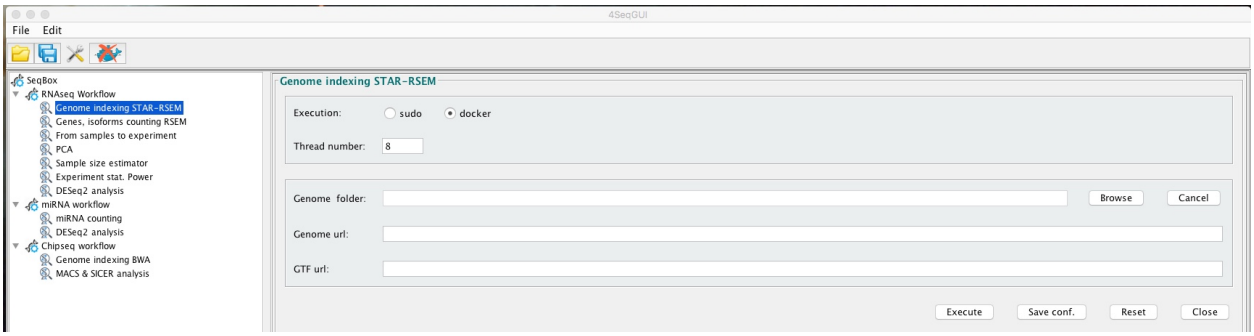

Figure 5: Creating a STAR genome index

- Creating a genome index for STAR (see end of this paragraph)
- Running removing sequencing adapters
- Mapping reads to the reference genome
- Quantify gene and transcript expression level
- Annotating genes.

All the parameters can be setup using 4SeqGUI

Creating a STAR index file for mRNAseq:

The index can be easily created using the graphical interface:

A detailed description of the parameters is given below.

Creating a STAR index file by line command

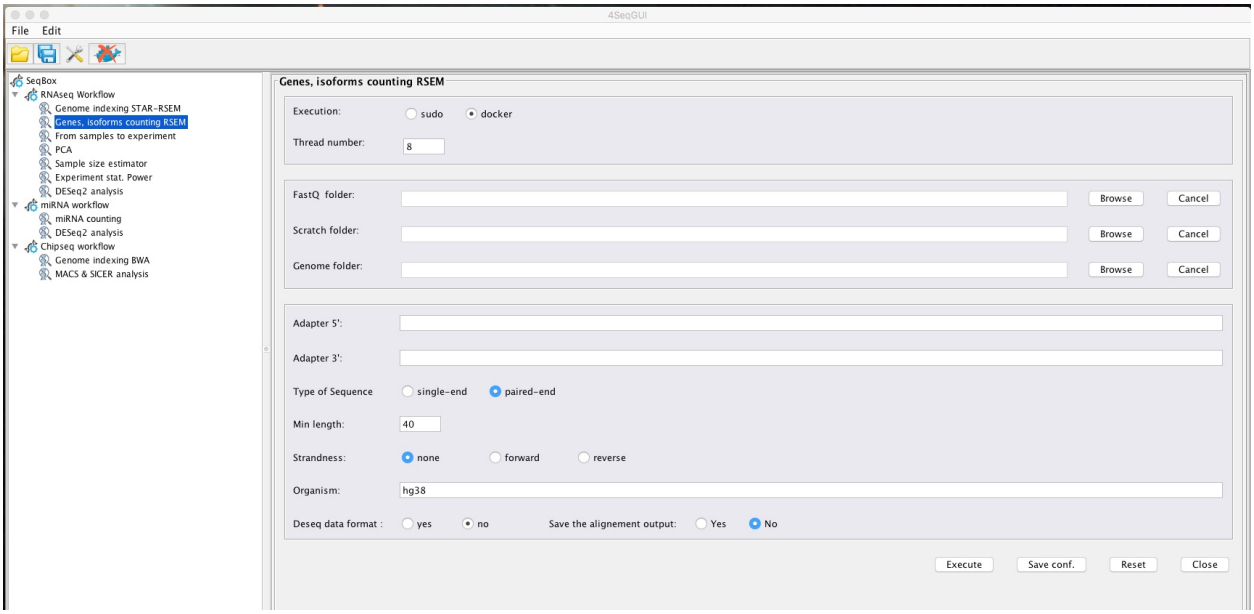

Figure 6: Gene, Isoform counting

```
rsemstarIndex(group="docker",genome.folder="/data/scratch/hg38star",
ensembl.urlgenome="ftp://ftp.ensembl.org/pub/release-87/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.toplevel.fa.gz",
ensembl.urlgtf="ftp://ftp.ensembl.org/pub/release-87/gtf/homo_sapiens/Homo_sapiens.GRCh38.87.gtf.gz")
```

In brief, **rsemstarIndex** uses ENSEMBL genomic data. User has to provide the URL (**ensembl.urlgenome**) for the file XXXXX_dna.toplevel.fa.gz related to the organism of interest, the URL (**ensembl.urlgtf**) for the annotation GTF XXX.gtf.gz and the path to the folder where the index will be generated (**genome.folder**). The parameter **threads** indicate the number of cores dedicated to this task.

Quantifying genes/isoforms:

A detailed description of the parameters is given below.

Sample quantification by line command

The sample quantification can be also executed using R and it is completely embedded in a single function:

```
#test example
system("wget http://130.192.119.59/public/test.mrnaCounts.zip")
unzip("test.mrnaCounts.zip")
setwd("./test.mrnaCounts")
library(docker4seq)
rnaseqCounts(group="docker",fastq.folder=getwd(), scratch.folder=getwd(),
adapter5="AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT",
adapter3="AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT",
seq.type="se", threads=8,  min.length=40,
genome.folder="/data/scratch/mm10star", strandness="none", save.bam=FALSE,
org="mm10", annotation.type="gtfENSEMBL")
```

User needs to create the **fastq.folder**, where the fastq.gz file(s) for the sample under analysis are located. The **scratch.folder** is the location where temporary data are created. The results will be then saved in the **fastq.folder**.

User needs to provide also the sequence of the sequencing adapters, **adapter5** and **adapter3** parameters. In case Illumina platform is used the adapters sequences can be easily recovered **here**.

seq.type indicates if single end (se) or pair-end (pe) data are provided, **threads** indicates the max number of cores used by *skewer* and *STAR*, all the other steps are done on a single core.

The **min.length** refers to the minimal length that a reads should have after adapters trimming. Since today the average read length for a RNAseq experiment is 50 or 75 nts would be better to bring to 40 nts the min.length parameter to increase the precision in assigning the correct position on the genome.

The **genome.folder** parameter refers to the location of the genomic index generated by STAR using the *docker4seq* function **rsemstarIndex**. The generation of the genome index is very simple and it is highlited at the end of this paragraph.

annotation.gene_id	annotation.gene_biotype	annotation.gene_name	annotation.source	transcript_id.s.	length	effective_length	expected_count	TPM	FPKM
ENSMUSG00000000001	protein_coding	Gnai3	ensembl_havana	ENSMUST00000000001	3262	3213.06	67	48.66	36.48
ENSMUSG00000000003	protein_coding	Pbsn	ensembl_havana	ENSMUST00000000003,ENSMUST00000114041	799.5	750.56	0	0	0
ENSMUSG00000000028	protein_coding	Cdc45	ensembl_havana	ENSMUST00000000028,ENSMUST00000096990,ENSMUST00000115585	1874.36	1825.42	43	54.97	41.21
ENSMUSG00000000031	lincRNA	H19	ensembl_havana	ENSMUST00000132294,ENSMUST00000136359,ENSMUST00000140716,ENSMUST00000149974,ENSMUST00000152754	817	768.06	1	3.04	2.28
ENSMUSG00000000037	protein_coding	Scml2	ensembl_havana	ENSMUST00000019101,ENSMUST00000074802,ENSMUST00000077375,ENSMUST00000087090,ENSMUST00000101113,ENSMUST00000112345,ENSMUST00000124775	3297.14	3248.21	0	0	0
ENSMUSG00000000049	protein_coding	Apoh	ensembl_havana	ENSMUST00000000049,ENSMUST00000133383,ENSMUST00000146050,ENSMUST00000152958	665.5	616.56	0	0	0
ENSMUSG00000000056	protein_coding	Narf	ensembl_havana	ENSMUST00000103015,ENSMUST00000151088,ENSMUST00000154047	4395	4346.06	24	12.89	9.66
ENSMUSG00000000058	protein_coding	Cav2	ensembl_havana	ENSMUST00000000058,ENSMUST00000115459,ENSMUST00000115462	2733	2684.06	38	33.04	24.77

Figure 7: gtf_annotated_genes.results

	sa_Cov.1	sb_Cov.1	sc_Cov.1	sd_Cov.1	se_Cov.1	sf_Cov.1	sg_Cov.1	ra_Cov.2	rb_Cov.2	rc_Cov.2	rd_Cov.2	re_Cov.2	rf_Cov.2	rg_Cov.2
TSPAN6:ENSG00000000003	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TNMD:ENSG00000000005	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DPM1:ENSG00000000419	161	205	163	56	91	58	225	179	118	222	161	145	187	253

Figure 8: counts table with covariates

strandness, is a parameter referring to the kit used for the library prep. If the kit does not provide strand information it is set to "none", if provides strand information is set to "forward" for Illumina stranded kit and it set to "reverse" for Illumina ACCESS kit. **save.bam** set to TRUE indicates that genomic bam file and transcriotomic bam files are also saved at the end of the analysis. **annotation.type** refers to the type of available gene-level annotation. At the present time is only available ENSEMBL annotation defined by the gtf downloaded during the creation of the indexed genome files, see paragraph *at the end*Creating a STAR index file for mRNAseq*.

Sample quantification output files

The mRNAseq workflow produces the following output files:

```
+ XXXXX-trimmed.log, containing the information related to the adapters trimming
+ gtf_annotated_genes.results, the output of RSEM gene quantification with gene-level annotation
+ Log.final.out, the statistics of the genome mapping generated by STAR
+ rsem.info, summary of the parameters used in the run
+ genes.results, the output of RSEM gene quantification
+ isoforms.results, the output of RSEM isoform quantification
+ run.info, some statistics on the run
+ skewerd_xxxxxxxxxxxx.log, log of the skewer docker container
+ stard.yyyyyyyyyyyy.log, log of the star docker container
```

The first column in **gtf_annotated_genes.results** is the ensembl gene id, the second is the biotype, the 3rd is the annotation source, the 4th contains the set of transcripts included in the ensembl gene id. Then there is the length of the gene, the lenght of the gene to which is subtracted the average length of the sequenced fragments, the expected counts are the couts to be used for differentiual expression analysis. TPM and FPM are normalized gene quantities to be used only for visualization purposes.

From samples to experiment

The RSEM output is sample specific, thus it is necessary to assemble the single sample in an experiment table including in the header of the column both the covariates and the batch, if any. The header sample name is separated by the covariate with an underscore, e.g. mysample1_Cov1, mysample2_Cov2.

In case also a batch is present also this is separated by a further underscore, e.g. mysample1_Cov1_batch1, mysample2_Cov_batch2.

The addition of the covariates to the various samples can be done using the **4seqGUI** using the button: *From samples to experiment*. Covariates are added to the column name, i.e. sample name, using _, e.g. mysample_Cov.1. Batches are added after covariates with _, e.g. mysample_Cov.1_batch.1.

From samples to experiments by line command

	sa_Cov.1_1	sb_Cov.1_2	sc_Cov.1_3	sd_Cov.1_4	se_Cov.1_5	sf_Cov.1_6	sg_Cov.1_7	ra_Cov.2_1	rb_Cov.2_2	rc_Cov.2_3	rd_Cov.2_4	re_Cov.2_5	rf_Cov.2_6	rg_Cov.2_7
TSPAN6:ENSG00000000003	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TNMD:ENSG00000000005	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DPM1:ENSG00000000419	161	205	163	56	91	58	225	179	118	222	161	145	187	253

Figure 9: counts table with covariates and batch

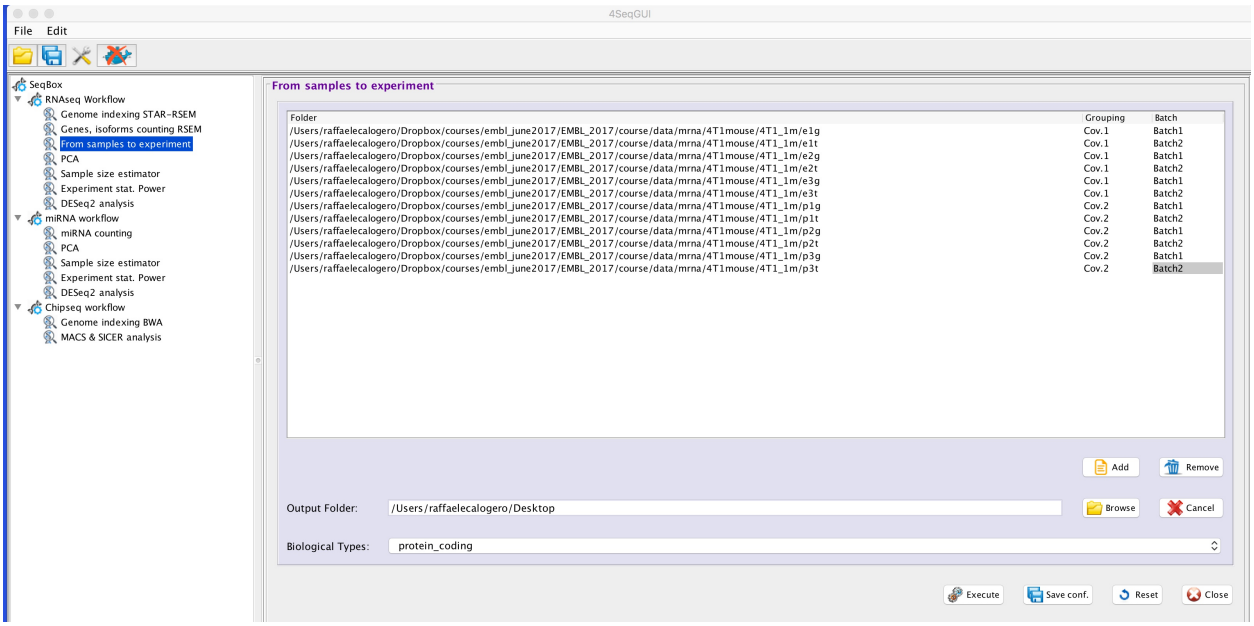

Figure 10: generating a table with covariates

```
#test example
system("wget http://130.192.119.59/public/test.samples2experiment.zip")
unzip("test.samples2experiment.zip")
setwd("test.samples2experiment")
library(docker4seq)
sample2experiment(sample.folders=c("./e1g","./e2g","./e3g",
"./p1g", "./p2g", "./p3g"),
covariates=c("Cov.1","Cov.1","Cov.2","Cov.2","Cov.2"),
bio.type="protein_coding", output.prefix=".")
```

User needs to provide the paths of the samples, **sample.folder** parameter, a vector of the covariates, **covariates**, and the biotype(s) of interest, **bio.type** parameter. The parameter **output.prefix** refers to the path where the output will be created, as default this is the actual R working folder.

From samples to experiments output files

The samples to experiments produces the following output files:

```
+ _counts.txt: gene-level raw counts table for differential expression analysis
+ _isoforms_counts.txt: isoform-level raw counts table for differential expression analysis
+ _isoforms_log2TPM.txt: isoform-level log2TPM for visualization purposes
+ _log2TPM.txt: gene-level log2TPM for visualization purposes
+ _isoforms_log2FPKM.txt: isoform-level log2FPKM for visualization purposes
+ _log2FPKM.txt: gene-level log2FPKM for visualization purposes
+ XXXXX.Rout: logs of the execution
```

Visualizing experiment data with PCA

PCA finds the principal components of data. Principal components are the underlying structure in the data. They are the directions where there is the most variance, the directions where the data is most spread out. **4SeqGUI** provides an interface to the generation experiment samples PCA

The plot is saved in **pca.pdf** in the selected folder.

PCA by line command

```
#test example
system("wget 130.192.119.59/public/test.analysis.zip")
unzip("test.analysis.zip")
setwd("test.analysis")
library(docker4seq)
pca(experiment.table="_log2FPKM.txt", type="FPKM", legend.position="topleft", covariatesInNames=FALSE, principal.components=c(1,
```

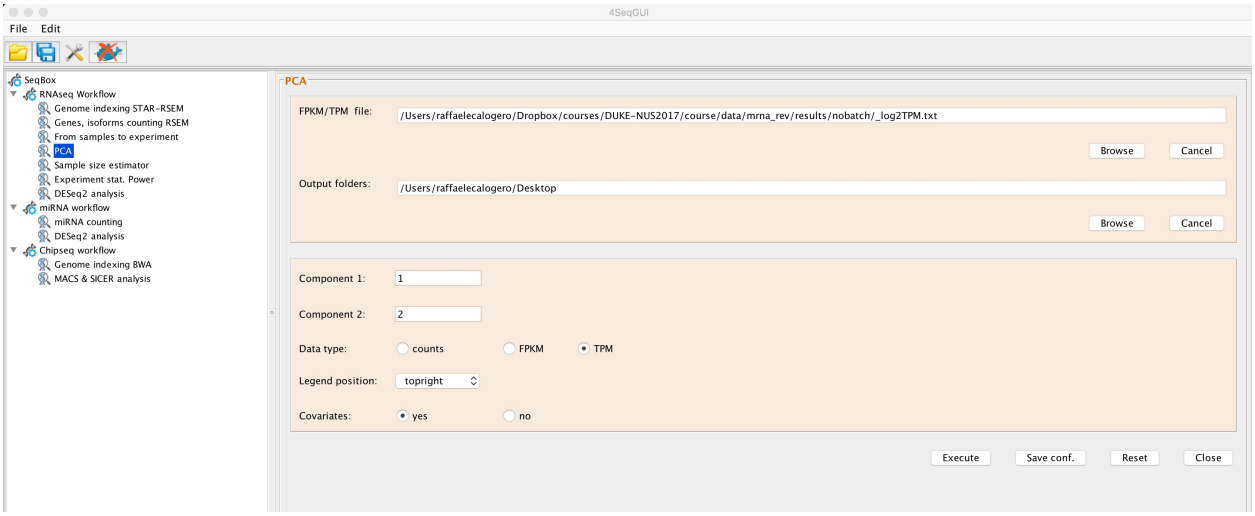

Figure 11: PCA

User needs to provide the paths of experiment table, **experiment.table** parameter, i.e. the file generated using the samples2experiment function. The **type** parameter indicates if FPKM, TPM or counts are used for the PCA geenration. The parameter **legend.position** defines where to locate the covariates legend. The parameter **covariatesInNames** indicates if the header of the experiment table contains or not covariate information. The parameter **principal.components** indicates which principal components should be plotted. **output.folder** indicates where to save the pca.pdf file.

The values in parentesis on x and y axes are the amount of variance explained by each principal component.

IMPORTANT: The above analysis is suitable also for miRNAseq data

Evaluating experiment power and sample size

RnaSeqSampleSize Bioconductor package provides the possibility to calculate, from a pilot experiment, the statistical power and to define the optimal sample size. **4SeqGUI** provides an interface to sample size estimation:

and to statistical power estimation:

Sample size estimation by line command

Sample size estimation is an important issue in the design of RNA sequencing experiments. We have implemented a wrapper function for sample size estimation using the bioconductor package RnaSeqSampleSize.

```
#test example
system("wget 130.192.119.59/public/test.analysis.zip")
unzip("test.analysis.zip")
setwd("test.analysis")
library(docker4seq)
sampleSize(group="docker", filename="_counts.txt", power=0.80, FDR=0.1, genes4dispersion=200, log2fold.change=1)
```

The requested parameters are the path to the counts experiment table generated by **samples2experiment** function. The param **power** indicates the expecte fraction of differentially expressed gene, e.g 0.80. **FDR** and **log2fold.change** are the two thresholds used to define the set of differentially expressed genes of interest.

The output file is **sample_size_evaluation.txt** is saved in the R working folder, below an example of the file content:

IMPORTANT: The above analysis is suitable also for miRNAseq data

Experiment statistical power estimation by line command

Experiment power provides an indication of which is the fraction of differentially expressed genes that can be detected given a specific number of samples and differential expression detection thresholds. We have implemented a wrapper function for experiment power estimation using the bioconductor package RnaSeqSampleSize.

```
#test example
system("wget 130.192.119.59/public/test.analysis.zip")
unzip("test.analysis.zip")
setwd("test.analysis")
library(docker4seq)
experimentPower(group="docker", filename="_counts.txt",replicatesXgroup=7, FDR=0.1, genes4dispersion=200, log2fold.change=1)
```

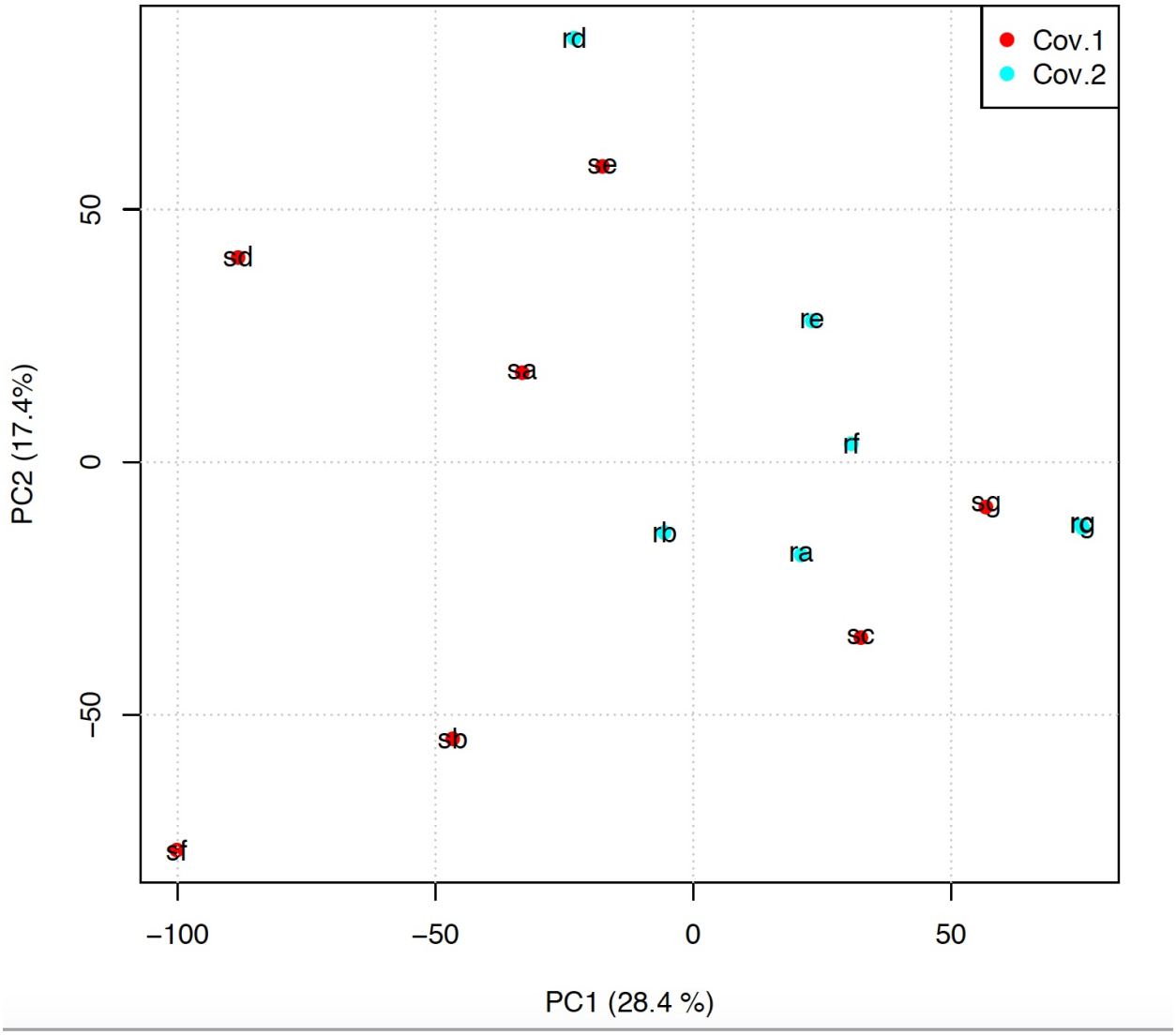

Figure 12: pca.pdf

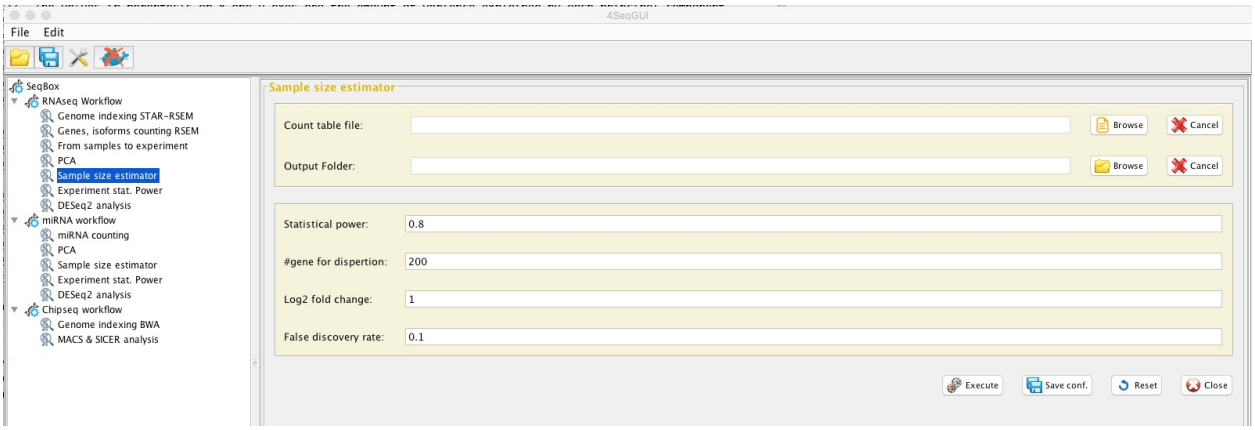

Figure 13: sample size estimation

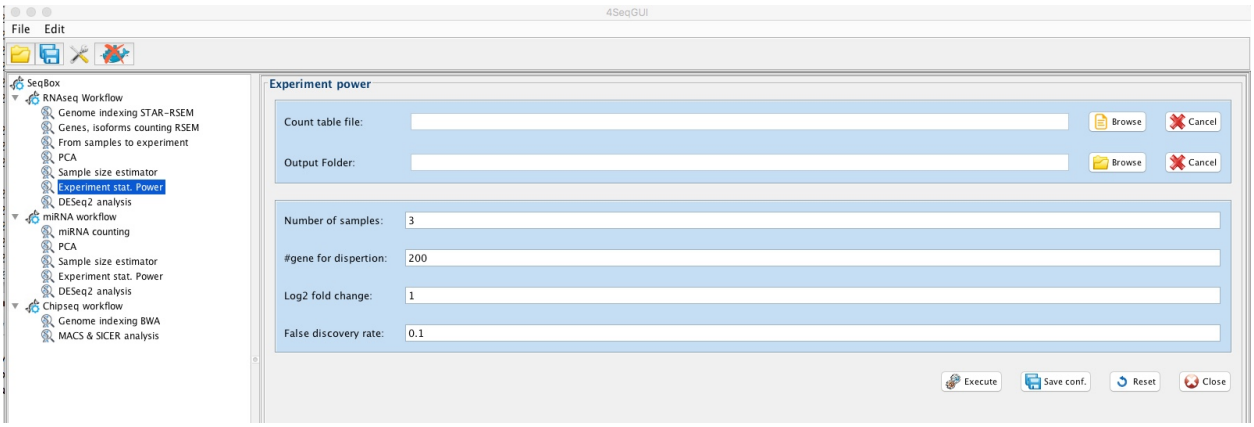

Figure 14: stat power estimation

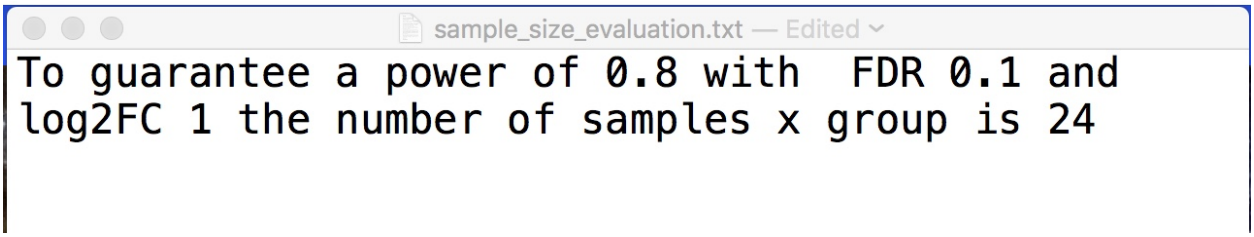

Figure 15: sample_size_evaluation.txt

The requested parameters are the path to the counts experiment table generated by **samples2experiment** function. The param **replicatesXgroup** indicates the number of sample associated to each of the two covariates. **FDR** and **log2fold.change** are the two thresholds used to define the set of differentially expressed genes of interest. **genes4dispersion** indicates the number of genes used in estimation of read counts and dispersion distribution.

The output file is **power_estimation.txt** is saved in the R working folder, below an example of the file content:

IMPORTANT: The above analysis is suitable also for miRNAseq data

Differential expression analysis with DESeq2

A basic task in the analysis of count data from RNA-seq is the detection of differentially expressed genes. **4SeqGUI** provides an interface to DESeq2 to simplify differential expression analysis.

The output files are:

DEfull.txt containing the full set of results generated by DESeq2

DEfiltered_log2fc_X_fdr_Y.Y.txt containing the set of differentially expressed genes passing the indicated thresholds

genes4david.txt a file containing only the gene symbols to be used as input for DAVID or ENRICHR

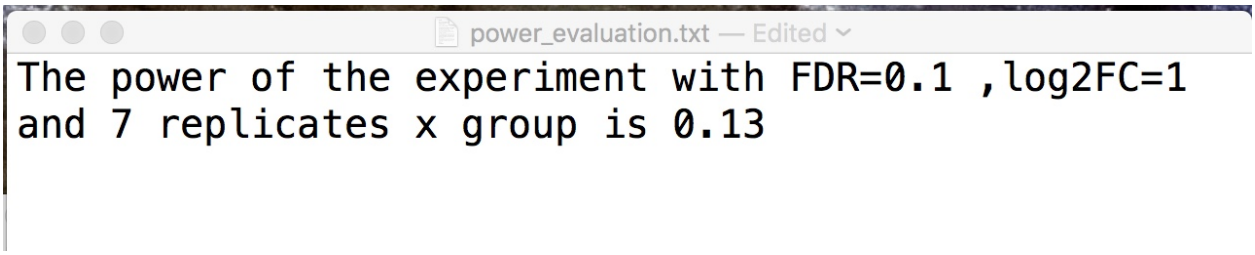

Figure 16: power_estimation.txt

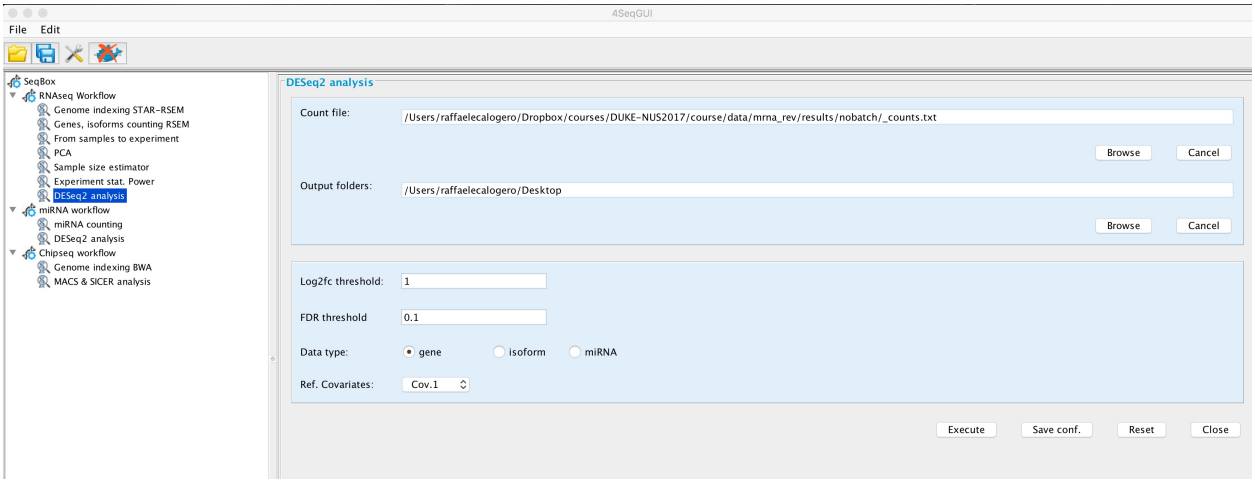

Figure 17: DESeq2

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
TSPAN6:ENSG00000000003	0	NA	NA	NA	NA	NA
TNMD:ENSG00000000005	0	NA	NA	NA	NA	NA
DPM1:ENSG00000000419	151.2813052	0.229109109	0.281353207	0.814311347	0.415466611	0.654910102
SCYL3:ENSG00000000457	9.579249027	-0.409918944	0.370807831	-1.105475425	0.268953637	0.521645453
C1orf112:ENSG00000000460	41.97662811	-0.24214877	0.248599391	-0.974052143	0.33003065	0.582892889
FGR:ENSG00000000938	0.404790498	-0.377526098	0.396378555	-0.952438253	0.340874767	NA
CFH:ENSG00000000971	329.6621973	0.083214521	0.556259849	0.14959649	0.881082977	0.947442296
FUCA2:ENSG00000001036	13.75729193	-0.247735458	0.560111923	-0.442296347	0.658274774	0.830452815
GCLC:ENSG00000001084	60.38724968	0.309693536	0.380151483	0.814658234	0.415267967	0.654779367

Figure 18: DEfull.txt

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
CFLAR:ENSG00000003402	39.9725599	-1.3390809	0.32914578	-4.06835204	4.73E-05	0.00243639
WDR54:ENSG00000005448	41.1342173	-1.0498896	0.31243125	-3.360386	0.00077834	0.01470356
KMT2E:ENSG00000005483	142.828476	-1.20951634	0.31531729	-3.83587064	0.00012512	0.00441174
TRAPPC6A:ENSG00000007255	7.27624305	-1.25031688	0.51531214	-2.42632917	0.01525243	0.09759974
RPUSD1:ENSG00000007376	3.22069221	1.28290362	0.510833	2.51139536	0.01202549	0.08408484
LUC7L:ENSG00000007392	22.4784933	1.07737824	0.30939729	3.4821838	0.00049734	0.01090749
SYN1:ENSG00000008056	68.2773928	1.65920689	0.52085936	3.18551807	0.00144495	0.02172925
IDS:ENSG00000010404	37.2144825	-1.19958366	0.2985641	-4.01784293	5.87E-05	0.00267413
CALCOCO1:ENSG00000012822	4.22352463	-1.74891951	0.52455504	-3.33410102	0.00085576	0.01552904

Figure 19: DEfiltered_log2fc_1_fdr_0.1.txt

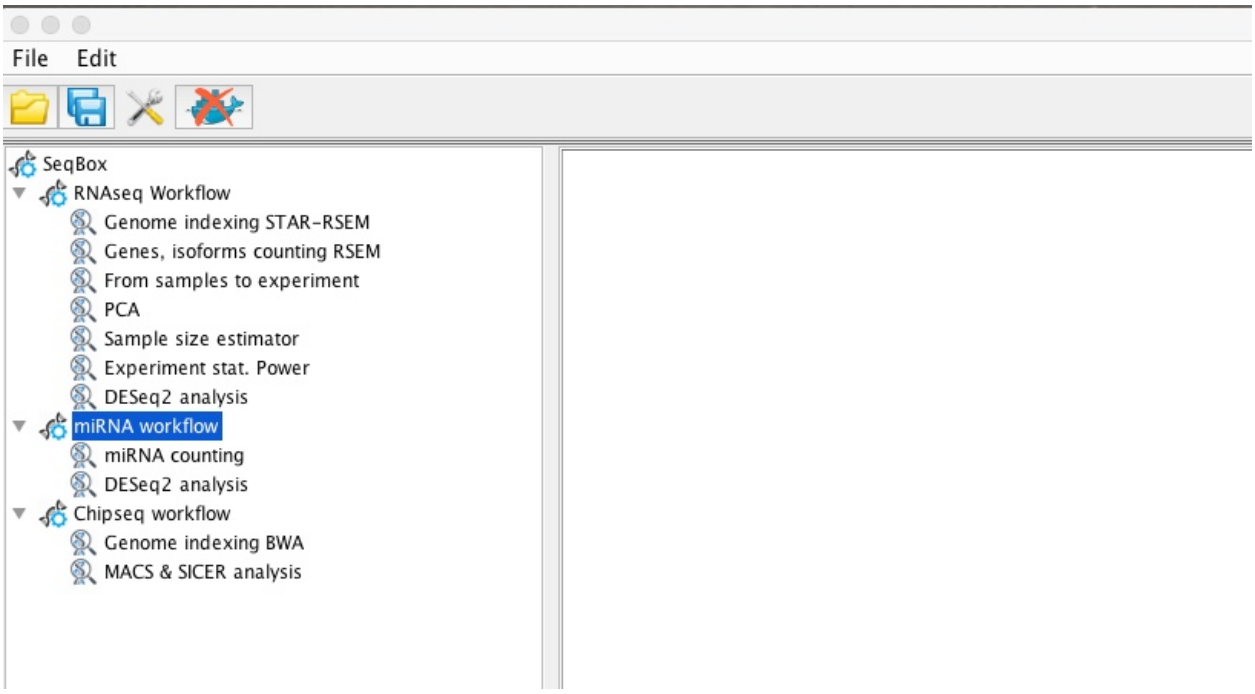

Figure 20: miRNAseq workflow

log2normalized_counts.txt, DESeq2 log2 library size normalized counts to be used for visualizaiton only instead of log2 FPKM or log2 TPM.

DESeq2 by line command

```
#test example
system("wget 130.192.119.59/public/test.analysis.zip")
unzip("test.analysis.zip")
setwd("test.analysis")
library(docker4seq)
wrapperDeseq2(output.folder=getwd(), group="docker", experiment.table="_counts.txt", log2fc=1, fdr=0.1, ref.covar="Cov.1", type=
```

User needs to provide experiment table, **experiment.table** param, i.e. the counts table generated with **samples2experiment** function, the thresholds for the differential expression analysis, **log2fc** and **fdr** params, the reference covariate, **ref.covar** param, i.e. the covariate that is used as reference for differential expression detection, the **type** param, whihc refers to the type of experiment table in use: *gene*, *isoform*, *mirna*, **batch** parameter that indicates, it it is set to **TRUE** that the header of the experiment table also contains the extra information for the batch effect (see above).

IMPORTANT: the above analysis can be also applied to miRNAseq data.

miRNAseq workflow

The miRNAseq workflow can be run using **4SeqGUI** graphical interface:

The miRNAseq docker container executes the following steps:

The full workflow is described in Cordero et al. Plos ONE 2012. In brief, fastq files are trimmed using cutadapt and the trimmed reads are mapped on miRNA precursors, i.e. harpin.fa file, from miRBase using SHRIMP. Using the location of the mature miRNAs in the precursor, countOverlaps function, from the Bioconductor package GenomicRanges is used to quantify the reads mapping on mature miRNAs.

All the parameters needed to run the miRNAseq workflow can be setup using 4SeqGUI

A detailed description of the parameters is given below.

miRNAseq workflow by line command

The miRNAseq workflow can be also executed using R and it is completely embedded in a unique function:

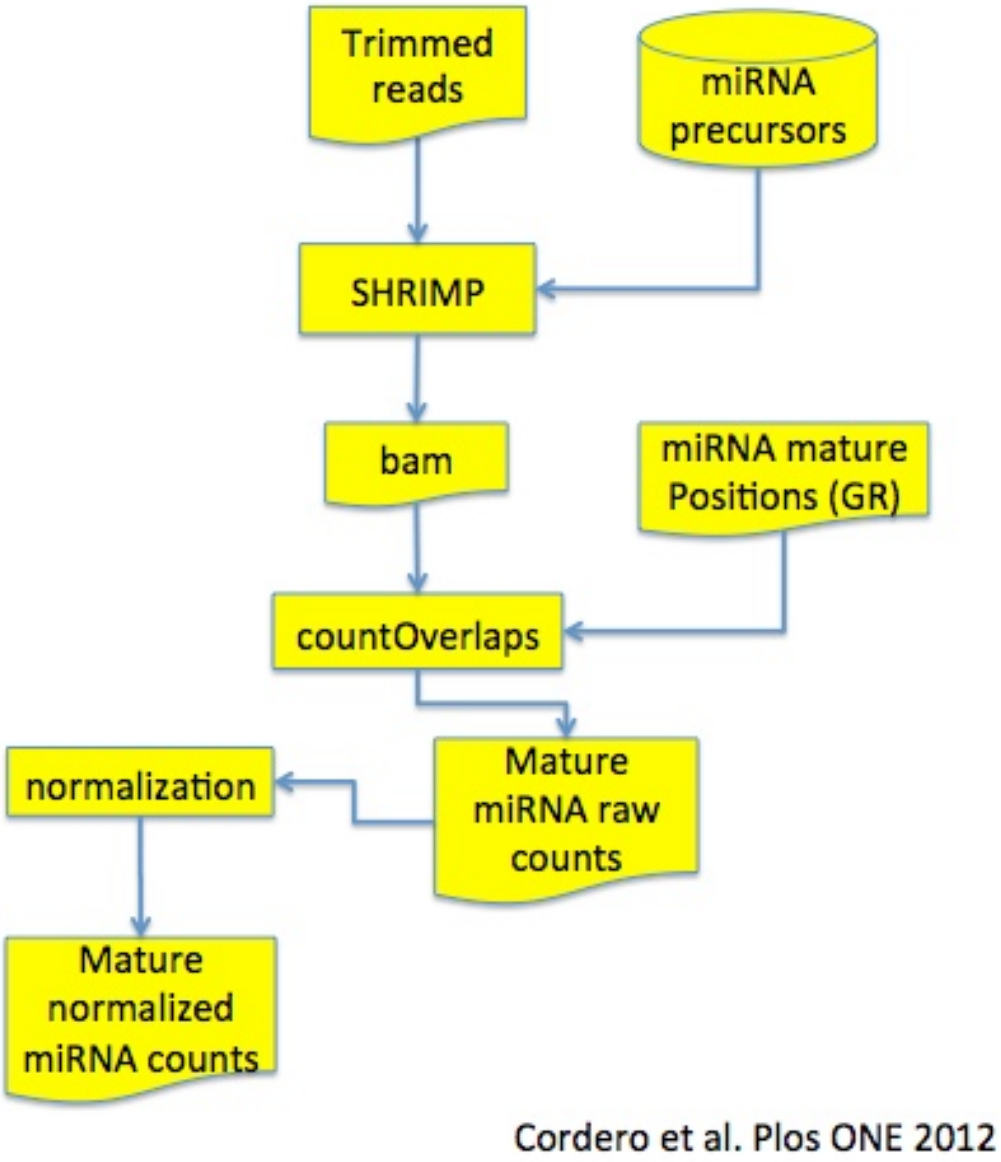

Figure 21: miRNAseq workflow

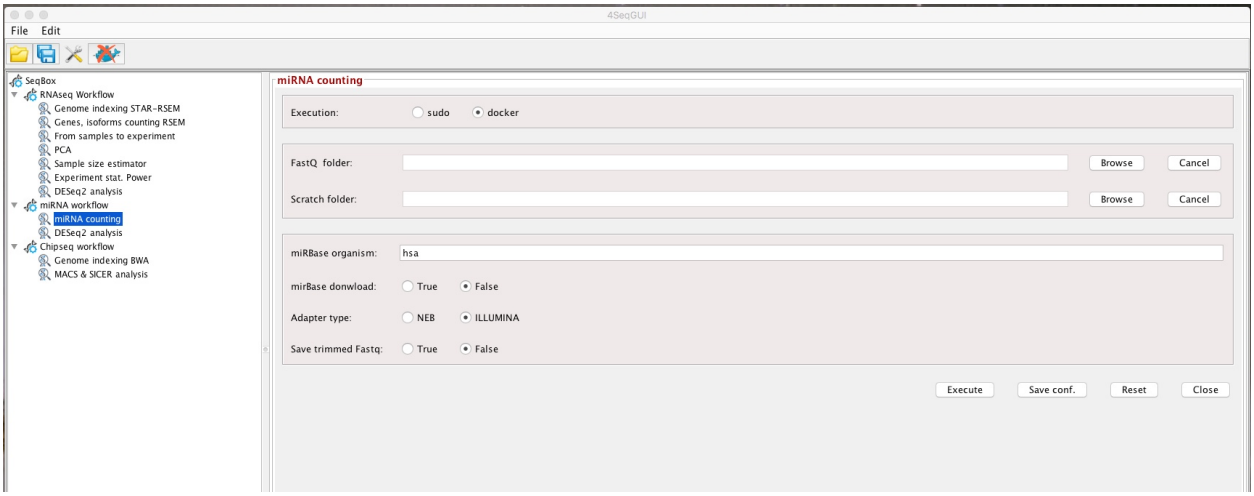

Figure 22: miRNAseq parameters

```
#test example
system("wget 130.192.119.59/public/test.mirnaCounts.zip")
unzip("test.mirnaCounts.zip")
setwd("test.mirnaCounts")
library(docker4seq)
mirnaCounts(group="docker",fastq.folder=getwd(), scratch.folder="/data/scratch",
            mirbase.id="hsa",download.status=FALSE, adapter.type="NEB", trimmed.fastq=FALSE)
```

User needs to create the **fastq.folder**, where the fastq.gz files for all miRNAs under analysis are located. The **scratch.folder** is the location where temporary data are created. The results will be then saved in the **fastq.folder**. User needs to provide also the identifier of the miRBase organism, e.g. **hsa** for Homo sapiens, **mmu** for Mus musculus. If the **download.status** is set to FALSE, mirnaCounts uses miRBase release 21, if it is set to TRUE the lastest version of precursor and mature miRNAs will be downloaded from miRBase. Users need to provide the name of the producer of the miRNA library prep kit to identify which adapters need to be provided to cutadapt, **adapter.type** parameter. The available adapters are NEB and Illumina, but, upon request, we can add other adapters. Finally, if the **trimmed.fastq** is set to FALSE the trimmed fastq are not saved at the end of the analysis.

miRNAseq workflow output files

The miRNAseq workflow produces the following output files:

```
+ README: A file describing the content of the data folder
+ all.counts.txt: miRNAs raw counts, to be used for differential expression analysis
+ trimmimg.log: adapters trimming statistics
+ shrimp.log: mapping statistics
+ all.counts.Rda: miRNAs raw counts ready to be loaded in R.
+ analysis.log: logs of the full analysis pipeline
```

Adding covariates and batches to mirnaCounts output: all.counts.txt

The funnction **mirnaCovar** add to the header of all.counts.txt covariates and batches or covariates only.

```
#test example
system("wget 130.192.119.59/public/test.mirna.analysis.zip")
unzip("test.mirna.analysis.zip")
setwd("test.mirna.analysis")
library(docker4seq)
mirnaCovar(experiment.folder=getwd(),
     covariates=c("Cov.1", "Cov.1", "Cov.1", "Cov.1", "Cov.1", "Cov.1",
                  "Cov.2", "Cov.2", "Cov.2", "Cov.2", "Cov.2", "Cov.2"),
     batches=c("bath.1", "bath.1", "bath.2", "bath.2", "batch.1", "batch.1",
               "batch.2", "batch.2","batch.1", "batch.1","bath.2", "bath.2"))
```

chipseq workflow

The chipseq workflow can be run using **4SeqGUI** graphical interface:

Figure 23: ChIPseq workflow

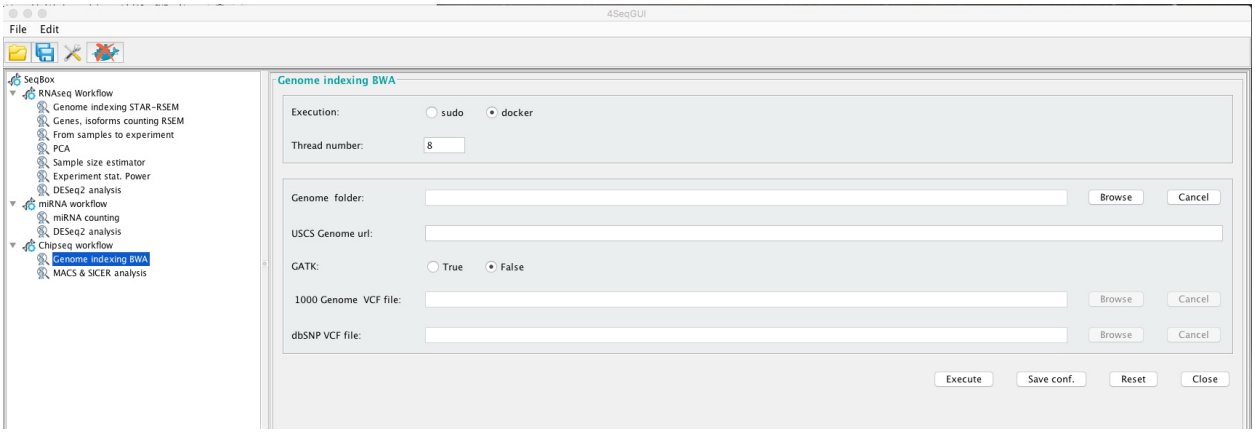

Figure 24: Creating a BWA index with Genome indexing BWA

The ChIPseq is made of two main steps:

- Creating a genome index for BWA (see end of this paragraph)
- Running MACS or SICER analysis

Creating a BWA index file for Chipseq:

The index can be easily created using the graphical interface:

```
bwaIndexUcsc(group="sudo",genome.folder="/sto2/data/scratch/mm10bwa", uscs.urlgenome=
"http://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/chromFa.tar.gz",
gatk=FALSE)
```

In brief, **bwaIndexUcsc** uses UCSC genomic data. User has to provide the URL (**uscs.urlgenome**) for the file chromFa.tar.gz related to the organism of interest and the path to the folder where the index will be generated (**genome.folder**). The parameter **gatk** has to be set to FALSE because is not used for a genomic index used for ChIPseq.

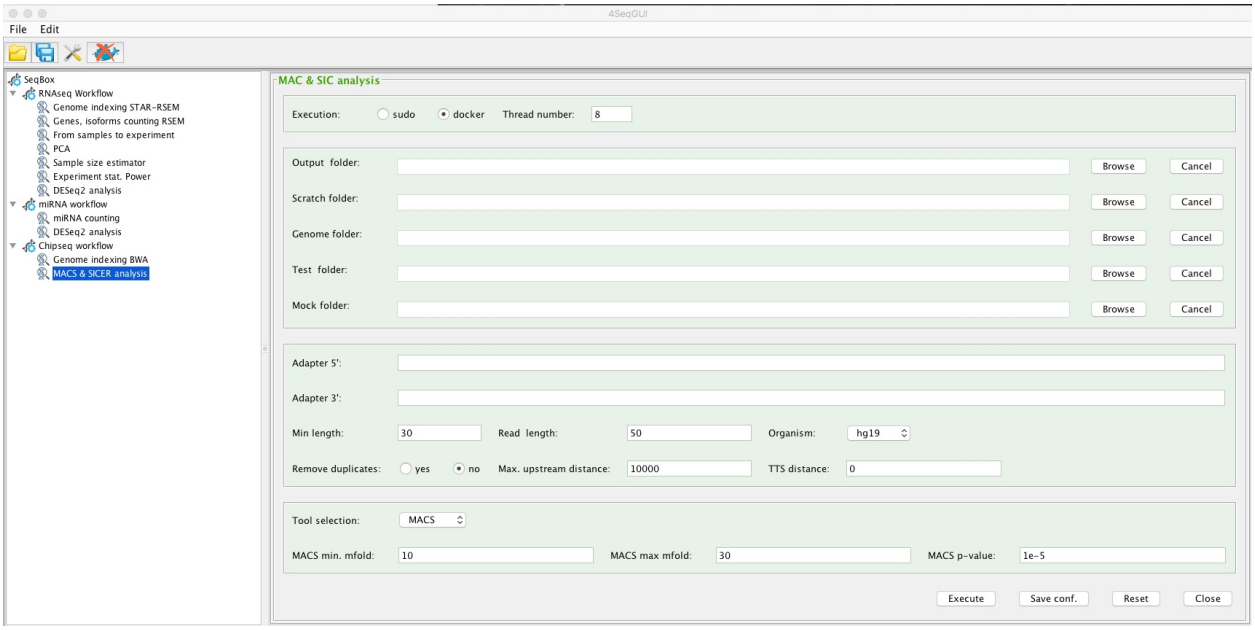

Figure 25: MACS and SICER analysis

Calling peaks and annotating:

All the parameters needed to run MACS or SICER can be setup using 4SeqGUI

A detailed description of the parameters is given below.

Chipseq workflow by line command

The chipseq workflow can be also executed using R and it is completely embedded in a unique function:

```
system("wget 130.192.119.59/public/test.chipseqCounts.zip")
unzip("test.chipseqCounts.zip")
setwd("test.chipseqCounts")
library(docker4seq)
chipseqCounts(group = "docker", output.folder = "./prdm51.igg",
  mock.folder="./igg", test.folder="./prdm51", scratch.folder=getwd(),
  adapter5 = "AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT",
  adapter3 = "AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT",
  threads = 8, min.length = 30, genome.folder,
  mock.id = "igg", test.id = "tf", genome, read.size = 50,
  tool = "macs", macs.min.mfold = 10, macs.max.mfold = 30,
  macs.pval = "1e-5", sicer.wsize = 200, sicer.gsize = 200,
  sicer.fdr = 0.1, tss.distance = 0, max.upstream.distance = 10000,
  remove.duplicates = "N")
```

Specifically user needs to create three folders:

```
+ mock.folder, where the fastq.gz file for the control sample is located. For control sample we refer to ChIP with IgG only or i
+ test.folder, where the fastq.gz file for the ChIP of the sample to be analysed.
+ output.folder, where the R script embedding the above script is located.
```

The **scratch.folder** can be the same as the **output.folder**. However, if the system in use has a high speed disk for temporary calculation, e.g. a SSD disk, the location of the scratch.folder on the SSD will reduce significantly the computing time.

User needs to provide also the sequence of the sequencing adapters, **adapter5** and **adapter3** parameters. In case Illumina platform is used the adapters sequences can be easily recovered here.

Threads indicates the max number of cores used by *skewer* and *bwa*, all the other steps are done on a single core. The **min.length** refers to the minimal length that a reads should have after adapters trimming. Since today the average read length for a ChIP experiment is 50 or 75 nts would be better to bring to 40 nts the min.length parameter to increase the precision in assigning the correct position on the genome.

The **genome.folder** parameter refers to the location of the genomic index generated by bwa using the *docker4seq* function **bwaIndexUcsc**. The generation of the genome index for ChIP experiment is very simple and it is highlited at the end of this paragraph.

mock.id and **test.id** identify the type of sample and are assigned to the ID parameter in the RG field of the bam file.

genome is the parameter referring to the annotation used to associate ChIP peaks to genes. In the present implemetation are available hg38, hg19 for human and mm10 and mm9 for mouse annotations.

read.size is a parameter requested by MACS and SICER for their analysis. **macs.min.mfold**, **macs.max.mfold**, **macs.pval** are the deafult parameters requested for peaks definition for more info please refer to the documetation of MACS 1.4. **sicer.wsize**, **sicer.gsize**, **sicer.fdr** are the deafult parameters requested for peaks definition for more info please refer to the documetation ofSICER 1.1. **Important**: The optimal value for **sicer.gsize** in case of H3K4Me3 ChIP is 200 and in case of ChIP H3K27Me3 is 600.

tss.distance and **max.upstream.distance** are parameters required by ChIPseqAnno, which is the bioconducto package used to assign the peaks to specific genes. Specifically max.upstream.distance refers to the max distance in nts that allow to associate a peak to a gene.

remove.duplicates is the parameter that indicates if duplicates need to be removed or not. It has two options: **N** duplicates are not removed, **Y** duplicates are removed.

Chipseq workflow output files

The chipseq workflow produces the following output files:

```
+ README: A file describing the content of the data folder
+ mypeaks.xls: All detected peaks alongside the nearest gene and its annotation
+ mytreat.counts: The total reads count for the provided treatment file
+ mycontrol.counts: The total reads count for the provided control/background file
+ peak_report.xls: Aggregate information regarding the peak and their position relative to the nearest gene
+ chromosome_distribution.pdf: Barplot of the distribution of the peaks on the chromosomes
+ relative_position_distribution.pdf: Barplot of the distribution of the peaks positions relative to their nearest gene
+ peak_width_distribution.pdf: Histogram of the distribution of the width of the peaks
+ distance_from_nearest_gene_distribution.pdf: Histogram of the distribution of the distance of each peak from its nearest gene
+ cumulative_coverage_total.pdf: Cumulative normalized gene coverage
+ cumulative_coverage_chrN.pdf: Cumulative normalized gene coverage for the specific chromosome
+ mycontrol_sorted.bw: bigWig file for UCSC Genome Browser visualization
+ mytreat_sorted.bw: bigWig file for UCSC Genome Browser visualization
```