General Overview

HTS-flow main folder is located in:

```
/data/BA/public_html/HTS-flow/
```

This folder contains the php files composing the web interface (completed.php, external.php, footer.php, header.php, index.php, menu.php, merging.php, primary.php, run.php, secondary.php, usage.php), as well the javascript files JSfunction.js and those in the folder TableFilter/. These files contain the functions allowing the web interface to work.

Files with extension .R contain the R functions to call the pipelines (pipeline.R, pipeFunctions.R, primary.R, secondary.R, merging.R, peakcalling.R, deletePrimaryFromHTS-flow.R).

There are several subfolders in the HTS-flow main folder, which store pipelines outputs, support files, logs files for tracing errors occurred during the analyses.

These folders are:

DB/: contains the pipeline outputs as well as the reference genomes and the MySQL schema of the database used by HTS-flow.

users/: contains user-specific files used by HTS-flow for submitting user-specific jobs via qsub.

logs/: contains the log files produced by the pipelines.

HTS-flow is based on a MySQL database called *vbianchi* that is used by the web interface and by pipelines for retrieving pieces of information about jobs submitted by users.

All data produced by the pipelines are stored in

```
/data/BA/public_html/HTS-flow/DB/
```

This folder contains several subfolders containing the output of the pipelines and third party software needed by HTS-flow for running the pipelines.

ALN/: contains the alignment and index files produced by primary.R.

COUNT/: contains the files with read counts per gene produced by primary.R. These files are generated only for RNA-Seq samples.

external_data/: this folder is intended as a repository for external data to be analysed with HTS-flow. This is a convenience folder: it is not mandatory to store external data here.

QC/: contains results from FastQC tool on alignment files stored in the ALN/ folder.

Secondary/: contains results generated from secondary.R. The subfolders are labelled with the Secondary identification number.

TMP/: contains temporary data from primary analyses if a user decides to keep intermediate processing steps produced by primary.R.

preprocess/: contains temporary data produced by primary.R. Its content is removed at the end of each run.

pathFiles/: contains the genomes/ folder with all the reference genomes used by HTS-flow, as well the contaminants_list.txt file used by FastQC tool for quality control and by fastqSorter2.py after the trimming of the reads in primary analysis, the random_line_extraction.using_ration.pl script used for downsampling alignment files and the runWellington.py script, used by footprint analysis for calling the wellington tool.

scr/: contains two files, the MySQL DB schema that can be used to recreate the database from the scratch, and CheckForUpdatesInLIMS2.py used for updating the database with new entries in the LIMS.

External tools provided with HTS-flow are the aligners bowtie (**bowtie-1.1.1/**), bismark (**bismark_v0.14.0/**) and tophat (**tophat-2.0.8/**); FastQC (**FastQC/**) for quality assessment on aligned reads, featureCounts (**subread-1.4.5-p1/**) for counting the number of reads per gene, and R statistical tool (**R-3.2.0/**)

MySQL database

Database vbianchi can be reached by the following command:

```
MySQL -h MySQL.application.ieo.eu -u vbianchi -p108be6054c2bc36d835f020a80f71a05
vbianchi
```

The main entities of this database are:

diff_gene_expression: contains information for running a Secondary Analysis DEG calling pipeline.

expression_quantification: contains information for running Secondary Analysis pipeline for expression quantification.

footprint: contains information for running a Secondary Analysis footprint pipeline.

merge: contains information for running a Merging pipeline.

pa_options: contains the set of options for running a Primary Analysis pipeline.

paths: contains paths for all the programs and files used by HTS-flow.

peak_calling: contains information for running a Secondary Analysis peak calling pipeline.

pre_analysis: contains information for running the Primary Analysis pipeline.

sample: contains the entries obtained from the LIMS database.

secondary: contains information for running Secondary Analysis pipeline.

totComp: a MySQL view for reporting all the completed analysis.

uploaded: contains the entries from external data defined by the users.

users: the list of users in HTS-flow with and without the developer status, needed for viewing some parts of the web interface.

A schema of the database is stored in:

```
/data/BA/public_html/HTS-flow/DB/scr/Dbschema.sql
```

The backup of the database has not been yet automatized.

R Pipelines

HTS-flow uses R scripts to recover information added by the users on the web interface and perform the requested jobs. These scripts are in the HTS-flow main folder and can be roughly divided in scripts for primary analysis, scripts for secondary analysis and support scripts.

Support scripts are:

pipeline.R: this is the header script that controls if a user asks for primary, secondary or merging samples analyses.

pipeFunctions.R: contains a set of functions needed by other R scripts, mainly functions for updating or retrieving information from the MySQL database.

deletePrimaryFromHTS-flow.R: contains the set of functions to remove a primary analysis job from MySQL database and the corresponding files stored in HTS-flow folders.

Primary analysis script is:

primary.R: contains the set of functions to complete a primary analysis, from unzipping fastq files to quality controls of reads before the alignment, reads alignment to reference genome and generation of big wig files.

Secondary Analysis scripts are:

secondary.R: contains the set of functions to run secondary analyses: Expression Quantification, DEG calling, Peak calling and Footprints calling.

merging.R: contains the set of functions to perform the merge of two or more BAM files.

peakcalling.R: contains the set of function to perform NARROW and BROAD peak calling.

R installation

The official release of R is available on the web page

http://cran.r-project.org/sources.html

A new installation of R must include additional packages from bioconductor repository (listed below) for annotation in peak calling and differentially expressed genes analyses as well packages for plotting heatmaps.

For installing R please follow the rules in R '**Installation and Administration**' manual following the link

http://cran.r-project.org/doc/manuals/r-release/R-admin.html

When R is correctly installed, new packages can be added. When installing new packages the library path must be provided (see example below), otherwise R will load the libraries only for the user that installed the package. In addition, this procedure stores all the installed packages in a single folder, making the entire process more clean for debugging. The R installation comes with a default library folder called 'library'. It is suggested to use that folder as path for packages.

The following code shows how to install the basic packages required by HTS-flow. Substitute X.X.X with the installed version of R.

```
# INSTALLING A NEW VERSION OF R.

# Required packages

# default packages from bioconductor

source("http://bioconductor.org/biocLite.R")

biocLite(lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')

# annotation part

biocLite('org.Mm.eg.db', lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')

biocLite('org.Hs.eg.db', lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')

biocLite('org.Dm.eg.db', lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')

biocLite('org.Rn.eg.db', lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')
biocLite('TxDb.Mmusculus.UCSC.mm9.knownGene', lib='/data/BA/public_html/HTS-
flow/DB/R-X.X.X/library')

biocLite('TxDb.Hsapiens.UCSC.hg19.knownGene', lib='/data/BA/public_html/HTS-
flow/DB/R-X.X.X/library')

biocLite('TxDb.Mmusculus.UCSC.mm10.knownGene', lib='/data/BA/public_html/HTS-
flow/DB/R-X.X.X/library')

biocLite('TxDb.Hsapiens.UCSC.hg18.knownGene', lib='/data/BA/public_html/HTS-
flow/DB/R-X.X.X/library')

biocLite('TxDb.Rnorvegicus.UCSC.rn5.refGene', lib='/data/BA/public_html/HTS-
flow/DB/R-X.X.X/library')

# compEpiTools

biocLite("compEpiTools", lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')

# DESeq2

biocLite("DESeq2", lib='/data/BA/public_html/HTS-flow/DB/R-X.X.X/library')

# Heatmaps

install.packages( "RColorBrewer", lib='/data/BA/public_html/HTS-flow/DB/R-
X.X.X/library' )

install.packages( "gplots", lib='/data/BA/public_html/HTS-flow/DB/R-
X.X.X/library' )
```

Add a user to HTS-flow

Computational scientists have the task to set the computer environment suitable for running HTS-flow to Wet Lab Scientists.

To add a user to HTS-flow there are two steps:

(1) Access the cluster opening the app Terminal (OS X and Linux users) and log with the SSH client by typing:

```
ssh your_user_name@grid.ieo.eu
```

your_user_name must be the username provided by the Service Desk; when you are prompted for a password, type your password and press Enter.

(2) To submit your jobs you have to modify the execution of crontab. crontab is a program running on Linux machines allowing the execution of a specific command every given amount of time.

In order to avoid submitting jobs to HTS-flow from the Terminal, the user must modify the user-specific crontab.

After login to the cluster, run the following command:

```
nano cronfile
```

then paste this three lines at the end of the file:

```
MAILTO=USER_EMAIL

HOME=/data/BA/public_html/HTS-flow/

* * * * * python /data/BA/public_html/HTS-flow/HTSflowSubmitter.py USER_NAME
```

Then CTRL-O to save and CTRL-X to exit. Check with command ls that your file has been correctly created.

Please remember to change USER_EMAIL with the email where you want to receive messages from crontab and USER_NAME with your campus username that you use to log in.

```
############ Example ############

User Valerio Bianchi wants to create the crontab file for submitting jobs with HTS-
flow. These are the two variables he need to change accordingly

USER_NAME: vbianchi
```

```
USER_EMAIL: vbianchi@ieo.eu

He creates the file ‘cronfile’ with nano and paste:

MAILTO=vbianchi@ieo.eu

HOME=/data/BA/public_html/HTS-flow/

* * * * * python /data/BA/public_html/HTS-flow/HTSflowSubmitter.py vbianchi

Then he saves the file.

These lines will automatically run the script HTSflowSubmitter.py, which retrieves
and submits to the cluster each job the user vbianchi is submitting with HTS-flow.

############## End ##############
```

The final step is to associate this lines to crontab. So run the following line:

```
crontab cronfile
```

It is recommended to modify the crontab execution from the head of the computer cluster, for being sure which node is running the crontab (the head in this case).

If no errors were generated, the user will now be able to submit jobs using HTS-flow web interface, otherwise none of his/her jobs will be executed. In case errors were generated, contact the administrator (arnaud.ceol@iit.it).

Primary Analysis

Submitting a Job

From this page a bioinformatician can submit a primary analysis on a sample or on a group of samples. A primary analysis consists of filtering, quality control and alignment to reference genome of the reads from a specific sample.

The reference genomes present in HTS-flow are the following:

mm9 - Mus musculus

mm10 - Mus musculus

hg18 - Homo sapiens

hg19 - Homo sapiens

rn5 - Rattus norvegicus

dm6 - Drosophila melanogaster

Sequencing technologies implemented in HTS-flow are RNA-Seq, ChIP-Seq, DNaseI-Seq, BS-Seq.

Samples submitted for primary analysis need to have one of the following label as REF_GENOME: mm9, mm10, hg18, hg19, rn5, dm6.

METHOD label accepted are the following: RNA-Seq, ChIP-Seq, DNaseI-Seq, BS-Seq.

If the samples submitted do not follow these rules, the analysis will not be completed returning an error. For this purpose the user can change the REF_GENOME and METHOD labels using the appropriate “Edit” button on the web-interface.

After the selection of sample(s) the user can proceed to select the set of options for the primary analysis pressing the button ‘PRIMARY ANALYSIS SETTINGS’:

Remove Bad Reads: default TRUE - removal of reads that has been labelled bad from the sequencer (grep -A 3 '^@.* [^:]*:N:[^:]*:' | grep -v -- '^--$' | sed 's/[0-9]:N:[0-9]*:[A-Z]*$//g').

Trimming: default FALSE - if TRUE trim the reads starting at 5' if nucleotide quality Q is below 20. Phred quality scores Q are defined as a property which is logarithmically related to the base-calling error probabilities P. To be used if exists the possibility of high degradation of quality at 5' ends of reads.

Masking: default TRUE - if TRUE will mask nucleotides along the whole reads with N if their Q quality score is below 20.

Alignment: default TRUE - if TRUE will align the reads to the reference genome.

Program: tophat/bwa/bismarck - tophat is used for aligning RNA-Seq reads, bwa for ChIP-Seq and DNaseI-Seq, bismarck for BS-Seq.

Alignment Options: this line is intended for changing the options provided to the aligners.

Paired: TRUE/FALSE - if TRUE will be used for both tophat and bwa the set of options that treats paired-end reads.

Remove Temp Files: default TRUE - removes the temporary files generated during the filtering and alignment processes.

Remove Duplicates: default TRUE - if TRUE the final alignment file is processed for removing PCR duplicates (reads that align on the same genomic location).

This job can be submitted pressing the button ‘SUBMIT PRIMARY ANALYSIS’. This action will automatically redirect the web-browser to running page that will show the list of jobs now yet completed.

After the completion of job it will be visible on completed page.

Search for a sample

The search for samples in this page can be accomplished using both the ‘Refine your search’ tool and/or the input fields present at the top of the samples table.

The former is used to specify Sequencing method, User ID and Reference Genome of the samples to be viewed in the table, while the latter is used to fill with the query keys the separate fields in order to subset the table.

External Data

To load external data into HTS-flow, the user must store FASTQ file(s) on a specific location on the cluster (see Preparing the FASTQ file(s) below) and load it into the web interface (see Loading samples through the web-interface below). HTS-flow does not accept BAM files as external data. If the starting file is in BAM format, it must first be converted in FASTQ format (for example, using bamtools available here https://github.com/pezmaster31/bamtools).

Preparing the FASTQ file(s)

Each sample must be stored in HTS-flow's external_data folder:

```
/data/BA/public_html/HTS-flow/DB/external_data/
```

In this folder, the user must create a subfolder with his/her name (e.g. vbianchi) and, within this, a subfolder for each sample he/she wants to analyse in HTS-flow (e.g. vbianchi/Myc).

For each sample, FASTQ file(s) must be zipped and copied to the corresponding folders. The name of the FASTQ file(s) must respect the following pattern: sample_name_R1.extension (single end)

or

sample_name_R1.extension and sample_name_R2.extension (paired end).

```
############ Example ############

User vbianchi wants to load a sample on HTS-flow consisting in an single-end RNA-
Seq file named ‘sample1_wt_rep0’, currently stored in /home/vbianchi/ folder. This
sample has to be aligned to reference genome mm9.

The FASTQ files must first be renamed ‘sample1_wt_rep0_R1.fq’ and then compressed
with gzip:

gzip  sample1_wt_rep0_R1.fq.gz

to obtain ‘sample1_wt_rep0_R1.fq.gz’.

Then the folder for the sample in HTS-flow must be created:

mkdir -p /data/BA/public_html/HTS-flow/DB/external_data/vbianchi/sample1_wt_rep0/

and the compressed and renamed FASTQ file must be copied to the new folder:

mv /home/vbianchi/sample1_wt_rep0_R1.gz /data/BA/public_html/HTS-
flow/DB/external_data/vbianchi/sample1_wt_rep0/
```

```
############## End ##############
```

Loading samples through the web interface

Go with your genome browser to the web interface of HTS-flow:

```
http://www.bioinfo.ieo.eu/BAgroup/HTS-flow/external.php
```

In the panel 'UPLOAD EXTERNAL DATA', the user must fill the form, which contains the following fields:

Sequencing Method: mandatory: one among RNA-Seq, ChIP-Seq, DNaseI-Seq, BS-Seq

Read Length: optional: length of the reads

Read Type: mandatory: either Single End or Pair End

Reference Genome: mandatory: one among mm9, mm10, hg18, hg19, rn5, dm6

Sample(s): mandatory: sample names, one per line, matching the order used in Path(s) field.

Path(s): mandatory: path for the folder(s) corresponding to each sample, one per line, matching the order used in Sample(s) field.

When all the mandatory fields are filled, the sample(s) can be submitted to HTS-flow using the 'SUBMIT' button. The page will reload and the new samples will appear at the top of the table showing all the external available samples.

```
############ Example ############

User vbianchi has created a compressed FASTQ file containing single-end reads from
an RNA-Seq experiment that must be aligned to reference genome mm9.

This FASTQ file has been renamed as:

sample1_wt_rep0_R1.gz

and stored in:

/data/BA/public_html/HTS-flow/DB/external_data/vbianchi/sample1_wt_rep0/

To load this file through the web-interface the user vbianchi must fill the 'UPLOAD
EXTERNAL DATA' form:

Sequencing Method: RNA-Seq

Reads Length: 0 (optional)

Reads Type: Single End
```

```
Reference Genome: mm9

Sample(s): sample1_wt_rep0

Path(s):/data/BA/public_html/HTS-flow/DB/external_data/vbianchi/sample1_wt_rep0/

############## End ##############
```

Submitting a primary analysis on external data

After selecting the desired samples from the table, the user must press the “primary analysis setting” and use the bottom panel to select the desired parameters (see PRIMARY ANALYSIS).

Check analyses from shell

Log files

Every analysis submitted through the web interface will create a shell command line into a user-specific file whose name correspond to the user name, inside the users/ folder.

The PHP page run.php retrieves the information selected by users both for primary and secondary analyses and creates different shell files with extension ".sh" containing the command line script that has to be executed for running the appropriate pipeline. The name of the shell files depend on the kind of analysis HTS-flow has to perform. In addition, a command line containing the absolute path of each sh file is written inside a user-specific file in users/ folder.

run.php in running Primary analysis will generate an sh file with name

run_PRIMARYID_primary.sh

where PRIMARYID is the primary ID provided by HTS-flow to the analysis, which will contains the following lines:

```
#! /bin/bash

#$ -S /bin/sh

#$ -cwd

#$ -l h_stack=128m

#$ -l h_vmem=24g

/data/BA/public_html/HTS-flow/DB/R-X.X.X/bin/Rscript /data/BA/public_html/HTS-
flow/pipeline.R PRIMARYID primary > /data/BA/public_html/HTS-
flow/PRIMARYID_primary.out
```

Then run.php writes the command line for submitting the job (qsub) on a file in users/ folder that has the name of the user submitting the job.

The command line script contained in the sh file redirects also the output of the R script to a file called PRIMARYID_primary.out, which will contain then all the message() lines in the R script.

Suppose that user **vbianchi** is submitting a primary analysis from the web interface with primary ID **1000**; run.php will generate an sh file in HTS-flow main folder called **run_1000_primary.sh** containing the command line script to run the primary analysis in R. At the same time, it will create in the users/ folder a file called **vbianchi** with a single line:

```
qsub /data/BA/public_html/HTS-flow/run_1000_primary.sh
```

The python script HTSflowSubmitter.py checks every minute if something has been written on the user file vbianchi. This script has been put in crontab during the user creation as explained in chapter '**Add a user to HTS-flow**' .

After running the command(s) found in the user-specific file, HTSflowSubmitter.py renames the file to avoid to run the same commands multiple times. The file is renamed in the form

```
USER_lastDone
```

where USER is the user name.

This strategy allows to check the last command executed by a user.

The same applies for the others pipelines executed by HTS-flow.

The name of the sh files changes accordingly to the pipeline:

Primary Analysis: run_PRIMARYID_primary.sh

Secondary Analysis: run_SECONDARYID_secondary.sh

Merging Analysis: run_MERGINGID_merging.sh

Peak Calling: run_PEAKID_peak.sh

where PRIMARYID, SECONDARYID, MERGINGID and PEAKID are respectively primary analysis ID, secondary analysis ID, merging analysis ID and peak calling ID provided by HTS-flow.

Each sh file creates two files, collecting the standard error and standard output. All the print() lines from an R script are redirected in the standard error file. They can be easily recognized because have the same name of the sh file followed by '.eXXXXXX' where XXXXXX is the job ID assigned by the cluster for that particular run.

When a pipeline is finished, HTS-flow moves the sh files, the standard error/output file and the out file in folder **logs/**.

In case a user selects to keep temporary files, these files are moved to **DB/TMP/** folder.

Rerun a Job

Before rerunning a job, you have to follow these rules:

1a – if the job is a primary analysis or a merging analysis, you have to delete the BAM file from **DB/ALN/** folder because HTS-flow will check for the existence of the BAM file.

1b – if the job is a secondary analysis, you have to delete the folder created in **DB/secondary/** followed by the secondary analysis ID.

2 – remove the sh file from the main folder if job has not reached the end.

3 – create the sh file from scratch using the appropriate python scripts:

Primary analysis

```
python creaSHprimary.py PRIMARYID
```

Secondary Analysis

```
python creaSHsecondary.py SECONDARYID RAM
```

Merging Analysis

```
python creaSHmerging.py MERGINGID
```

4 – You can either qsub the sh file generated or create the user specific file in **users/** folder filling the file with the qsub line and wait HTSflowSubmitter.py which will run the command line. The user specific file has to be named with the same user's user name.