NEAT for ChIPseq

NGS pipelines made easy

Patrick Schorderet

Patrick.schorderet@molbio.mgh.harvard.edu

Jan 2015

1 Introduction

NExt generation ***A***nalysis ***T***oolbox (NEAT) is a perl/R package that supports users during the analysis of next generation sequencing (NGS)

NEAT provides an easy, rapid and reproducible exploratory data analysis (EDA) tool that allows users to assess their data in less than 24 hours (based on a 200mio read Highseq run). NEAT was developed in two main sections, ChIPpip (RNApip) and ChIPmE (RNAmE). The first section (ChIPpip) helps users with demanding computationally (mapping, filtering, etc). The second section (ChIPmE) consists of two standalone applications that provide users with human readable data.

2 Getting started with UNIX

2.1 Notes

In the following tutorial, all unix/perl/R command lines will be bold, italicized and highlighted in blue. Most will be embedded in tables. The command line is the text following, but not including, the dollar sign ($).

---

*[~]$ **this is a unix command***

This tutorial is intended to run on MacOS/LINUX environments. For MacOS users, we suggest to use the *Terminal* (applications/Terminal) for all following steps. The terminal/shell output will be depicted in black.

Copy pasting the *unix* commands should allow you to follow all steps of this tutorial. Please be aware that *unix* commands are case sensitive, including white spaces.

2.2 Remote servers and SSH

Secure Shell (SSH) is a cryptographic network protocol for secure data communication. In brief, it is a way for users to access remote computers (and their content) using a secure channel (a tunnel) through an insecure network (the internet). To access your computer cluster, you will need to establish an SSH connection. In analogy to an access card to your building, each user should be provided with an SSH username and password. Finally, the last essential parameter to access your computer cluster is the virtual *address* of the server. However, before accessing the remote server, you will need to copy the ChIPpip directory from your local computer to the remote server.

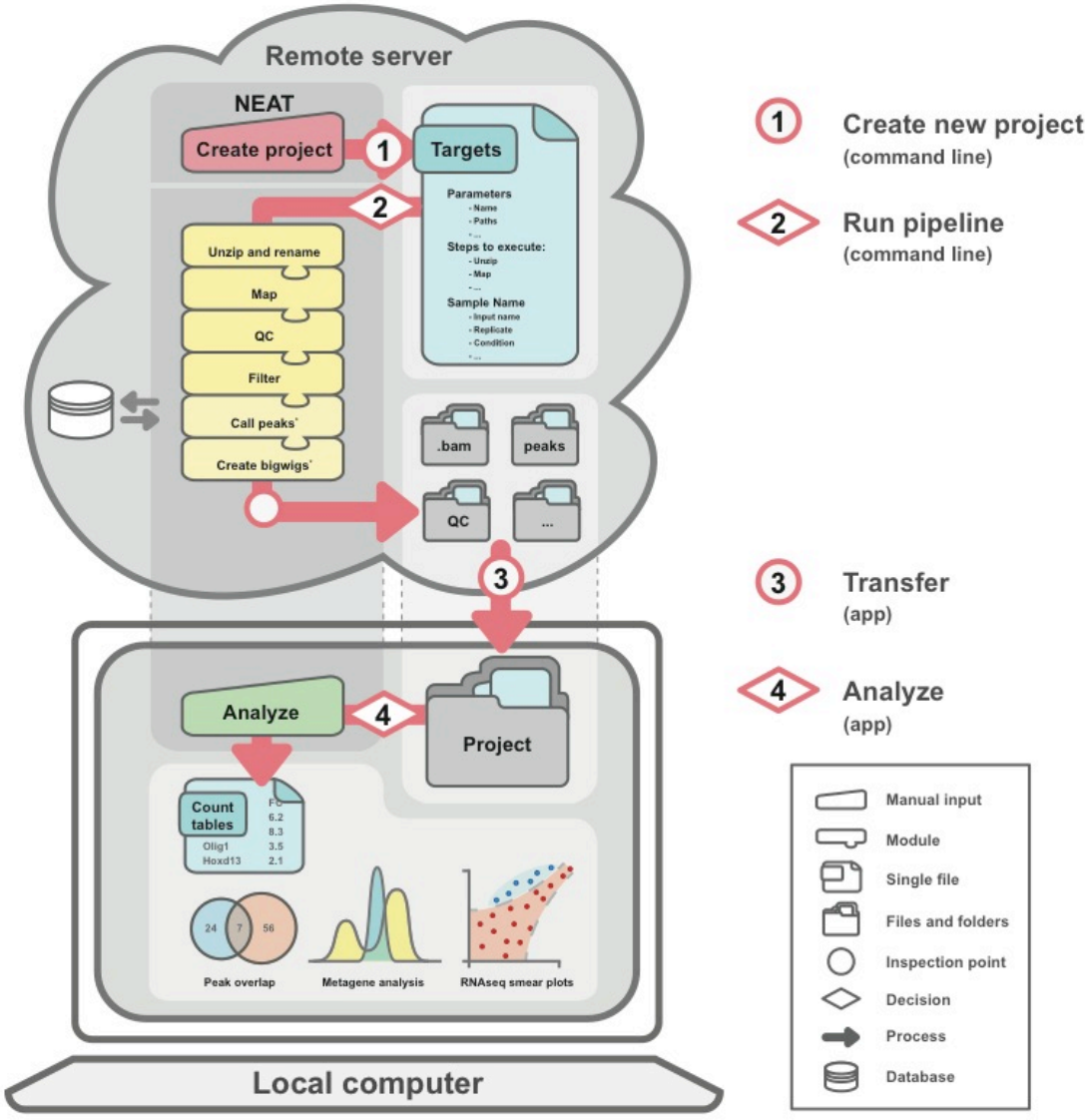


Fig.1 NEAT architecture. NGS data can be analyzed using NEAT in less than a day. Users follow a logical 4-step process, including the creation of a new project, running the pipeline on a remote server or in the cloud, transferring the data to a local computer and proceeding to the analysis.

3 NEAT

3.1 Introduction to NEAT

A central feature of NEAT is its ability to perform repetitive tasks on complex sample setups while managing batch submissions and cluster queuing. NEAT can easily be implemented in any institution with limited to no programming experience. The workflow has been designed to efficiently run on a computer cluster using a distributed resource manager such as TORQUE (qsub commands) or LSF (bsub commands). NEAT has been developed by and for wet-lab scientists as well as bioinformaticiens to ensure user-friendliness, management of complicated experimental setups and reproducibility in the big data era.

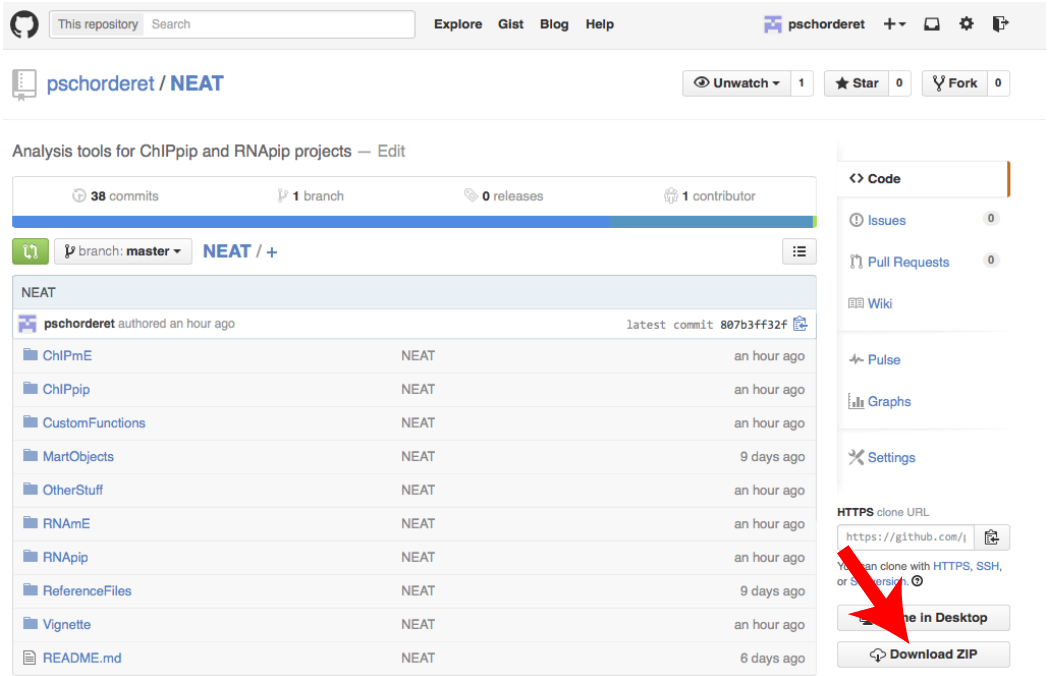
To start using NEAT for ChIPseq data analysis, please follow the tutorial. This will walk you through the analysis of a small test dataset (provided as part of ChIPpip) using your own computer cluster. This will also ensure NEAT and its dependencies are correctly installed before submitting large, memory-savvy analysis.

All fastq files from the test data have been subsetted to ca. 15'000 reads. This data comes from an unpublished 50bp single end (SE) sequencing experiment although NEAT can deal with paired-end (PE) sequencing as well. For more information on the test data provided in this tutorial, please read below.

Although NEAT can be run by scientists with limited to no programming experience, this tutorial requires access to a remote server. Users are thereby required to have SSH accessibility with a username and a password. Please refer to your system administrator to obtain such credentials. For more information on how to access a remote server through SSH, please read below.

3.2 Install NEAT

To install NEAT, download the NEAT repository from GitHub to any directory on your computer cluster.



In this tutorial, we will suppose the *NEAT/* directory was saved to the user's desktop on a local computer (*~/Desktop*) and that it will be saved to their home directory (*/home/*) on the remote server.

Make sure the folder is named *NEAT* and not *NEAT-master*. Start the transfer to the remote folder by typing the following command line in a terminal window:

```
[~Desktop]$ rsync -avz ~/Desktop/NEAT username@serveradress.edu:/home/
```

Enter your password and allow for the folders to transfer.

An alternative for copying files and folders from your local computer to a remote server are free softwares such as *FileZilla* or *Cyberduck*.

Finally, to install the required R packages, copy paste the content of R_packages.txt into an R terminal (how to download and install R).

Once it has finished, the NEAT directory should be saved to your remote server. Check this by accessing your remote server. In a *Terminal* window, type the following:

[MY_COMPUTE~]$ ***ssh username@serveradress.edu***
password
[username@setrveradress ~]$

These commands bring you to your <HOME> directory on the remote server. If you are not sure of your current working directory, type ***pwd*** to *print* your current *working directory*. Navigate to the section 1 (ChIPpip) directory of NEAT using the ***cd*** and list files using the ***ls*** command. If NEAT was properly copied to your <HOME> directory, you should see the following:

[username@setrveradress ~]$ ***cd /home/NEAT/ChIPpip***
[username@setrveradress ChIPpip]$ ***ls -l***

```
 3405   ChIPpipCreateNewProject.pl
12170   ChIPpipRunPipeline.pl
 1933   README.md
 4096   scripts
```

Note here that *./* is a short cut to your working directory. In this tutorial, because we have set our working directory to */home/NEAT/ChIPpip/* by using the command ***cd /home/NEAT/ChIPpip/***, *'./'* will be a short cut for */home/NEAT/ChIPpip/* and these can and will be used interchangeably.

3.3 Test data

3.3.1 Structure and significance of test data

The unpublished test data provided for this tutorial is part of the ChIPpip folder. If you have followed the previous steps, the compressed fastq files required for the tutorial can be found in the test data folder */home/NEAT/ChIPpip/scripts/testdata/*.

[ChIPpip]:$ ***ls -l ./scripts/testdata***

```
922725    PSa36-1_R1.fastq.gz
927057    PSa36-2_R1.fastq.gz
886915    PSa37-3_R1.fastq.gz
875985    PSa37-4_R1.fastq.gz
```

The data consists of a 50 base pair, single end ChIPseq experiment on a histone mark (K4me3) in two growing conditions (noDox, Dox) with their corresponding inputs. No replicates were generated.

4 NEAT Part 1 : ChIPpip

4.1 Running NEAT part 1.1 (ChIPpip)

4.1.1 Creating a new ChIPpip project

The first step to run NEAT on ChIPseq data is to create a new ChIPpip project. Navigate to the ChIPpip directory and run the *ChIPpipCreateNewProject.pl* script. We need to add ***perl*** in front of this command to tell the computer it should deal with this file as a perl script. This script requires the user to specify the path where the new ChIPpip project will be created including the name of the new project. In this example, we will create a project in the ChIPpip directory named EXAMPLE.

```
[ChIPpip]:$ perl ChIPpipCreateNewProject.pl ~/NEAT/ChIPpip/EXAMPLE

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-

Creating a new ChIPpip project

--------------------------------------------------------

New ChIPpip project has been built as follows:

        /home/ChIPpip/EXAMPLE
                └─ DataStructure
                        └─ Targets.txt
                └─ fastq
                └─ scripts

--------------------------------------------------------------------------------------------------

IMPORTANT:    Fill in the Targets.txt file before running ChIPpipRunPipeline.pl
              To modify Targets.txt, copy paste the following command in your terminal:

              nano /home/ChIPpip/EXAMPLE/DataStructure/Targets.txt

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
```

Running the *ChIPpipCreateNewProject.pl* script should prompt the message above. If not, please troubleshoot before proceeding to the next step.

4.1.2 The Targets.txt file structure

4.1.2.1 Filling in the Targets.txt file

The Targets.txt file found in the */home/EXAMPLE/DataStructure/* directory is the backbone of ChIPpip. It contains all the information specific to your experiment and your computer cluster, including the names of files, the paths to the reference genomes, the steps to execute, the name of your samples, their relationships, etc.

IMPORTANT: Sample names including inputs and fastq files cannot start with an 'n' (small or capital) as this is the universal perl symbol for carriage return.

The *Targets.txt* file is the most important piece of NEAT and users are expected to invest the time to ensure all paths and parameters exist and are correctly set. However, once set, most of these parameters will not be changed on a specific computer cluster (users from a same institute will use the same paths). Therefor, we suggest modifying the *original* Targets.txt template file (see below).

All parameters of the Targets file should be self-explanatory. Here is a brief summary:

My_personal_email	:	If users would like to be notified by emailed when the cluster has finished. This will only work if your computer cluster has activated the emailing feature (please check with system administrator). To ensure servers are not overwhelmed by email services, ChIPpip is configured in such a way as to notify users only if the pipeline has terminated properly (with no error). Users may change this parameter by modifying the QSUB_header.sh template file found in *./ChIPpip/scripts/*.
My_personal_ssh	:	SSH parameters. Refer to your computer core facility
My_project_title	:	This is the name of the folder of your project on the remote server [automatically generated by ChIPpipCreateNewProject].

Reference_genome	:	The genome your data will be aligned to. Make sure your core faility has this genome reference installed on your cluster and that the extensions of the files are '.fa'. In addition, a .fai file will be required for the filtering step.
Local_path_to_proj	:	[Automatically generated by ChIPpipCreateNewProject].
Local_path_to_NEAT	:	[Automatically generated by ChIPpipCreateNewProject].
Proj_TaxonDatabase	:	See NEAT for RNAseq.
Proj_TaxonDatabaseDic	:	See NEAT for RNAseq.
Proj_TaxonBSgenome	:	See NEAT for RNAseq
Remote_path_to_proj	:	Full path to your project folder (without the project name) [automatically generated].
Remote_path_to_NEAT	:	Full path to your NEAT folder. Note that in our example, we have created our project within the ChIPpip folder itself, but users can freely decide to create a dedicated folder for all of their ChIPpip projects.
Remote_path_to_orifastq_gz		Full path to where your .fastq.gz files are. Usually, your sequencing core facility will let you know where they store these files. Note that all .fastq.gz files can be kept in a single location, they do not need to be copied to your folder.
Remote_path_to_chrLens.dat		Path to a .dat file containing chromosome information for your reference genome. Refer to your computer core facility.
Remote_path_to_RefGen.fasta		Path to folder containing your reference genome files. Refer to your computer core facility
Aligner_algo_short	:	"BWA" for standard alignment. If other algorithms are used, modify the *AdvancedSettings.txt* file accordingly.
Paired_end_seq_run	:	"0" for single end sequencing. "1" for paired end sequencing.
Remove_from_bigwig	:	Many softwares are incapable to load tracks because they countain unrecognized lines. For the mouse mm9 genome, these correspond to lines starting with 'random' and 'chrM'. You can easily find these for your preferred genome by attempting to load them to a genome browser, which will tell you which lines are 'unrecognized'.
Aligner_algo_short	:	The R script that will be used to call peaks. Several R scripts for peakcalling are provided *de facto* (see below).
Steps_to_execute_pipe	:	Users can choose from the following tasks: *unzip*, *qc*, *map*, *filter*, *peakcalling*, *cleanbigwig* and *granges*. If you do not want to run all of these, simply delete them for the Targets.txt or rename them. Once ran, ChIPpip will change the value of these from '*unzip*' to '*unzip_DONE*'. Obviously, a certain hierarchy has to be followed, e.g. attempting to filter reads without having previously

mapped them (in the same run or in a previous run) will not work. Note that '*qc*' requires Thomas Girke's systemPipeR package; map requires bwa; the default *peacalling* requires the R package SPP; *filter* and *cleanbigwig* require samtools. Refer below for exact requirements.

Please modify the *Targets.txt* file according to your needs. The paths to the reference genomes should be obtained from your computer core facility (system administrator), as they are the ones usually maintaining these up to date. Note that the reference genome files should have an '*.fa*' extension (e.g. mm9.fa). Please make sure that your core has named these files accordingly as any other extension will lead the pipeline to abort prematurely. To modify the Targets.txt file, we suggest users get accustomed to using a terminal text editor such as ***vi*** or ***nano*** as it will avoid including spaces and special characters.

Fill in your *Targets.txt* fill using the following command:

```
[ChIPpip]:$  nano ./EXAMPLE/DataStructure/Targets.txt

#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
#                                                                                     #
#                    ChIPseq                                                          #
#                                                                                     #
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
#
#
# My project Title        =       "<PROJECT_NAME>"
# Date of creation        =       "2015-03-06"
#
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-|
#
#-----------------------------------------------
#---------------- GENERAL INFO -----------
#
# My_personal_email       =       "<my.email@harvard.edu>"
# My_personal_ssh         =       "-"
#
# My_project_title        =       "<PROJECT_NAME>"
# Reference_genome        =       "mm9"
# Reference_genome_rx     =       "dm9"
#
#-----------------------------------------------
#----------------- LOCAL INFO ------------
#
# Local_path_to_proj              =       "<LOCAL_PATH_TO_PROJECT>"
# Local_path_to_NEAT              =       "/Users/patrick/Desktop/NEAT"
# Proj_TaxonDatabase              =       "TxDb.Mmusculus.UCSC.mm9.knownGene"
# Proj_TaxonDatabaseDic           =       "org.Mm.eg.db"
# Proj_TaxonBSgenome              =       "BSgenome.Mmusculus.UCSC.mm9"
#
```

```
#
#-------------------------------------------------
#---------------- REMOTE INFO -----------
#
# Remote_path_to_proj                   =       "/data/schorderet/NEAT/ChIPpip"
# Remote_path_to_NEAT                   =       "/data/schorderet/NEAT/"
#
# Remote_path_to_orifastq_gz            =       "/data/schorderet/NEAT/ChIPpip/scripts/testdata"
# Remote_path_to_chrLens_dat            =       "/data/schorderet/reference_files/mm9/chr_lens.dat"
# Remote_path_to_RefGen_fasta           =       "/data/ref/mm9/bwa/mm9.fa"
#
# Remote_path_to_chrLens_dat_ChIP_rx    =       "/data/schorderet/reference_files/dm3/chr_lens.dat"
# Remote_path_to_RefGen_fasta_ChIP_rx   =       "/data/ref/dm3/bwa/dm3.fa"
#
#-------------------------------------------------
#---------------- PARAMETERS -----------
#
# Aligner_algo_short    =       "BWA"
# Paired_end_seq_run    =       "0"
# Remove_from_bigwig    =       "random, chrM"
# PeakCaller_R_script   =       "PeakCaller_SPP.R"
#
# Steps_to_execute_pipe =       "unzip + qc + chiprx + map + filter + peakcalling + cleanbigwig + granges"
#
#
#
#-------------------------------------------------
#---------------- SAMPLES INFO -----------
#
OriFileName     FileName                OriInpName      InpName             Factor  Replicate  FileShort  Experiment  Date
PSa36-1_R1      PSa36-1_noDox_K4me3     PSa37-3_R1      PSa37-3_noDox_Inp   K4me3       1       noDox      1       2015-01-01
PSa36-2_R1      PSa36-2_Dox_K4me3       PSa37-4_R1      PSa37-4_Dox_Inp     K4me3       2       Dox        1       2015-01-01
```

Note here that *#Remote_path_to_RefGen.fasta* refers to the path to the file *<reference_genome>.fa* and not to the folder in which '*.fa*' files are located. This is different than in RNApip.

Once all information has been modified, hit ***ctrl x*** to save the file. Confirm by hitting the ***y*** and ***enter***. This will save all changes to the file.

To avoid repeating these steps at each new NEAT (ChIPpip) project creation, we suggest you modify the *original* Targets.txt file that is used as template when creating a new ChIPpip project. Modify it using the same approach as above:

```
[ChIPpip]:$ nano ./scripts/NewChIPpipProject/DataStructure/Targets.txt
```

4.2 Running NEAT Part 1.2 (ChIPpip)

Once the *Targets.txt* file is correctly set up, the *ChIPpipRunPipeline.pl* script can be run. This script will execute the tasks specified in the *Targets.txt* file. Users can choose to perform the following tasks: *unzip, qc, map, filter, peakcalling* and *cleanbigwig*. If one or several tasks should not be run, simply discard them from the Targets.txt file under *# Steps to execute*.

As does the *ChIPpipCreateNewProject.pl*, the *ChIPpipRunPipeline.pl* script requires the user to specify the path to the ChIPpip project folder (users will obviously feed the same path to both scripts). In our example, the path is *./EXAMPLE*.

```
[ChIPpip]:$ perl ChIPpipRunPipeline.pl ~/NEAT/ChIPpip/EXAMPLE

#########################################################################
#                                                                       #
#                        ChIPseq pipeline v1.0.1 (Jan 2015)             #
#                                                                       #
#########################################################################

My email:               your.email@harvard.edu

expFolder:              EXAMPLE
genome:                 mm9
userFolder:             /home/NEAT/ChIPpip
path2ChIPpip:           /home/NEAT/ChIPpip
path2expFolder:         /home/NEAT/ChIPpip/EXAMPLE
path2fastq.gz:          /home/NEAT/ChIPpip/EXAMPLE/scripts/testdata/
Targets:                /home/NEAT/ChIPpip/EXAMPLE/DataStructure/Targets.txt
chrlens:                /.../reference_files/mm9/chr_lens.dat
refGenome:              /.../mm9.fa

Paired end sequencing:      0
Aligner algorithm:          BWA
Align command 1:            bwa aln
Align command 2:            bwa samse

Remove pcr dupl:            1
Make unique reads:          1
PeakCaller.fdr:             0.01
```

```
.....
Performing following tasks:
.....
unzip:                  TRUE
map:                    TRUE
qc:                     TRUE
filter:                 TRUE
peakcalling:            TRUE
cleanbigwig:            TRUE        (remove: random chrM)
granges                 TRUE
.....
Samples:
        PSa36-1_noDox_K4me3     -       PSa37-3_noDox_Inp
        PSa36-2_Dox_K4me3       -       PSa37-4_Dox_Inp

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Exiting INITIAL section with no known error

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

This will launch the pipeline and will prompt a summary of the user's parameters. ChIPpip automatically manages all creations and batch submissions of jobs, dependencies, ordering of files, queuing, etc. If the cluster is using TORQUE, the processes can be followed by the ***qstat*** command (type ***qstat*** in your terminal). Briefly, *Q* stands for queuing, *R* for running, *E* for exiting and *H* for holding.

From this step on, the user will NOT need to intervene further. The pipeline is completely automated.

Once the pipeline has finished, it will notify the user of its status by email. The first step is to check whether everything ran smoothly. To this end, please open the Targets.txt file and check whether all jobs are marked as *DONE* under the *# Steps to execute* tag. If not, please follow the 'Troubleshooting' section below.

```
[ChIPpip]:$ less ./EXAMPLE/DataStructure/Targets.txt
```

The mock data provided as a test example should take no more than one hour to run, usually a lot less.

Note: There is a reported bug for running samtools on some cluster. Please refer below to the '*Reported bugs*' section to fix the '*cleanbigwig*' jobs.

4.2.1 ChIPpip workflow

4.2.1.1 Unzipping and renaming fastq files

Using the Targets.txt file, ChIPpip will unzip fastq.gz files found under '*OriFileName*' /'*OriInpName*' and will rename them to names found under '*FileName*'/'*InpName*'. ChIPpip will use the virtual path to the compressed files and will save the unzipped fastq files in the project folder. This minimizes file transfer and ensures all original fastq files can be kept in a central directory.

4.2.1.2 Quality control (QC)

Quality control of fastq files uses the elegant systemPipeR package developed by Thomas Girke (Girke, 2014).

4.2.1.3 Mapping and filtering reads

ChIPpip utilizes BWA (Li, 2013; Li and Durbin, 2009), a well-established and commonly accepted algorithm for ChIPseq data. The most common parameters such max edit distance (corresponds to the –n parameter), remove pcr duplications and make unique reads as well as the command line for bwa can be changed in the *AdvancedSettings.txt* file (*./EXAMPLE/DataStructure/AdvancedSettings.txt*).

Filtering is done using the samtools package. Users can enforce size of fragments, minimum and maximum size and split by chromosome in the AdvancedSettings.txt file.

4.2.1.4 Peakcalling

ChIPpip is based on the SPP peak caller algorithm (Kharchenko et al. 2008) and its architecture. Users can specify several options including the false discovery rate (*PeakCaller.fdr*), the position option (*PeakCaller.posopt*) and the density option (*PeakCaller.densityopt*) in the AdvancedSettings.txt file.

However, users can customize this step by simply wrapping their favorite peakcaller into the PeakCaller.R code. Note here that PeakCaller.R will be called for each pair of sample/input.

4.2.1.5 Cleanbigiwg

The cleanbigwig will sort out different files and folders and will transform the wiggle files (wig), which are often relatively large and cumbersome, to bigwig files, a format widely accepted by third party software. In addition, users can specify the sequences that are unrecognized by some of the aforementioned programs. Refer to the *Targets.txt* file description for more information.

4.2.1.6 Granges

The granges module will produces .GRanges.RData files, which will be used in downstream analysis. In addition, this module will produce the wig files that can be opened and visualized in genome browsers such as IGV (for more details including bin size, please refer to the *Advanced.txt* file).

4.3 Outputs

4.3.1 Architecture

ChIPpip will generate many file of which the majority will not be used in further analysis. We should note that the aligned, filter .bam files are stored in

./EXAMPLE/aligned/<sample>/<sample>.bam. Various files including peaks and bigwigs can be found in *./EXAMPLE/peakcalling/*. Quality reports (if applicable) are found in *./EXAMPLE/QC/*.

4.4 Next steps: ChIPmE

Although some users may prefer to take over the analysis from this step, we suggest using ChIPmE. ChIPmE is part of the NEAT toolkit and has been developed as a downstream module for ChIPpip. ChIPmE accompanies users from a ChIPpip output to metagene analysis in as few as two double clicks. It automatically transfers files from remote server to local computer to create wet-lab scientist readable data including pdf graphs of metagene analysis (enrichments over features), Venn diagrams (overlap of peaks) and count tables.

4.5 Troubleshooting

4.5.1 Output and error files

ChIPpip is broken down into distinct job sections. For example, all files corresponding to the 'map' section can be found in the scripts folder (*~/ChIPpip/EXAMPLE/scripts/map/*). In each job folder, the qsub directory contains all output (.o.jobID) and error (.e.jobID) files for individual jobs, which makes it easy to troubleshoot any possible errors. Files are named as follows:

```
[ChIPpip]:$ ls -l ./EXAMPLE/scripts/map/
1018    map.sh
 759    PSa36-1_noDox_K4me3_map.sh
 749    PSa36-1_noDox_K4me3_map.sh
  ...
```

```
        4096    qsub
```

[ChIPpip]:$ ***ls –l ./EXAMPLE/scripts/map/qsub***

```
           0    Iterate_map.o<jobID>
           0    Iterate_map.e<jobID>
         354    PSa36-1_noDox_K4me3_map.sh.e<jobID>
           0    PSa36-1_noDox_K4me3_map.sh.o<jobID>
         ...
```

Most .o.jobID and .e.jobID qsub files should be empty. Exceptions to this are the map.e.jobID and the filter.e.jobID files, which contain terminal outputs. Most of these can be disregarded.

In the scenario where ChIPpip cannot proceed to all jobs, it will stop. Users can modify the email settings in the QSUB_header.sh to be notified in case of an error (refer to the Advanced settings section below). Users can follow up which jobs induced the stop by looking at the Targets.txt file. The last <job>_DONE is the <job> that induced the premature stop.

4.5.2 Example

As an example, lets suppose ChIPpip crashed while analyzing the test data. Troubleshoot the error by looking at the Targets.txt file:

[ChIPpip]:$ ***less ./EXAMPLE/DataStructure/Targets.txt***

```
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#
# Project ID: "EXAMPLE"
#
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#
# Remote Server
#
# My_email          =       "your.email@harvard.edu"
#
# My_project_title        =       "EXAMPLE"
# Reference_genome        =       "mm9"
# Path_to_proj_folder     =       "/home/NEAT/ChIPpip"
# Path_to_ChIPpip         =       "/home/NEAT/ChIPpip"
# Path_to_orifastq.gz     =       "/home/NEAT/ChIPpip/fastq"
# Path_to_chrLens.dat     =       "/.../reference_files/mm9/chr_lens.dat"
```

```
#
# Path_to_RefGen.fa       =       "/.../mm9.fa"
# Paired_end_run          =       "0"
#
# Steps_to_execute        =       " unzip_DONE+ qc_DONE + map_DONE + filter + peakcalling + cleanbigwig "
#
# Remove_from_bigwig      =       "random, chrM"
#
#
# Local
#
# TaxonDatabaseKG         =       "TxDb.Mmusculus.UCSC.mm9.knownGene"
# TaxonDatabaseDict       =       "org.Mm.eg.db"
#
#-----------------------------------------------------------------------------------------------------$
#
#
OriFileName   FileName                OriInpName  InpName  Factor Replicate  FileShort  Experiment  Date
PSa36-1_R1    PSa36-1_noDox_K4me3     PSa37-3_R1  PSa37-3_noDox_Inp  K4me3    1  noDox  1  2015-01-01
PSa36-2_R1    PSa36-2_Dox_K4me3       PSa37-4_R1  PSa37-4_Dox_Inp    K4me3    2  Dox    1  2015-01-01
```

The last <job>_DONE is the map_DONE, which indicates the source of the error. Troubleshoot the origin by looking at the qsub error files corresponding to the 'map' section.

```
[ChIPpip]:$ ls -l ./scripts/map/qsub

790     PSa36-1_noDox_K4me3_map.sh.e<jobID>
  0     PSa36-1_noDox_K4me3_map.sh.o<jobID>
```

The .e.<jobID> file is a lot bigger than usual. Use the unix ***less*** command to open it in a read mode. Error messages should be self-explanatory. To exit, hit ***Enter***. In our example, we have the following error message:

```
[ChIPpip]:$ less ./EXAMPLE/scripts/map/qsub/

[bwt_restore_bwt] fail to open file '/data/ref/mm9/bwa/mm9.fa.bwt'. Abort!
```

ChIPpip tells you it could not open the '/data/ref/mm9/bwa/mm9.fa.bwt' file. Check the existence of the file in this path. Once the source of the error is determined, modify the Targets.txt file accordingly and move on.

4.6 Advanced settings

4.6.1 AdvancedSettings.txt

Users can modify advanced settings (resource manager extension and parameters, queue file, wall time, etc), map and filter parameters, bin size for wig files, etc) in the *AdvanceSettings.txt* file found in the *DataStructure* directory.

Would users decide to modify the bwa aligner command (*#Aligner_algo_short*), additional parameters need to be fitted between the '*aln*' and the '*-n*'. If any additional parameters are added elsewhere, ChIPpip will terminate with an error. Changing this command to another aligner is presently in beta-mode and is not supported.

5 NEAT Part 2 : ChIPmE

5.1 Introduction

ChIPmE is a package of two applications. Its main goal is to accompany users during the analysis of next generation sequencing (NGS) data as part of the NEAT toolkit (NGS easy analysis toolkit). ChIPpip, in conjuncture with the NEAT package, provides an easy, rapid and reproducible exploratory data analysis (EDA) tool that allows users to assess their data in as few as a couple. The primary goal of ChIPmE is to provide metagene analysis, peak overlaps and countables. ChIPmE has been developed as a downstream analysis tool for NGS data that has been analyzed using the NEAT (ChIPpip) package.

ChIPmE runs through the MacOS automator software. This software should be installed by default on most modern Apple computers. Please ensure this is installed on your computer in the *applications* directory. In addition, ChIPmE is an R script that requires a few widely recognized R packages. For details on these, please refer to the *Version information and required packages* section below.

5.2 Before running ChIPmE

Please make sure all R packages required for ChIPmE are installed on your computer (refer to the R manual) prior to running ChIPmE. Refer to the *Version information and required packages* section below for more information.

Finally, make sure your *Terminal* software is closed before launching ChIPmE.

5.3 Running NEAT Part 2.1 (ChIPmE)

5.3.1 Step 1: Download a ChIPpip project

To transfer a ChIPpip project from a remote server to a local computer, double click the ChIPmE *3_Transfer.app*.

This app is can be found in the NEAT directory *~/Desktop/NEAT/ChIPmE/*. Users will be asked to locate the NEAT directory and where they want to save their ChIPpip project. In this example, the NEAT directory is on our desktop and we will save our ChIPpip project on the desktop as well.

The next step is to provide the path to the ChIPpip directory on the remote server. In the ChIPpip tutorial, users were invited to save their ChIPpip project to the ChIPpip directory (*/HOME/NEAT/ChIPpip/EXAMPLE*), so this is the path we will enter.

Finally, ChIPmE will prompt you to enter your SSH information. Once again, in consistency with the ChIPpip tutorial, we will enter: *username@server-address.edu*.

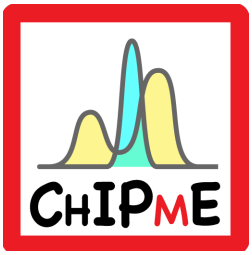
Starting the download will launch the *Terminal* to open and start running R. It will require users to enter their password several times (for each call to the remote servers). Please, be aware and follow this process as failing to enter your password will break the connection after some time.

Downloading an entire project should not take more than a few minutes.

5.4 Running NEAT Part 2.2 (ChIPmE)

5.4.1 Step 2: Run a ChIPmE analysis

Once the project has been downloaded, users can run the proper ChIPmE analysis. To this end, double click the ChIPmE *4_Analyze.app*.



Users will be asked to locate the ChIPpip folder (the one you just downloaded). In our case, the ChIPpip project was downloaded to the desktop.

Users will then have to locate the NEAT folder (also saved on the desktop in this example). Finally, users will have to choose a mart object. These are .bed files of regions you are interested in aligning your data to. Examples of mart objects are transcriptional start sites (TSS), all transcripts, enhancers, etc. Some mart objects are provided as part of the NEAT package in the MartObject folder. For this example, we will choose to align our data over TSS. The provided mart object (*mm9_TSS_10kb.bed*) is comprises of 10kb around all TSS of the mouse genome. Please note here that care should be brought to match the mart objects with the reference genome initially used. In our case, our data was mapped to the mouse mm9 genome, hence the *mm9_TSS_10kb.bed*. Several parameters can be set before running the analysis including *binNumber*, *strand*, *runmeank*, *Venn* and *normInp*. Values are set as a reference, but we suggest users experiment to find the best values for their own need.

Running the analysis using the test data should take less than a minute.

5.5 Outputs

5.5.1 Logs

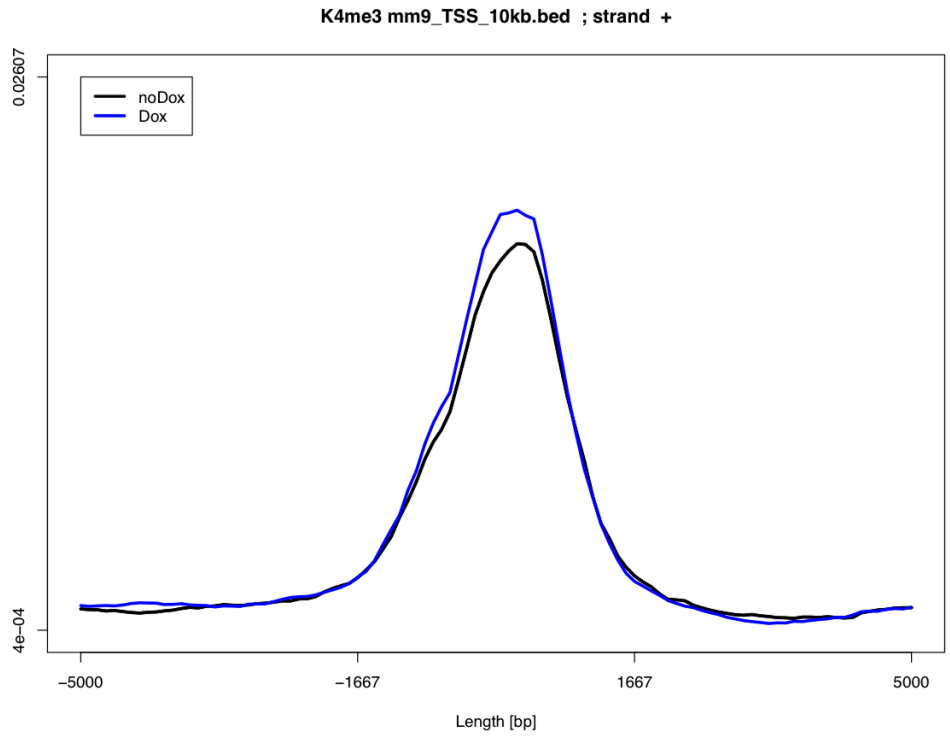
Each time ChIPmE is run, a log file is created and named using the date and time. This file is save in the *./EXAMPLE/logs/* directory. We strongly encourage users to initially look at these files, as any error that might have occurred will be saved there. Usually, if no error is prompted from the terminal, ChIPmE has terminated correctly. Also, please note that if there are unrecognized chromosome names such as random chromosomes, warning messages will appear. In the test data, there are 50 or more warnings. These can usually be disregarded.

5.5.2 Count tables

ChIPmE will generate count tables that should be self-explanatory. In brief, rows correspond to a mart object line and columns correspond to bins. Please note here that bins will be of same length for centered mart objects (for example TSSs) and will have names corresponding to base pairs, but will be of varied length for non-centered objects (for example transcripts), hence the columns will arbitrarily be named V1, V2, …, VN. Users should not worry about this to interpret graphs as it is accounted for by normalizing the values per bin by the bin length.

5.5.3 Metagene plots

ChIPmE generates pdf plots that are saved in */EXAMPLE/plots/*. Below is an examples of the plot generated for K4me3 over TSSs.



5.6 Advanced settings

5.6.1 Custom mart objects

Custom mart objects can easily be created with your favorites genes/regions. The files are simple .bed files that can either be manually or automatically created using various online tools or can be directly downloaded from genome browsers such as USCS or Ensembl. ChIPmE has been developed to ensure consistency between and within labs. In addition the relative small size of these bed files makes it easy to email/share them. We therefore suggest keeping an up-to-date, centralized folder containing all recurrent mart object files.

5.6.2 Bam files and GRanges

Once Granges objects have been created, bam files are no longer required locally. Users are thus free to delete these files as they are often two orders of magnitude larger than GRanges objects (respectively several Gb vs tens of Mb). We do suggest that users backup their .bam files on the remote server.

5.6.3 Consolidating projects

Consolidating projects is very easy. Users who intend to do so will simply need to change the *Targets.txt* file and copy-paste the Granges objects (or bam files) from one folder to the other. Other files and folder can be left as is.

6 Version information and required packages

Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.5.9-r16

Program: samtools (Tools for alignments in the SAM format)
Version: 0.1.18 (r982:295)

R version 3.1.0 (2014-04-10)
Platform: x86_64-redhat-linux-gnu (64-bit)

R packages (with dependencies) required to run ChIPpip:

- SPP (spp_1.11)
- systemPipeR (systemPipeR_0.99.0)
- …

R packages (with dependencies) required to run ChIPmE:

- Rsamtools
- GenomicRanges
- GenomicAlignments
- caTools
- VennDiagram

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:

[1] grid parallel stats4 stats graphics grDevices utils

[8] datasets methods base

other attached packages:

[1] VennDiagram_1.6.9 caTools_1.17.1 GenomicAlignments_1.2.1

[4] Rsamtools_1.18.2 Biostrings_2.34.1 XVector_0.6.0

[7] GenomicRanges_1.18.3 GenomeInfoDb_1.2.3 IRanges_2.0.1

[10] S4Vectors_0.4.0 BiocGenerics_0.12.1

loaded via a namespace (and not attached):

[1] base64enc_0.1-2 BatchJobs_1.5 BBmisc_1.8 BiocParallel_1.0.0

[5] bitops_1.0-6 brew_1.0-6 checkmate_1.5.0 codetools_0.2-9

[9] DBI_0.3.1 digest_0.6.4 fail_1.2 foreach_1.4.2

[13] iterators_1.0.7 RSQLite_1.0.0 sendmailR_1.2-1 stringr_0.6.2

[17] tools_3.1.2 zlibbioc_1.12.0

7 Reported bugs

7.1 Cleanbigwig

Some clusters have settings that do not allow ChIPpip to pipe certain samtool commands such as the *wigToBigwig* tool. To check if your cluster is under these rules, list files from the qsub folder. The numbers following the .o and .e are the job id. Choose one and open the fie using ***less***, which will show you the error.

[ChIPpip]:$ ***ls -l ./EXAMPLE/scripts/cleanbigwig/qsub/***

```
216    PSa36-1_noDox_K4me3_cleanbigwig.sh.e471534
  0    PSa36-1_noDox_K4me3_cleanbigwig.sh.o471533
216    PSa36-2_Dox_K4me3_cleanbigwig.sh.e471534
  0    PSa36-2_Dox_K4me3_cleanbigwig.sh.o471534
```

[ChIPpip]:$ ***less ./EXAMPLE/scripts/cleanbigwig/qsub/PSa36-1_noDox_K4me3_cleanbigwig.sh.e471534***

./EXAMPLE/scripts/cleanbigwig/wigToBigwig.sh: line 2: wigToBigWig: command not found

---

This should be an easy fix by manually running the *./EXAMPLE/scripts/wigToBigwig.sh* script. To this, simply type:

```
[ChIPpip]:$ sh ./EXAMPLE/scripts/cleanbigwig/wigToBigwig.sh
```

8 Funding

This pipeline was developed with funding from the Swiss National Science Foundation and various funding sources from the Kingston Lab at the department of molecular biology at MGH and the department of Genetics at Harvard Medical School.

9 References

- Kharchenko P, Tolstorukov M & Park P. (2008) Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nature Biotechnology* **26**, 1351 - 1359
- Girke T. (2014) systemPipeR: NGS workflow and report generation environment. URL https://github. com/tgirke/systemPipeR.
- Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics*, 25:1754-60.
- Michael Lawrence, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, Morgan MT, and Carey VJ. (2013) Software for computing and annotating genomic ranges. *PLoS Comput. Biol.*, 9(8):e1003118.
- Li H and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14): 1754–1760.
- Li H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. URL http: //arxiv.org/abs/1303.3997.