ChIPpip : NGS pipelines made easy

Patrick Schorderet

Patrick.schorderet@molbio.mgh.harvard.edu

Jan 2015

1. Introduction

ChIPpip is a perl/R package that supports 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 less than 12 hours (based on a 200mio read Highseq run). As such, ChIPpip manages many of the repetitive, error-prone tasks required for NGS data analysis. It is versatile and easily configurable to meet each users' preferences in terms of mapping, filtering and peak calling. ChIPpip accompanies the user from compressed fastq files (.fastq.gz), usually provided by the sequencing core facility, to bigwigs using a single command line.

A central feature of ChIPpip is its ability to perform repetitive tasks on complex sample setups while managing batch submissions and cluster queuing. ChIPpip 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. ChIPpip 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 ChIPpip, please read the README file. In addition, before analyzing your own data, we suggest you follow this tutorial, which will help you better understand the logic of ChIPpip. You will be able to follow the analysis of a small test dataset (provided as part of ChIPpip) using your own computer cluster. This will also ensure ChIPpip 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 This data comes from an unpublished 50bp single end (SE) sequencing experiment although ChIPpip can deal with paired-end (PE) sequencing as well. For more information on the test data provided in this tutorial, please read below.

Although ChIPpip can be run by scientists with limited to no programming experience, this tutorial require 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.

2. Getting started with UNIX

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 for MacOS/LINUX. For MacOS users, we suggest you use the *Terminal* (applications/Terminal) for all the 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.

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 analogous 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.

Install ChIPpip

To install ChIPpip, download the ChIPpip repository from GitHub to any directory on your computer cluster. As an example for the following tutorial, we will suppose you have the *ChIPpip/* directory on your desktop (local computer) and that you will save it to your */HOME/* directory on the remote server.

[~Desktop]$ ***scp –r ~/Desktop/ChIPpip/ username@serveradress.edu:~/***

Enter your password and wait for the folders to transfer.

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

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

These commands should bring you to your home directory on the remote server. For the following tutorial, we will suppose the name of this directory is */HOME/*. Navigate to your */HOME/* directory using the ***cd*** and list files using the ***ls*** command.

```
[username@setrveradress ~]$ cd ChIPpip
[username@setrveradress ChIPpip]$ ls -l
ChIPpipCreateNewProject.pl
ChIPpipRunPipeline.pl
README
README.md
Scripts
```

3. Test data

Structure and significance of test data

The unpublished test data provided for this tutorial is part of the ChIPpip folder and can be downloaded on GitHub. If you have followed the previous steps, the

compressed fastq files required for the tutorial can be found in the test data folder /ChIPpip/scripts/testdata/.

```
[~]:$ ls -l ./scripts/testdata
1153158    PSa36-1_R1.fastq.gz
1158276    PSa36-2_R1.fastq.gz
1113658    PSa36-5_R1.fastq.gz
1112601    PSa36-6_R1.fastq.gz
1109556    PSa37-3_R1.fastq.gz
1094951    PSa37-4_R1.fastq.gz
```

The data consists of four ChIPseq experiments for two histone marks (K4me3, K27me3) in two growing conditions (noDox, Dox). Each ChIP has been made from a single batch of cells (per condition), so only one input files is provided per sample. No replicates were generated.

4. Running ChIPpip PART 1

Creating a new ChIPpip project

The first step to run ChIPpip is to create a new ChIPpip project. Navigate to the ChIPpip directory and run the ChIPpipCreateNewProject.pl script. This script requires the user to specify the path where the new ChIPpip project will be created and the name of the new project. In this example, we will create a project in the ChIPpip directory named EXAMPLE.

```
[~]:$ perl ChIPpipCreateNewProject ./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.

5. The Targets.txt file structure

Filling in the Targets.txt file

The Targets.txt file found in the ~/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. This file is the most important piece of ChIPpip 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), so 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_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_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'.
Path_to_proj_folder	:	Full path to your project folder (without the project name) [automatically generated by ChIPpipCreateNewProject].
Path_to_ChIPpip	:	Full path to your ChIPpip 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.
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.
Path_to_chrLens.dat	:	Refer to your computer core facility.
Path_to_RefGen.fa	:	Refer to your computer core facility.
Paired_end_run	:	"0" for single end sequencing. "1" for paired end sequencing.
Steps_to_execute	:	Users can choose from the following tasks: *unzip*, *qc*, *map*, *filter*, *peakcalling* and *cleanbigwig*. 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 tehR package SPP and *cleanbigwig* requires samtools. Refer below to exact requirements.
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'.

# Local	:	These parameters are only necessary for users who go on to use the NEAT tookit for metagene analysis, etc on their local computer. If you will not use this package, please disregard (leave as is).
TaxonDatabaseKG	:	Database of preferred feature such as known gene for RNAseq.
TaxonDatabaseDict	:	Idem

Please modify the Targets.txt file for 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. Moreover, the reference genome files should have a .fa extension (e.g. mm9.fa). Please check 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:

[~]:$ **nano ./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/ChIPpip"
# Path_to_ChIPpip       =      "/HOME/ChIPpip"
# Path_to_orifastq.gz   =      "/HOME/ChIPpip/fastq"
# Path_to_chrLens.dat   =      "/HOME/.../reference_files/mm9/chr_lens.dat"
# Path_to_RefGen.fa     =      "/HOME/.../mm9.fa"
# Paired_end_run        =      "0"
#
# Steps_to_execute      =      " unzip + qc + map + 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
PSa36-5_R1    PSa36-5_noDox_K27me3  PSa37-3_R1   PSa37-3_noDox_Inp   K27me3   1   noDox   1   2015-01-01
PSa36-6_R1    PSa36-6_Dox_K27me3    PSa37-4_R1   PSa37-4_Dox_Inp     K27me3   2   Dox     1   2015-01-01
```

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

To avoid repeting this evry time you create a new ChIPpip project, we suggest you alter the *original* Targets.txt file that is used as template when creating a new ChIPpip project. Modify it using the same approach as above:

```
[~]:$ nano ./scripts/NewChIPseqProject/DataStructure/Targets.txt
```

6. Running ChIPpip PART 2

Run the analysis

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, the ChIPpipRunPipeline 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*.

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

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

My email:               your.email@harvard.edu

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

Paired end sequencing:      0
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)
.....
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Exiting INITIAL section with no known error

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

This will launch the pipeline and will prompt a summary of the parameters you have specified. 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).

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.

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

7. ChIPpip workflow

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.

Quality control (QC)

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

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 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.

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.

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 specifiy the

sequences that are unrecognized by some of the aforementioned programs. Refer to the Targets.txt file description for more information.

8. Generated data and next steps

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

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. Users interested in such analysis can download the NEAT package from GitHUB and follow the tutorial.

9. Troubleshooting

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 (*/HOME/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:

```
[~]:$ ls -l ./EXAMPLE/scripts/map/
   1018   map.sh
    759   PSa36-1_noDox_K4me3_map.sh
    749   PSa36-1_noDox_K4me3_map.sh
     ...
   4096   qsub

[~]:$ 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 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.

Example

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

[~]:$ 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/ChIPpip"
# Path_to_ChIPpip       =     "/HOME/ChIPpip"
# Path_to_orifastq.gz   =     "/HOME/ChIPpip/fastq"
# Path_to_chrLens.dat   =     "/HOME/.../reference_files/mm9/chr_lens.dat"
# Path_to_RefGen.fa     =     "/HOME/.../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
PSa36-5_R1   PSa36-5_noDox_K27me3   PSa37-3_R1  PSa37-3_noDox_Inp  K27me3  1  noDox  1  2015-01-01
PSa36-6_R1   PSa36-6_Dox_K27me3     PSa37-4_R1  PSa37-4_Dox_Inp    K27me3  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.

```
[~]:$  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:

```
[~]:$  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.

10. Advanced settings

AdvancedSettings.txt

Users can modify advanced settings (map, filter, etc) in the AdvanceSettings.txt file found in the DataStructure directory.

QSUB parameters

The qub header can be modified to meet the requirements of specific clusters, including queuing times, nodes, number of CPUs, etc. If this is of interest, please modify the QSUB_header.sh template file in ./ChIPpip/scripts/QSUB_header.sh such as to apply personalized settings to all jobs (this needs to be only once). Please refer to your computer core facility systems administrator for further details.

11. 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:

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

12. Funding

This pipeline was developed with funding from the Swiss National Science Foundation.

13. 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.