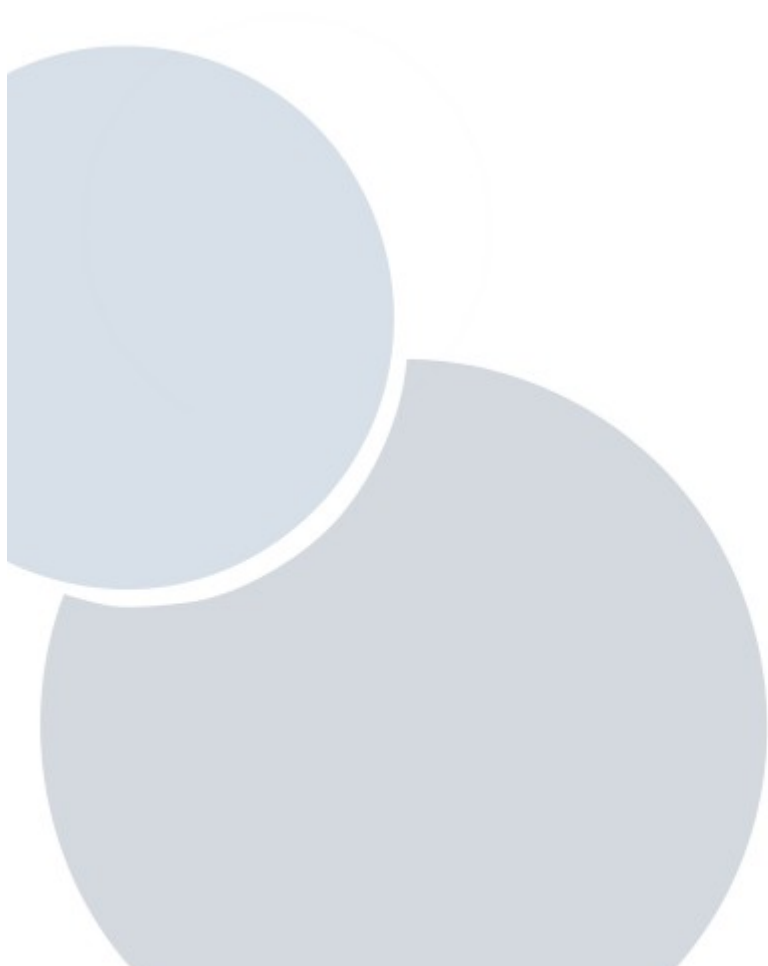




RUBIOSeq 3.7

Manual

Miriam Rubio Camarillo



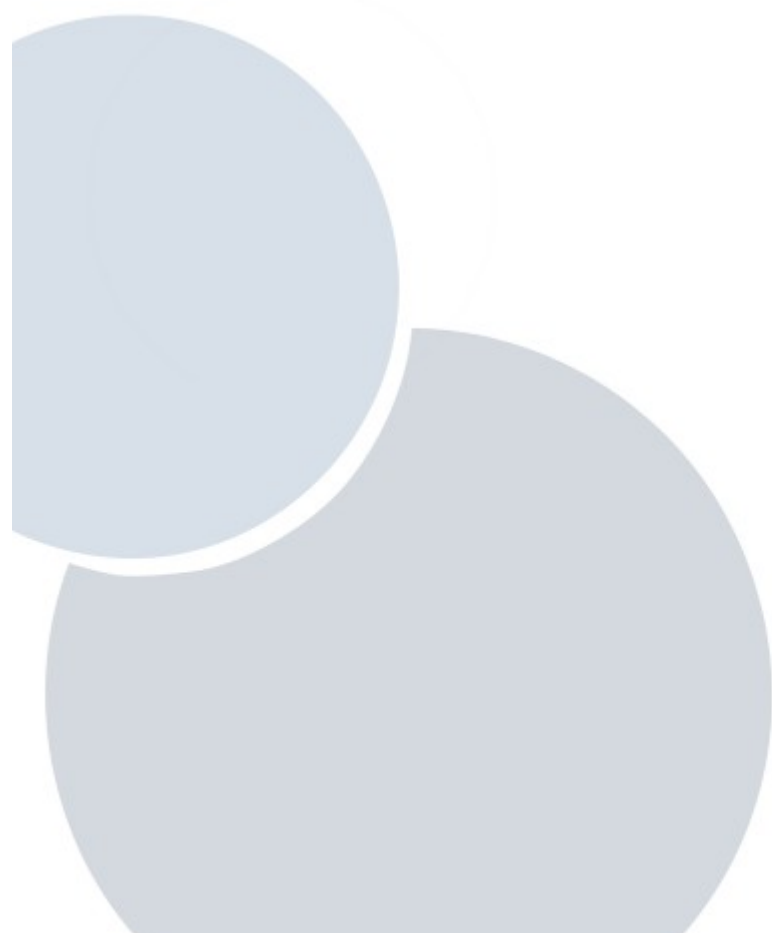
Index

- 1.Preface.....7
- 2.License.....7
- 3.Installation.....7
- 4.Prerequisites.....8
- 5.RUbioSeq. Variant Caller.....9
 - 1.Introduction.....9
 - 2.Prerequisites.....9
 - 3.Before running a variant analysis for first time.....10
 - 1.Reference genome indexing.....10
 - 1.Illumina:.....10
 - 2.SOLiD:.....11
 - 2.Configure program paths.....11
 - 4.Configuration files.....11
 - 1.configProgramPaths.xml.....12
 - 2.Experiment.xml.....13
 - 3.VEP script configuration.....22
 - 4.CONDEL PLUGIN.....22
 - 5.Program execution.....22
 - 1.Program workflow.....22
 - 1.Input Data.....23
 - a)Sample files:.....23
 - Raw input data.....23
 - Illumina.....23
 - SOLiD.....24
 - Processed input data.....27
 - Example XML experiment file configuration.....27
 - b)Reference and annotation files.....29
 - Example:.....29
 - Reference.fa:.....29
 - 2.Program outputs.....30
 - a)Output files.....30
 - 3.Quality and Control.....31
 - a)Fastqc analysis.....31
 - b)Bam files validation.....31
 - 4.RubioSeq filters.....32
 - a)Depth filter and minQUAL filter.....32
 - b)dbSNP filter.....32
 - c)Change Hard Filters.....32
 - 5.Variant Quality Score Recalibration or Hard filters?.....32
 - 6.Execution modes.....32

- 6.RUbioSeq. CnvCaller.....34
- 1.Introduction.....34
- 2.Prerequisites.....34

3. Before running a cnv analysis for first time.....	35
1. Index reference.....	35
2. Configure program paths.....	35
4. Configuration files.....	35
1. configProgramPaths.xml.....	35
2. Experiment.xml.....	37
5. Program execution.....	42
1. Program workflow.....	42
1. Program outputs.....	43
a) Output files.....	43
2. Quality and Control.....	43
7. RUBioSeq. ChIPseq Caller.....	44
1. Introduction.....	44
2. Prerequisites.....	44
3. Before running a ChIPseq analysis for first time.....	45
1. Index reference.....	45
2. Configure program paths.....	45
4. Configuration files.....	45
1. configProgramPaths.xml.....	45
2. Experiment.xml.....	47
5. Program execution.....	52
1. Program workflow.....	53
1. Program outputs.....	53
2. Quality and Control.....	54
8. RUBioSeq. Methylation caller.....	55
1. Introduction.....	55
2. Prerequisites.....	55
3. Before running a methylation analysis for first time.....	55
1. Configure program paths.....	55
4. Configuration files.....	55
1. configProgramPaths.xml.....	55
2. Experiment.xml.....	57
5. Program execution.....	60
1. Program workflow.....	60
1. Program inputs.....	61
2. Program outputs.....	61
a) Files.....	61
SAM files.....	61
Methylation calls files.....	61
Intervals methylation.....	62
3. Quality analysis.....	63
a) Fastqc analysis.....	63
4. User filters.....	64
a) Depth filter.....	64

5.Execution modes.....	64
9.Quick Examples.....	66
1. Variant Calling.....	66
1.Raw Data.....	66
2.Steps.....	66
3.Results.....	68
2.Copy Number Variation Analysis.....	68
1.Raw Data.....	68
2.Steps.....	68
3.Methylation Calling.....	69
1.Raw Data.....	69
2.Steps.....	69
3.Results.....	70
4.ChipSeq Calling.....	70
1.Raw Data.....	70
2.Steps.....	70
3.Results.....	71
10.Bibliography.....	72



Figures

Figure 1: SOLiD Color space.....	25
Figure 2: SOLiD read types.....	25
Figure 3: Parallel execution variant calling workflow.....	33
Figure 4: Methylcalls file.....	62
Figure 5: Multiexecution workflow.....	65

1. Preface

Primary and secondary data analyses of next-generation sequencing studies (NGS) consists on a set of successive stages that are repetitive and routinely executed using a wide collection of tools (e.g. quality control tools, read aligners, variant callers, etc.). These tools have different origins and usually lack of straight interoperability. This issue has driven computational biologists to claim for intuitive, efficient and integrated pipelines to facilitate routine NGS analysis avoiding manual steps and improving the reproducibility of the results. Several remarkable efforts have been done in this sense.

We present RUBioSeq, an automated and parallelized software suite for primary and secondary analysis of Illumina, ION and SOLiD experiments. Using standard input and output file formats and an intuitive XML configuration file, the application offers an integrated framework to run parallelized pipelines for variant detection in exome enrichment, methylation studies and ChIPseq analyses.

This document is meant to serve as a guide for the practical use of RUBioSeq. It includes explanations of all command-line options for each type of analyses to give an idea of basic usage. Input and output file formats are also detailed. Also, many examples of use are given.

This document does not try to explain the underlying algorithms or data-structures used in RubioSeq (i.e. we will not explain the GATK algorithms, or the Bismark algorithm).

2. License

RUBioSeq by Miriam Rubio is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License (<http://creativecommons.org/licenses/by-nc-nd/3.0/>).

3. Installation

As RUBioSeq suite depends on more than 20 different software packages, some of them difficult to install and setup, a customized 64-bit LiveDVD (based on Ubuntu 14.04 Desktop LiveCD) has been created, it bundles RUBioSeq_v3.7 plus all its dependencies, ready to be used on any computer. You can even install the contents of this customized Ubuntu on any computer, so you have RUBioSeq+Ubuntu installed at once!

Also, user can install manually the software and the dependencies.

4. Prerequisites

RUBioSeq is distributed via source code. We assume that the RUBioSeq tarball has been downloaded from <http://rubioseq.biocnio.es> and unpacked.

All programs listed below and their corresponding dependencies must be correctly installed.

1. Perl v5.10.0 or higher. <http://www.perl.org/get.html>. Executable directory must be added to the PATH environment variable.
2. Java version 1.7 . <http://www.java.com/>. Executable directory must be added to the PATH environment variable.
3. Samtools 0.1.19. <http://samtools.sourceforge.net/>
4. Picardtools 1.107. <http://picard.sourceforge.net/>
5. Fastqc v0.10.1. <http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/>
6. If running on an HPC with SGE: SGE version 6.2u5. (Experimentally on: PBS version PBSPro_10.1.0.91350)
7. Perl modules:
 1. XML::LibXML
 2. Carp
 3. FindBin
 4. File::Basename
 5. File::Spec
 6. File::Copy
 7. Getopt::Long
 8. Class::Inspector

Specific programs to the different analysis types supported by RUBioSeq are detailed in the prerequisites section of each analysis.

Most of the RUBioSeq's required programs can be downloaded at https://sourceforge.net/projects/rubioseq/files/Required_programs/ .

5. RUBIOSEQ. Variant Caller.

1. Introduction

The variant calling workflow developed in this program automatically executes all involved stages in this process using state-of-art software.

Our process includes four main stages:

1. Short-read alignment with a combination of BWA and BFAST aligners.
2. Duplicates marking, GATK-based recalibration and realigning.
3. GATK-based variant calling and filtration protocol for *indels* and SNPs.
4. User customized filters and variant effect predictor to annotate the variants.

Our program accepts raw data in Fastq format from Illumina platform, raw data from SOLiD platform (F3, F5-P2 reads) or BAM files. It generates a result file with the variants and its biological impact prediction ranked by detection quality score.

Quality and errors check points for input data and files created during the execution are also performed.

This program has been design to execute on a HPC, scheduled by an SGE system, this design allows a parallel multiple sample execution in order to reduce the processing time. It can be also executed on a HPC with PBS system and on a standard workstation in sequential mode.

2. Prerequisites

Operating System: UNIX.

All programs listed below and their corresponding dependencies must be correctly installed.

1. Bwa 0.7.10. <http://bio-bwa.sourceforge.net/>
2. Bfast-bwa 0.7.0b . <https://sourceforge.net/p/bfast/bfast/ci/master/tarball>
3. GenomeAnalysisTK-3.1-1-g07a4bf8
4. R environment 2.14 or higher. <http://www.r-project.org/>
5. RScript 2.14.0 or higher. <http://www.rforge.net/rscript/files> . Executable directory must be added to the PATH environment variable.
6. Perl modules:
 1. Bioperl (1.2.3)
 2. DBI
 3. DBD::mysql

3. Before running a variant analysis for first time

User have to perform some previous steps before running the program for first time, because reference genome indexes for BWA and BFAST aligners must be prepared. These steps depends on the raw data we are going to analyzed: Illumina or ABI SOLiD platform.

These steps must be done only once.

User can execute by his own or in case of SGE scheduled systems, user can adapt the auxiliary scripts provided in RUBioSeq code in order to facilitate the task.

Reference genome and its index files must be located in the same directory.

1. Reference genome indexing

Users must index the genome adapted to the data they are going to use. The BFAST masks used below are the recommended masks to align human genome reads equals or greater than 40 bp.

1. Illumina:

1. Indexing reference for BWA:

```
#bwa index -a bwtsv reference.fasta
```

2. Steps reference for BFAST:

```
#bfast fasta2brg -f reference.fasta -A 0
```

Execute:

```
#bfast index -f reference.fasta -m <mask> -w 14 -i <index number> -n 16
```

for each of these masks:

```
mask[1]="11111111111111111111111111111111"
mask[2]="111110111011101010010101101111"
mask[3]="10111101011010010110000110100011111111"
mask[4]="101110011010011001001111010100010111111"
mask[5]="111110110111011110111111111111"
mask[6]="111111100101001000101111101110111"
mask[7]="11110101110010100010101101010111111"
mask[8]="111101101011011001100000101101001011101"
mask[9]="1111011010001000110101100101100110100111"
```

```
mask[10]="1111010010110110101110010110111011"
```

2. SOLiD:

We have to execute Illumina indexing steps and also perform the “color space” indexing:

1. Indexing reference for BWA:

```
#bfast bwtdindex -a bwtsw -A 1 reference.fasta
#bfast bwtdindex -a bwtsw -A 0 reference.fasta
```

2. Steps for BFAST:

```
#bfast fasta2brg -f reference.fasta -A 1
```

And then execute:

```
#bfast index -f reference.fasta -A 1 -m <mask> -t -w 14 -i <index number> -n 16
```

for each of these masks:

```
mask[1]="11111111111111111111111111111111"
mask[2]="11111010011111001111111111111111"
mask[3]="1011111101100110001111100011111111"
mask[4]="1111111100101111000001100011111011"
mask[5]="11111111000111111001111111111111"
mask[6]="111110110100110000110001100111111111"
mask[7]="11111111111100111011111111111111"
mask[8]="11101100001111111100111101111111"
mask[9]="1110110001011010011100101111101111"
mask[10]="1111110010001100010111001100011000111111"
```

2. Configure program paths

User must configure configProgramPaths.xml file with the corresponding paths to the programs. Configuration program file is explained in the next section.

4. Configuration files

There are four configuration files: configProgramPaths.xml to configure applications paths and queue schedulers management. Experiment.xml, with all specific parameters for the execution and vepConfig files to configure Ensembl database connection and Condol plugin.

1. configProgramPaths.xml

This XML format file configures applications paths used by the program, it also initializes

queue schedulers parameters for our specific system.

It must be located in *installation_path/RUBioSeq_v3.7/variantCalling/config/*.

User must configure this file properly to adapt the application paths and scheduler.

Configuration file example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RUBioSeq variantCaller CONFIG FILE -->
<?xml version="1.0" encoding="UTF-8"?>
<!-- RUBioSeq variantCaller CONFIG FILE -->
<configData>
  <bwaPath>/opt/NGS/bwa/0.7.10/</bwaPath>
  <samtoolsPath>/opt/NGS/samtools/0.1.19/</samtoolsPath>
  <gatkpath>/home/mrubioc/soft/GenomeAnalysisTK-3.1-1/</gatkpath>
  <picardPath>/home/mrubioc/soft/picard-tools-1.60/</picardPath>
  <BFASTPath>/opt/NGS/bfast+bwa/0.7.0b/bin/</BFASTPath>
  <fastqcPath>/home/mrubioc/soft/FastQC/</fastqcPath>
  <nthr>4</nthr>
  <javaRam>-Xmx16G</javaRam>
  <queueSystem>SGE</queueSystem>
  <queueName>ngs</queueName>
  <multicoreName>multicore</multicoreName>
  <multicoreNumber>4</multicoreNumber>
</configData>
```

Fields:

All this fields are mandatory. They initialize the directory where binary or script of the corresponding program is located:

bwaPath, **samtoolsPath**, **gatkpath**, **picardPath**, **BFASTPath**, **fastqcPath**.

queueSystem: represents queue scheduler type. The accepted values are: SGE, PBS and none (for execution in non-queue controlled systems).

If *queueSystem* is initialized to *none* the multisample execution is not performed in a parallel way, but in a secuencially execution, because there is no scheduler to synchronize the programs.

These other fields are optional:

nthr : represents thread number parameter for all compatible applications. Default value: 1.

javaRam: Maximum RAM memory limit for java applications. Default value: -Xmx16G.

queueName: Queue's name in which tasks are going to run. Default value: normal.

multicoreName: Only valid for SGE systems. It represents the name of the SGE's parallel environment. Default value: multicore (disabled). To use this parameter, SGE manager must create a parallel environment with the multicoreName feature.

multicoreNumber: Only valid for SGE systems. Represents the number of slots the user want to keep for his execution. Default value: -1 (disabled). To use this feature, SGE manager must create a parallel environment called *multicore*.

2. Experiment.xml

Experiment.xml configures all the variant caller parameters to adapt the script to the specific input data and user desired configuration.

This file must follow the format `installation_path/RUBioSeq_v3.7/variantCalling/config/Template.xml`.

It must have at least all the mandatory fields. Name and location of this file is a user choice. This file is one of the two parameters of the variant caller.

User has also to specify the branch type in the XML main element configData:

```
<configData branch="SNV">
```

File fields:

GenRef: Mandatory. It initializes the absolute path to reference genome. Genome indexes must be located in the same directory.

Format:

```
<GenRef>ReferenceGenomePath</GenRef>
```

Example:

```
<GenRef>/Volumes/RAID/Soft/NGS/GATK-1.0.4388/resources/Homo_sapiens_assembly18.fasta</GenRef>
```

DbSnpAnnot: Mandatory. It represents the absolute path to the variants annotation file (dbSNP). It must be in ROD or VCF format.

Format:

```
<DbSnpAnnot>DbSnpPath</DbSnpAnnot>
```

Example:

```
<DbSnpAnnot>/Volumes/RAID/Soft/NGS/GATK-  
1.0.4388/resources/dbsnp_129_hg18.rod</DbSnpAnnot>
```

Genomes1000Annot: Mandatory. It represents the absolute path to 1000 Genome project variant annotations. It must be in ROD or VCF format.

Format:

```
<Genomes1000Annot>1000GAnnotationFilePath</Genomes1000Annot>
```

Example:

```
<Genomes1000Annot>/Volumes/RAID/Soft/NGS/HG19/1000G_omni2.5.hg19.vcf</Ge  
nomes1000Annot>
```

IndelAnnot: Mandatory. It represents the absolute path to REFSEQ annotation file. It must be in ROD or VCF format.

Format:

```
<IndelAnnot>RefseqFilePath</IndelAnnot>
```

Example:

```
<IndelAnnot>/Volumes/RAID/Soft/NGS/GATK-  
1.0.4388/resources/refSeqFinal_hg18.rod</IndelAnnot>
```

Intervals: Optional. If exists, it represents the genomic intervals over which to operate the GATK functions, if it doesn't, the tools operated over the whole genome. File must be in BED format.

Format:

```
<Intervals>AbsolutePathToIntervalsfile</Intervals>
```

Example:

```
<Intervals>/Volumes/RAID/Homes/mrubio/SS_AllExonv2_GATK.bed</Intervals>
```

KnownIndels: Optional. It represents input VCF files with known indels. This field can appear one or more times. These files will be used in GATK realigner phase.

Format:

```
<KnownIndels>AbsolutePathToKnownIndels</KnownIndels>
```

Example:

```
<KnownIndels>/Volumes/RAID/NGS/HG19/Mills_Devine_2hit.indels.hg19.sites.vcf</K  
nownIndels>
```

Platform: Mandatory. Represents NGS technology that generates input raw data. Valid values: illumina, solid or ion.

Format:

```
<Platform>platform</Platform>
```

checkCasava: Optional. Only apply on Illumina's fastq input files. Set to 0 **only** if fastq reads don't have standard Casava format. Reads must have ASCII33 qualities. Default value :1.

Format:

```
<checkCasava>1</checkCasava>
```

dirOutBase: Mandatory. Absolute path to output directory.

Format:

```
<dirOutBase>OutputDirectory</dirOutBase>
```

ProjectId: Mandatory. This is the project's name, it indicates the project basename directory and it's also the prefix of all files generated in the execution. This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<ProjectId>Name</ProjectId>
```

UserName: Optional. Default value: "Undefined". This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<UserName>User</UserName>
```

dataInDirpreProcess: Mandatory. It defines the absolute location of raw data directory.

Format:

```
<InDirPreProcess>RawDataDirectoryPath</InDirPreProcess>
```

Example:

```
<InDirPreProcess>/home/MyData</InDirPreProcess>
```

Sample: Mandatory. It defines all related characteristics of a sample. There must be one per analyzed sample. It can be more than one instance of Sample in the experiment configuration file, (i.e.: one per sample).

SampleName: Mandatory. This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

`<SampleName>Name</SampleName>`

SampleFiles: Mandatory. It represents the variable part on the input data name of the raw data files or the BAM basename filename without '.bam' suffix. If the sample is splitted in several files, user can append each of them with the separator ':'.

If RUBioSeq is executed on a cluster, the alignment steps will be parallelized by SampleFile.

Format:

`<SampleFiles>FileNameWithoutSuffix</SampleFiles>`

splitted case:

`<SampleFiles>FileNameWithoutSuffixAA:FileNameWithoutSuffixAB</SampleFiles>`

SampleSuffix: Mandatory. It represents the raw data name common suffix. If the input data are BAM files, this suffix must be '.bam'. If input data are SOLiD files (*.csfasta and *_QV.qual), this suffix must be 'solid'.

For example:

If we have two samples called s_4_sequence.txt and s_5_sequence.txt respectively, the common suffix will be '_sequence.txt'. If the sample are paired-end, raw suffix must not include the paired information (i.e. : _1/_R1 or _2/_R2 in Illumina case).

Format:

`<SampleSuffix>RawSuffix</SampleSuffix>`

Example:

`<SampleSuffix>_sequence.txt</SampleSuffix>`

SampleType: Mandatory. It represents reads type, single-end or paired-end. Values: 1(single-end), 2(paired-end).

Format:

`<SampleType>readType</SampleType>`

Example of complete Sample feature:


```
<Sample>  
  <SampleName>LC60RT215</SampleName>  
  <SampleFiles>LC60RT215-DNA_A01</SampleFiles>  
  <SampleSuffix>.fastq</SampleSuffix>  
  <SampleType>2</SampleType>  
</Sample>
```

CallingType: Optional. This is the genotype likelihoods calculation model to employ in GATK functions. Values: SNP, INDEL or BOTH. Default value: BOTH.

Format:
<CallingType>CallingType</CallingType>

GATKoutputMode: Optional. It represents the type of calls the GATK functions is going to output. Values: EMIT_VARIANTS_ONLY(default), EMIT_ALL_SITES and EMIT_ALL_CONFIDENT_SITES.

Format:
<GATKoutputMode>EMIT_VARIANTS_ONLY</GATKoutputMode>

rsFilter: Optional. This is a user-filter parameter to apply to the calling output file. If it is enabled (value = 1), dbSNP variants are removed from the output file, if it isn't, no filter is performed. Default: 0.

Format:
<rsFilter>0 or 1</rsFilter>

RUBioSeq_Mode: Optional. It enables the joint multisample execution. Values: 0, standalone execution and 1, multisample execution. Default value: 0.

Format:
<RUBioSeq_Mode>1 or 0</RUBioSeq_Mode>

fastqc: Optional. It activates FastQC module. This module performs this analysis for all samples in the experiment configuration file. The results are stored in Quality/FastQC subdirectory. Values: 1, activates FastQC; 0, disable the option. Default value: 0.

Format:
<fastqc>0 or 1</fastqc>

VEPFlag: Optional. If it is enabled (value = 1), Ensembl Variant Effect Predictor (VEP) over GATK output variants, is performed. Default value: 0. This analysis is only performed when reference genome is from hg19 release and the reference filename

contains the string *hg19* in its name, in other case, the analysis will be automatically disabled.

Format:

<VEPFlag>0 or 1</VEPFlag>

TCFlag: Optional. This flag enables Tumor/control analyses. Values: 1, enables Tumor/control comparison analysis. Default: 0.

Format:

<TCFlag>0 or 1</TCFlag>

Correct use of this module:

– Number of Sample elements must be even, and they must be sorted in Tumor/Control pairs.

Example:

```
<Sample>
  <SampleName>LC60RT215_Mutant</SampleName>
  <SampleFiles>LC60RT215-DNA_A01YT</SampleFiles>
  <SampleSuffix>.fastq</SampleSuffix>
  <SampleType>2</SampleType>
</Sample>
<Sample>
  <SampleName>LC60RT215_Control</SampleName>
  <SampleFiles>LC60RT215-DNA_A01GG</SampleFiles>
  <SampleSuffix>.fastq</SampleSuffix>
  <SampleType>2</SampleType>
</Sample>
```

The Tumor/control result vcf files will be stored in Tumor sample *calling/TC* subdirectory.

MDFlag: Optional. Only for advanced users. This flag enables or disables the mark duplicates step.

Values: 1, enables Markduplicates step; 0, disables Markduplicates step. Default: 1. It is recommended to disable this step if we have an targeted resequencing of a small region(a few hundred genes).

Format:

<MDFlag>0 or 1</MDFlag>

standCallConf: Optional. The minimum phred-scaled confidence threshold at which

variants not at 'trigger' track sites should be called. Default: 30.0.

Format:

`<standCallConf>Value</standCallConf>`

standEmitConf: Optional. The minimum phred-scaled confidence threshold at which variants not at 'trigger' track sites should be emitted (and filtered if less than the calling threshold). Default: 30.0.

Format:

`<standEmitConf>Value</standEmitConf>`

queueSGEProject: Optional. Only for SGE controlled systems. It represents the project name associated with the execution. This name must match with someone on the valid project names list. This list can be consulted executing the command: *qconf -sprjl*.

Format:

`<queueSGEProject>Name</queueSGEProject>`

VQSR block: Optional. It activates the VQSR step, in any other case, default Hard filter step will be applied. This parameter are composed of 3 compulsory sub-parameters that represent a list of sites for which VQSR applies a prior probability of being correct.

For SNPs we use both HapMap v3.3 and the Omni chip array from the 1000 Genomes Project as training data.

When modeling indels with the VQSR we use a training dataset that was created at the Broad by strictly curating the (Mills, Devine, Genome Research, 2011) dataset as well as adding in very high confidence indels from the 1000 Genomes Project.

Sub-parameters

Mills: Compulsory. It is the complete path to Mills_and_1000G_gold_standard.indels.hg19.sites.vcf file.

HapMap: Compulsory. It is the complete path to hapmap_3.3.hg19.sites.vcf file.

ThousandG: Compulsory. It is the complete path to 1000G_omni2.5.hg19.sites.vcf file.

Format:

`<VQSR>`

```
<Mills>/Volumes/RAID/NGS/HG19/VQSR/Mills_and_1000G_gold_standard.indels.hg19
.sites.vcf</Mills>
    <HapMap>/Volumes/RAID/NGS/HG19/VQSR/hapmap_3.3.hg19.sites.vcf</Ha
pMap>
    <ThousandG>/Volumes/RAID/NGS/HG19/VQSR/1000G_omni2.5.hg19.sites.vcf
</ThousandG>
</VQSR>
```

Hard filters block:

RUBioSeq provides four parameters to configure the GATK hard filtering of the pipeline:

DPmin and **minQual**: If they exist, these filters will be applied with the default GATK hard filters parameters and also with the customized hard filters.

DPmin: Optional. This is a user-filter parameter to apply to the calling output file. If it is enabled, value (>0) indicates the depth cutoff for the variants, if it isn't, no depth filter is performed.

Format:

```
<!-- HARD FILTERS block -->
<HardFilters>
    <DPmin>value</DPmin>
</HardFilters>
```

Example:

```
<!-- HARD FILTERS block -->
<HardFilters>
    <DPmin>30</DPmin>
</HardFilters>
```

minQual: Optional. This is a user-filter parameter to apply to the calling output file. If it is enabled, value (>0) indicates the minimum quality cutoff for the variants, if it isn't, no quality filter is performed.

Format:

```
<!-- HARD FILTERS block -->
<HardFilters>
    <minQual>value</minQual>
</HardFilters>
```

HfilterName[SNP/INDEL] and **HfilterRule[SNP/INDEL]**: These parameters are only recommended for advanced users. If these parameters are defined in the experiment config file, RUBioSeq's default GATK filters will be overwritten with the new user hard filters.

If the elements have the suffix 'SNP' the rules are applied to the SNPs variants, and if they have the suffix 'INDEL', the filters will be applied to the indels.

HfilterNameSNP/HfilterNameINDEL: Only for advanced users. Optional. This name must be a set of alphanumeric characters, no other character will be accepted. It must be followed by **HfilterRule[SNP/INDEL]** parameter. It represents the hard filter name associated with its corresponding hard filter rule.

HfilterRuleSNP/HfilterRuleINDEL: Only for advanced users. Optional. This parameter must be preceded by **HfilterName[SNP/INDEL]** parameter. It represents the hard filter rule associated with the previously declared hard filter name. This rule must be a JEXL expression, with the exception that '>' and '<' characters must be substituted respectively by '>' and '<'.

It can be more than one pair of **HfilterName[SNP/INDEL]/HfilterRule[SNP/INDEL]** in the experiment configuration file.

Format:

```
<!-- HARD FILTERS block -->
<HardFilters>
  <HfilterNameSNP>Name</HfilterNameSNP>
  <HfilterRuleSNP>JEXL</HfilterRuleSNP>
  <HfilterNameINDEL>Name</HfilterNameINDEL>
  <HfilterRuleINDEL>JEXL</HfilterRuleINDEL>
</HardFilters>
```

Example:

```
<!-- HARD FILTERS block -->
<HardFilters>
  <HfilterNameSNP>QDFilter</HfilterNameSNP>
  <HfilterRuleSNP>QD&gt; 2.0</HfilterRuleSNP>
  <HfilterNameSNP>ReadPosRankSum</HfilterNameSNP>
  <HfilterRuleSNP>ReadPosRankSum &lt; -20.0</HfilterRuleSNP>
</HardFilters>
```

3. VEP script configuration

User has to configure to files: registry.local and condel_SP.conf

registry.local

This file is located in *installation_path/RUBioSeq_v3.7/variantCalling/VEP*/registry.local*, it configures host, port, specie and user to create the connection to Ensembl database required by VEP script application. User must configure it properly.

Example file content:

```
species homo_sapiens
host     ensembl.db.ensembl.org
port     5306
user     anonymous
```

4. CONDEL PLUGIN

condel_SP.conf

This file is located in *installation_path/RUBioSeq_v3.7/variantCalling/VEP*/.vep/Plugins/config/Condel/config*,

User has to change the 'condel.dir' parameter to the corresponding location of the Condel's config directory, that is only change the path to RUBioSeq source code.

```
condel.dir='installationPath/RUBioSeq_v3.7/variantCalling/VEP73/.vep/Plugins/config/Condel/'
```

5. Program execution

Once we have the samples data, and our XML config files done, we can now execute the variant caller.

Command:

```
# pathToRUBioSeq/RUBioSeq.pl --analysis variantCalling --config <ExperimentConfigFile> ,
```

where *ExperimentConfigFile* is the absolute path to our XML experiment config file.

1. Program workflow

If all files, directories and parameters of the configuration files are well established, program will begin to execute samples in parallel if we are running on a HPC system scheduled by a SGE or PBS system (experimental) or sequentially if we aren't. (Corresponding parameter must be properly initialized).

The variant caller has been designed in a modular way, it divides the execution in four main levels:

- First level: Alignment phase and FastQC analysis.
- Second level: Duplicates marking, GATK realignment and recalibration.
- Third level: Variant calling,
- Fourth level: Tumor/control somatic/germline calls detection and variant effect predictor.

It accepts standalone level execution adding the *level* parameter to the command:

```
#pathToRUBioSeq/RUBioSeq.pl --analysis variantCalling --config <FicheroPrueba.xml>
--level <num> ,
```

where num is a number between 1 and 4, corresponding respectively to the stages explained above.

It is important to remark that when we execute level 2, 3 or 4, the previous level must have been executed previously.

1. *Input Data*

User must provide the workflow two kinds of input data files : the samples reads files (raw or aligned format) and the reference/annotation files.

a) Sample files:

The variant calling analysis accepts raw input data from Illumina and SOLiD platform or BAM files.

Raw input data.

The program accepts single-end or paired-end *.fastq files from Illumina platform and SOLiD F3 (single-end) or F3,F5-P2(paired-end) *.csfasta *.qual files.

Illumina

A minimal FASTQ file might look like this:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTGTTCAACTCACAGTTT
+
!*(((((***+))%%%+))%%%%).1***-+*)**55CCF>>>>>>CCCCCCC65
```

Sequences from the Illumina software use a systematic identifier:

@HWUSI-EAS100R:6:73:941:1973#0/1

HWUSI-EAS100R	the unique instrument name
6	flowcell lane
73	tile number within the flowcell lane
941	'x'-coordinate of the cluster within the tile
1973	'y'-coordinate of the cluster within the tile
#0	index number for a multiplexed sample (0 for no indexing)
/1	the member of a pair, /1 or /2 (<i>paired-end or mate-pair reads only</i>)

Versions of the Illumina pipeline since 1.4 appear to use #NNNNNN instead of #0 for the multiplex ID, where NNNNNN is the sequence of the multiplex tag.

With Casava 1.8 the format of the '@' line has changed:

@EAS139:136:FC706VJ:2:2104:15343:197393 1:Y:18:ATCACG

EAS139	the unique instrument name
136	the run id
FC706VJ	the flowcell id
2	flowcell lane
2104	tile number within the flowcell lane
15343	'x'-coordinate of the cluster within the tile
197393	'y'-coordinate of the cluster within the tile
1	the member of a pair, 1 or 2 (<i>paired-end or mate-pair reads only</i>)
Y	Y if the read fails filter (read is bad), N otherwise
18	0 when none of the control bits are on, otherwise it is an even number
ATCACG	index sequence

Note: With Casava 1.8, quality values must be coded in ASCII33; with previous Casava versions quality values must be coded in ASCII64.

SOLiD

CSFASTA files are FASTA files code in color space. QUAL files contain the corresponding qualities for the reads.

Accepted reads : F3 (single-end) y F3, F5-P2 (paired-end) in CSFASTA and QUAL format.

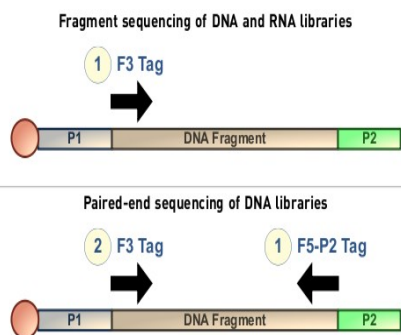


Figure 2: SOLiD read types

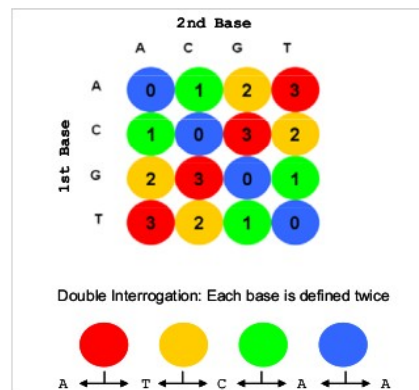


Figure 1: SOLiD Color space

For this kind of data the specific parameters are:

Platform: solid

RawSuffix: solid (the program only accepts files *.csfasta y *_QV.qual).

The others parameters must be configurated as usual.

Example: XML experiment file configuration.

Raw data directory location: /home/myRawData

Samples name: raw_1.fastq, raw_2.fastq.

Type: paired-end.

Platform: illumina

For this kind of data, the specific values(in bold) for the samples are the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- EXAMPLE RUBIOSEQ EXPERIMENT CONFIG FILE -->
<configData branch="SNV">
  <!-- GENOME REFERENCE PATH :: MANDATORY -->
  <GenRef>/Volumes/RAID/NGS/HG19/hg19.fa</GenRef>
  <!-- DBSNP ANNOTATION PATH :: MANDATORY -->
  <DbSnpAnnot>/Volumes/RAID/NGS/HG19/dbsnp_137.hg19.vcf</DbSnpAnnot>
  <!-- 1000 Genomes ANNOTATION PATH :: MANDATORY -->
  <Genomes1000Annot>/Volumes/RAID/NGS/HG19/1000G_omni2.5.hg19.vcf</Genomes1000Annot>
  <!-- REFSEQ ANNOTATION PATH :: MANDATORY -->
  <IndelAnnot>/Volumes/RAID/NGS/HG19/refseqhg19.rod</IndelAnnot>
  <!-- INTERVALS PATH :: OPTIONAL -->
  <Intervals>/Volumes/RAID/NGS/HG19/SureSelect_All_Exon_50mb_with_annotation.hg19.bed</Intervals>
  <!-- KNOWN INDELS FOR REALIGNING :: OPTIONAL -->
```

```

<KnownIndels>/Volumes/RAID/NGS/HG19/1000G_biallelic.indels.hg19.vcf</KnownIndels>
<KnownIndels>/Volumes/RAID/NGS/HG19/Mills_Devine_2hit.indels.hg19.sites.vcf</KnownIndels>
<!-- PLATFORM :: MANDATORY -->
<Platform>illumina</Platform>
<!-- checkCasava :: OPTIONAL. Set to 0 only if fastq reads have ascii33 qualities and they don't have standard
Casava format -->
<checkCasava>1</checkCasava>
<!-- OUTPUT DIRECTORY :: OPTIONAL, DEFAULT: Home directory -->
<dirOutBase>/home/ngs/</dirOutBase>
<!-- PROJECT NAME :: MANDATORY -->
<ProjectId>MyId</ProjectId>
<!-- USER NAME :: OPTIONAL(default Undefined) -->
<UserName>MyUser</UserName>
<!-- RAW DATA PATH :: MANDATORY -->
<InDirPreProcess>/home/myRawData</InDirPreProcess>
<Sample>
  <!-- SAMPLE NAME :: MANDATORY -->
  <SampleName>Sample</SampleName>
  <SampleFiles>raw</SampleFiles>
  <!-- SUFFIX :: MANDATORY -->
  <SampleSuffix>.fastq</SampleSuffix>
  <!-- READ TYPE - 1: single-end 2:paired-end :: MANDATORY -->
  <SampleType>2</SampleType>
</Sample>
<!-- CALL TYPE :: OPTIONAL (default BOTH) -->
<CallingType>BOTH</CallingType>
<!-- GATKoutputMode - variants:EMIT_VARIANTS_ONLY, others:EMIT_ALL_SITES, EMIT_ALL_CONFIDENT_SITES
::default (EMIT_VARIANTS_ONLY) -->
<GATKoutputMode>EMIT_VARIANTS_ONLY</GATKoutputMode>
<!-- Clean dbSNP output entries :: 1, clean dbSNPs (default 0) :: OPTIONAL -->
<rsFilter>1</rsFilter>
<!-- RUBioSeq_Mode Values: 0: standalone multisample, 1: joint multisample execution (default 0) :: OPTIONAL-->
<RUBioSeq_Mode>0</RUBioSeq_Mode>
<!-- Run fastqc analysis :: 1, run analysis (default 0) :: OPTIONAL -->
<fastqc>1</fastqc>
<!-- VEP analysis :: 1,execute analysis (default 0) :: OPTIONAL -->
<VEPFlag>1</VEPFlag>
<!-- Tumor/Control flag :: OnlyTumor and Germline analyses:: 1(default 0) :: OPTIONAL -->
<TCFlag>0</TCFlag>
<!-- Markduplicates flag (WARNING:: ONLY FOR ADVANCED USERS) Enable markduplicates step :: 1, Disable
markduplicates step :: (default 1) :: OPTIONAL -->
<MDFlag>1</MDFlag>
<!-- Minimum phred-scaled confidence threshold for calling (default 30.0) ::OPTIONAL -->
<standCallConf>30.0</standCallConf>
<!-- Minimum phred-scaled confidence threshold for emitting (default 30.0) ::OPTIONAL -->
<standEmitConf>30.0</standEmitConf>
<!-- Queue project :: OPTIONAL (default none) -->
<queueSGEProject>none</queueSGEProject>
<!-- Whole genome analyses filtering -->
<!--
<VQSR>
  <Mills>/Volumes/RAID/NGS/HG19/VQSR/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf</Mills>
  <HapMap>/Volumes/RAID/NGS/HG19/VQSR/hapmap_3.3.hg19.sites.vcf</HapMap>
  <ThousandG>/Volumes/RAID/NGS/HG19/VQSR/1000G_omni2.5.hg19.sites.vcf</ThousandG>
</VQSR-->
<!-- Whole exome and target sequencing analyses filtering -->
<HardFilters>
  <!-- VCF Depth filter :: OPTIONAL -->
  <DPmin>15</DPmin>

```

```

    <!-- VCF min quality filter :: OPTIONAL -->
    <minQual>100</minQual>
    <!-- Optional. ONLY FOR ADVANCED USERS. Hard filter custom name -->
    <!--<HfilterNameSNP>QDFilter</HfilterNameSNP>-->
    <!-- Optional. ONLY FOR ADVANCED USERS. Hard filter custom rule -->
    <!--<HfilterRuleSNP>QD<2.0</HfilterRuleSNP>-->
  </HardFilters>
</configData>

```

Processed input data.

The program also accepts input BAM files.

For this kind of data `RawSuffix` param must be `'.bam'`.

Example XML experiment file configuration.

Bam data directory location: `/home/myBamData`

Samples name: `bamfile1.bam`, `bamfile2.bam`.

Type: `paired-end`.

Reads platform: `illumina`

For this kind of data, the specific values(in bold) for the samples are the following:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- EXAMPLE RUBIOSEQ EXPERIMENT CONFIG FILE -->
<configData branch="SNV">
  <!-- GENOME REFERENCE PATH :: MANDATORY -->
  <GenRef>/Volumes/RAID/NGS/HG19/hg19.fa</GenRef>
  <!-- DBSNP ANNOTATION PATH :: MANDATORY -->
  <DbSnpAnnot>/Volumes/RAID/NGS/HG19/dbsnp_137.hg19.vcf</DbSnpAnnot>
  <!-- 1000 Genomes ANNOTATION PATH :: MANDATORY -->
  <Genomes1000Annot>/Volumes/RAID/NGS/HG19/1000G_omni2.5.hg19.vcf</Genomes1000Annot>
  <!-- REFSEQ ANNOTATION PATH :: MANDATORY -->
  <IndelAnnot>/Volumes/RAID/NGS/HG19/refseqhg19.rod</IndelAnnot>
  <!-- INTERVALS PATH :: OPTIONAL -->
  <Intervals>/Volumes/RAID/NGS/HG19/SureSelect_All_Exon_50mb_with_annotation.hg19.bed</Intervals>
  <!-- KNOWN INDELS FOR REALIGNING :: OPTIONAL -->
  <KnownIndels>/Volumes/RAID/NGS/HG19/1000G_biallelic.indels.hg19.vcf</KnownIndels>
  <KnownIndels>/Volumes/RAID/NGS/HG19/Mills_Devine_2hit.indels.hg19.sites.vcf</KnownIndels>
  <!-- PLATFORM :: MANDATORY -->
  <Platform>illumina</Platform>
  <!-- checkCasava :: OPTIONAL. Set to 0 only if fastq reads have ascii33 qualities and they don't have standard
Casava format -->
  <checkCasava>1</checkCasava>
  <!-- OUTPUT DIRECTORY :: OPTIONAL, DEFAULT: Home directory -->
  <dirOutBase>/home/ngs/</dirOutBase>
  <!-- PROJECT NAME :: MANDATORY -->
  <ProjectId>MyId</ProjectId>
  <!-- USER NAME :: OPTIONAL(default Undefined) -->
  <UserName>MyUser</UserName>
  <!-- RAW DATA PATH :: MANDATORY -->
  <InDirPreProcess>/home/myRawData</InDirPreProcess>
  <Sample>
    <!-- SAMPLE NAME :: MANDATORY -->
    <SampleName>Sample 1</SampleName>
  </Sample>
</configData>

```

```

    <SampleFiles>bamfile1</SampleFiles>
    <!-- SUFFIX :: MANDATORY -->
    <SampleSuffix>.bam</SampleSuffix>
    <!-- READ TYPE - 1: single-end 2:paired-end :: MANDATORY -->
    <SampleType>2</SampleType>

</Sample>
<Sample>
    <!-- SAMPLE NAME :: MANDATORY -->
    <SampleName>Sample2</SampleName>
    <SampleFiles>bamfile2</SampleFiles>
    <!-- SUFFIX :: MANDATORY -->
    <SampleSuffix>.bam</SampleSuffix>
    <!-- READ TYPE - 1: single-end 2:paired-end :: MANDATORY -->
    <SampleType>2</SampleType>

</Sample>
<!-- CALL TYPE :: OPTIONAL (default BOTH) -->
<CallingType>BOTH</CallingType>
<!-- GATKoutputMode - variants:EMIT_VARIANTS_ONLY, others:EMIT_ALL_SITES, EMIT_ALL_CONFIDENT_SITES
::default (EMIT_VARIANTS_ONLY) -->
<GATKoutputMode>EMIT_VARIANTS_ONLY</GATKoutputMode>
<!-- Clean dbSNP output entries :: 1, clean dbSNPs (default 0) :: OPTIONAL -->
<rsFilter>1</rsFilter>
<!-- RUBioSeq_Mode Values: 0: standalone multisample, 1: joint multisample execution (default 0) :: OPTIONAL-->
<RUBioSeq_Mode>0</RUBioSeq_Mode>
<!-- Run fastqc analysis :: 1, run analysis (default 0) :: OPTIONAL -->
<fastqc>1</fastqc>
<!-- VEP analysis :: 1,execute analysis (default 0) :: OPTIONAL -->
<VEPFlag>1</VEPFlag>
<!-- Tumor/Control flag :: OnlyTumor and Germline analyses:: 1 (default 0) :: OPTIONAL -->
<TCFlag>0</TCFlag>
<!-- Markduplicates flag (WARNING:: ONLY FOR ADVANCED USERS) Enable markduplicates step :: 1, Disable
markduplicates step :: 0 (default 1) :: OPTIONAL -->
<MDFlag>1</MDFlag>
<!-- Minimum phred-scaled confidence threshold for calling (default 30.0) ::OPTIONAL -->
<standCallConf>30.0</standCallConf>
<!-- Minimum phred-scaled confidence threshold for emitting (default 30.0) ::OPTIONAL -->
<standEmitConf>30.0</standEmitConf>
<!-- Queue project :: OPTIONAL (default none) -->
<queueSGEProject>none</queueSGEProject>
<!-- Whole genome analyses filtering -->
<!--
<VQSR>
    <Mills>/Volumes/RAID/NGS/HG19/VQSR/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf</Mills>
    <HapMap>/Volumes/RAID/NGS/HG19/VQSR/hapmap_3.3.hg19.sites.vcf</HapMap>
    <ThousandG>/Volumes/RAID/NGS/HG19/VQSR/1000G_omni2.5.hg19.sites.vcf</ThousandG>
</VQSR>-->
<!-- Whole exome and target sequencing analyses filtering -->
<HardFilters>
    <!-- VCF Depth filter :: OPTIONAL -->
    <DPmin>15</DPmin>
    <!-- VCF min quality filter :: OPTIONAL -->
    <minQual>100</minQual>
    <!-- Optional. ONLY FOR ADVANCED USERS. Hard filter custom name -->
    <!--<HfilterNameSNP>QDFilter</HfilterNameSNP>-->
    <!-- Optional. ONLY FOR ADVANCED USERS. Hard filter custom rule -->
    <!--<HfilterRuleSNP>QD<2.0</HfilterRuleSNP>-->

</HardFilters>
</configData>

```

b) Reference and annotation files.

These files represent the reference sequence and annotation files that RUBioSeq passes to the different programs in the workflow. The corresponding parameters in the experiment configuration file are: GenRef, DbSnpAnnot, Genomes1000Annot, IndelAnnot, Intervals and KnownIndels.

Look *Configuration Files* section of this manual for further information.

In order to avoid GATK software warnings and errors, user must follow these tips:

- Annotation and interval files identifiers (chromosomes) must be a subset of sequence identifiers list on reference sequences FASTA file.
- All annotation and intervals files must be sorted by position in the same reference sequence identifiers order.
- Be careful with files formatting, error extra-characters, empty lines or missing headers can produce internal program's parsing errors.

User can download the HG19 bundle at https://sourceforge.net/projects/rubioseq/files/HG19_bundle/.

Example:

Reference.fa:

```
>chr1
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...
>chr2
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...
>chr3
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...
```

Sequence identifiers: chr1, chr2, chr3

dbSNP_132.hg19.vcf:

```
[vcf header]
chr1 79050 rs62641299 G T . PASS dbSNPBuildID=129
chr1 79418 rs2691296 G C . PASS dbSNPBuildID=100
chr2 10361 rs114064093 G C . PASS dbSNPBuildID=132
chr2 10380 rs114647447 G T . PASS dbSNPBuildID=132
chr4 10052 rs113041763 T C . PASS dbSNPBuildID=132 ERROR!!
```

2. Program outputs

The variant caller has an output file organization in directories.

It creates a directory based on XML experiment file, on the location: /dirOutBase/ProjectId
If we are running on standalone multisample execution, one subdirectory per sample will be created. If we are running on joint multisample execution one subdirectory called *multi* will be created.

Every stage explained in 'Program workflow' section, generates a directory or group of directories, where files are stored:

preprocess directory: It stores the bwa + bfast reads alignment.

MD directory: It stores BAM file with duplicates removed.

realign directory : It stores BAM file after GATK realignment.

recal directory: It stores BAM file after GATK recalibration.

calling directory: It stores output vcf files.

a) Output files

When the variant calling analysis finishes, many VCF files have been created. In this section we are going to clarify what every file located on the *calling* directory represents.

Files:

***_annotated.vcf** - It represents the output file of the GATK's UnifiedGenotyper function. This file isn't a final output, it can contain false positives. It may be useful if user wants to compare the original variant without any filtering and the final results.

***.cleanNOTPASS*.vcf** - This is a final output file, it contains the variants that pass both GATK and user filters.

***.cleanNOTPASS.vep.vcf** - This is a final output file, it contains the variants on the *.cleanNOTPASS.vcf with the variant effect predictor(VEP) annotation.

***_recalibrated_somaticIndelTvsC.vcf** - This is a final output file, this file contains the Somatic Indel Detector analysis output, that is, the somatic indels Tumor versus Control detected by the tool.

***.eval_INDEL.gatkreport / *.eval_SNP.gatkreport** - Quality control metrics calculated by GATK's VarianEval function. These metrics include the number of raw or filtered SNP counts; ratio of transition mutations to transversions. These metrics are calculated from GATK-filtered VCF files.

***_recalibrated_indels.to_mask.vcf** - This is an auxiliary file. Do not remove, because an standalone level 3 execution wouldn't work.

Tumor/control files: If user has performed a Tumor/control analysis. These files will be created in Tumor sample *calling/TC* directory:

mergedTC.vcf - Germline and Somatic mutations without removing the non-PASS entries.

onlyTumor*cleanNOTPASS.vcf - only PASS Somatic mutations.

onlyTumor*.vep.vcf - PASS Somatic mutations with VEP information.

bothTC*cleanNOTPASS.vcf - only PASS Germline mutations with VEP information.

bothTC*.vep.vcf - PASS Germline mutations with VEP information.

onlyControl*cleanNOTPASS.vcf - only PASS mutations in control samples.

onlyControl*.vep.vcf - PASS control mutations with VEP information.

***_summary.html** - HTML with VEP summary and statistics.

3. *Quality and Control*

Different types of quality and control analyses are done in the variant calling analysis execution. Some of them are enabled by the corresponding parameter in the experiment configuration file, and others are automatically performed.

a) Fastqc analysis

Enabled by the user. FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

FastQC data outputs are located in: /dirOutBase/ProjectId/SampleName/Quality/FastQC.

b) Bam files validation

Automatic. Read a SAM or BAM file and report on its validity. This control analysis is performed by ValidateSamFile function (picardTools) and it's executed over /dirOutBase/ProjectId/preprocess/*_first_aligned.bam and over /dirOutBase/ProjectId/recal/*_recalibrated.bam. This function generates an output file in the input file directory with the name *_validation.

4. RUBIOSEQ filters

a) Depth filter and minQUAL filter

User can filter program result variants that don't have a minimum depth-filter or a minimum quality (VCF QUAL column) . These filters are activated initializing DPmin and minQUAL parameters respectively to a value greater than zero.

b) dbSNP filter

User can filter dbSNP known variants from VCF output file. It is activated initializing rsFilter parameter to one.

c) Change Hard Filters

RUBIOSEQ provides two parameters (HfilterName[SNP/INDEL] and HfilterRule[SNP/INDEL]) to configure the default GATK hard filtering of the pipeline. These parameters are only recommended for advanced users. If these parameters are defined in the experiment config file, RUBIOSEQ's default GATK filters will be overwritten with the new user hard filters. See *Configuration files* section for further information.

5. Variant Quality Score Recalibration or Hard filters?

These steps are designed to separate out the false positive machine artifacts from the true positive genetic variants. The selection of one or another depends of the experiment type.

Following the recommendations of GATK developers, VQSR should be used on whole-genome experiments, or in whole-exome projects with at least 30 cases. For small whole-exome projects or small target experiments, the hard filters are the recommended step.

6. Execution modes

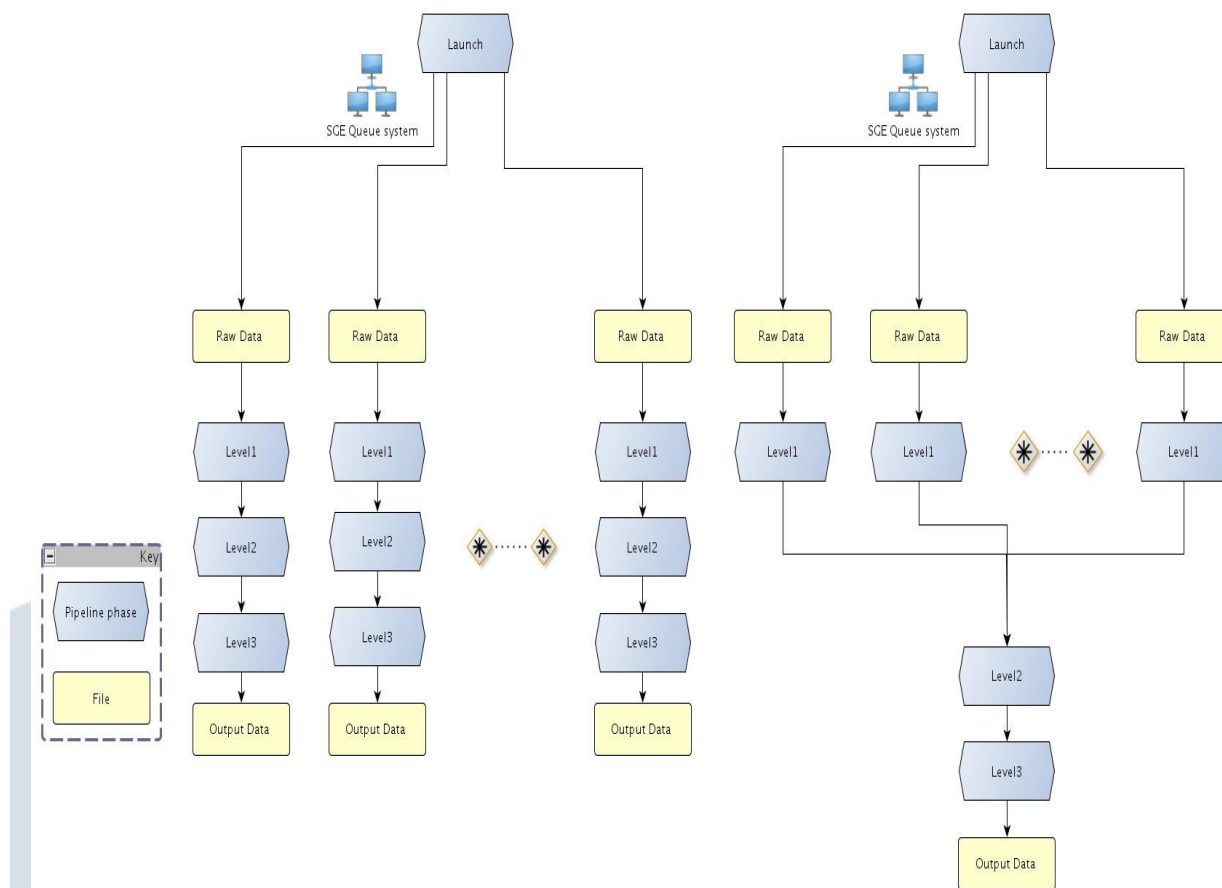
If user is running the program on a HPC system, SGE scheduled or PBS scheduled, the variant caller permits two execution modes:

Standalone multisample execution: Parallel execution of one or more samples. In this mode, each sample is treated independly and we get one VCF output file per sample.

Joint multisample execution: Parallel execution of two or more samples. In this mode, all

sample information is used together to enrich the VCF output file. We get only one VCF output file.

Figure 3: Parallel execution variant calling workflow



The next figure represents graphically the execution modes:

If user is running on a typical workstation, both standalone multisample and joint multisample execution can be performed, but it will be done sequentially.

6. RUBioSeq. CnvCaller.

1. Introduction

The cnv calling workflow developed in this program automatically executes all involved stages in this process using state-of-art software.

Our process includes three main stages:

1. Short-read alignment with a combination of BWA and BFAST aligners.
2. GATK-based realign, marks duplication and recalibration steps.
3. Cnvs detection(CONTRA).

Our program accepts raw data in Fastq format from Illumina/ion platform, raw data from SOLiD platform (F3, F5-P2 reads) or BAM files. It generates an vcf result file with the detected cnvs.

Quality and errors check points for input data and files created during the execution are also performed.

This program has been design to execute on a HPC, scheduled by an SGE system, this design allows a parallel multiple sample execution in order to reduce the processing time. It can be also executed on a HPC with PBS system and on a standard workstation in sequential mode.

2. Prerequisites

Operating System: UNIX.

All programs listed below and their corresponding dependencies must be correctly installed.

1. Bwa 0.7.10. <http://bio-bwa.sourceforge.net/>
2. Bfast-bwa 0.7.0b . <https://sourceforge.net/p/bfast/bfast/ci/master/tarball>
3. CONTRA.V2.0.3. <https://sourceforge.net/apps/mediawiki/contra-cnv/index.php?title=CONTRA: Copy Number Analysis for Targeted Resequencing>
4. Bedtools(v2.11.2)(provided inside CONTRA tarball). <http://code.google.com/p/bedtools/>
5. GenomeAnalysisTK-3.1-1
6. R environment 2.14 or higher. <http://www.r-project.org/>
7. RScript 2.14.0 or higher. <http://www.rforge.net/rscript/files> . Executable directory must be added to the PATH environment variable.

3. *Before running a cnv analysis for first time*

User have to perform some previous steps before running the program for first time:

1. Index reference

Reference genome indexes for BWA and BFAST aligners must be prepared.

These steps depends on the raw data we are going to analyzed: Illumina/Ion or ABI SOLiD platform.

These steps must be done only once.

User can execute by his own or in case of SGE scheduled systems, user can adapt the auxiliary scripts provided in RUBioSeq code in order to facilitate the task.

Reference genome and its index files must be located in the same directory.

Steps:

See section *Before running a snv analysis for first time*.

2. Configure program paths

User must configure configProgramPaths.xml file with the corresponding paths to the programs. Configuration program file is explained in the next section.

4. *Configuration files*

There are two configuration files: configProgramPaths.xml to configure applications paths and queue schedulers management and Experiment.xml, with all specific parameters for the execution.

1. configProgramPaths.xml

This XML format file configures applications paths used by the program, it also initializes queue schedulers parameters for our specific system.

It must be located in *RUBioSeqPath/variantCalling/config/cnv*.

User must configure this file properly to adapt the application paths and scheduler.

Configuration file example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RUBioSeq variantCaller CONFIG FILE -->
<configData>
```

```

<bwaPath>/opt/NGS/bwa/0.7.10/</bwaPath>
<samtoolsPath>/opt/NGS/samtools/0.1.19/</samtoolsPath>
<gatkpath>/home/guest/soft/GenomeAnalysisTK-3.1-1/</gatkpath>
<picardPath>/home/guest/soft/picard-tools-1.60/</picardPath>
<BFASTPath>/opt/NGS/bfast+bwa/0.7.0b/bin/</BFASTPath>
<BEDToolsPath>/home/guest/soft/CONTRA.v2.0.3/BEDTools-Version-
2.11.2/bin/</BEDToolsPath>
<CONTRAPath>/home/guest/soft/CONTRA.v2.0.3/</CONTRAPath>
<fastqcPath>/home/guest/soft/FastQC/</fastqcPath>
<nthr>4</nthr>
<javaRam>-Xmx16G</javaRam>
<queueSystem>none</queueSystem>
<queueName>none</queueName>
<!--
<multicoreName>multicore</multicoreName>
<multicoreNumber>4</multicoreNumber>
-->

</configData>

```

Fields:

All this fields are mandatory. They initialize the directory where binary or script of the corresponding program is located:

bwaPath, **samtoolsPath**, **gatkpath**, **picardPath**, **BFASTPath**, **fastqcPath**,..

Note: BEDTools must be the version included in CONTRA sources (BEDTools Version-2.11.2).

queueSystem: represents queue scheduler type. The accepted values are: SGE, PBS and none(for execution in non-queue controlled systems).

ensemblPath: Base directory to ensembl, ensembl-compara, ensembl-variation and ensembl-functgenomics.

If queueSystem is initialized to *none* the multisample execution is not performed in a parallel way, but in a secuencially execution, because there is no scheduler to synchronize the programs.

These other fields are optional:

nthr : represents thread number parameter for all compatible applications. Default value: 1.

javaRam: Maximum RAM memory limit for java applications. Default value: -Xmx16G.

queueName: Queue's name in which tasks are going to run. Default value: normal.

multicoreName: Only valid for SGE systems. It represents the name of the SGE's parallel environment. Default value: multicore (disabled). To use this parameter, SGE manager must create a parallel environment with the multicoreName feature.

multicoreNumber: Only valid for SGE systems. Represents the number of slots the user wants to keep for his execution. Default value: -1 (disabled). To use this feature, SGE manager must create a parallel environment called *multicore*.

2. Experiment.xml

Experiment.xml configures all the cnv caller parameters to adapt the script to the specific input data and user desired configuration.

This file must follow the format *RUBioSeqPath/variantCalling/config/cnv/Template.xml*.

It must have at least all the mandatory fields. Name and location of this file is a user choice. This file is one of the two parameters of the cnv caller.

User has also to specify the branch type in the XML main element configData:

```
<configData branch="CNV">.
```

File fields:

GenRef: Mandatory. It initializes the absolute path to reference genome. Genome indexes must be located in the same directory.

Format:

```
<GenRef>ReferenceGenomePath</GenRef>
```

Example:

```
<GenRef>/Volumes/RAID/Soft/NGS/resources/Homo_sapiens_assembly18.fasta</GenRef>
```

DbSnpAnnot: Mandatory. It represents the absolute path to the variants annotation file (dbSNP). It must be in ROD or VCF format.

Format:

```
<DbSnpAnnot>DbSnpPath</DbSnpAnnot>
```

Example:

```
<DbSnpAnnot>/Volumes/RAID/Soft/NGS/resources/dbsnp_129_hg18.rod</DbSnpAnnot>
```

IndelAnnot: Mandatory. It represents the absolute path to REFSEQ annotation file. It must be in ROD or VCF format.

Format:

```
<IndelAnnot>RefseqFilePath</IndelAnnot>
```

Example:

```
<IndelAnnot>/Volumes/RAID/Soft/NGS/GATK-1.0.4388/resources/refSeqFinal_hg18.rod</IndelAnnot>
```

Intervals: Mandatory. Target region definition file. File must be in BED format. Four columns must be specified: Chromosome, ChromStart, ChromEnd, Name. Name can be anything, and is used for annotation. No header line is expected.

Format:

```
<Intervals>AbsolutePathToIntervalsfile</Intervals>
```

Example:

```
<Intervals>/Volumes/RAID/Homes/target.bed</Intervals>
```

KnownIndels: Optional. It represents input VCF files with known indels. This field can appear one or more times. These files will be used in GATK realigner phase.

Format:

```
<KnownIndels>AbsolutePathToKnownIndels</KnownIndels>
```

Example:

```
<KnownIndels>/Volumes/RAID/NGS/HG19/Mills_Devine_2hit.indels.hg19.sites.vcf</KnownIndels>
```

MDFlag: Optional. Only for advanced users. This flag enables or disables the mark duplicates step.

Values: 1, enables Markduplicates step; 0, disables Markduplicates step. Default: 1.

Format:

```
<MDFlag>0 or 1</MDFlag>
```

Platform: Mandatory. Represents NGS technology that generates input raw data. Valid values: illumina, solid or ion.

Format:

```
<Platform>platform</Platform>
```

checkCasava: Optional. Set to 0 **only** if fastq reads don't have standard Illumina's Casava format. Reads must have ASCII33 qualities. Default value :1.

Format:

```
<checkCasava>1</checkCasava>
```

dirOutBase: Mandatory. Absolute path to output directory.

Format:

```
<dirOutBase>OutputDirectory</dirOutBase>
```

ProjectId: Mandatory. This is the project's name, it indicates the project basename directory and it's also the prefix of all files generated in the execution. This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<ProjectId>Name</ProjectId>
```

UserName: Optional. Default value: "Undefined". This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<UserName>User</UserName>
```

InDirPreProcess: Mandatory. It defines the absolute location of raw data directory.

Format:

```
<InDirPreProcess>RawDataDirectoryPath</InDirPreProcess>
```

Example:

```
<InDirPreProcess>/home/MyData</InDirPreProcess>
```

Sample: Mandatory. It defines all related characteristics of a sample. There must be one per analyzed sample. It can be more than one instance of Sample in the experiment configuration file, (i.e.: one per sample).

SampleName: Mandatory. This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<SampleName>Name</SampleName>
```

SampleFiles: Mandatory. It represents the variable part on the input data name of the raw data files or the BAM basename filename without '.bam' suffix. If

the sample is splitted in several files, user can append each of them with the separator ':'. If RUBioSeq is executed on a cluster, the alignment steps will be parallelized by `SampleFile`.

Format:

```
<SampleFiles>FileNameWithoutSuffix</SampleFiles>
```

Splitted case:

```
<SampleFiles>FileNameWithoutSuffixAA:FileNameWithoutSuffixAB</SampleFiles>
```

SampleSuffix: Mandatory. It represents the raw data name common suffix. If the input data are BAM files, this suffix must be '.bam'. If input data are SOLiD files (*.csfasta and *_QV.qual), this suffix must be 'solid'.

For example:

If we have two samples called `s_4_sequence.txt` and `s_5_sequence.txt` respectively, the common suffix will be `'_sequence.txt'`. If the sample are paired-end, raw suffix must not include the paired information (i.e. : `_1/_R1` or `_2/_R2` in Illumina case).

Format:

```
<SampleSuffix>RawSuffix</SampleSuffix>
```

Example:

```
<SampleSuffix>_sequence.txt</SampleSuffix>
```

SampleType: Mandatory. It represents reads type, single-end or paired-end.
Values: 1(single-end), 2(paired-end).

Format:

```
<SampleType>readType</SampleType>
```

Example of complete Sample feature:

```
<Sample>
  <SampleName>LC60RT215</SampleName>
  <SampleFiles>LC60RT215-DNA_A01</SampleFiles>
  <SampleSuffix>.fastq</SampleSuffix>
  <SampleType>2</SampleType>
</Sample>
```


If no baseline parameter is provided, then the samples must be sorted in Test-Control pairs and that will be the pairs to perform the cnv calling with CONTRA.

Example:

```
<Sample>
  <SampleName>LC60RT215_Mutant</SampleName>
  <SampleFiles>LC60RT215-DNA_A01YT</SampleFiles>
  <SampleSuffix>.fastq</SampleSuffix>
  <SampleType>2</SampleType>
</Sample>
<Sample>
  <SampleName>LC60RT215_Control</SampleName>
  <SampleFiles>LC60RT215-DNA_A01GG</SampleFiles>
  <SampleSuffix>.fastq</SampleSuffix>
  <SampleType>2</SampleType>
</Sample>
```

fastqc: Optional. It activates FastQC module. This module performs this analysis for all samples in the experiment configuration file. The results are stored in `dirOut/projectID/SampleName/Quality/FastQC` subdirectory. Values: 1, activates FastQC; 0, disable the option. Default value: 0.

Format:

```
<fastqc>0 or 1</fastqc>
```

extraContra: Optional. With this option user can pass any **optional** parameters to CONTRA software.

Format:

```
<extraContra>CONTRA parameters</extraContra>,
```

where CONTRA parameters are, CONTRA **optional** parameters.

Consecutive hyphens in the original parameters must be preceded in the experiment XML file by a backslash in order to avoid XML constraints.

Example:

```
<extraContra>\-\plot \-\minNBases 20</extraContra>
```

baseline: Optional. User can define a baseline file (in BED format) as the control sample. User can create their own baseline files from a set of BAMfiles with the baseline script provided in CONTRA software.

If a baseline file is set in the experiment XML file, then all samples defined in `Sample_Seed` are considered as independent test samples, and the baseline will be their control file.

queueSGEProject: Optional. Only for SGE controlled systems. It represents the project name associated with the execution. This name must match with someone on the valid project names list. This list can be consulted executing the command: `qconf -sprjl`.

Format:

`<queueSGEProject>Name</queueSGEProject>`

5. Program execution

Once we have the samples data, and our XML config files done, we can now execute the snv caller.

Command:

```
# pathToRUBioSeq/RUBioSeq.pl --analysis cnvCalling --config <ExperimentConfigFile> ,
```

where *ExperimentConfigFile* is the absolute path to our XML experiment config file.

1. Program workflow

If all files, directories and parameters of the configuration files are well established, program will begin to execute samples in parallel if we are running on a HPC system scheduled by a SGE or PBS system (experimental) or sequentially if we aren't. (Corresponding parameter must be properly initialized).

The cnv caller has been designed in a modular way, it divides the execution in three main stages:

- First stage: Alignment phase and FastQC analysis.
- Second stage: Duplicates marking, GATK realignment and recalibration.
- Third stage: Cnv calling with CONTRA software.

It accepts standalone level execution adding the *level* parameter to the command:

```
#pathToRUBioSeq/RUBioSeq.pl --analysis cnvCalling --config <FicheroPrueba.xml> --level <num> ,
```

where num is a number between 1 and 3, corresponding respectively to the stages

explained above.

It is important to remark that when we execute level 2 or 3, the previous level must have been executed previously.

1. *Program outputs*

The cnv caller has an output file organization in directories.

It creates a directory based on XML experiment file, on the location: /dirOutBase/ProjectId/Sample

Every stage explained in 'Program workflow' section, generates a directory or group of directories, where files are stored:

preprocess directory: It stores the bwa + bfast reads alignment.

MD directory: It stores BAM file with duplicates removed.

realign directory: It stores BAM file after GATK realignment.

recal directory: It stores BAM file after GATK recalibration.

calling directory: It stores output vcf files.

a) **Output files**

When the cnv calling analysis finishes, the output CONTRA files generated are stored in calling/cnv directory.

2. *Quality and Control*

Different types of quality and control analyses are done in the snv calling analysis execution. Some of them are enabled by the corresponding parameter in the experiment configuration file, and others are automatically performed.

- Fastqc analysis
- Bam files validation
- Recalibration control

See section Quality and Control on Snv Caller part for further information.

7. RUBIOSEQ. CHIPSEQ Caller.

1. Introduction

The CHIPseq calling workflow developed in this program automatically executes all involved stages in this process using state-of-art software.

Our process includes three main stages:

1. Short-read alignment with a combination of BWA aligner.
2. Duplication marking steps.
3. Normalization step
4. Peak-calling with MACS2 and CCAT for sharp and broad peaks respectively and annotation with PeakAnnotator.

Our program accepts raw data in Fastq format from Illumina/ion platform, raw data from SOLiD platform (F3, F5-P2 reads) or BAM files. It generates the BED files with the detected peaks.

Quality and errors check points for input data and files created during the execution are also performed.

This program has been design to execute on a HPC, scheduled by an SGE system, this design allows a parallel multiple sample execution in order to reduce the processing time. It can be also executed on a HPC with PBS system and on a standard workstation in sequential mode.

2. Prerequisites

Operating System: UNIX.

All programs listed below and their corresponding dependencies must be correctly installed.

1. Bwa 0.7.10. <http://bio-bwa.sourceforge.net/>
2. MACS 2.0.10.20130712 (tag:beta) <https://github.com/taoliu/MACS>
3. CCAT 3.0. <http://cmb.gis.a-star.edu.sg/CHIPSeq/paperCCAT.htm>
4. BEDTools 2.16.2. <http://code.google.com/p/bedtools/>
5. BedGraphToBigWig. <http://hgdownload.cse.ucsc.edu/admin/exe/>
6. IDR. <https://sites.google.com/site/anshulkundaje/projects/idr/>
7. PeakAnnotator 1.4 Java.
<http://www.bioinformatics.org/peakanalyzer/wiki/Main/Download>
8. R environment 2.14 or higher. <http://www.r-project.org/>
9. RScript 2.14.0 or higher. <http://www.rforge.net/rscript/files> .Executable directory

must be added to the PATH environment variable.

3. *Before running a ChIPseq analysis for first time*

User have to perform some previous steps before running the program for first time:

1. *Index reference.*

Reference genome indexes for **BWA** aligner must be prepared.

This step must be done only once.

Reference genome and its index files must be located in the same directory.

Steps:

See section *Before running a snv analysis for first time*.

2. *Configure program paths*

User must configure configProgramPaths.xml file with the corresponding paths to the programs. Configuration program file is explained in the next section.

4. *Configuration files*

There are two configuration files: configProgramPaths.xml to configure applications paths and queue schedulers management and Experiment.xml, with all specific parameters for the execution.

1. **configProgramPaths.xml**

This XML format file configures applications paths used by the program, it also initializes queue schedulers parameters for our specific system.

It must be located in *RUBioSeqPath/ChIPseq/config/*

User must configure this file properly to adapt the application paths and scheduler.

Configuration file example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RUBioSeq ChIPSeq CONFIG FILE -->
<configData>
  <bwaPath>/Volumes/RAID/Soft/Linux_x86_64/NGS/bwa/0.7.10/</bwaPath>

  <samtoolsPath>/Volumes/RAID/Soft/Linux_x86_64/NGS/samtools/0.1.19/</samtoolsPath>
```

```

<picardPath>/Volumes/RAID/Soft/Linux_x86_64/NGS/picardtools/1.104/</picardPath>
  <MACSPath>/home/mrubioc/soft/bin/</MACSPath>
  <PythonPath>/home/mrubioc/soft/lib/python2.7/site-packages</PythonPath>

<BEDToolsPath>/Volumes/RAID/Soft/Linux_x86_64/NGS/BEDTools/2.16.2/bin/</BEDToolsPath
>
  <CCATPath>/home/mrubioc/soft/CCAT3.0/bin/</CCATPath>
  <BedgraphtoWigPath>/home/mrubioc/soft/</BedgraphtoWigPath>
  <IDRPath>/home/mrubioc/soft/idrCode/</IDRPath>

<PeakAnnotatorPath>/home/mrubioc/soft/PeakAnnotator_Java_1.4/</PeakAnnotatorPath>
  <fastqcPath>/Volumes/RAID/Soft/Linux_x86_64/NGS/FastQC/0.10.0/</fastqcPath>
  <nthr>4</nthr>
  <javaRam>-Xmx16G</javaRam>
  <queueSystem>SGE</queueSystem>
  <queueName>ngs</queueName>
  <!-<multicoreName>multicore</multicoreName>
  <multicoreNumber>4</multicoreNumber>-->
</configData>

```

Fields:

All program paths fields are mandatory. They initialize the directory where binary or script of the corresponding program is located:

bwaPath, **samtoolsPath**, **picardPath**, **fastqcPath**, etc.

queueSystem: represents queue scheduler type. The accepted values are: SGE, PBS and none(for execution in non-queue controlled systems).

If *queueSystem* is initialized to *none* the multisample execution is not performed in a parallel way, but in a sequentially execution, because there is no scheduler to synchronize the programs.

These other fields are optional:

nthr : represents thread number parameter for all compatible applications. Default value: 1.

javaRam: Maximum RAM memory limit for java applications. Default value: -Xmx16G.

queueName: Queue's name in which tasks are going to run. Default value: normal.

multicoreName: Only valid for SGE systems. It represents the name of the SGE's parallel environment. Default value: multicore (disabled). To use this parameter, SGE manager must create a parallel environment with the multicoreName feature.

multicoreNumber: Only valid for SGE systems. Represents the number of slots the user want to keep for his execution. Default value: -1 (disabled). To use this feature, SGE manager must create a parallel environment called *multicore*.

2. Experiment.xml

Experiment.xml configures all the ChIPseq caller parameters to adapt the script to the specific input data and user desired configuration.

An example of a ChIPseq experiment XML file can be found in: *RUBioSeqPath/ChIPseq/config/Template.xml*.

It must have at least all the mandatory fields. Name and location of this file is a user choice. This file is one of the two parameters of RubioSeq's ChIPseq caller.

User has also to specify the branch type in the XML main element configData:

```
<configData branch="ChipSeq">.
```

File fields:

GenRef: Mandatory. It initializes the absolute path to reference genome. Genome indexes must be located in the same directory.

Format:

```
<GenRef>ReferenceGenomePath</GenRef>
```

Example:

```
<GenRef>/Volumes/RAID/Soft/NGS/resources/Homo_sapiens_assembly18.fasta</GenRef>
```

Platform: Mandatory. Represents NGS technology that generates input raw data. Valid values: illumina, solid or ion.

Format:

```
<Platform>platform</Platform>
```

checkCasava: Optional. Set to 0 only if fastq reads don't have standard Illumina's Casava format. Reads must have ASCII33 qualities. Default value :1.

Format:

```
<checkCasava>1</checkCasava>
```

dirOutBase: Mandatory. Absolute path to output directory.

Format:

```
<dirOutBase>OutputDirectory</dirOutBase>
```

ProjectId: Mandatory. This is the project's name, it indicates the project basename directory and it's also the prefix of all files generated in the execution. This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<ProjectId>Name</ProjectId>
```

UserName: Optional. Default value: "Undefined". This name must be a set of alphanumeric characters, no other character will be accepted.

Format:

```
<UserName>User</UserName>
```

InDirPreProcess: Mandatory. It defines the absolute location of raw data directory.

Format:

```
<InDirPreProcess>RawDataDirectoryPath</InDirPreProcess>
```

Example:

```
<InDirPreProcess>/home/MyData</InDirPreProcess>
```

Experiment

Mandatory. There must be at least one Experiment per file. This element represent a complete ChipSeq experiment. It is composed by a set of ChipSeqUnits (at least one), all samples described in an Experiment, will be normalized together.

If ReplicatesFlag is activated, the program will assume all the defined SampleTreatments are replicates and the IDR analysis will be perform.

If RUBioSeq is executed on a cluster, execution will be parallelized by Experiment.

```
<Experiment>
  <ChipSeqUnit>
    ...
  </ChipSeqUnit>
  <ChipSeqUnit>
    ...
  </ChipSeqUnit>
  <ReplicatesFlag>0</ReplicatesFlag>
</Experiment>
```

ReplicatesFlag:

Mandatory. If it's activated, samples inside the Experiment element will be treated as replicates and RUBioSeq will execute IDR Analysis for replicates. Values: 1, activates Replicates analysis; 0, disable the option.

Format:

```
<ReplicatesFlag>0</ReplicatesFlag>
```

Important note: If the peaks number called by MACS2 is less than 100K, protocol recommends to add the parameters: ' -p 1e-3 -to-large' to perform a correct IDR analysis:

```
<MACS_extraArgs>-p 1e-3 -to-large</MACS_extraArgs>
```

ChipSeqUnit:

Mandatory. It defines the basic data for a ChipSeq analysis. It is composed by SampleTreatment (compulsory) and the SampleInput (optional)

<ChipSeqUnit>

```

  <SampleTreatment>
  <!-- SAMPLE NAME :: MANDATORY -->
  <SampleName>MA9-K4S</SampleName>
  <SampleFiles>MA9-K4</SampleFiles>
  <!-- SUFFIX :: MANDATORY -->
  <SampleSuffix>_sequence.txt</SampleSuffix>
  <!-- READ TYPE - 1: single-end 2:paired-end :: MANDATORY -->
  <SampleType>1</SampleType>
</SampleTreatment>
  <SampleInput>
  <!-- SAMPLE NAME :: MANDATORY -->
  <SampleName>inputT</SampleName>
  <SampleFiles>input</SampleFiles>
  <!-- SUFFIX :: MANDATORY -->
  <SampleSuffix>_sequence.txt</SampleSuffix>
  <!-- READ TYPE - 1: single-end 2:paired-end :: MANDATORY -->
  <SampleType>1</SampleType>
</SampleInput>
</ChipSeqUnit>

```

SampleTreatment:

Mandatory. Only one, per ChipSeqUnit. It defines the treatment sample for ChIPseq analysis. The type of this element is SampleInfo.

Example of complete SampleInput feature:

```

  <SampleTreatment>
    <SampleName>LC60RT215</SampleName>
    <SampleFiles>LC60RT215-DNA_A01</SampleFiles>
    <SampleSuffix>.fastq</SampleSuffix>
    <SampleType>2</SampleType>
  </SampleTreatment>

```

SampleInput:

Optional. It defines the input/control sample for ChIPseq analysis. The type of this element is SampleInfo.

Note: SampleInput must be set for broad peak-calling.

Example of complete SampleInput feature:

```
<SampleInput>
  <SampleName>Input</SampleName>
  <SampleFiles>LC60RT215-DNA_A01</SampleFiles>
  <SampleSuffix>.fastq</SampleSuffix>
  <SampleType>2</SampleType>
</SampleInput>
```

SampleInfo Type:

It is composed by:

SampleName: Mandatory. This name must be a set of alphanumeric characters, no other character will be accepted. If SampleName is repeated inside the same Experiment element, the program will assume that it is the same sample, all steps (alignment, etc.) will be done only once for the sample.

Format:

```
<SampleName>Name</SampleName>
```

SampleFiles: Mandatory. It represents the variable part on the input data name of the raw data files or the BAM basename filename without '.bam' suffix. If the sample is splitted in several files, user can append each of them with the separator ':'.

If RUBioSeq is executed on a cluster, the alignment steps will be parallelized by SampleFile.

Format:

```
<SampleFiles>FileNameWithoutSuffix</SampleFiles>
```

Splitted case:

```
<SampleFiles>FileNameWithoutSuffixAA:FileNameWithoutSuffixAB</SampleFiles>
```

SampleSuffix: Mandatory. It represents the raw data name common suffix. If the input data are BAM files, this suffix must be '.bam'. If input data are SOLiD files (*.csfasta and *_QV.qual), this suffix must be 'solid'.

For example:

If we have two samples called s_4_sequence.txt and s_5_sequence.txt respectively, the common suffix will be '_sequence.txt'. If the sample are paired-end, raw suffix must not include the paired information (i.e. : _1/_R1 or _2/_R2 in

Illumina case).

Format:

```
<SampleSuffix>RawSuffix</SampleSuffix>
```

Example:

```
<SampleSuffix>_sequence.txt</SampleSuffix>
```

SampleType: Mandatory. It represents reads type, single-end or paired-end.
Values: 1(single-end), 2(paired-end).

Format:

```
<SampleType>readType</SampleType>
```

chromSize: Mandatory. Path to a tabular file with chromosome name and size.

Format:

```
<chromSize>/home/mrubioc/Developing3.7/hg19.chromSize</chromSize>
```

Tab file format:

```
chr1 249250621
chr2 243199373
chr3 198022430
```

peakAnalysis: Optional. Values: sharp, that enables sharp peak analysis, broad, for broad peak analysis and both, that activates sharp and broad peak analyses. Sharp peaks will be calculated with macs2 and broad peaks, with CCAT. Default value: sharp

Format :

```
<peakAnalysis>sharp or broad or both</peakAnalysis>
```

CCAT_config_file: Optional. Complete path to CCAT software config file. Configure if you want to perform a broad peak analysis.

You can find an example of this file in: *RUBioSeqPath/ChIPseq/config/configCCAT.txt*

Format:

```
<CCAT_config_file>/home/mrubioc/Developing3.7/ChIPseqTest/dataTest/configCCAT.txt</CCAT_config_file>
```

MACS_extraArgs: Optional. MACS2 extra-options provided by the user. The different parameters will be separated with spaces. Note: parameters -bdg/-B and -control/-c are not allowed, they will be automatically passed to MACS2.

Example:

```
<MACS_extraArgs>\-\-nomodel \-\-shiftsize 90</MACS_extraArgs>
```

Consecutive hyphens in the original parameters must be preceded in the experiment XML file by a backslash in order to avoid XML constraints.

Important note: If the peaks number called by MACS2 is less than 100K, protocol recommends to add the parameters: ' -p 1e-3 -to-large' to perform a correct IDR analysis:

```
<MACS_extraArgs>-p 1e-3 \-\-to-large</MACS_extraArgs>
```

fastqc: Optional. It activates FastQC module. This module performs this analysis for all samples in the experiment configuration file. The results are stored in `dirOut/projectID/SampleName/Quality/FastQC` subdirectory. Values: 1, activates FastQC; 0, disable the option. Default value: 0.

Format:

```
<fastqc>0 or 1</fastqc>
```

annotFile: Optional. Complete path to a GTF/BED peak annotation file. If it exists RUBioSeq performs the peak annotation.

Example:

```
<annotFile>/home/mrubioc/Developing3.7/ChIPseqTest/Homo_sapiens.GRCh37.58.gtf</annotFile>
```

queueSGEProject: Optional. Only for SGE controlled systems. It represents the project name associated with the execution. This name must match with someone on the valid project names list. This list can be consulted executing the command: `qconf -sprjl`.

Format:

```
<queueSGEProject>Name</queueSGEProject>
```

5. Program execution

Once we have our XML config files done, we can now execute the ChIPseq caller.

Command:

```
1. # pathToRUBioSeq/RUBioSeq.pl --analysis ChIPseq --config <ExperimentConfigFile> ,
```

where *ExperimentConfigFile* is the absolute path to our XML experiment config file.

1. Program workflow

If all files, directories and parameters of the configuration files are well established, program will begin to execute samples in parallel if we are running on a HPC system scheduled by a SGE or PBS system (experimental) or sequentially if we aren't. (Corresponding parameter must be properly initialized).

The ChIPseq caller has been designed in a modular way, it divides the execution in four main stages:

- First stage: Alignment phase and FastQC analysis.
- Second stage: Duplicates marking.
- Third stage: Normalization steps.
- Fourth stage: ChIPseq calling , Peaks annotation and IDR control (if it corresponds).

It accepts standalone level execution adding the *level* parameter to the command:

```
#pathToRUBioSeq/RUBioSeq.pl --analysis ChIPseq --config <TestFile.xml> --level <num> ,  
where num is a number between 1 and 4, corresponding respectively to the stages  
explained above.
```

It is important to remark that when we execute level 2 , 3 or 4, the previous level must have been executed previously.

1. Program outputs

The ChIPseq pipeline has an output file organization in directories.

It creates a directory based on *dirOutBase* and *ProjectID* elements from XML experiment file, on the location: */dirOutBase/ProjectId/* and one subdirectory per *Experiment* element. So if we have only one experiment, output files will be stored in: */dirOutBase/ProjectId/Experiment1*

A copy of our XML configuration file will be store in */dirOutBase/ProjectId/* as a backup.

Every stage explained in 'Program workflow' section, generates a directory or group of directories, where files are stored:

preProcess directory: It stores the bwa reads alignment.

MD directory: It stores BAM file with duplicates removed.

Norm directory: It stores BED files from normalization steps and log files.

sharp_peaks directory: It stores output MACS2 peaks and an annotation directory with the annotated peaks with PeakAnnotator.

broad_peads: It stores CCAT peaks and an annotation directory with the annotated peaks with PeakAnnotator.

Quality directory: It stores the results from FASTQC program and IDC analysis.

2. *Quality and Control*

Different types of quality and control analyses are done in the ChIPseq calling analysis execution. Some of them are enabled by the corresponding parameter in the experiment configuration file, and others are automatically performed.

- FastQC analysis (See section Quality and Control on Snv Caller part for further information.)
- Bam files validation (See section Quality and Control on Snv Caller part for further information.)
- IDR analysis

This quality analysis is design to measure consistency between replicates in high-throughput experiments and is and it is activated when ReplicatesFlag is enabled and peakAnalysis is set to sharp or both.

The method was developed by Qunhua Li and Peter Bickel's group and is extensively used by the ENCODE and modENCODE projects. A publication describing the statistical details of the IDR framework is <http://www.stat.washington.edu/qli/IDR-AOAS.pdf> .

The results are stored in /dirOutBase/ProjectId/Experiment#Number/Peaks/IDR.

Important note: If the peaks number called by MACS2 is less than 100K, protocol recommends to add the parameters: ' -p 1e-3 -to-large' to perform a correct IDR analysis.

<MACS_extraArgs>-p 1e-3 -to-large</MACS_extraArgs>

8. RUBIOSEQ. Methylation caller

1. Introduction

RUBIOSEQ's Methylation caller is a workflow to analyze Bisulfite-seq data.

Two library protocols have been developed for constructing bisulfite-converted libraries. Cokus et al. protocol and Lister et al. protocol. The methylation caller uses Bismark program for mapping BS reads generated from either experimental protocol. Then a postprocess phase is applied to the mapped reads in order to extract the methylation calls, context, methylation coverage percentage and graphic representation.

This workflow includes three execution levels: (a) Sequence alignment and methylation calling with Bismark program, (b) methylation calls extraction and (c) an optional intervals methylation percentages calculation.

2. Prerequisites

Operating System: UNIX.

All programs listed below and their corresponding dependencies must be correctly installed.

1. Bedtools(v2.16.2). <http://code.google.com/p/bedtools/>
2. Bismark 0.10.1. <http://www.bioinformatics.babraham.ac.uk/projects/bismark/>
3. Bowtie 0.12.7. <http://bowtie-bio.sourceforge.net/index.shtml>
4. Fastx Toolkit 0.0.13.2. http://hannonlab.cshl.edu/fastx_toolkit/
5. FiloTools 1.1.0. <https://github.com/arq5x/filo>

3. Before running a methylation analysis for first time

1. Configure program paths.

User must configure configProgramPaths.xml file with the corresponding paths to the programs. Configuration program file is explained in the next section.

4. Configuration files

There are two configuration files: configProgramPaths.xml to configure applications paths and queue schedulers management and Experiment.xml, with all specific parameters for the execution.

1. configProgramPaths.xml

This XML format file configures applications paths used by the program, it also initializes queue schedulers parameters for our specific system.

It must be located in *installation_path/RUBioSeq_v3.7/MethylCSeq/MethylBis/config/*.

User must configure this file properly to adapt the application paths and scheduler.

Configuration file example:

```
<!-- RUBioSeq-MethylBis CONFIG FILE -->
<configData>
  <bismarkPath>/home/mrubioc/soft/bismark_v0.10.1/</bismarkPath>
  <bowtiePath>/home/mrubioc/soft/bowtie-0.12.7/</bowtiePath>
  <fastqcPath>/home/mrubioc/soft/FastQC/</fastqcPath>
  <bedToolsPath>/home/mrubioc/soft/BEDTools-Version-2.16.2/bin/</bedToolsPath>
  <picardPath>/home/mrubioc/soft/picard-tools-1.60/</picardPath>
  <fastxPath>/home/mrubioc/soft/bin/</fastxPath>
  <filoPath>/home/mrubioc/soft/filo-master/bin/</filoPath>
  <queueSystem>none</queueSystem>
  <queueName>ngs</queueName>
  <!--<multicore>16</multicore>-->
</configData>
```

Fields:

All these fields are mandatory. They set the absolute directory to the binary or script of the corresponding program:

bismarkPath, **bowtiePath**, **bedToolsPath**, **fastxPath**, **picardPath** and **fastqcPath**

queueSystem: It represents queue scheduler type. The accepted values are: SGE, PBS and none(for execution in non-queue controlled systems).

If queueSystem is initialize to *none* the multisample execution is not performed in a parallel way, but in a secuencially execution, because there is no scheduler to synchronize the programs.

These other fields are optional:

nthr : represents thread number parameter for all compatible applications.

javaRam: Maximum RAM memory limit for java applications.

queueName: Queue name in which tasks are going to run.

multicore: Only valid for SGE systems. Represents the number of slots, user want to keep for his execution. To use this feature SGE manager must create a parallel

environment called *multicore*.

2. Experiment.xml

Experiment.xml configures all methylation caller parameters to adapt the program to the specific input data and user desired configuration.

This file must follow the format `installation_path/RUbioSeq_v3.7/MethylCSeq/MethylBis/config/exampleExperimentConfig.xml`.

It must have at least all mandatory fields. Name and location of this file is a user choice. This file is one of the two parameters of RubioSeq.

User has also to specify the branch type in the XML main element configData:

```
<configData branch="Methylation">
```

File fields:

referencePath: Mandatory. It initializes the absolute path to the directory that contains the reference genome. This genome must be in FASTA format.

Format:

```
<referencePath>AbsoluteDirectory</referencePath>
```

Example:

```
<referencePath>/home/ngs/bismark_ESLIF/reference/</referencePath>
```

intervals: Optional. If exists, it represents the genomic intervals in which we are interested in studying the methylation profiles, and a methylation per intervals is calculated. The results are stored in `/projectCompletePath/intervals` directory. If it doesn't exist the tools operate over the whole genome. File must be in BED format.

Format:

```
<intervals>AbsolutePathToIntervalsfile</intervals>
```

Example:

```
<intervals>/home/ngs/BedFiles/UCSC_CpG_islands_RESORTED.bed</intervals>
```

platform: Mandatory. It represents NGS technology that generates input raw data. Valid values: `illumina` or `fastq` (for other fastq files in ASCII33 and not-casava headers).

Format:

```
<platform>Platform</platform>
```

projectCompletePath: Mandatory. This is the complete project's path. The directory

name will be the project's name, that will be the prefix of all files generated in the execution.

Format:

```
<projectCompletePath>AbsoluteProjectPath</projectCompletePath>
```

sample1: Mandatory. It represents the input data filename. If data is paired-end, it represents the paired-1 file.

Format:

```
<sample1>Filename</sample1>
```

Example:

```
<sample1>ES_LIF_s_8_TGtATT-sequence.txt</sample1>
```

It can be more than one instance of *sample1* in the experiment configuration file, (i.e.: one per sample).

sample2: Optional. It represents the paired-2 input data filename.

Format:

```
<sample2>Filename</sample2>
```

Example:

```
<sample2>ES_LIF_s_8_TGtATT_paired2-sequence.txt</sample2>
```

Note: It must be the same instances of *sample2* in the experiment configuration file that *sample1* instances.

readsPath: Mandatory. It defines the absolute location of raw data directory.

Format:

```
<readsPath>RawDataDirectoryPath</readsPath>
```

Example:

```
<readsPath>/home/mrubioc/READS_2nd_ROUND/</readsPath>
```

seed_length: Optional. (Bowtie configuration parameter). It represents the number of bases on the high-quality end of the read to which the `-n` ceiling applies. The lowest permitted setting is 5 and the default is 28.

Format:

```
<seed_length>seed_length</seed_length>
```

num_mis: Optional. (Bowtie configuration parameter). Alignments may have no more than `seed_length` mismatches. `num_mis` is a number between 0 and 3. Default value is 2.

Format:

```
<seed_length>seed_length</seed_length>
```

trimTagLength: Optional. Trim N bases from the beginning of the reads. Useful for trimming non-directional BS-reads tags.

Format:

```
<trimTagLength>tag_length</trimTagLength>
```

minQual: Optional. Reads with less than 50% bases with a minimum quality Q are filtered.

Format:

```
<minQual>minQuality</minQual>
```

depthFilter: Optional. Calls with depth less than value are filtered from results.

Format:

```
<depthFilter>value</depthFilter>
```

methylType: Mandatory. It represents the methylation protocol. Values: Lister, Cokus.

Format:

```
<methylType>Lister or Cokus</methylType>
```

context: Optional. This param selects the methylation contexts we are interested in. Values: ALL, CpG, CHG and CHH. Default value: ALL.

Format:

```
<context>ALL|CpG|CHG|CHH</context>
```

multiExec: Optional. It selects the application execution mode (see 'Execution Modes' section). Values: 0, standalone multisample execution and 1, joint multisample execution. Default value: 0.

Format:

```
<multiExec>1 or 0</multiExec>
```

fastqc: Optional. It activates FastQC module. This module performs this analysis for all samples in the experiment configuration file. The results are stored in

dirOut/projectId/SampleName/Quality/FastQC subdirectory. Values: 1, activates FastQC; 0, disable the option. Default value: 0.

Format:

`<fastqc>0 or 1</fastqc>`

queueSGEProject: Optional. Only for SGE controlled systems. It represents the project name asociated with the execution. This name must match with someone on the valid project names list. This list can be consulted executing the command: `qconf -sprjl`.

Format:

`<queueSGEProject>Name</queueSGEProject>`

5. Program execution

Once we have the samples data, and our XML config files done, we can now execute the methylation caller.

Command:

```
#pathToRUBioSeq/RUBioSeq.pl --analysis methylationCalling --config
<ExperimentConfigFile> ,
```

where *ExperimentConfigFile* is the absolute path to our XML experiment config file.

1. Program workflow

If all files, directories and parameters of the configuration files are well established, program will begin to execute in parallel if we are running on a HPC system scheduled by a SGE or PBS system or secuencially if we aren't. (Corresponding parameter must be properly initialized).

The methylation caller has been design in a modular way, it divides the execution in three main stages:

- First stage: Sequence alignment and methylation calling.
- Second stage: Methylation calls extraction.
- Third stage: Intervals methylation calculation.

The methylation caller accepts standalone level execution adding the *level* parameter to the command:

```
#pathToRUBioSeq/RUBioSeq.pl --analysis methylationCalling --config <FicheroPrueba.xml>
--level <num> ,
```

where num is a number between 1 and 3, corresponding respectively to the stages

explained above.

It is important to remark that when we execute level 2 or 3, the previous level must have been executed previously.

1. *Program inputs*

The input program data must be FASTQ files from Bisulfite-seq experiments. The program accepts the two library types: directional (Lister) or undirectional (Cokus).

If raw files are directly from Illumina platform, the platform parameter must be illumina.

If FASTQ files don't have Illumina reads identifier, the platform parameter must be set to NCBI.

2. *Program outputs*

The methylation caller has an output file organization in directories.

It creates a directory based on XML experiment file, in the location: /projectCompletePath.

If we are running on standalone multisample execution, one subdirectory per sample will be created. If we are running on joint multisample execution, a subdirectory called *multi* will be created.

Every stage explained in the previous section, generates a directory or group of directories, where files are stored:

methylCalls directory : It stores the methylation calls.

graphFiles directory: This is a subdirectory of methylCalls. It stores bedGraph files of the methylation calls. These files can be visualized in a genome browser like UCSC.

intervals directory : It stores the intervals methylation percentages.

a) **Files**

SAM files

These files are located in projectCompletePath directory. This file is the Bismark application output.

Methylation calls files

In projectCompletePath/methylCalls directory we find the methylation calls files (*.vcf).

The methylation calls files information are distributed by chromosome and they are on VCF format.

Every line of these files contains the following information:

- Chromosome
- Position
- Methylation context (CpG, CHG, CHH)
- Nucleotide in the reference genome (always 'C')
- Alteration: It may be 'C', then the cytosine at this position is methylated or '.' (unmethylated).
- QUAL: Only a compatibility column to VCF format.
- FILTER: Applied filters.
- INFO:
 - NS: Number of samples with data.
 - DP: Total Depth at the position.
 - CD: Methylated cytosine depth.
 - PER: Methylation percentage.
 - STR: Strand Alignment.

```
##fileformat=VCFv4.1
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=CD,Number=1,Type=Integer,Description="Cytosine Depth">
##INFO=<ID=PER,Number=1,Type=Float,Description="Methylation percentage">
##INFO=<ID=STR,Number=1,Type=String,Description="Strand Alignment">
#CHROM POS ID REF ALT QUAL FILTER INFO
chr 17 1502 CHG C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1508 CHG C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1517 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1518 CHG C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1521 CpG C C . . NS=1;DP=1;CD=1;PER=100;STR=+
chr 17 1525 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1526 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1527 CHG C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1530 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1531 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1534 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1536 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1541 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1543 CHG C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1567 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1569 CHG C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1572 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1574 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
chr 17 1575 CHH C . . . NS=1;DP=1;CD=0;PER=0;STR=+
```

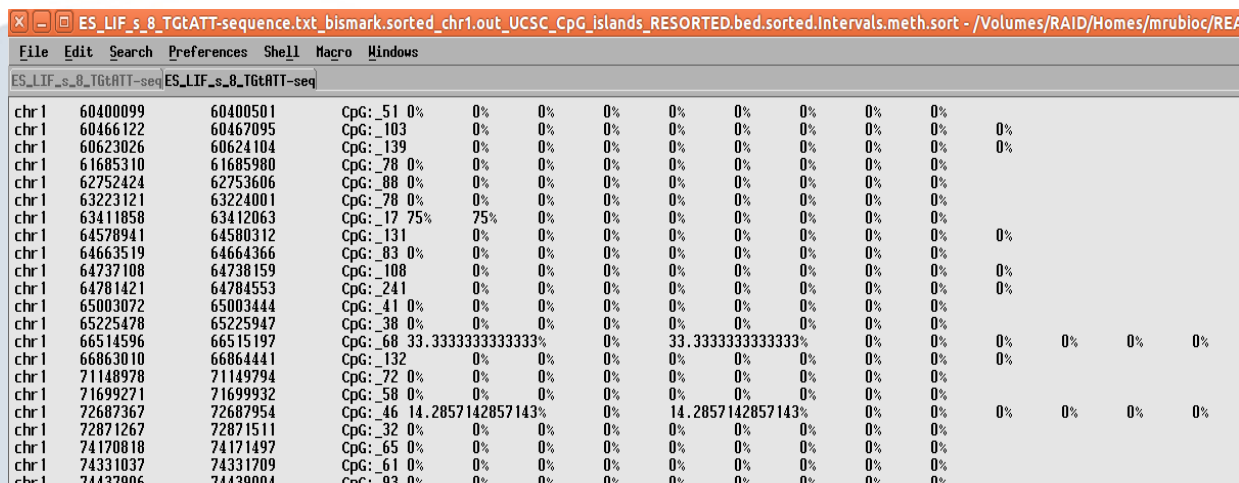
Figure 4: Methycalls file

Intervals methylation

If we have provided an intervals file in order to study the methylation on this intervals, program generates an interval methylation file per chromosome. These files are located in: projectCompletePath/intervals.

The files with the suffix 'Intervals.meth.sort' contain the following information:

- Chromosome
- Start interval position
- End interval position
- Interval's name (if it exists in the original intervals file).
- CpG methylation percentage.
- CHG methylation percentage.
- CHH methylation percentage.
- CpG methylation percentage on forward strand.
- CHG methylation percentage on forward strand.
- CHH methylation percentage on forward strand.
- CpG methylation percentage on reverse strand.
- CHG methylation percentage on reverse strand.
- CHH methylation percentage on reverse strand.



chr	start	end	CpG	CHG	CHH	CHG	CHH	CHG	CHH	CHG	CHH
chr1	60400099	60400501	CpG: .51 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	60466122	60467095	CpG: .103	0%	0%	0%	0%	0%	0%	0%	0%
chr1	60623026	60624104	CpG: .139	0%	0%	0%	0%	0%	0%	0%	0%
chr1	61685310	61685980	CpG: .78 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	62752424	62753606	CpG: .88 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	63223121	63224001	CpG: .78 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	63411858	63412063	CpG: .17 75%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	64578941	64580312	CpG: .131	0%	0%	0%	0%	0%	0%	0%	0%
chr1	64663519	64664366	CpG: .83 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	64737108	64738159	CpG: .108	0%	0%	0%	0%	0%	0%	0%	0%
chr1	64781421	64784553	CpG: .241	0%	0%	0%	0%	0%	0%	0%	0%
chr1	65003072	65003444	CpG: .41 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	65225478	65225947	CpG: .38 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	66514596	66515197	CpG: .68 33.33333333333333%	0%	0%	33.33333333333333%	0%	0%	0%	0%	0%
chr1	66863010	66864441	CpG: .132	0%	0%	0%	0%	0%	0%	0%	0%
chr1	71148978	71149794	CpG: .72 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	71699271	71699932	CpG: .58 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	72687367	72687954	CpG: .46 14.2857142857143%	0%	0%	14.2857142857143%	0%	0%	0%	0%	0%
chr1	72871267	72871511	CpG: .32 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	74170818	74171497	CpG: .65 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	74331037	74331709	CpG: .61 0%	0%	0%	0%	0%	0%	0%	0%	0%
chr1	74437906	74439004	CpG: .93 0%	0%	0%	0%	0%	0%	0%	0%	0%

The files ended with 'sorted.inter' contain the same information as the VCF files in methylCalls intersected with intervals file information.

3. *Quality analysis*

a) *Fastqc analysis*

Enabled by the user. FastQC aims to provide a simple way to do some quality control

checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

FastQC data outputs are located in: /dirOutBase/ProjectId/SampleName/Quality/FastQC.

4. User filters

a) Depth filter

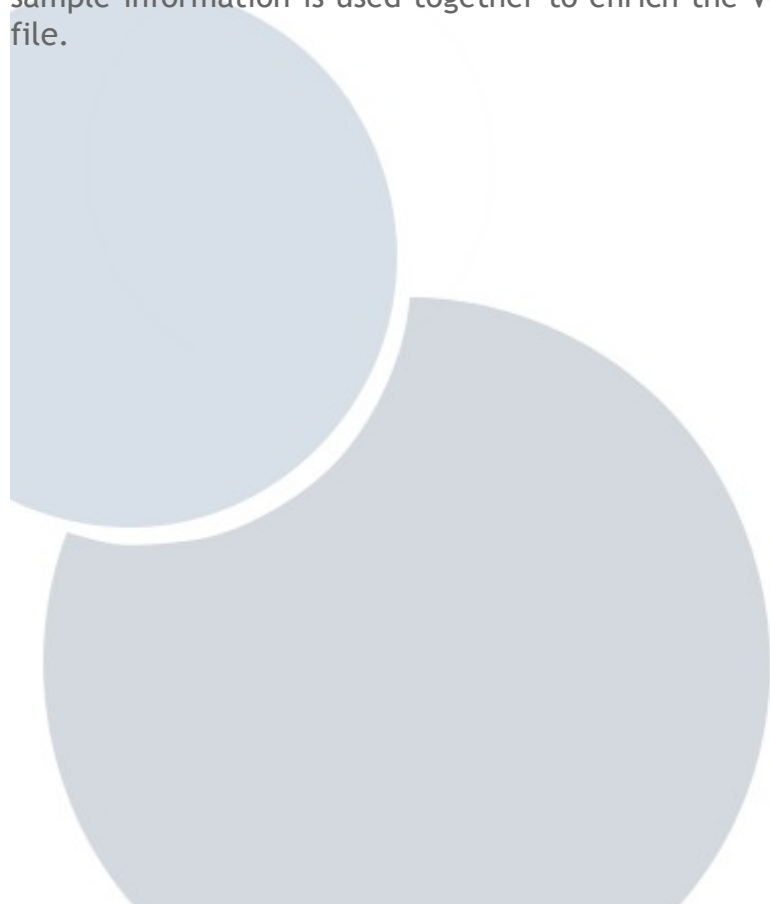
User can filter program result methylation calls that don't have a minimum depth-filter. It is activated initializing `depthFilter` parameter to a value greater than zero.

5. Execution modes

If user is running the program on a HPC system, SGE scheduled or PBS scheduled, the methylation caller permits two execution modes:

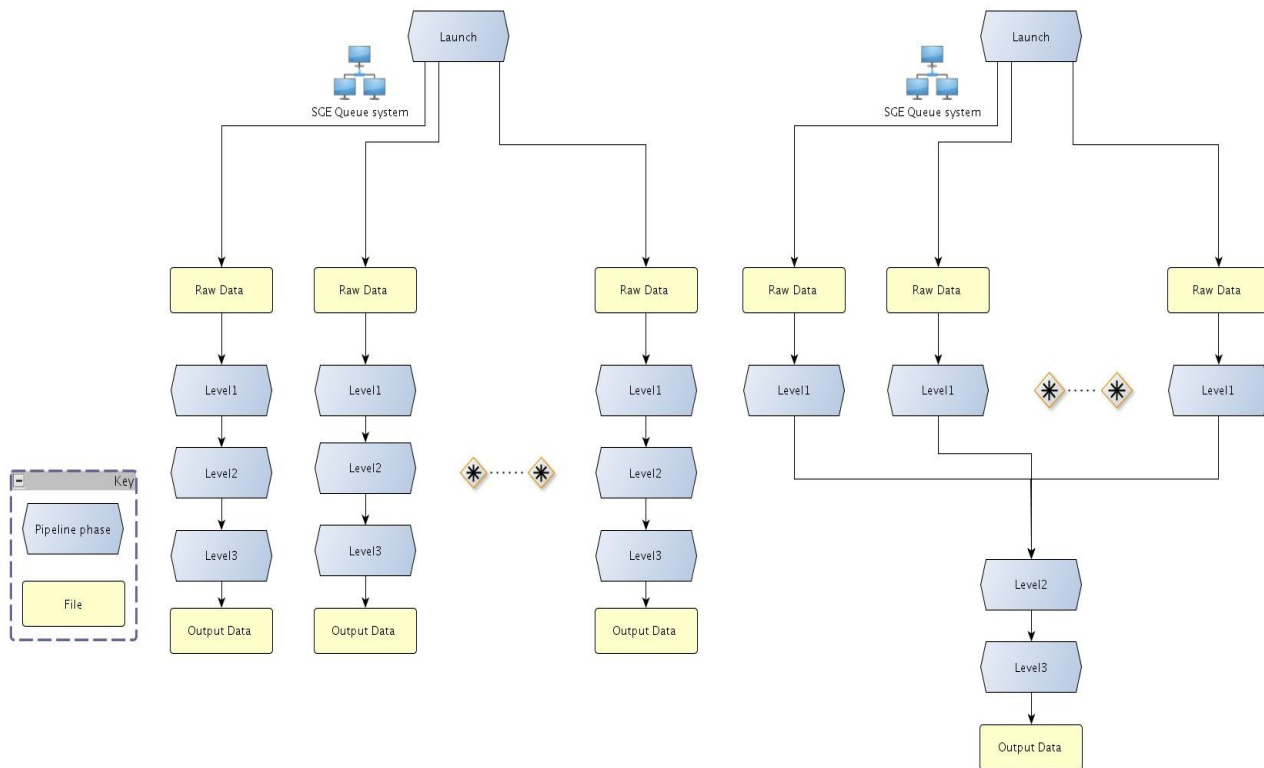
Standalone multisample execution: Parallel execution of one or more samples. In this mode, each sample is treated independently and we get one VCF output file per sample.

Joint multisample execution: Parallel execution of two or more samples. In this mode, all sample information is used together to enrich the VCF output file. We get only one output file.



The next figure represents graphically the execution modes:

Figure 5: Multiexecution workflow



If user is running on a typical workstation, both standalone multisample and joint multisample execution can be performed, but it will be done sequentially.

9. Quick Examples

<http://rubioseq.sourceforge.net/quickSamples.shtml>

Note: These examples are prepared for RUBioSeqV.3.2.1 version. To test with the current version, user will have to adapt only the experiment.xml files for variant calling and CNV calling.

1. Variant Calling

1. Raw Data

Reads type: Fastq, single-end.

This file is a subset of public available sample LB901 from PLoS ONE 2012 7(6): e38158. doi:10.1371/journal.pone.0038158.

Reads number: 6326241

Reads length: 42

2. Steps

Download source code.

Install all programs and prerequisites for variant calling analysis especified in RubioSeq's manual.

Download variant calling test data.

Decompress: >tar xvfz VariantTestData.tgz

>cd YOUR_INSTALLATION_PATH/VariantTestData/reference

Execute BWA indexing:

>YOUR_BWA_PATH/bwa index -a bwtsv hg19.chr20.fa

Execute BFAST+BWA:

```
>YOUR_BFAST+BWA_PATH/bfast fasta2brg -f hg19.chr20.fa -A 0
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m "11111111111111111111111111111111" -w 14  
-i 1 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"1111101110111010100101011011111" -w 14 -i 2 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"1011110101101001011000011010001111111" -w 14 -i 3 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"10111001101001100100111101010001011111" -w 14 -i 4 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m "11111011011101111011111111111111"  
-w 14 -i 5 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"111111100101001000101111101110111" -w 14 -i 6 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"11110101110010100010101101010111111" -w 14 -i 7 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"111101101011011001100000101101001011101" -w 14 -i 8 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"1111011010001000110101100101100110100111" -w 14 -i 9 -n [threads number]
```

```
>YOUR_BFAST+BWA_PATH/bfast index -f hg19.chr20.fa -m  
"1111010010110110101110010110111011" -w 14 -i 10 -n [threads number]
```

Configure experiment XML file:

```
YOUR_INSTALLATION_PATH/VariantTestData/experimentTest.xml
```

You only need to configure the filenames paths to your corresponding directory.

Configure programs XML file located in

```
PathToRubioSeq/variantCalling/config/configProgramPaths.xml.
```

Run RUBioSeq:

```
>PathToRubioSeq/RUBioSeq.pl --config  
YOUR_INSTALLATION_PATH/VariantTestData/experimentTest.xml
```

3. Results

Final results with VEP annotations are available at
\$PROJECT_DIRECTORY/SRR397777/calling/*.cleanNOTPASS.merged.

Please read the manual in order to locate and understand the output files.

2. Copy Number Variation Analysis

1. Raw Data

Input formats and experiment type: BAM, .txt (baseline file), single-end.

These files are publicly available at CONTRA web site. Download.

Reads number: 10230604

Reads length: 75

2. Steps

Use RUBioSeq's LiveDVD or [Download](#) source code and install manually all programs and prerequisites for CNVs analysis specified in RUBioSeq's manual. Most of RUBioSeq's required programs can be downloaded here.

Download CNV test data.

Decompress:

```
>tar xvfz CNVTestData.tgz
```

Configure experiment XML file:

```
YOUR_TEST_PATH/CNVTestData/experimentTest.xml
```

You only need to configure the filenames paths to your corresponding directory.

Configure programs XML file located at:

RUBioSeqPath/variantCalling/config/cnv/configProgramPaths.xml.

RUN RUBioSeq:

```
>PathToRUBioSeq/RUBioSeq.pl          -analysis          cnvCalling          --config  
YOUR_TEST_PATH/CNVTestData/experimentTest.xml
```

Please read the manual to locate the output files.

3. Methylation Calling

1. Raw Data

Reads type: Fastq, Lister, paired-end

Reads number: 1000000 per file

Reads length: 50

Synthetic CpG rate: ~ 30%

Synthetic CHH,CHG rate: ~ 5%

2. Steps

Download source code.

Install all programs and prerequisites for methylation calling analysis specified in RubioSeq's manual.

Download methylation calling test data.

Decompress:>tar xvfz MethylationTestData.tgz

Configure experiment XML file:

YOUR_INSTALLATION_PATH/MethylationTestData/experimentTest.xml

You only need to configure the filenames paths to your corresponding directory.

Configure programs XML file located in
PathToRubioSeq/MethylCSeq/MethylBis/config/configProgramPaths.xml

Run RUBioSeq:

```
>PathToRubioSeq/RUBioSeq.pl --analysis methylationCalling --config  
YOUR_INSTALLATION_PATH/MethylationTestData/experimentTest.xml
```

Please read the manual in order to locate and understand the output files.

3. Results

RUBioSeq calculates the same percentages as we have synthetically generated.

4. ChipSeq Calling

1. Raw Data

Input formats and experiment type: FASTQ, single-end.

These files are a subset of public available: [SRR402848](#): GATA-1 ChIPseq and [SRR402850](#):
Input sample from Wontakal SN, Guo X, Smith C, MacCarthy T et al. A core erythroid
transcriptional network is repressed by a master regulator of myelo-lymphoid
differentiation. Proc Natl Acad Sci USA 2012 Mar 6;109(10):3832-7 .[doi:
10.1073/pnas.1121019109](#).

2. Steps

Download source code and install manually all programs and prerequisites for ChIPSeq
analysis especified in RUBioSeq's manual.

Download ChIPSeq test data.

Decompress:>tar xvfz ChIPSeqTestData.tgz

Configure experiment XML file:

YOUR_INSTALLATION_PATH/ChIPSeqTestData/experimentTest.xml

You only need to configure the filenames paths to your corresponding directory.

Configure programs XML file located in:

PathToRubioSeq/ChIPseq/config//configProgramPaths.xml

Indexing:

Open a terminal:

```
cd YOUR_TEST_PATH/ChIPSeqTestData/reference
```

Execute BWA indexing:

```
>YOUR_BWA_PATH/bwa index -a bwtsv mm9.chr10.fa
```

Run RUBioSeq:

```
>PathToRUBioSeq/RUBioSeq.pl --analysis ChIPseq --config  
YOUR_TEST_PATH/ChIPSeqTestData/experimentTest.xml
```

Please read the manual in order to locate and understand the output files.

3. Results

Final results with peaks are available at PROJECT_DIRECTORY/Experiment1/sharp_peaks/.

10. Bibliography

1. RUBioSeq: a suite of parallelized pipelines to automate exome variation and bisulfite-seq analyses. Miriam Rubio-Camarillo; Gonzalo Gomez-Lopez; Jose M. Fernandez; Alfonso Valencia; David G. Pisano. *Bioinformatics* 2013; doi: 10.1093/bioinformatics/btt203
2. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data .Aaron McKenna .*Genome Res.* 2010 .
3. A framework for variation discovery and genotyping using next-generation DNA sequencing data .Mark A DePristo .*Nature genetics.* 10 April 2011 .
4. GATK:
http://www.broadinstitute.org/gsa/wiki/index.php/The_Genome_Analysis_Toolkit
5. PicardTools: <http://picard.sourceforge.net/>
6. Sequencing technologies – the next generation .Michael L. Metzker .*Nature Reviews.* January 2010 .
7. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants .Peter J. A. Cock .*Nucleic Acids Research*, 2010 .
8. CASAVA Software Version 1.7 User Guide .Illumina.
9. CASAVA v1.8 Changes .Illumina.
10. SOLiDTM 4 System Product Selection Guide .AppliedBiosystems.
11. SOLiD Data Format and File Definitions Guide .NCBI.
12. The SAM Format Specification (v1.4-r962) .The SAM Format Specification Working Group .April 17, 2011 .
13. samtools : <http://samtools.sourceforge.net/>
14. BFAST: An Alignment Tool for Large Scale Genome Resequencing .Nils Homer .*PlosOne* 2009.

15. BFAST: BLAT-like Fast Accurate Search Tool .Nils Homer .
16. A survey of sequence alignment algorithms for next-generation sequencing .Heng Li and Nils Homer .Bioinformatics.May 11, 2010 .
17. Fast and accurate long-read alignment with Burrows-Wheeler Transform. Li H. and Durbin R. Bioinformatics.2010. bwa : <http://bio-bwa.sourceforge.net/>
18. Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor
19. .William McLaren .Bioinformatics.June 18, 2010 .
20. VEP: http://www.ensembl.org/info/docs/variation/vep/vep_script.html
21. Improving the Assessment of the Outcome of Nonsynonymous SNVs with a Consensus
22. Deleteriousness Score, Condel .Abel Gonzalez-Perez .The American Journal of Human Genetics. April 8, 2011 .
23. The Variant Call Format and VCFtools .Petr Danecek .Bioinformatics. June 7, 2011. Vcftools: <http://vcftools.sourceforge.net/>
24. BEDtools-Usage.v24.Aaron R. Quinlan and Ira M. Hall .17-July-2010.
25. fastqc : <http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/> .
26. PileLine: http://sing.ei.uvigo.es/pileline/index.php/Main_Page .
27. 1000 Genomas: <http://www.1000genomes.org/> .
28. http://www.nilshomer.com/index.php?title=BFAST_with_BWA .
29. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. Kumar P. et al, Nat Protoc. 2009 .
30. Polyphen: <http://genetics.bwh.harvard.edu/pph/>
31. http://www.illumina.com/technology/sequencing_technology.ilmn

32. <http://wikis.sun.com/display/GridEngine/>
33. <http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing.html>
34. Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics*. 2011 Jun 1;27(11):1571-2. Epub 2011 Apr 14.
35. Human DNA methylomes at base resolution show widespread epigenomic differences .Lister et al. *Nature*. Vol 462.19 November 2009 .
36. Shotgun bisulfite sequencing of the Arabidopsis genome reveals DNA methylation patterning . Cokus et al. *Nature*. 2008 March 13; 452(7184): 215-219.
37. DNA methylome analysis using short bisulfite sequencing data . *Nature methods*. vol.9 no.2. February 2012 .
38. Li et al. (2012) CONTRA: copy number analysis for targeted resequencing. *Bioinformatics*. 28(10):1307-13.