Project Acronym: MuG

Project title: Multi-Scale Complex Genomics (MuG)

Call: H2020-EINFRA-2015-1

Topic: EINFRA-9-2015

Project Number: 676556

Project Coordinator: Institute for Research in Biomedicine (IRB Barcelona)

Project start date: 1/11/2015

Duration: 36 months

Deliverable 4.6: Benchmarks and documentation

Lead beneficiary: The European Bioinformatics Institute (EMBL-EBI)

Dissemination level: PUBLIC

Due date: 31/10/2018

Actual submission date: 31/10/2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676556.

Document History

Version	Contributor(s)	Partner	Date	Comments
0.1	Mark McDowall	EMBL-EBI	08/10/2018	First draft
0.2	Andrew Yates	EMBL-EBI	10/10/2018	Revised version
1.0			30/10/2018	Approved by Supervisory Board.

Table of Contents

1 EXECUTIVE SUMMARY

The following document describes the tools and pipelines that have been integrated into the VRE based on D4.3 along with relevant steps take to optimise the code. In total there are 29 tools that have been wrapped as part of this Work Package or in collaboration with other packages along with documentation, unit testing and matching the coding standards set by the consortium. All of the tools are based on the Tool and DM APIs that had been developed as part of WP5 and D4.4 respectively to allow for a harmonious integration pathway. The document also describes the efforts that have been taken to document the VRE and requirements for external developers that was to integrate new tools.

To test the integration documentation and expand the functionality of the site the CHiCAGO tool has been wrapped based on the templates and documentation. Groups not involved with the initial development of the Tool API framework have also started to use the documentation to wrap their own tools and pipelines using the documentation and provide feedback when information is not clear.

2 INTRODUCTION

From user to developer, providing accurate and relevant information for the task they wish to complete, whether that is how to align a sequence, knowing how long a process should take to complete or creating a new tool, documentation plays a vital role. Clear documentation and benchmarks provides the user with knowledge about what they can do and how long it will take. This prevents users leaving when they are lost, from become impatient when pipelines take time and informs users about the what they are doing.

Developers are also a key part and require targeted documentation, especially for the long term maintainability of code. Providing developers with clear documentation and common frameworks about how they are able to participate with the project and enhance the virtual research environment (VRE) with their own tools and pipelines is key. This will encourage others to get involved and keep the service relevant as new techniques are developed.

To address both the users and developers a large resource of documentation has been created to guide researchers that want to interact with the VRE. Work has also been done to try and address benchmarking the tools and pipelines and feeding this back to the users. To maintain commonality and reduce technical debt between tools and pipelines created by different developers, common frameworks and standards have been put in place.

3 PIPELINE DESIGN UPDATES

Based on the work done in D4.3, along with collaboration with teams in WP5 and WP6, the Tool API was developed by WP6. The Tool API is a python library created to formalise the description of a pipeline. It provides a wrapping framework that can be used to easily incorporate tools into the VRE via a common interface and utilises the pyCOMPSs infrastructure to provide a convenient way to enable parallelisation of the tools.

The Tool API framework encompasses the interface with the VRE as well as defining a consistent structure for how to wrap tools and how they are integrated into a workflow. A pipeline is what runs a workflow and handles the inputs from the VRE; it requires 2 input JSON files and the location of an output metadata JSON file for files created by the workflow, defined in D5.2. For the input JSON files the first lists the input files, what output files are expected and any parameters defined in the VRE and required by the tools run within a workflow. The second JSON file contains the location and metadata about all of the input files. At runtime the Tool API handles imports both JSON files making all of the metadata, files and arguments available within the workflow. Once the workflow has completed the Tool API exports the metadata for each file that is required as a JSON file in the location specified.

3.1 Metadata Class

The Metadata class encapsulates the descriptive information about a file. This information is either extracted from the input JSON files when the workflow is initiated or created by each tool when the file they are to return has been generated. The parameters stored within the Metadata class match with the parameters defined in for each entry in the DM API as defined in D4.5 Table 3.1.1.

3.2 Tool Class

The Tools class wraps a single tool or function where all processing is self contained. With the use of pyCOMPSs breaking down of pipelines into separate tools means that it is possible to run multiple tasks at the same time if there are 2 tools that can be run independently of each other (ie in parallel).

Each tool class minimally consists of an initialisation function for loading tool configuration parameters and a run function. The run function should be able to take 3 dictionary objects defining the input files, the input file metadata and the required output files. The run function controls the running of a tool defined within its own function. If the tool can be run in parallel, python decorators should be used to define the function as a pyCOMPSs @task.

3.3 Workflow Class

The Workflow class is used to wrap one or more tools and manages the merging of the output files and metadata from tools. The workflow must contain an initialisation function for loading configuration information and a run function for calling each of the tools. A pipeline is a script that contains the workflow class, but also takes the input files from the VRE. When a pipeline is run it automatically loads the JSON files provided from the VRE into Metadata classes and passes the information to the workflow run class. The structure of the Workflow class has been designed so that it mirrors the interface of the Tool, this is to allow for other pipelines to be inherited as tools.

3.4 Pipeline

The pipeline is a script that handles the inputs from the VRE and initialises of the Workflow class. The pipeline can contain the whole workflow. These are the main scripts that are run by the VRE when a task is run via the VRE.

3.5 Code Quality, Testing and Documentation

As defined in the report D4.5 section 3.3 all the code for the MuG project conforms to a common set of standards for documentation (see MS14) and for code quality:

- https://multiscale-genomics.readthedocs.io/en/latest/coding_standards.html

All the code for the Tool API is provided via GitHub (https://github.com/Multiscale-Genomics/mg-tool-api) and the API documentation is available on ReadTheDocs (https://mg-tool-api.readthedocs.io/en/latest/). As part of good coding practice, unit tests have also been developed to ensure that if there are changes to the API this does not break the functionality. To enforce the coding standard, check that the documentation builds correctly and that changes to the code have not broken any of the tests Travis-CI is used to automatically build and test the library for every commit to the GitHub repository.

4 TOOL AND PIPELINE INTEGRATION

Based on the framework defined by the Tool API, the pipelines reported in D4.3 have been refactored using the new infrastructure and made accessible in a repository called mg-process-fastq on GitHub at the following URL:

- https://github.com/Multiscale-Genomics/mg-process-fastq

4.1 Tools

Since the D4.3 report the pipelines were analysed to identify the units of work that were performed within each pipeline, where there was commonality between pipelines and whether the units of work could also be split down further to provide greater flexibility. Table 1 lists the tools that have been created based on this review. Reformatting of the overall structure of the pipelines has reduced code duplication allowing for greater code reuse, especially with the pipelines that use the same aligners or filtering steps.

Tool	Description	Reference
BioBamBam2	Highlighting of duplicates in Bam files	https://github.com/gt1/biobambam2
Bowtie - Indexer	Generate genome indexes and package them within an archive file ready for the aligners	[Langmead2012]
Bowtie - Aligner	Aligner of single and paired end data	[Langmead2012]
BS Seeker2 - Indexer	Generate genome indexes and package them within an archive file ready for the aligner	[Guo2013]
BS Seeker2 - Filter	Filtering of duplicates and artifacts from the FASTQ files	[Guo2013]
BS Seeker2 - Aligner	Alignment of single and paired end data ready for methylation peak calling	[Guo2013]
BS Seeker2 - Methylation Caller	Peak caller to identify regions that have been methylated	[Guo2013]
BWA - Indexer	Generate genome indexes and package them within an archive file ready for the aligners	[Li2010]
BWA - ALN Aligner	BWA aligner for single and paired end data using the ALN algorithm	[Li2010]
BWA - MEM Aligner	BWA aligner for single and paired end data using the MEM algorithm	[Li2010]
FASTQ Splitter	Tool for splitting large FASTQ files	

	(single or paired end) into chunks of a defined size	
GEM Indexer	Generate genome indexes and package them within an archive file ready for the aligners	[Marco-Sola2012]
iDEAR	Peak caller for iDamID-seq data	[Gutierrez-Triana2016]
iNPS	Peak caller for MNase-seq data	[Chen2014]
Kallisto Indexer	Indexer of cDNA for for use in RNA-seq analysis	[Bray2016]
Kallisto Quant	Tool for quantifying the level of expression of transcripts	[Bray2016]
MACS2	Peak caller for ChIP-seq data	[Zhang2008]
Sleuth	RNA-seq differential analysis tool that uses the output from Kallisto	[Pimentel2017]
TADbit - bin	Binning of the adjacency matrix	[Serra2016]
TADbit - Filter	Filter technical errors	[Serra2016]
TADbit - Full Mapper	Mapping reads to the genome	[Serra2016]
TADbit - Generate TADS	TAD prediction	[Serra2016]
TADbit - Model	Generate 3D models	[Serra2016]
TADbit - Normalise	Normalise the matrix	[Serra2016]
TADbit - Parse Mapping	Generate the adjacency lists	[Serra2016]
TADbit - Save HDF5 Matrix	Save matrix in HDF5 index file	[Serra2016]
TADbit - Segment	TAD prediction	[Serra2016]
TrimGalore	FASTQ read quality trimming. Is a wrapper around cutadapt [Martin2011] and FASTQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/).	http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/
Validate FASTQ (FASTQC)	Analyse the quality of reads in a FASTQ file(s)	https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Table 1: *Individual tools that have been wrapped in the mg-process-fastq repository*

There is an additional range of ancillary modules to wrap functionality that is required by multiple tools, but not required as a tool in its own right. This includes the wrapping of functions for handling bam and fastq files using SamTools [Li2009] (via the python library pysam and calls to BedTools [Quinlan2002]) and functions for the reading and manipulation of FASTQ files. Initial work has been done to start to move this functionality into a separate repository to allow for easier re-use of the code.

- https://github.com/Multiscale-Genomics/mg-common

New tools that have been added since D4.3 include iDEAR [Gutierrez-Triana2016], Sleuth [Pimentel2017], TrimGalore and FastQC. iDEAR allows for the analysis of iDamID-seq data and Sleuth is a differential analysis package for identifying differentially expressed genes based on the results of the Kallisto analysis. TrimGalore and FastQC are tools for the analysis and trimming of FASTQ reads to help users to get the most accurate alignments.

4.1.1 Capture HiC Analysis

Capture Hi-C is a technique for identifying the genomic interactions around specific regions of DNA. This allows researchers to focus in on regions of interest rather than analysing the nucleus as a whole. The analysis of this data is not directly covered by the TADbit pipelines so CHiCAGO [Cairns2016] is being integrated as a pipeline for the normalisation of Capture Hi-C data. As part of the integration, HiCUP [Lieberman-Aiden2009,Wingett2015] is also being wrapped for the alignment and interaction assignment as part of the initial phase. However, TADbit will be investigated for the identification of interactions follow by using CHiCAGO for the normalisation of the datasets. Further work is going to be done to optimise the CHiCAGO algorithm to take advantage of the pyCOMPSs infrastructure.

4.1.2 Externally Developed Tools

Effort has also been made to reach out to tool developers outside of the MuG Consortium to encourage them to integrate their tool within the MuG VRE. The first has been Vera Pancaldi and Miguel Madrid who are wrapping their ChA tool [Pancaldi2016]. Efforts have also been made to reach out to others that have tools that are beneficial to the community as a whole. These include:

- Carl Barton - ChromoTrace
- Sean O’Donoghue - Rondo.ws
- Katie Pollard - ShapeMF
- Shamith Samarajiwa - Deep Neural Network predictor of TADs and long range interactions

4.2 Pipelines

For all of the individual tools there is a matching pipeline ready for integration into the VRE. For the pipelines that were highlighted in D4.3 these have been refactored to use the Tool API and the new tool framework (Table 2).

Pipeline	Description	Tools	Repository
Process Genome	Single administrative process for the pre-indexing of genomes	Bowtie2 Indexer BWA Indexer GEM Indexer	mg-process-fastq
Process ChIP-seq	Analysis of ChIP-seq data. Aligns single or paired end FASTQ data, filters out experimental artifacts and performs the peak calling	BWA ALN BioBamBam2 MACS2	mg-process-fastq
Process DamID-seq	Analysis of iDamID-seq data from FASTQ to peak calling	BWA MEM BioBamBam2 BSforge iDEAR	mg-process-fastq
Process MNase-seq	Analysis of MNase-seq data from FASTQ to peak calling	BWA MEM BioBamBam2 iNPS	mg-process-fastq
Process RNA-seq	Alignment and expression level calculated	Kallisto Index Kallisto Quant	mg-process-fastq
Process WGBS	Analysis of WGBS data from FASTQ to peak calling		mg-process-fastq
ATAC-seq	Analysis of ATAC-seq data from FASTQ to peak calling	BWA MEM TrimGalore MACS2	ATAC-seq
CHi-C	Alignment and normalisation of Capture Hi-C data	HiCUP CHiCAGO	CHi-C

Table 2: *Pipelines that use more than one tool. All of the repositories are available from the Multiscale Complex Genomics GitHub account (https://github.com/Multiscale-Genomics). Pipelines that are highlighted in bold were identified as new additions since the D4.3 report.*

5 BENCHMARKING

5.1 Tool Optimisation

The first major improvement has been the adoption of the Tool API. Independent of runtime speeds, this has had a positive effect on the development of tools and workflows increasing the efficiency of development and has made the process more transparent. This is beneficial for the long term development of the project making it simpler for future tool integration and maintenance or existing code.

For many of the tools that have been wrapped they worked in a linear fashion and require linear processing of the data, but there are certain steps that can be optimised to run more efficiently within an HPC environment. The major improvements have come from splitting up the input data in a sensible manner and then running the tool as many times as required in a parallel fashion and merging the final results.

5.1.1 Development of Luigi

So that optimisations to the code could be tested to see if they made a change to the runtimes of the pipelines work was done locally at the EMBL-EBI. Luigi (https://github.com/spotify/luigi) is a python library developed at Spotify for running complex pipelines within HPC environments. It was selected as an inplace library replacement of pyCOMPSs due to it being functionally very similar for workflow management using python decorators to wrap functions that can be run within an HPC environment. The major difference to pyCOMPSs is that it can be run on different cluster management architectures. The EMBL-EBI cluster uses the Load Sharing Facility (LSF), the closest variant in Luigi is the SunGrid Engine (SGE) cluster management software. The development of the SGE bindings in Luigi had been created from a branch of code developed for using LSF, so we reached out to the original developers and managed to update and improve their code and finally get it merged into the Luigi library.

5.1.2 Sequence Alignment

For workflows that use the BWA ALN, BWA MEM and Bowtie2 tools it is possible to split the input FASTQ files into chunks, align the reads and then merge the files into a single bam file at the end of the job. To test this pipelines were adjusted to use the Luigi library so that the tools could run on the LSF cluster at the EBI. The example datasets are the human datasets DRR000150 (21713314 single end reads).

5.1.2.1 File Splitting

Due to the requirements of (py)COMPSs there has to be a known number of input and output files, in this case a FASTQ as input and an archive (tar.gz) of all the subsets as output. For splitting the FASTQ file the number of output smaller FASTQ files has little effect on the time as shown in Table 3 using the DRR000150 FASTQ dataset. The time taken to split the file into FASTQ files containing 1M or 10M reads is only slightly slower than reading through the file.

Reads in output FASTQ Files	Time
21,713,314	1min 54.777sec
100,000	1min 11.596sec
1,000,000	1min 57.302sec
10,000,000	1min 56.396sec

Table 3: *The times taken to read through a FASTQ file and split it into smaller FASTQ files. The top row represents reading through the DRR000150 dataset and creating a single FASTQ file.*

5.1.2.2 BAM Merge

Increasing the number of alignments that have to happen also increases the number of merging steps that need to be performed. For this test the dataset SRR892982 was used as it had 55,435,326 aligned reads. As Table 4 shows the simplest methods is to merge everything at the same time.

Batch Size	Round 1	Round 2	Total
56	25m 48.644s		25m 48.644s
10	5m 12.175s 5m 7.482s 4m 57.378s 5m 0.048s 5m 3.393s 2m 44.943s	25m 30.003s	30m 42.178s

Table 4: *Times for the merging of 56 bam files. For the batch size of 10 this required 6 jobs running in parallel then a second job to perform the final merge. The total time for the batching is equivalent to the sum of the longest in each round*

Applying this to the pipeline, COMPSs requires that there are a known number of input and output files. So iteratively performing the merges in batches of 10 and having functions for handling 5, 4, 3 and 2 files is enough to cover all scenarios. Doing this does not drastically affect the run time taking only 5 minutes longer than merging everything in the same step.

5.1.2.3 FASTQ Alignment

For testing the alignment process DRR000150 was used along with BWA ALN for the alignment. This used a bam batch merge size of 10 across a range of FASTQ subset file sizes. Table 5 shows that subdivision of the original FASTQ file results in faster completion times, even with the merging penalty. The largest difference is reducing the number of reads that are present in a single job, however at the lower end there is then the potential for bam merging issues due to the number of bam files that need to be merged.

As a result of this the alignment modules were modified so that FASTQ files are initially split into FASTQ files of up to 1M reads and the merging is done in batches of 10.

Reads per fastq file	Time
21,713,314	62m 3.707s
100,000	18m 41.906s
1,000,000	16m 22.858s
10,000,000	42m 16.217s

***Table 5:** The times for each set include the splitting of the bam file, alignment and then merging the results.*

5.1.3 Peak Calling by Chromosome

Where it is possible, another area that it is possible to make changes to reduce the time for computation is when peak calling and the peaks can be called on a per chromosome independently without knowledge of other chromosomes. This was the case for MACS2 where the peaks could be called for each chromosome and then the results merged at the end.

5.2 Pipeline Performance

Based on the improvements identified using Luigi in the previous section, this was applied to all pipelines for where the highlighted changes were applicable. Once the changes were made and the final pipelines were tested in a local VM they were loaded onto a VM at the IRB and tests were run to determine how long the pipeline would take to complete on that VM with pyCOMPSs used to manage the jobs that get run. For this testing there was only a single VM, but this is enough to get an estimate of the performance for each of the pipelines.

For testing the pipelines the datasets used are shown in Table 6. Tables 7 and 8 show the times for running each of the pipelines and tools with human and yeast data to provide the user with a guide as to the required runtimes for each pipeline.

Dataset	Species	Data type	Single/ Paired	Reads	Avg. read length	%GC
DRR000150	Human	ChIP-seq	Single	21713314	36	43
SRR4420194	Human	ChIP-seq	Paired	5384541	25	46
DRR000386	Human	MNase-seq	Single	21713314	36	43
DRR000897	Human	RNA-seq	Single	22635328	37	54
ERR1227470	Human	WGBS	Single	18836903	20-51	35
ERR1094252	Yeast	ChIP-seq	Paired	1067084	51	38
ERR1380243	Yeast	RNA-seq	Single	9270157	50	41
SRR1916129	Yeast	WGBS	Single	5618700	51	19

Table 6: *Datasets used for benchmarking.*

Pipeline	Species	Dataset	Single/Paired	Time
Process Genome	Yeast	R64.1.1	NA	0m 54.289s
Process ChIP-seq	Yeast	ERR1094252	Paired	5m 1.638s
Process RNA-seq	Yeast	ERR1380243	Single	2m 0.866s
Process WGBS	Yeast	SRR1916129	Single	17m 22.185s
Process Genome	Human	GRCh38	NA	153m 18.459s
Process ChIP-seq	Human	DRR000150	Single	81m 6.452s
Process RNA-seq	Human	DRR000897	Single	20m 11.040s

Table 7: *Runtimes for pipelines that consist of multiple tools when run within a single VM at the IRB run from the command line. The "Process Genome" pipeline generates the Bowtie2, BWA and GEM indexes required by the other pipelines.*

Tool	Species	Dataset	Single/Paired	Time
FASTQC	Yeast	SRR1916129	Single	1m 15.583s
TrimGalore	Yeast	SRR1916129	Single	2m 53.577s
TrimGalore	Yeast	ERR1094252	Paired	1m 11.775s
Bowtie2	Yeast	SRR1916129	Single	8m 30.564s
BWA ALN	Yeast	SRR1916129	Single	9m 10.366s
BWA MEM	Yeast	SRR1916129	Single	9m 17.860s
Bowtie2 -	Yeast	ERR1094252	Paired	3m 52.173s
BWA ALN	Yeast	ERR1094252	Paired	3m 34.648s
BWA MEM	Yeast	ERR1094252	Paired	3m 30.990s
BioBamBam	Yeast	SRR1916129	Single	1m 40.622s
BioBamBam	Yeast	ERR1094252	Paired	0m 56.367s
MACS2	Yeast	ERR1094252	Paired	0m 44.463s
BS Seeker2 Indexer	Yeast	R64.1.1	Single	1m 2.456s
BS Seeker2 Filter	Yeast	SRR1916129	Single	1m 31.421s
BS Seeker2 Aligner	Yeast	SRR1916129	Single	14m 28.067s
BS Seeker2 Peak Caller	Yeast	SRR1916129	Single	2m 47.123s
FASTQC	Human	DRR000150	Single	3m 4.179s
TrimGalore	Human	DRR000150	Single	7m 35.523s
TrimGalore	Human	DRR046352	Paired	37m 8.073s
Bowtie2	Human	DRR000150	Single	
BWA ALN	Human	DRR000150	Single	61m 29.971s
BWA MEM	Human	DRR000150	Single	57m 49.779s
Bowtie2	Human	SRR4420194	Paired	
BWA ALN	Human	SRR4420194	Paired	
BWA MEM	Human	SRR4420194	Paired	42m 38.681s
BioBamBam	Human	DRR000150	Single	5m 28.875s
BioBamBam	Human	SRR4420194	Paired	2m 55.321s
MACS2	Human	DRR000150	Single	13m 55.766s
BS Seeker2 Indexer	Human	ERR1227470	Single	413m 15.687s
BS Seeker2 Filter	Human	ERR1227470	Single	4m 1.596s

Table 8: *Runtimes for pipelines for individual tools when run within a single VM at the IRB run from the command line.*

6 DOCUMENTATION

A crucial part to the success of a project reliant on code is the accessibility to documentation. In the case of the VRE there are several layers of documentation required:

- User level: This covers the functionality of the VRE, the interface and guiding the user.
- Developer level: Documentation and tutorials about tool integration.
- Admin level: Information about the requirements of tools and the infrastructure parameters

This covers the work done by developers as part of all work packages and act as part of the support network for users developed as part of Task 5.5 and extends the MS16 report.

6.1 User Documentation

There are several types of user documentation that is available on the VRE website. This covers the user of the VRE, tool specific documentation and tutorials about how to answer biological questions. The VRE and tool specific documentation provides details about the technicalities of using the site and the users workspace through to describing the tool and what are the available parameters for each of the tools. The biological tutorials were developed to guide researchers through which tools they required to answer a biological question, this was done to make the VRE more accessible and help those new to the field become comfortable with what is available.

As new tools are developed and integrated into the VRE, developers are then required to provide documentation formatted for the VRE describing the tool and what it does for use in the User Documentation sections. They are also encouraged to provide tutorials about how researchers can answer biological questions using their tool and the other tools already in the VRE. In doing this they are then able to introduce their tool to the community by showing how it can be used.

6.2 Developer Documentation

For developers that want to integrate a tool into the VRE there is a comprehensive set of documents covering what is required and example code to help get them started. From the VRE there is documentation taking the developer step-by-step through the process of wrapping their tool and making it visible in the VRE (https://www.multiscalegenomics.eu/MuGVRE/instructions/). It also extends beyond the wrapping of the code to include the creation of testing scripts using pytest and linters to ensure that the code they develop complies to the Coding Standard required (https://multiscale-genomics.readthedocs.io/en/latest/coding_standards.html). As the project uses the Apache 2.0 license there are documents covering how it is applied within tool code for running in the VRE. The Apache 2.0 license allows for the sharing of the code and is compatible with many other licenses.

To reduce the overhead with wrapping a tool we have developed a template repository (https://github.com/Multiscale-Genomics/mg-process-test). This is a basic pipeline and tool that matches the Coding Standard. As part of the template it includes documentation, and code testing. The template also has a predefined .travis.yml so that the developer is able to link their tool wrapper with Travis to automatically run the tests for the quality of the code, if the documentation builds and checks that the python tests pass to ensure functional code. The repository is also set up so that the code can be installed using the python package manager, pip.

Once the user has developed their code they are able to test it locally within a virtual machine that matches the one used within the VRE. From here they can optimise the code using pyCOMPSs and minimise issues arising at installation and runtime within the VRE.

The documentation and use of the template repository places emphasis on creating high quality, well documented code with full and repeatable installation instructions. The use of Travis and defining the installation procedures as part of the Travis CI YAML file ensures that the developer is fully able to describe the installation procedures within their documentation. This makes the installation within the VRE a much smoother process.

To further help developers we ran a workshop for tool integration at the EMBL-EBI site in September 2018. All slides and materials are available from the VRE website (https://www.multiscalegenomics.eu/MuGVRE/training/). Further details about the workshop are reported in D2.8.

7 CONCLUSIONS

The development of the Tool API and the template repository has acted to reduce the effort required to wrap a new tool as well as reducing the technical debt for maintenance if they were all implemented in different styles. Although this does not necessarily improve the performance from the users' perspective it provides a stable and predictable platform for those that maintain the infrastructure years after the tool was introduced.

Comprehensive and detailed documentation has been created to support the needs of both the user and the developer. As part of this steps have been taken to benchmark the tools and pipelines to help users understand what is happening behind the scenes and what to expect when they submit a job request.

During the time period that this report covers efforts have also been made to extend the repertoire of tools for the analysis of sequence data. There has been the implementation of FASTQ analysis tools, extra pipelines for analysing new *-seq data types and the introduction of CHiCAGO for the normalisation of Capture Hi-C data.

8 REFERENCES

[Bray2016] Bray, N. L., Pimentel, H., Melsted, P., & Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology, 34(5), 525–527. https://doi.org/10.1038/nbt.3519

[Cairns2016] Cairns, J., Freire-Pritchett, P., Wingett, S.W., Várnai, C., Dimond, A., Plagnol, V., Zerbino, D., Schoenfelder, S., Javierre, B.M., Osborne, C., Fraser, P. & Spivakov, M. (2016). CHiCAGO: robust detection of DNA looping interactions in Capture Hi-C data. Genome Biology 17:127 https://doi.org/10.1186/s13059-016-0992-2

[Chen2014] Chen, W., Liu, Y., Zhu, S., Green, C. D., Wei, G., & Han, J.-D. J. (2014). Improved nucleosome-positioning algorithm iNPS for accurate nucleosome positioning from sequencing data. Nature Communications, 5, 4909. https://doi.org/10.1038/ncomms5909

[Cheng2007] Cheng, T. M.-K., Blundell, T. L., & Fernandez-Recio, J. (2007). pyDock: Electrostatics and desolvation for effective scoring of rigid-body protein–protein docking. Proteins: Structure, Function, and Bioinformatics, 68(2), 503–515. https://doi.org/10.1002/prot.21419

[Flores2011] Flores, O., and Orozco, M. (2011). nucleR: a package for non-parametric nucleosome positioning. Bioinformatics *27*, 2149–2150. https://doi.org/10.1093/bioinformatics/btr345

[Guo2013] Guo, W., Fiziev, P., Yan, W., Cokus, S., Sun, X., Zhang, M. Q., *et al* (2013). BS-Seeker2: a versatile aligning pipeline for bisulfite sequencing data. BMC Genomics, 14, 774. https://doi.org/10.1186/1471-2164-14-774

[Gutierrez-Triana2016] Gutierrez-Triana, J.A., Mateo, J., Ibberson, D., Ryu, S. & Wittbrodt, J. (2016) iDamIDseq and iDEAR: an improved method and computational pipeline to profile chromatin-binding proteins. Development, 143:4272-4278. https://doi.org/10.1242/dev.139261

[Hospital2013] Hospital, A., Faustino, I., Collepardo-Guevara, R., González, C., Gelpí, J. L., & Orozco, M. (2013). NAFlex: a web server for the study of nucleic acid flexibility. Nucleic Acids Research, 41(W1), W47–W55. https://doi.org/10.1093/nar/gkt378

[Langmead2012] Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. Nature Methods, 9(4), 357–359. https://doi.org/10.1038/nmeth.1923

[Leinonen2011] Leinonen, R., Akhtar, R., Birney, E., Bower, L., Cerdeno-Tárraga, A., Cheng, Y., *et al* (2011). The European Nucleotide Archive. Nucleic Acids Research, 39(suppl 1), D28–D31. https://doi.org/10.1093/nar/gkq967

[Lieberman-Aiden2009] Lieberman-Aiden et al. (2009) Comprehensive mapping of long-range interactions reveals folding principles of the human genome. Science 326:289-293

[Li2009] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., *et al* (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics, 25(16), 2078–2079. https://doi.org/10.1093/bioinformatics/btp352

[Li2010] Li, H., & Durbin, R. (2010). Fast and accurate long-read alignment with Burrows–Wheeler transform. Bioinformatics, 26(5), 589–595. https://doi.org/10.1093/bioinformatics/btp698

[Marco-Sola2012] Marco-Sola, S., Sammeth, M., Guigó, R., & Ribeca, P. (2012). The GEM mapper: fast, accurate and versatile alignment by filtration. Nature Methods, 9(12), 1185–1188. https://doi.org/10.1038/nmeth.2221

[Martin2011] Martin, M. (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet journal 17 (1) https://doi.org/10.14806/ej.17.1.200

[Pancaldi2016] Pancaldi, V., Carrillo-de-Santa-Pau, E., Javierre, B.M., Juan, D., Fraser, P., Spivakov, M., Valencia, A. & Rico, D. (2016) Integrating epigenomic data and 3D genomic structure with a new measure of chromatin assortativity Genome Biology 17:152 https://doi.org/10.1186/s13059-016-1003-3

[Pimentel2017] Pimentl, H., Bray, N. L., Puente, S., Melsted, P. & Pachter, L. (2017). Differential analysis of RNA-seq incorporating quantification uncertainty. Nature Methods, 14, 687-690. https://www.nature.com/articles/nmeth.4324

[Quinlan2002] Quinlan, A. R. (2002). BEDTools: The Swiss-Army Tool for Genome Feature Analysis. In Current Protocols in Bioinformatics. John Wiley & Sons, Inc. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/0471250953.bi1112s47/abstract

[Serra2016] Serra, F., Baù, D., Filion, G., & Marti-Renom, M. A. (2016). Structural features of the fly chromatin colors revealed by automatic three-dimensional modeling. bioRxiv, 36764. https://doi.org/10.1101/036764

[Wingett2015] Wingett SW, Ewels, P., Furlan-Magaril M., Nagano, T., Schoenfelder, S., Fraser, P., Andrews, S. (2015) HiCUP: pipeline for mapping and processing Hi-C data F1000Research, 4:1310. https://doi.org/10.12688/f1000research.7334.1

[Zhang2008] Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., *et al* (2008). Model-based Analysis of ChIP-Seq (MACS). Genome Biology, 9(9), R137. https://doi.org/10.1186/gb-2008-9-9-r137

9 ANNEXES

9.1 Abbreviations

DAC: Data access committee

EGA: European Genome-phenome Archive

ENA: European Nucleotide Archive

FISH: Fluorescence *in-situ* hybridisation

MINSEQE: Minimum Information about a high-throughput Sequencing Experiment

OME: Open Microscopy Environment

PDB: Protein Data Bank

PMES: Programming Model Enactment Service

SRA: Sequence read archive

VRE: Virtual Research Environment

WGBS: Whole Genome Bisulphate Sequencing