Institut Pasteur

Sequana: a set of flexible genomic pipelines for processing and reporting NGS analysis

Thomas Cokelaer - Dimitri Desvillechabrol

Institut Pasteur

Oct 19th 2017, ENS, Paris (séminaire bioinfo)

Motivation

Jan 2015: provide NGS pipelines to Biomics sequencing platform https://research.pasteur.fr/en/team/biomics/ (Institut Pasteur)

- Genomics: QC + variant calling + de-novo
- Transcriptomics: RNA-seq + ChIP-seq
- Metagenomics
- Illumina but also Pacbio long reads technologies

How ?

a glue language, a scientific language

a pipeline framework mixing Python and Makefile

Köster, Johannes and Rahmann, Sven. Snakemake - A scalable bioinformatics workflow engine. Bioinformatics 2012.

Dedicated standalone such as genome coverage characterisation or a graphical user interface for Snakemake pipelines (Sequanix).

Snakemake as a workflow manager

= Makefile +

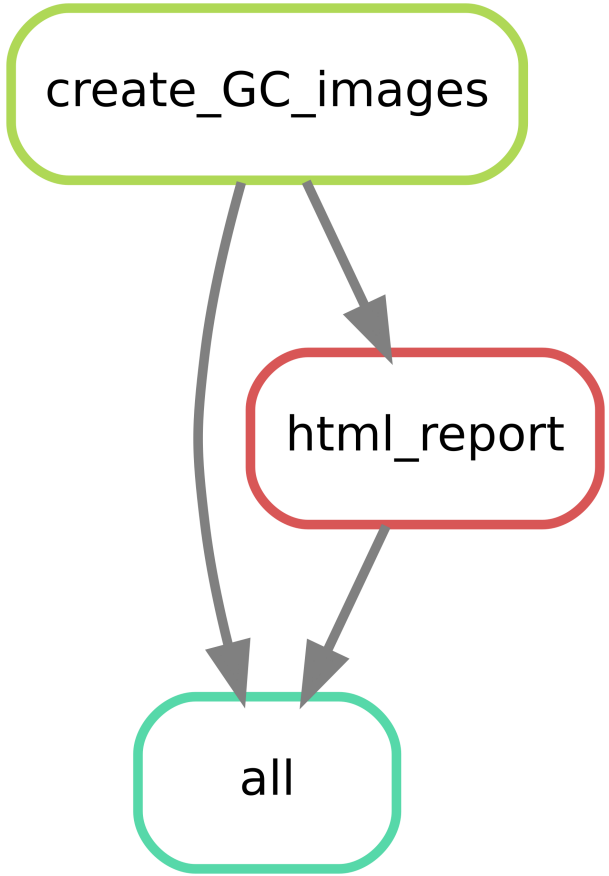
A minimalist snakemake workflow example (GC content)

```
# list of FastQ files without extension
files = ["A", "B", ....]

rule all:
    input: expand("{data}.png", data=files)
        "index.html"

rule create_GC_images:
    input: "{data}.fastq.gz"
    output: "{data}.png"
    run: # CODE TO COMPUTE IMAGES

rule html_report:
    input: expand("{data}.png", data=files)
    output: "index.html"
    run: # CODE TO CREATE HTML
```



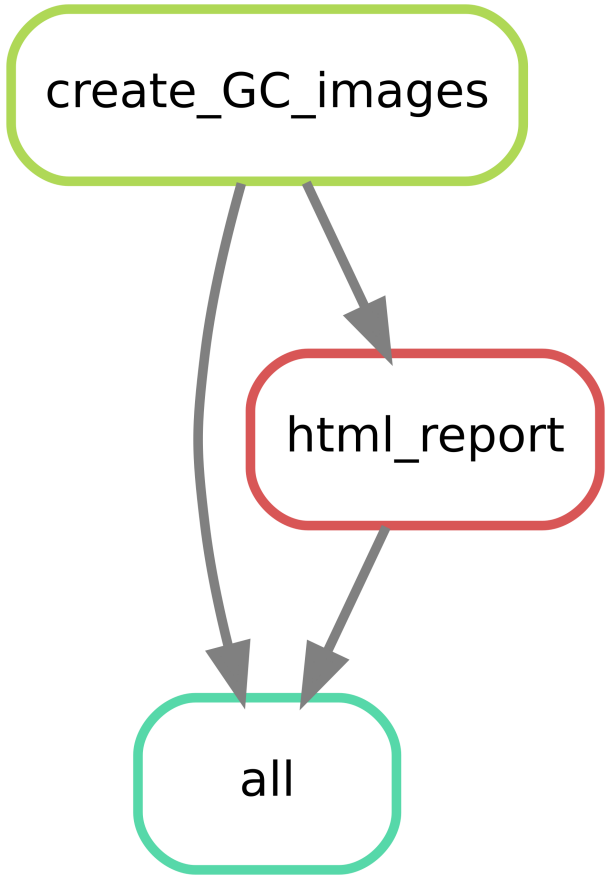
A minimalist snakemake workflow example (GC content)

```
# Use an external configuration file
import glob
files = glob.glob(config['gc']['directory']
files = [f.replace(".fastq.gz", "")
         for f in filenames]

rule all:
    input: expand("{data}.png", data=files)
        "index.html"

rule create_GC_images:
    input: "{data}.fastq.gz"
    output: "{data}.png"
    run: # CODE TO COMPUTE IMAGES

rule html_report:
    input: expand("{data}.png", data=files)
    output: "index.html"
    run: # CODE TO CREATE HTML
```



configuration file example in YAML format

```
##############################################################
# Input parameters for the fractal analysis
#
# :Parameters:
#
# - size: output image size formatted as NxM where N and M
#         are integers
# - depth: a integer (e.g. 200)
# - zoom: a positive value e.g. 0.5
# - N: number of random sets
gc:
    - window: 100
    - directory: /home/user/fastq_files
```

Execution

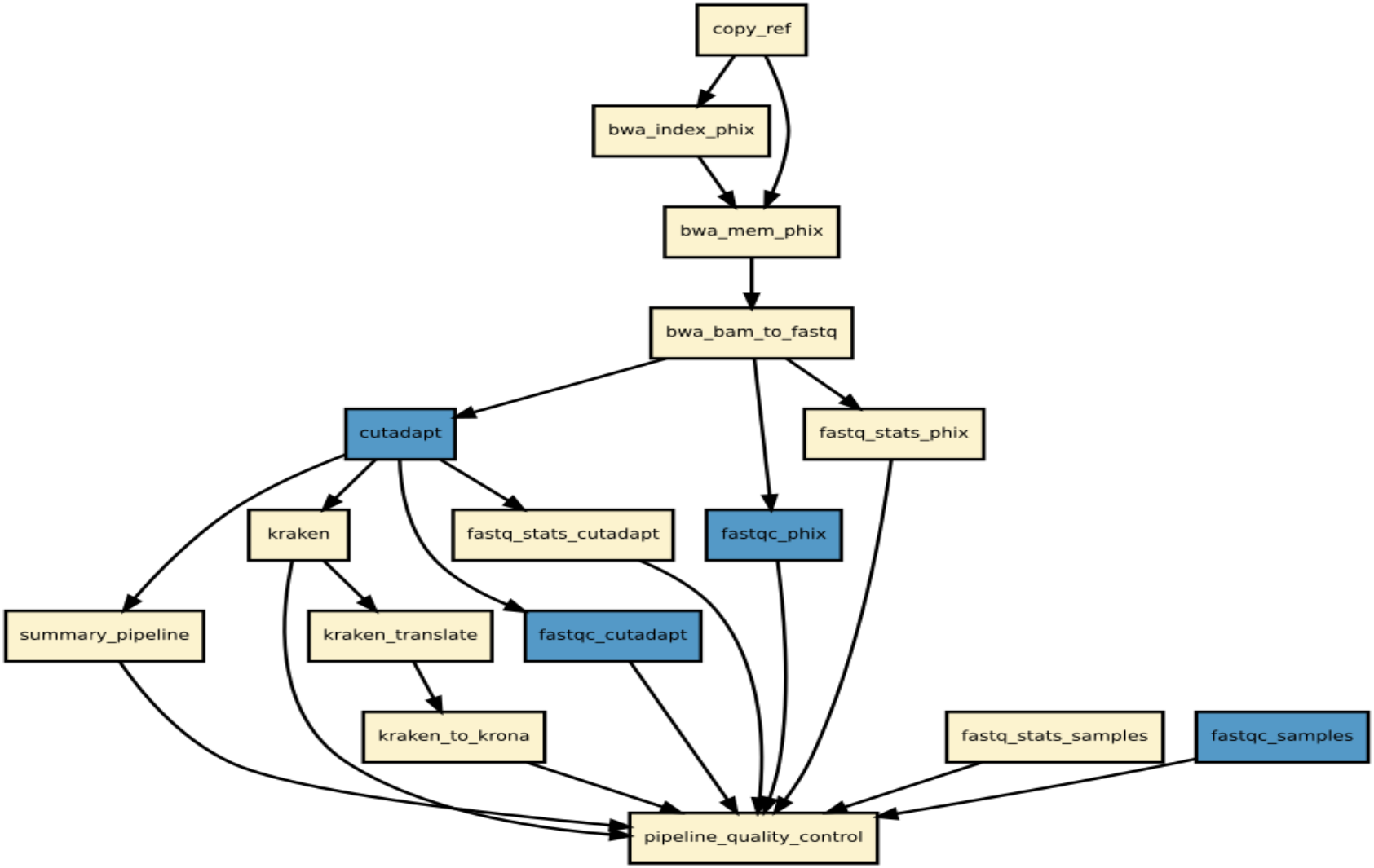
From a shell:

```
snakemake -s gc_minimalist.rules
```

More options if a configuration file is required, or execution is on a cluster, or . . . something goes wrong.

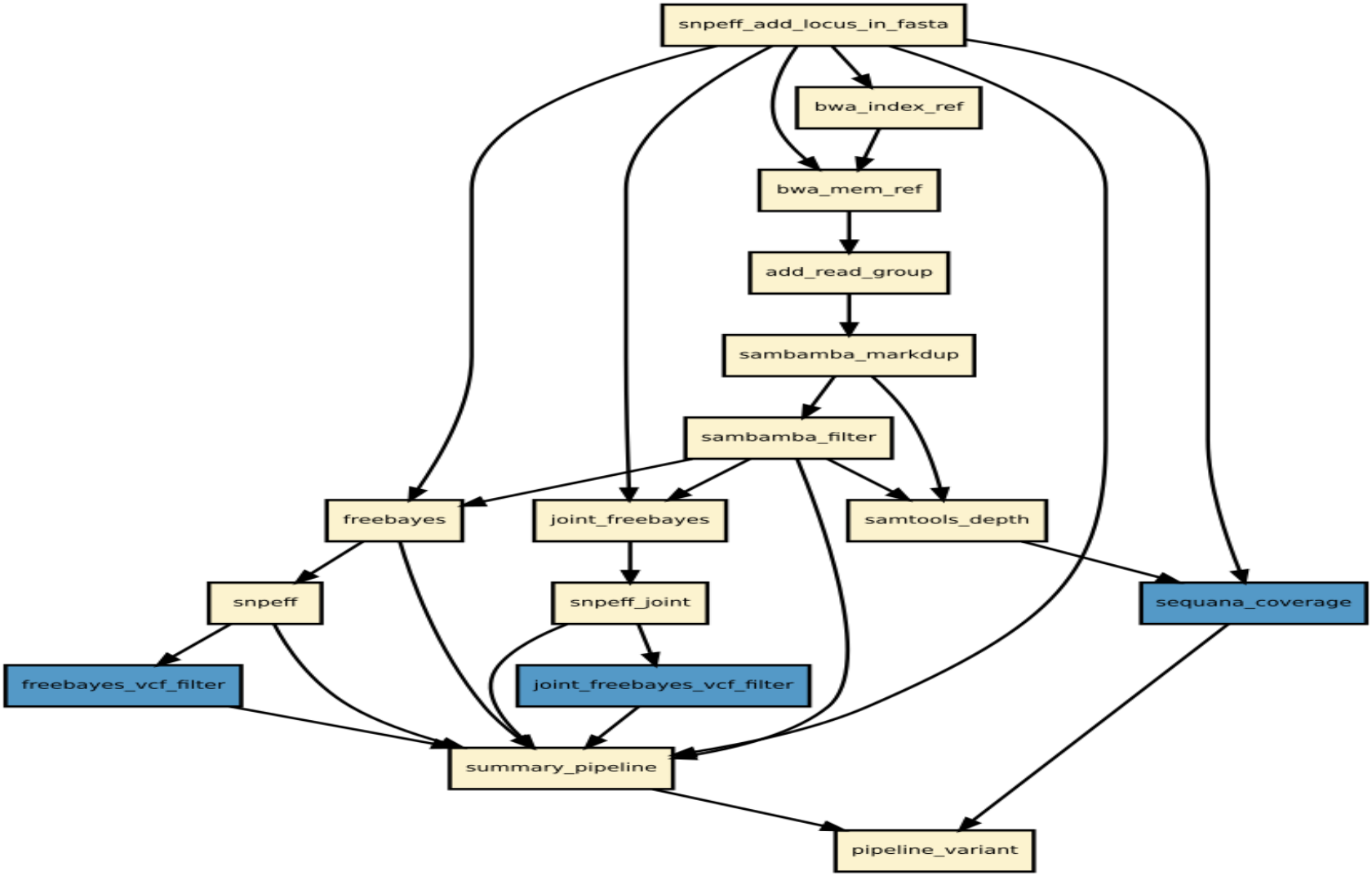
Sequana pipelines (an overview)

Pipeline example: quality control pipeline



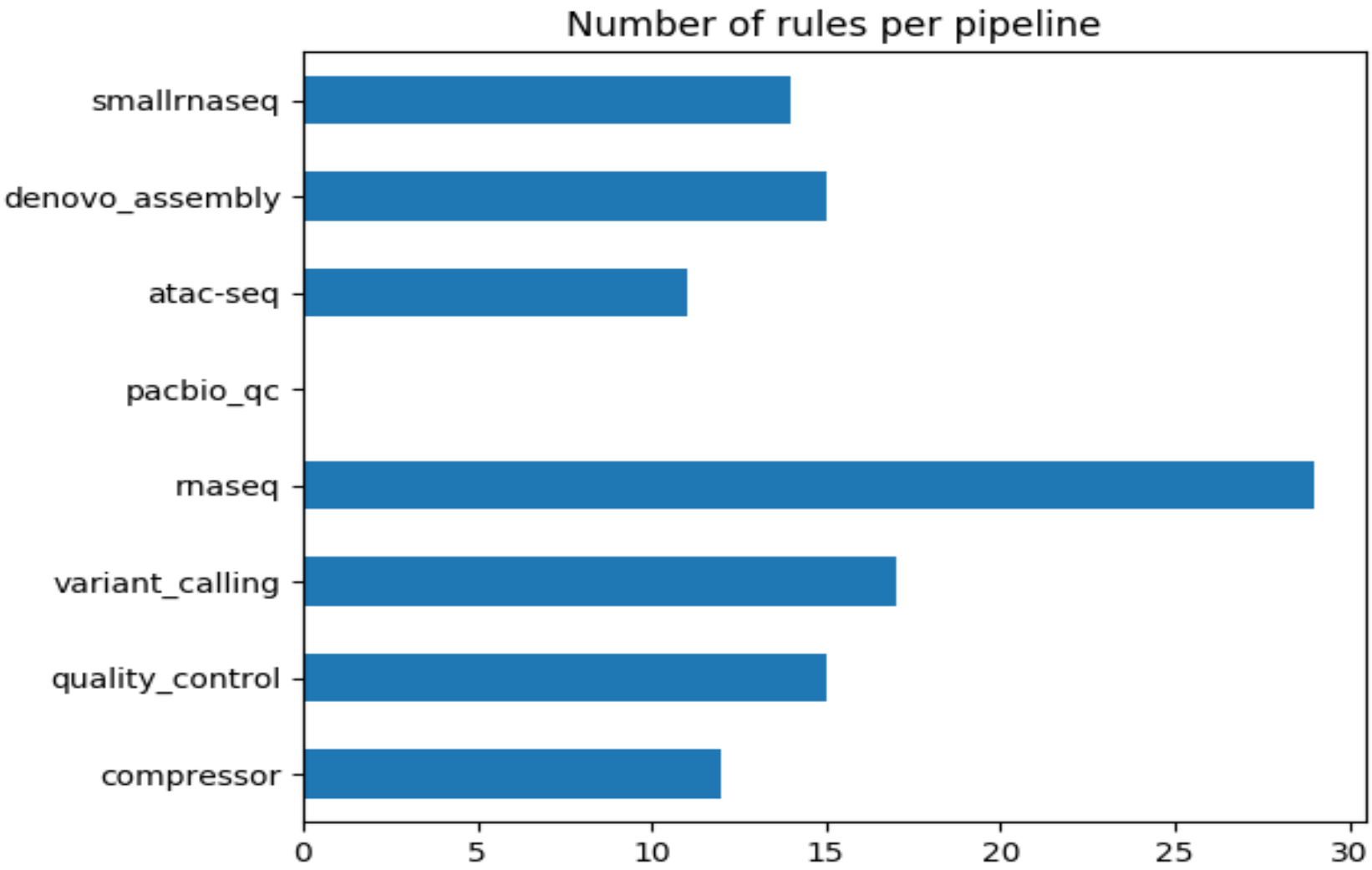
Quality control pipeline (sequana v0.5.2)

Pipeline example: variant calling



Variant calling pipeline (v0.5.2)

Pipeline complexity



Modularization: Factorise and reuse rules

Local standard rules

```
include: "path_to_rule_file"
```

Sequana rules

```
from sequana import snaketools as sm
include: sm.modules['rulegraph']
```

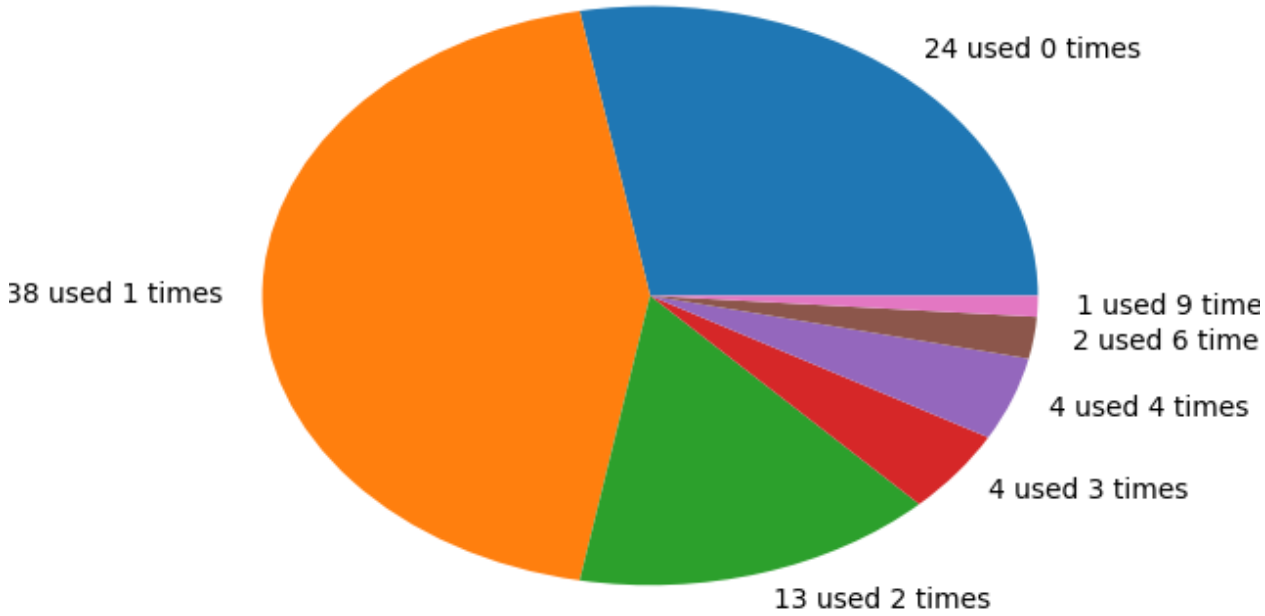
Dynamic rules:

```
sm.init("quality_control.rules", globals())
with open(sequana.modules["fastqc_dynamic"], "r")
    exec(dynrule.read())

manager = sm.PipelineManager("quality_control",
                             config)

include: fastqc_dynamic("example1", manager)
include: fastqc_dynamic("example2", manager)
```

Factorization



Once upon time there was a pipeline ...

Once upon time there was a pipeline ...

The Snakemake file assembles the rules together but all parameters can be externalised in a configuration file.

Once upon time there was a pipeline ...

The Snakemake file assembles the rules together but all parameters can be externalised in a configuration file.

One need to edit the configuration file ... without typos

Once upon time there was a pipeline ...

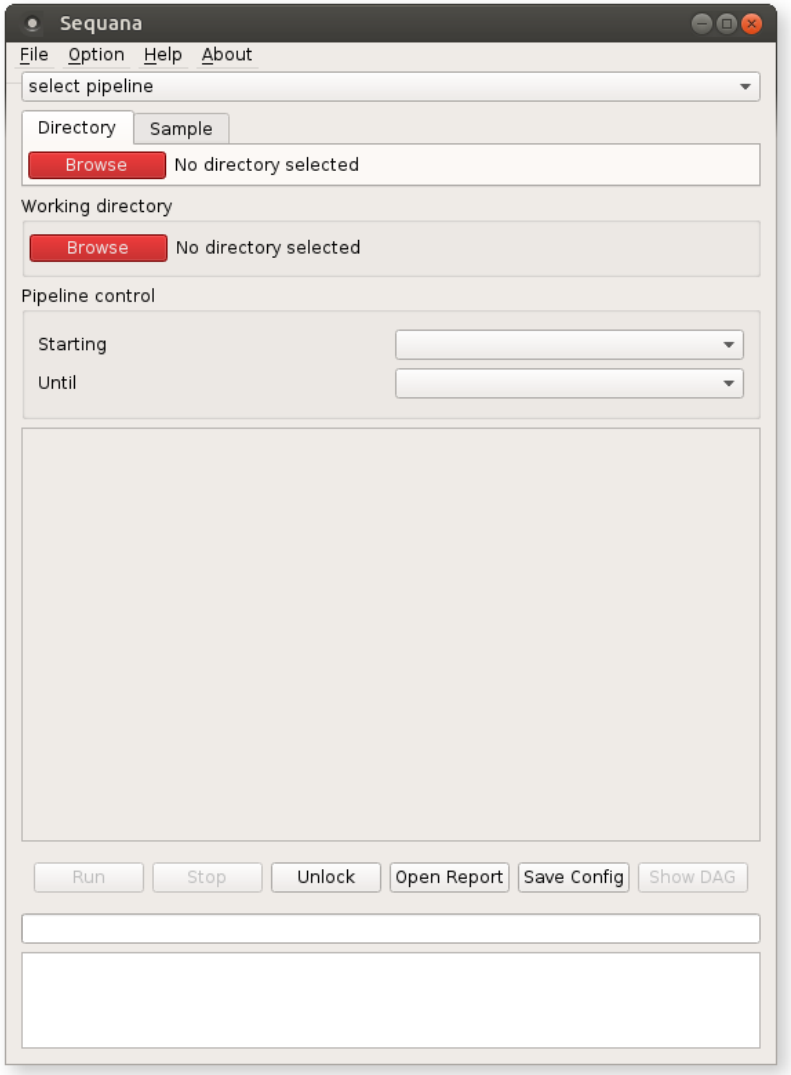
The Snakemake file assembles the rules together but all parameters can be externalised in a configuration file.

One need to edit the configuration file ... without typos

One need to launch the Snakemake command ... without typos

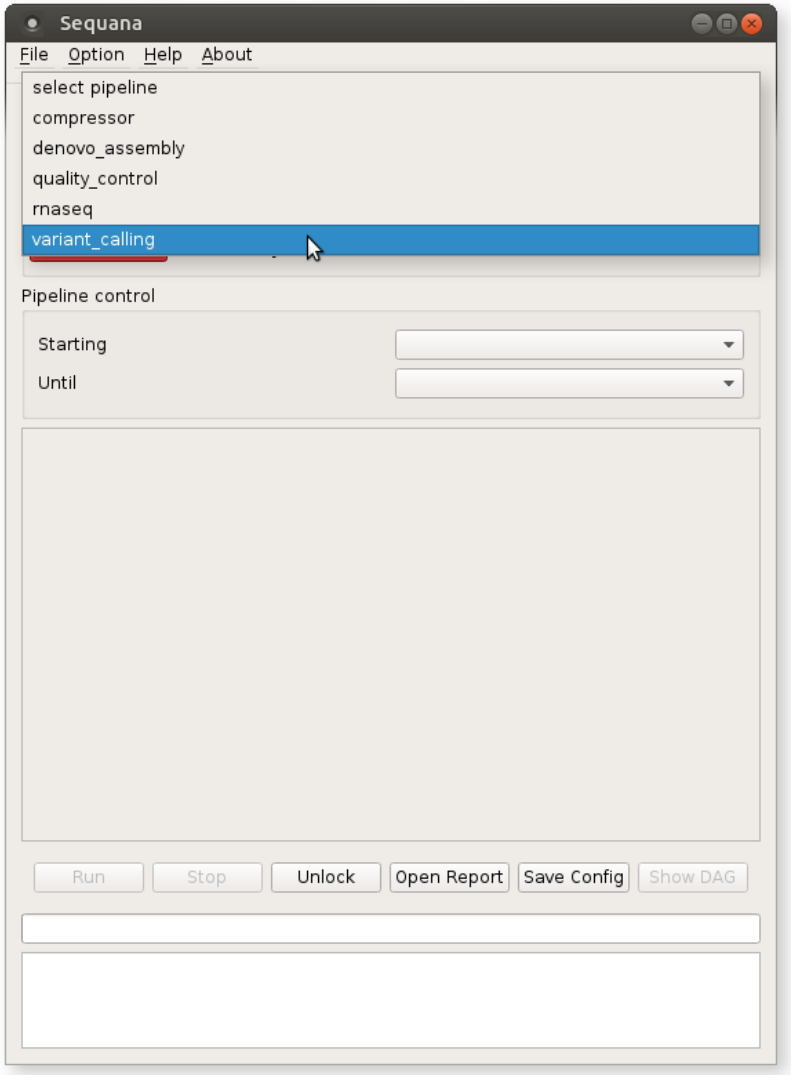
Sequanix

GUI to simplify the usage of snakemake



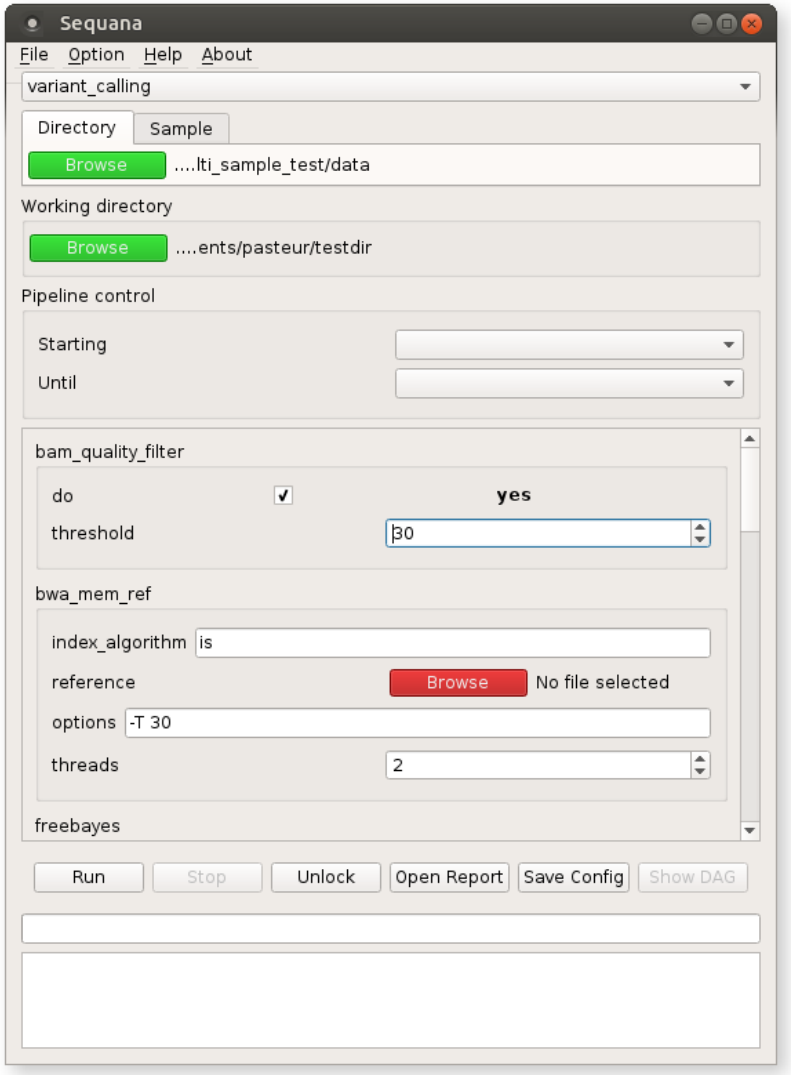
- Interface developed with PyQT5 and python
- Wrap our snakemake pipelines to ease the usage
- Usable on our cluster, which allows X11

GUI to simplify the usage of snakemake



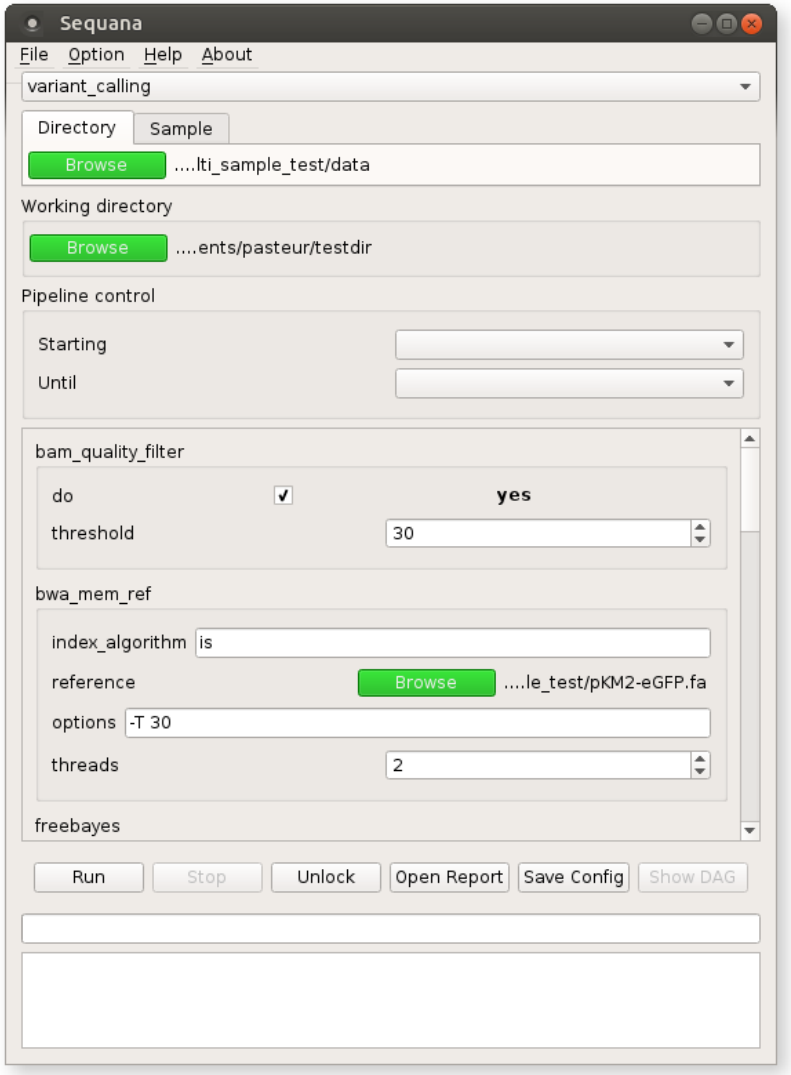
1. Choose a pipeline

GUI to simplify the usage of snakemake



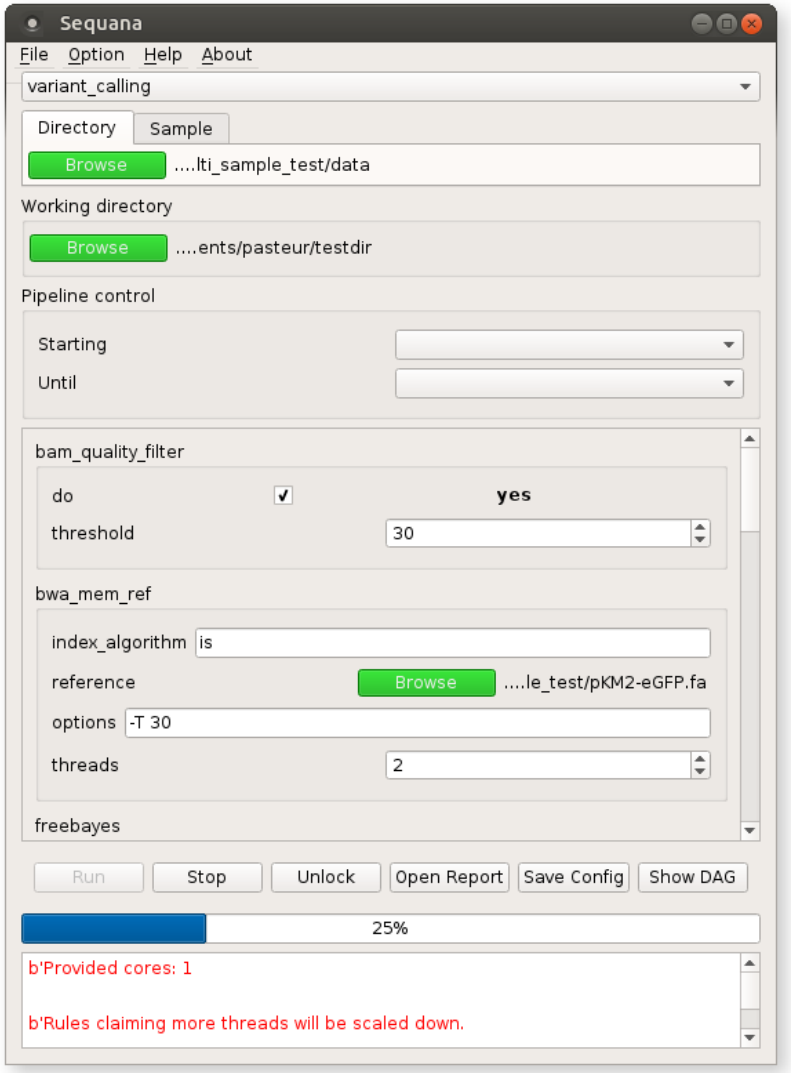
1. Choose a pipeline
2. Set input and output

GUI to simplify the usage of snakemake



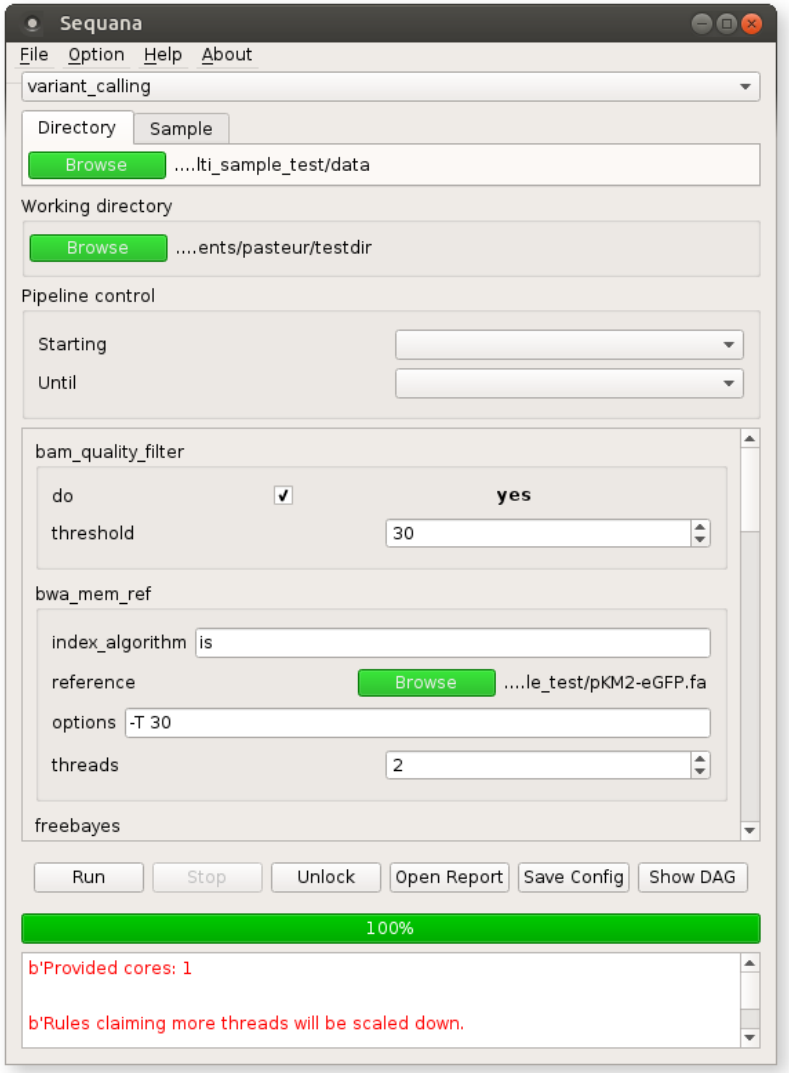
1. Choose a pipeline
2. Set input and output
3. Fill the config formular

GUI to simplify the usage of snakemake



1. Choose a pipeline
2. Set input and output
3. Fill the config formular
4. Run the pipeline

GUI to simplify the usage of snakemake



1. Choose a pipeline
2. Set input and output
3. Fill the config formular
4. Run the pipeline
5. Finished !

Reference

Sequanix: A Dynamic Graphical Interface for Snakemake Workflows

Dimitri Desvillechabrol, Rachel Legendre, Claire Rioualen, Christiane Bouchier, Jacques van Helden, Sean Kennedy, Thomas Cokelaer

https://www.biorxiv.org/content/early/2017/07/12/162701

Sequana coverage

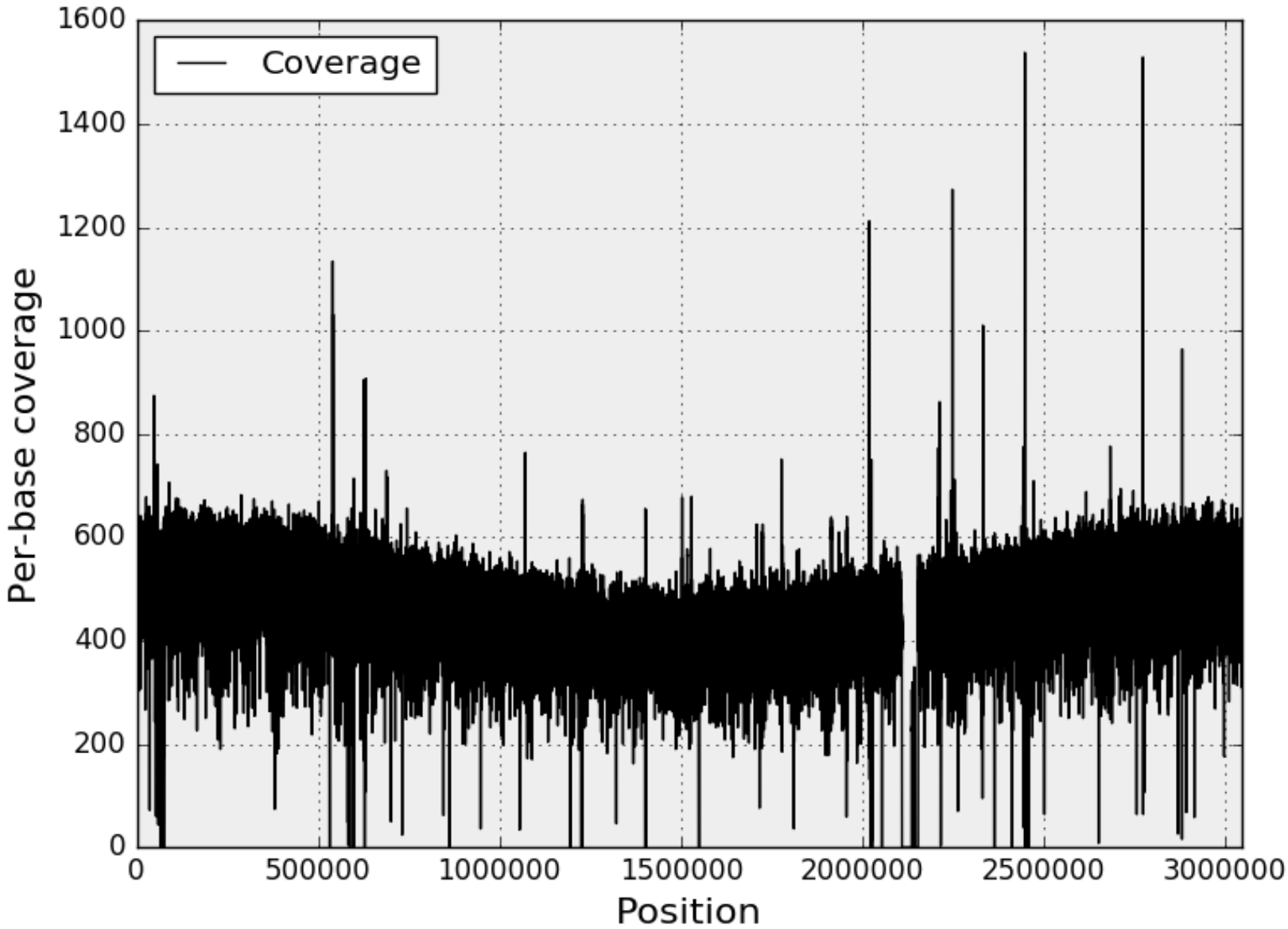
Genome coverage

Definition: The number of reads mapped to a specific position, b, within the reference genome.

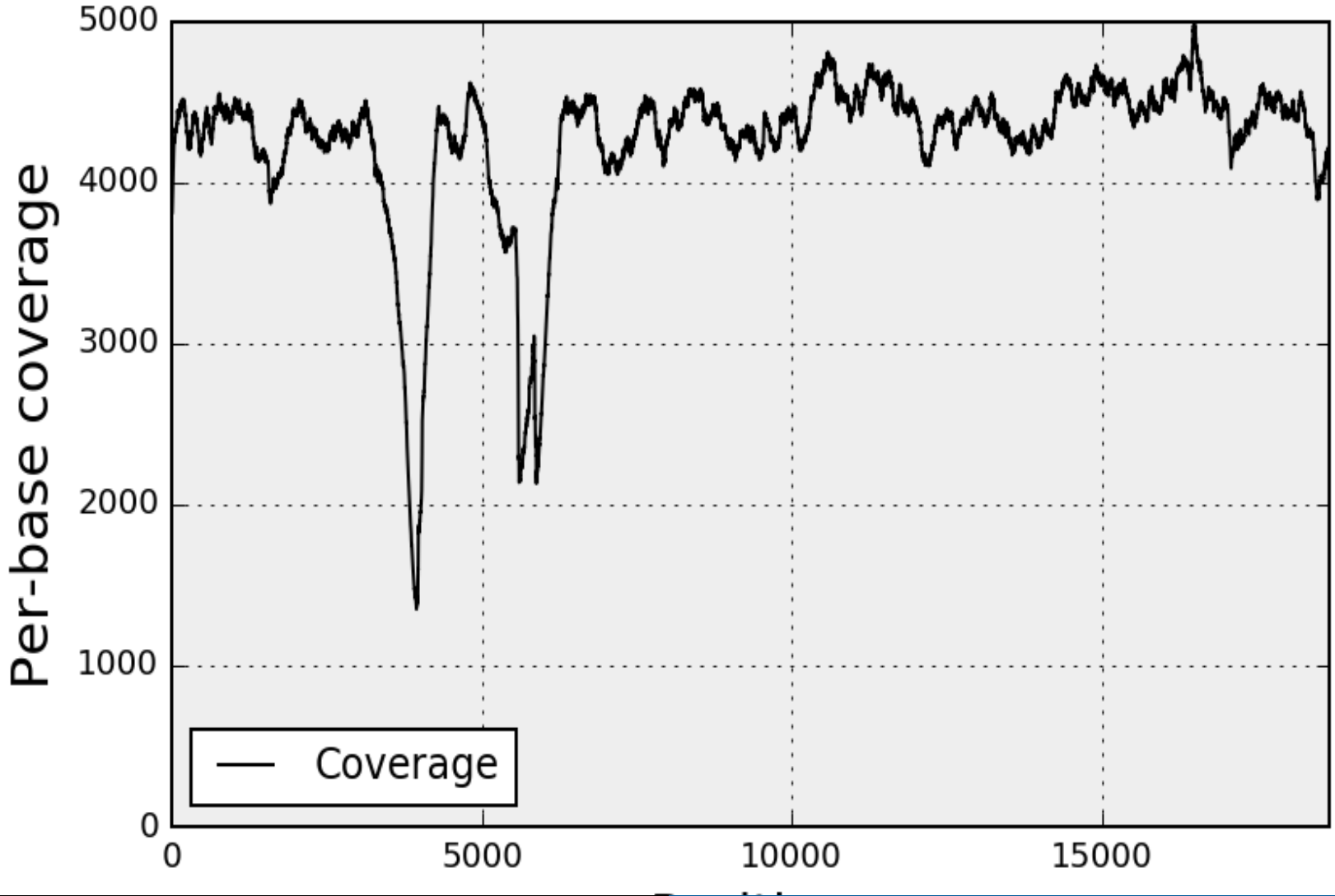
Notation: $C(b)$ also denoted C_b

Theoretical distribution: Poisson distribution but in practice over dispersed. The poisson parameter is distributed according to a Gamma hence leading to a negative binomial (See e.g., Linder et al 2013).

Bacteria case (low/high μ components and del. region)



Virus case


Per-base coverage
5000
4000
3000
2000
1000
0
0
5000
10000
15000
Coverage

Question: how to automatically detect and characterise under and over covered genomic regions

1. The algorithm
   1. Detrending (running median)
   2. Mixture model estimation (Gaussian approximation)
   3. Set a statistics (z-score)
   4. Clustering (double threshold)

1. Detrending

We divide C_b by its moving average (MA), or even better its running median (RM) defined as

$$RM_W(b) = \text{median}\left(C(b-V), \ldots, C(b+V)\right)$$

W is the running window and $V = (W-1)/2$.

The normalised genome coverage

$$\widetilde{C}_b = \frac{C_b}{RM_W(b)}$$

1. Detrending

We divide C_b by its moving average (MA), or even better its running median (RM) defined as

$$RM_W(b) = \text{median}\left(C(b-V), \ldots, C(b+V)\right)$$

W is the running window and $V = (W-1)/2$.

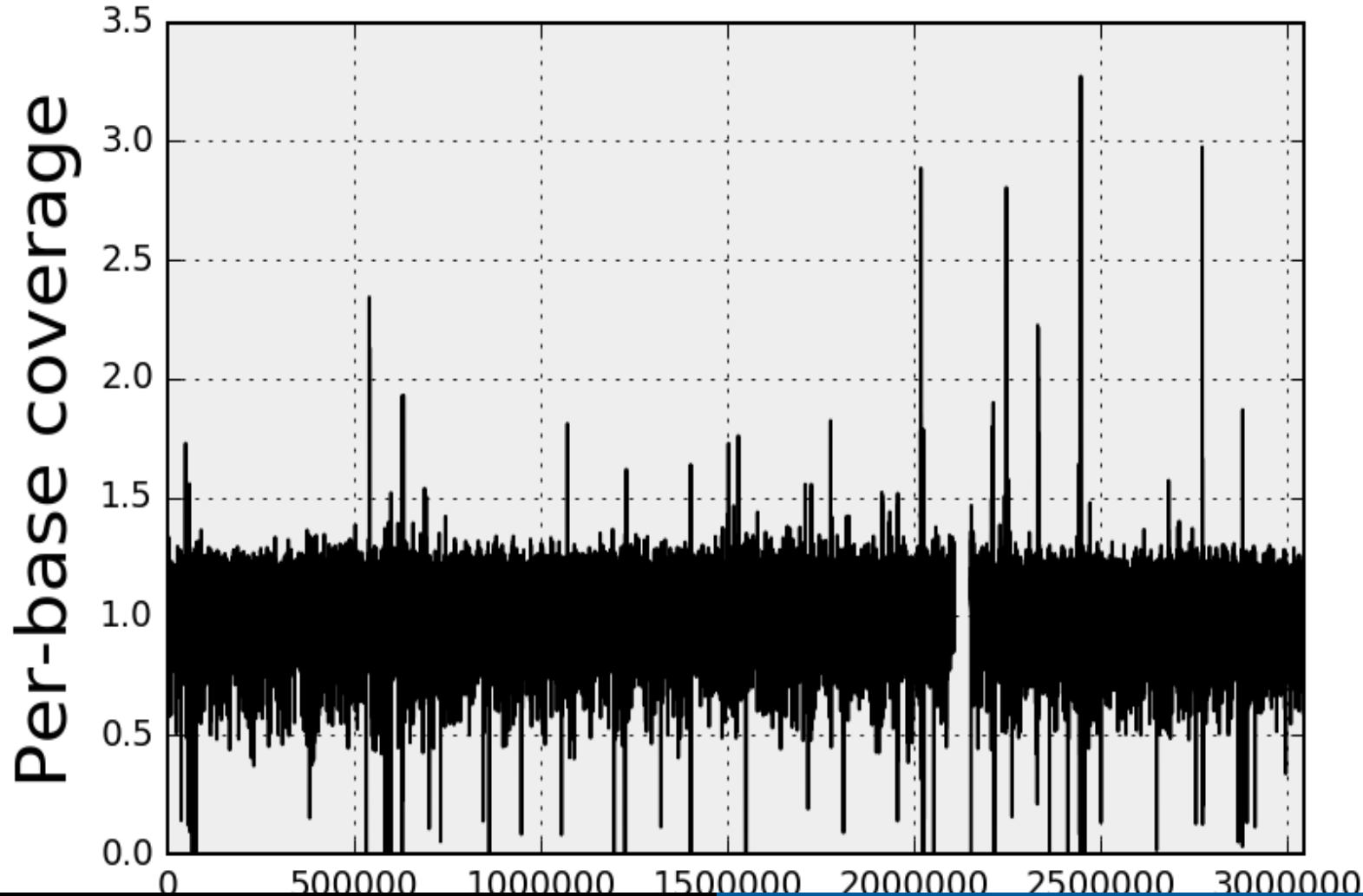
The normalised genome coverage

$$\widetilde{C}_b = \frac{C_b}{RM_W(b)}$$

Computational note

Running median is slow due to the sorting task.
Solution: a rolling window + a bisection method to insert new element in a sorted list + efficient insert/deletion in a list (B-tree) helps: **Only a few seconds to scan a 3Mb-length genome.**

Normalised coverage



2. Building a statistics

Definition: the normalised data can be decomposed into a **central** distribution $\widetilde{C}^0$ and a set of **outliers** $\widetilde{C}^1$ (above and below the central distribution)

$$\widetilde{C}_b = \left\{ \widetilde{C}_b^0, \{ \widetilde{C}_b^+, \widetilde{C}_b^- \} \right\}$$

Hypothesis 1: Central distribution is predominant.

$$\left| \widetilde{C}_b^0 \right| > \left| \widetilde{C}_b^1 \right|$$

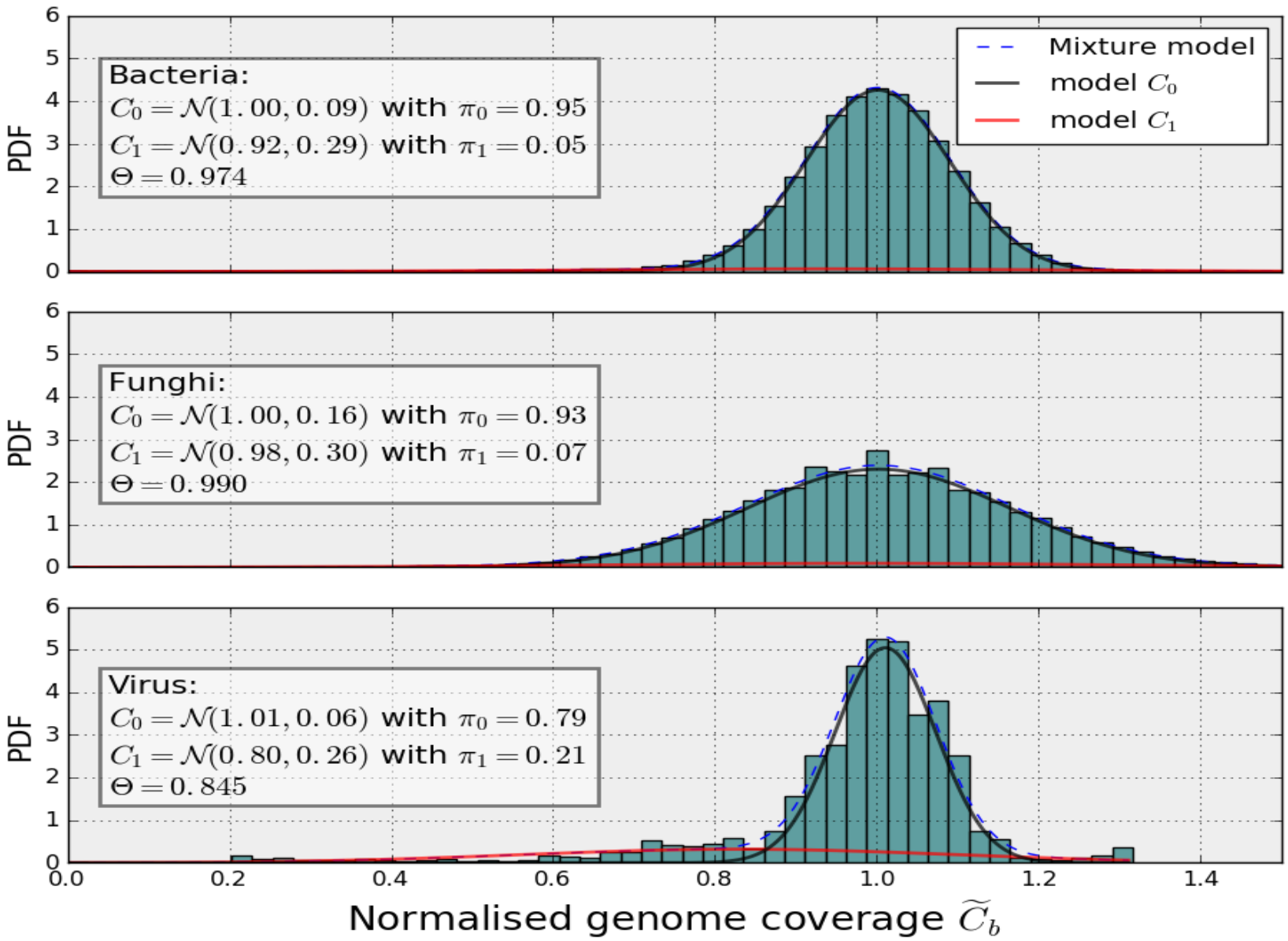
Hypothesis 2: The normalised genome coverage follows a Gaussian distribution in particular the central distribution

$$PDF(\widetilde{C}_b^0) \sim \mathcal{N}(\mu_0, \sigma_0)$$

We consider regime with $\delta \gg 1$. However, the algorithm works for low values: $\delta \sim 5 - 10$.

The central distribution C_0 will be fitted to a Gaussian distribution while the outliers will be fitted to another distribution.

- With an EM algorithm using $k = 2$ distributions we can estimate the parameters.
- The parameters of the outliers components are not used.
- The average of the central distribution is 1 (by definition)
- Note that the average of the outliers can be around 1 as well if the weights of C+ and C- are equivalent


Bacteria:
$C_0 = \mathcal{N}(1.00, 0.09)$ with $\pi_0 = 0.95$
$C_1 = \mathcal{N}(0.92, 0.29)$ with $\pi_1 = 0.05$
$\Theta = 0.974$
Mixture model
model C_0
model C_1
Funghi:
$C_0 = \mathcal{N}(1.00, 0.16)$ with $\pi_0 = 0.93$
$C_1 = \mathcal{N}(0.98, 0.30)$ with $\pi_1 = 0.07$
$\Theta = 0.990$
Virus:
$C_0 = \mathcal{N}(1.01, 0.06)$ with $\pi_0 = 0.79$
$C_1 = \mathcal{N}(0.80, 0.26)$ with $\pi_1 = 0.21$
$\Theta = 0.845$
PDF
Normalised genome coverage $\widetilde{C}_b$

C. From a constant to adaptative z-score

Assuming that the central distribution is the null hypothesis, we can now assign a z-score in the normalised space:

$$z(b) = \frac{\widetilde{C}(b) - \widetilde{\mu}_0}{\widetilde{\sigma}_0}$$

We can replace $\widetilde{C}_b$ by its expression $(C_b/RM_W(b))$ and express C_b as a function of the running median, the $z(b)$ and the parameters of the central distribution:

$$C(b) = (\widetilde{\mu}_0 + z(b)\widetilde{\sigma}_0)\, RM_W(b).$$

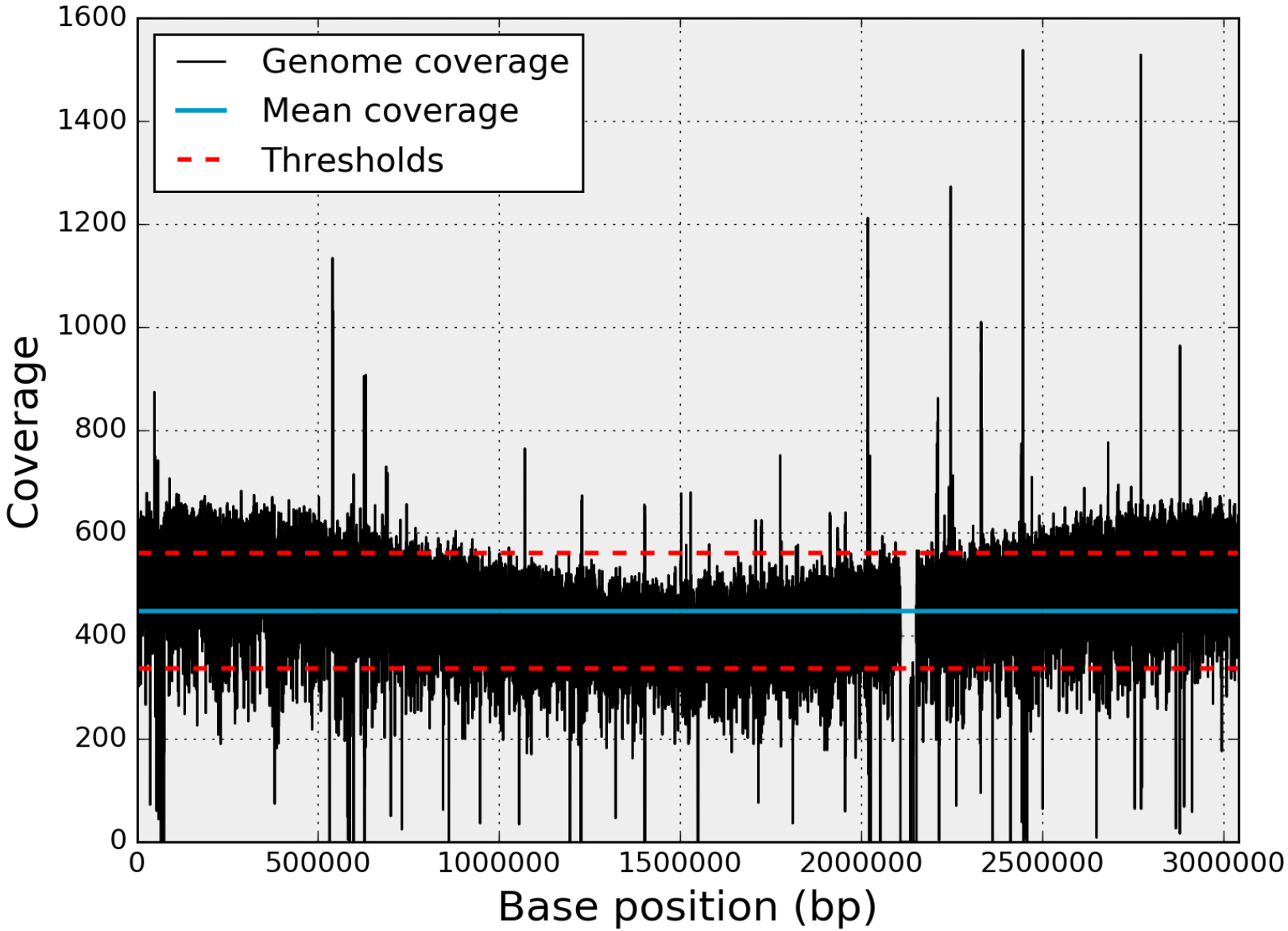
C. From a constant to adaptative z-score

We can now set a constant threshold in the normalised space (e.g. $\lambda \pm 3$) and get an adaptative threshold in the original space.

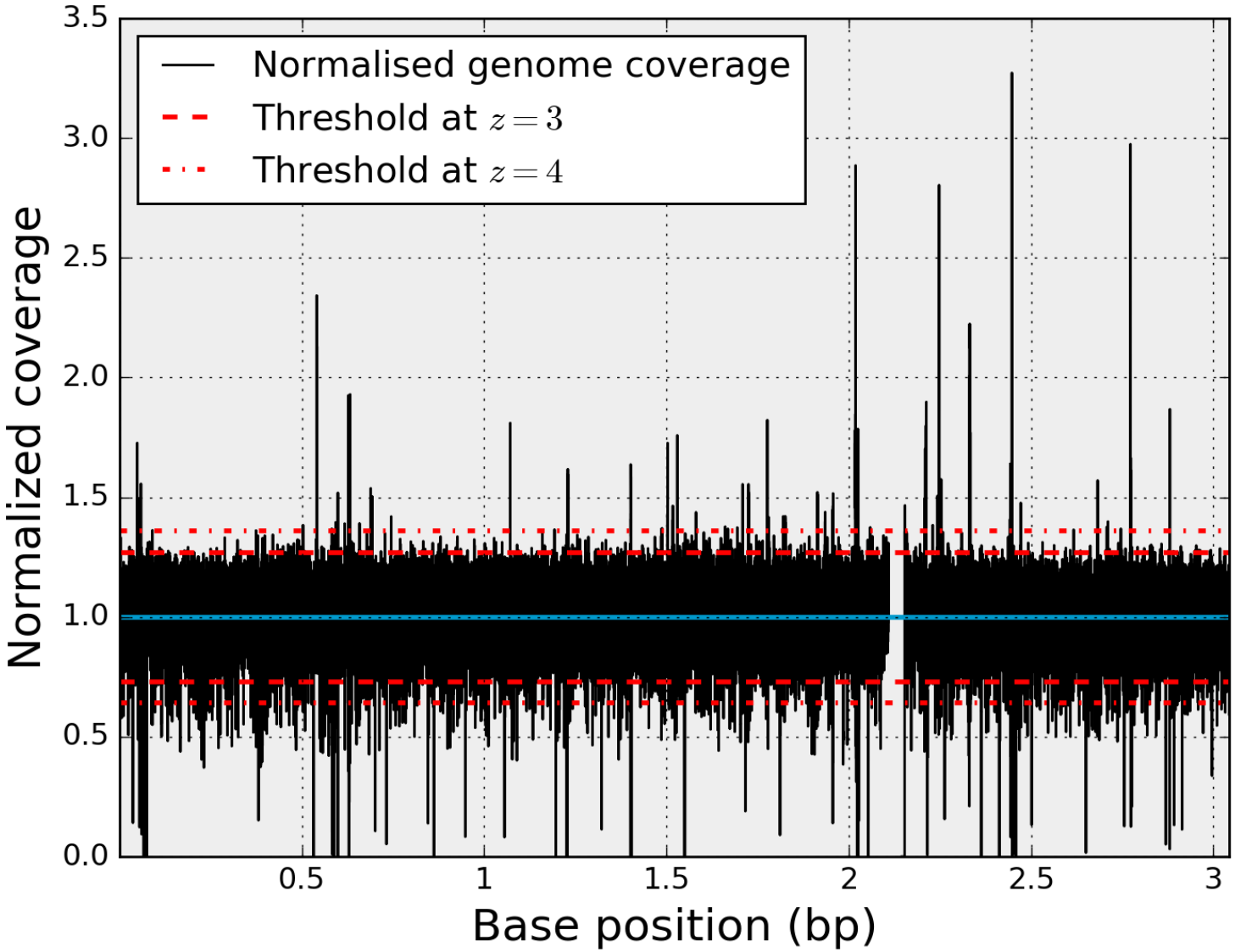
We can derive an adaptative threshold in the original space that is function of the genome position:

$$\tilde{\delta}^{\pm}(b) = (\widetilde{\mu}_0 \pm \lambda^{\pm} \times \widetilde{\sigma}_0)\, RM_W(b). \quad (1)$$

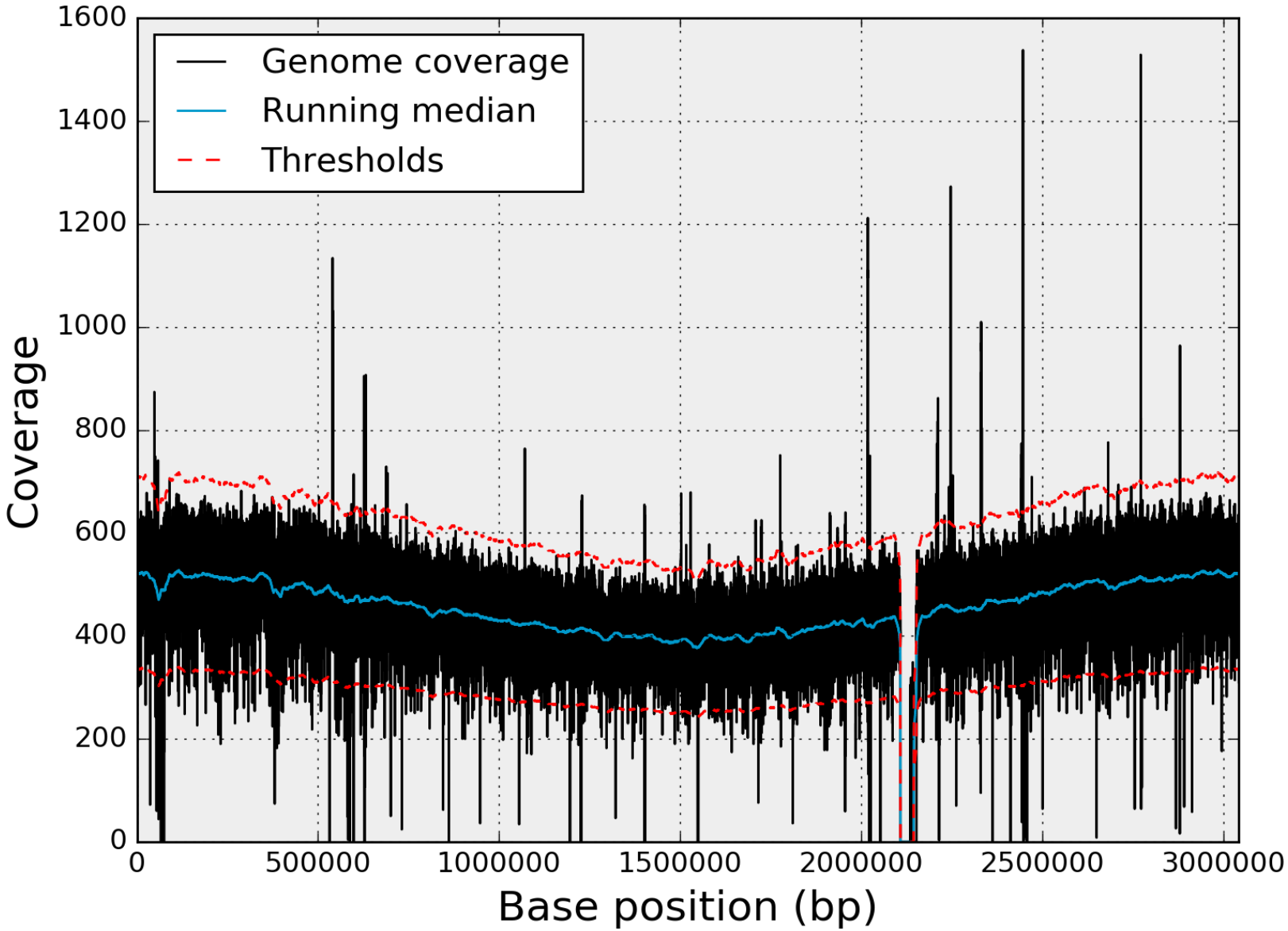
constant thresholds in the original space



constant thresholds in the normalised space



adaptative thresholds in the original space



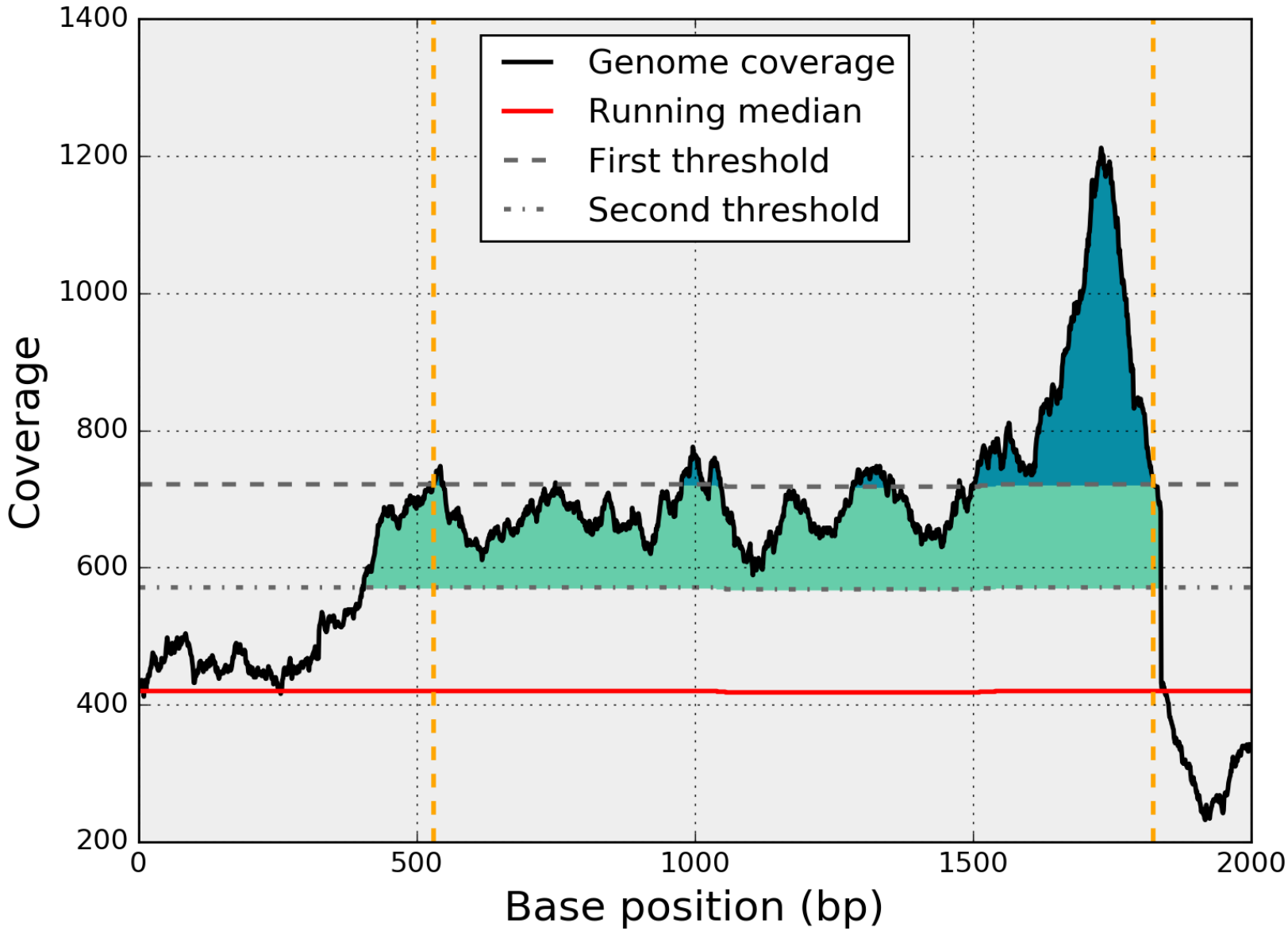
D. Regions of interest (clustering)

On a 3Mb genome with low thresholds the number of outliers are high. $\lambda = 3$ means about 4000 events by pure chance.

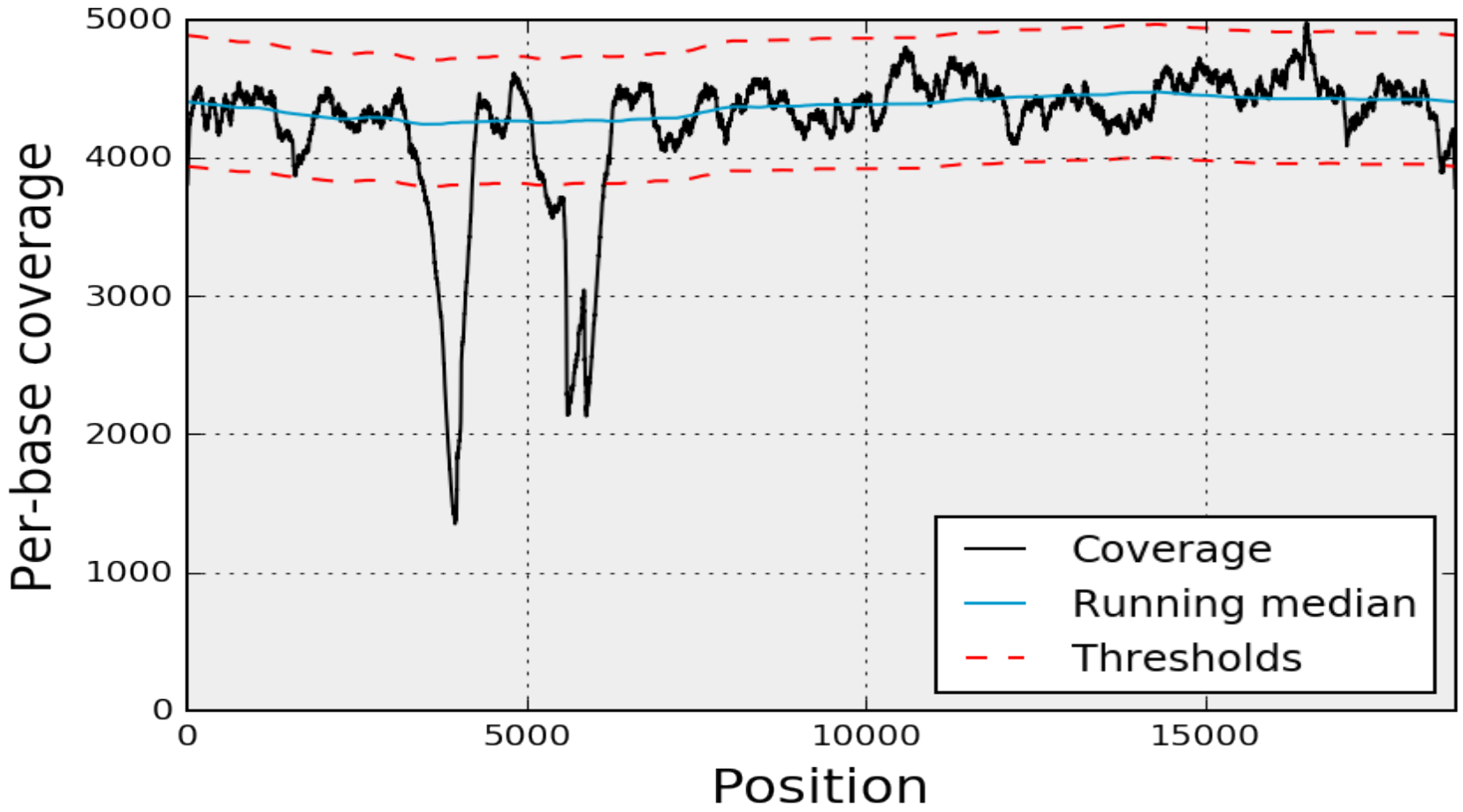
We need a strategy to reduce the number of interesting events. This is achieved by clustering the data.

Besides, to cluster close-by events further, we proceed with a double-thresholds approach.

Double thresholds



```
sequana_coverage --download-genbank JB409847
sequana_coverage --input JB409847.bed -w 3001 --circular
                 --genbank JB409847.gbk --show-html
```



Summary

- A robust and fast algo. to detect under/over covered regi

- The algorithm is made of 3 steps:
  1. Normalisation (running median)
  2. Set a statistics using EM to estimate mixture model
  3. Clustering of events in original space

- Implementation in Sequana as a standalone
  - HTML reports with ROIs provided as sortable tables
  - Identify repeated regions
  - A genbank can be provided to annotate the report

Dimitri Desvillechabrol, Christiane Bouchier, Sean Kennedy, Thomas Cokelaer *Detection and characterization of low and high genome coverage regions using an efficient running median and a double threshold approach.* bioRxiv 092478; doi: http://dx.doi.org/10.1101/092478

Sequana: Continuous integration

Versioning, Test and Documentation

https://github.com/sequana/sequana

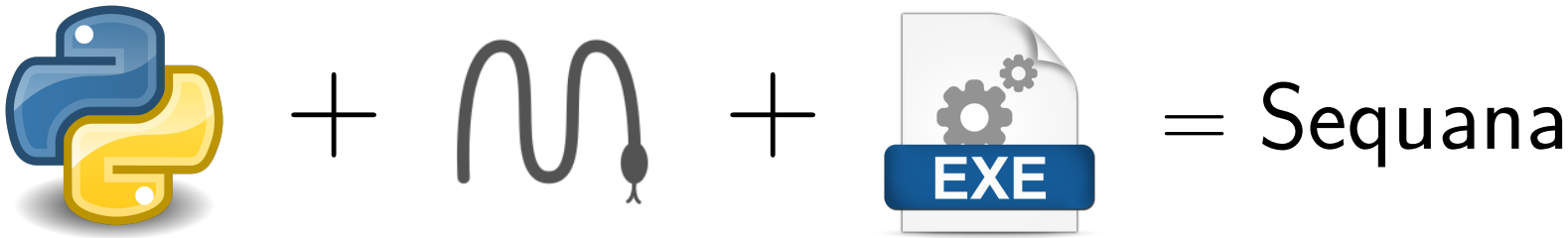
Continuous Integration on Travis with 185 tests with 85% coverage

Uses Sphinx (RST syntax) to document the source code and provides user guide.

Updated after each commits on sequana.readthedocs.io

Summary

Sequana in a nutshell



= Sequana

http://sequana.readthedocs.io

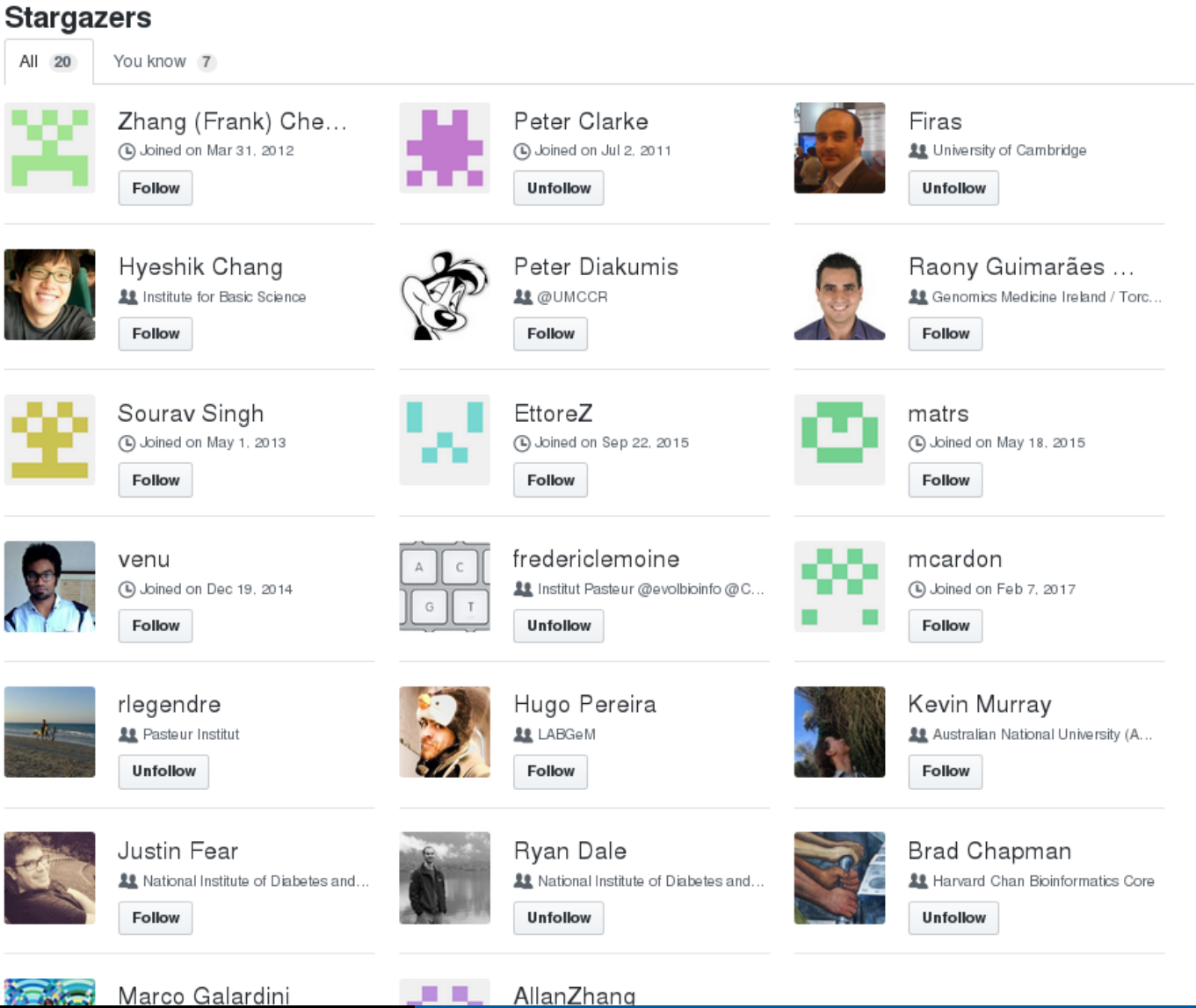
Summary

Sequana is a versatile tool that provides

1. A Python library dedicated to NGS analysis (e.g., tools to visualise standard NGS formats).
2. A set of snakemake workflows and rules dedicated to NGS
3. A GUI to execute them easily with Sequanix
4. HTML reports
5. Standalone applications:
   - **sequana_coverage** ease the extraction of genomic regions of interest and genome coverage information
   - **sequana_taxonomy** get a quick overview of read contents
   - ...

You like it ? Please, add a star on our github

https://github.com/sequana/sequana/stargazers



You like it ?

Join us ! add rules and pipelines !

Acknowledgements

- Dimitri Desvillechabrol (variant calling, denovo, sequana, sequanix)
- Rachel (Transcriptomics)
- Mélissa Cardon (pacbio)
- Biomics users (Institut Pasteur)

Thank you

Slides available on
http://github.com/sequana/sequana_presentations/2017/ENS