ngsCAT
Capture Assessment Tool for NGS data

ngscat.bioinfomgp.org

January 24, 2014

Contents

1. What is ngsCAT?

ngsCAT (Capture Assessment Tool for Next Generation Sequencing data) is a Linux command-line application written in Python which facilitates a comprehensive evaluation of the performance of the capture step in targeted high-throughput sequencing experiments in terms of:

- Sensitivity, which assesses the quality of the coverage on target regions.
- Specificity, which measures how much of the sequencing effort is wasted on sequencing off-target bases.
- Uniformity, which assesses sequencing biases due to specific genomic locations or nucleotide composition.

2. Obtaining ngsCAT

ngsCAT source code can be downloaded from the following link. ngsCAT is made available for non commercial research purposes only under the GNU General Public License. However, notwithstanding any provision of the GNU General Public License, ngsCAT software may not be used for commercial purposes without explicit written permission after contacting bioinfo@bioinfomgp.org

3. Requirements, dependencies and installation

3.1. Requirements

Although an efficient and multithreaded implementation of ngsCAT is provided, enabling a full execution for human exome data, some minimal requirements are required for a correct running of the tool:

- Operating system: Linux-based OS
- Computer processor: multi-core processor

- Computer memory: 4GB or more

3.2. Dependencies

ngsCAT requires a python interpreter (it has been tested in v.2.6 and v.2.7) and third-party software commonly used by the scientific community in the bioinformatics area. All of them are briefly described below:

- *samtools*: a set of utilities that manipulate alignments in the BAM format.
- *numpy*: a python package for scientific computing with Python.
- *scipy*: a python library which provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization.
- *xlwt*: a python library for generating spreadsheet files that are compatible with Excel 97/2000/XP/2003, OpenOffice.org Calc, and Gnumeric.
- *matplotlib*: a python library which produces high-quality 2D graphics for interactive graphing, scientific publishing, user interface development and web application servers targeting multiple user interfaces and hardcopy output formats.
- *pysam*: a python module for reading and manipulating Sam/Bam files.

3.3. Installation

Users may choose between either a manual installation (section 3.3.1.) or an automatic installation (section 3.3.2.).

3.3.1. Manual installation

Through a linux console, move to the directory where the tool has been downloaded and extract it to the desired directory:

```
> tar -xzvf ngs_cat.v0.1.tar.gz [-C /target/directory]
```

If desired, add ngsCAT to linux PATH and allow execution grants by means of the following command:

```
> PATH=\$\{PATH\}:[/target/directory/]ngs_cat.v0.1/ >> ~/.bashrc
> export PATH >> ~/.bashrc
> chmod +x [/target/directory/]ngscat.v0.1/ngscat.py
```

With regard to ngsCAT dependencies, samtools, numpy, scipy, xlwt and matplotlib can be installed from the software repositories provided by the corresponding O.S. distribution. For example, for a Debian/Ubuntu-based linux distribution, type on the terminal:

```
> apt-get install [python2.7] samtools python-numpy python-xlwt\
 python-scipy python-matplotlib
```

For a Fedora-based Linux distribution, type:

```
 > yum install samtools numpy scipy python-xlwt\
 python-matplotlib python-matplotlib-qt4
```

Pysam 0.7.5 can be downloaded from the pysam project home. Once it has been dowloaded, type on the terminal:

```
> tar -xzvf pysam-0.7.5.tar.gz
```

Build and install pysam 0.7.5:

```
>cd pysam-0.7.5
>python setup.py build
>python setup.py install
```

3.3.2. Automatic installation

With the aim of making the installation of ngsCAT and its dependencies easier, we provide in the downloads section scripts that automatically download and install ngsCAT source code and its dependencies for different Linux-based distributions:

- For Ubuntu-based linux distribution: *setup_ubuntu.sh* (it has been tested in Linux Mint 12, 13 and 15 and Ubuntu 12.04 and 13.10).
- For Debian-based linux distribution: *setup_debian.sh* (it has been tested in Debian 7).
- For Fedora-based linux distribution: *setup_fedora.sh* (it has been tested in Fedora 20).

To run such script, type on the terminal:

```
sudo -E ./setup_<your_distribution>.sh <targetAbsolutePath>
```

where $< your\_distribution >$ is either ubuntu, debian or fedora and $< targetAbsolutePath >$ is the absolute path of the directory where ngsCAT will be downloaded. For example:

```
sudo -E ./setup_ubuntu.sh /home/user/ngsCAT
```

After running this script, ngsCAT source code and its dependencies are automatically downloaded and installed in your computer. **If you experience problems with the installation or your distribution is not in the previous list, please, contact us**: bioinformatica at bioinfomgp.org

4. Running ngsCAT

Once the installation and the dependencies have been installed, *ngsCAT* can be run. All the options available for the tool are shown when the following command is run:

```
> ngscat.py -h
Usage:
********************************************************************************
```

```
Task: Assesses capture performance in terms of sensibility, specificity
              and uniformity of the coverage.
Output: An html report will be created at the path indicated
               with the --out option.
*************************************************************************


usage: ngscat.py --bams <filename> --bed <filename> --out <path>
               --reference <filename> --saturation <{y,n}> --depthlist <list>
               --tmp <path> --threads <integer>

Options:
  -h, --help            show this help message and exit
  --bams=BAMS           Required. Absolute paths of comma separated list of bam
                        files (2 maximum). E.g.: --bams
                        /home/user/bam1.sorted.bam,/home/user/bam2.sorted.bam
  --bed=BED             Required. Absolute path to the bed file containing the
                        target regions.
  --out=OUT             Required. Absolute path to the directory where results
                        will be saved.
  --extendtarget=EXTEND
                        Optional. Integer indicating the number of bases to
                        extend each target region up and down-stream.
                        Default=None.
  --reference=REFERENCE
                        Optional. String indicating the absolute path to a
                        .fasta file containing the reference chromosomes.
Default=None.
  --saturation=SATURATION
```

```
                        Optional. {y,n} to indicate whether saturation curve
                        should be calculated. Default=n.
  --depthlist=DEPTHLIST
                        Optional. Will only be used in case --saturation is
                        "y". Comma separated list of real numbers (do not
                        leave spaces between) indicating the number of
                        millions of reads to simulate for the saturation
                        curve. E.g.: 1,5,10 would indicate 1*10^6, 5*10^6 and
                        10*10^6. Default=auto.

  --coveragethrs=COVERAGETHRESHOLDS
                        Optional. Comma separated list of real numbers (do not
                        leave spaces between) indicating coverage thresholds
                        to be used when calculating percentages of covered
                        bases (first graph in the report).
                        Default=1,5,10,20,30.
  --onefeature=FEATURE  Optional. Use this option if just one of the
                        graphs/statistics should be calculated. String
                        indicating one of the following features:  {'percbases
                        ','saturation','specificity','coveragefreq',
                        'coveragedistr', 'coveragestd', 'gcbias'}.
  --tmp=TMP             Optional. String indicating the absolute path to a
                        temporary directory where temporary files will be
                        created. Default=/tmp/.
  --threads=NTHREADS    Optional. Integer indicating the number of concurrent
                        threads to launch. Default=2.
```

Some example datasets and reports generated by *ngsCAT* are given in the Downloads section together with the source code of the tool for a better understanding of the metrics used to evaluate the efficiency of targeted enrichment sequencing experiments (see Sections 7. and 8. below).

4.1. Important issues

- When the program is run for the first time for a given experiment (BAM file), an index file related to that BAM file is automatically created in the temporal directory (*–tmp argument*) through the *Pysam* package. This way, it is ensured that errors do not appear when running *ngsCAT* because of incompatibilities with indexes created with other packages (such as *samtools*). If the same sample (BAM file) is analyzed again through *ngsCAT* and the temporal directory does not change, the index will not be created again.
- BAM file must contain only mapped reads.
- Be aware of the resources available in your computer. The more number of threads are launched when running *ngsCAT* through the *–threads* argument, the more resources (memory, use of CPU, etc) of the computer are used.
- Be consistent with the identifiers of your chromosomes or contigs. For example, if your BED file contains regions with identifiers of type *chr*, be sure that your reads have been mapped against a reference genome with identifiers of type *chr*.
- There is no a limitation regarding the number of chromosomes and/or contigs present in an experiment.
- There is no a limitation related to the number of ROIs or their length.
- Reading off-target bases is inherent to the capture technology. ngsCAT provides the option of "extending" ROIs up/down-stream (*––extendtarget* parameter) and running the analysis with this extended target. This enables the user to assess whether metrics significantly differ from those calculated with the original target bed, and therefore to assess the importance of the set of off-target reads located at flanking regions of the ROIs.

5. Output generated

ngsCAT will generate in the output directory (–*out* argument) the HTML report named *captureQC.html* with all metrics used to assess the efficiency of targeted enrichment sequencing and two folders:

- *data*: contains figures and tables embebbed in the HTML report (will be explained below) and a set of bed files per chromosome. Two bed types can be found, both of which are formatted so that can be visualized in genome browsers such as the one of UCSC or IGV (bed files are in Bed-Graph Track Format):
  - .bed: that can be used to visualize per-base coverage along chromosomal positions
  - .off.bed: contain the coverage of off-target bases which are located far from ROIs (> 1 Kb and which present coverage $> 15\times$ by default). These tracks allow the user to easily locate clusters of off-target reads, which may be indicative of low-specific probes in capture design. Thresholds of coverage and minimum distance to the ROIs can be modified at the config.py file (see explanations below about this configuration file).
- *img*: some miscellaneous images needed by the HTML report (not of interest).

For each section in the HTML report, a minimum threshold value can be set to produce a warning if the values of the tested experiment are beyond the preset threshold. *ngsCAT* uses a configuration file named *config.py* which sets the different thresholds with default values, but can be changed by the user prior to the execution of ngsCAT. Each threshold will be explained in its corresponding section below. **Please, note that the criteria to decide whether a particular experiment was successful, or not, are dependent on the capture design, sequencing platform and data analysis pipeline.**

5.1. Input parameters and summary sections

The first two sections in the report generated by *ngsCAT* are:

- Input parameters: it is shown information about the input files (BAM and target files), execution date, information about the reference genome, temporary directory used and other miscellaneous information related to the different metrics used to assess the efficiency of targeted enrichment sequencing.
- Summary table, which shows a summary of the different metrics and a quick evaluation of whether the results of the metric seem normal (green tick) or are beyond the corresponding threshold (warning triangle) specified in the *config.py* configuration file. As a general guideline, sensitivity parameters are more relevant than specifity parameters and the latter more than uniformity ones according to their impact in the performance of target enrichment experiments. In addition, a json file named data/summary.json will be created containing these summary data.

5.2. Evaluation in terms of Sensitivity

ngsCAT provides two metrics to assess the quality of the coverage on target regions:

1. The percentage of target bases covered at different coverage thresholds and
2. The saturation curve of the percentage of target bases covered as a function of the number of mapped reads.

5.2.1. Percentage of target bases covered at different coverage thresholds

This metric allows to assess the sensitivity of the capture process by showing the degree of covered target region at a specific coverage (see the following figure).

In Fig 1, Y axis represents the % of target bases covered and X axis repre-

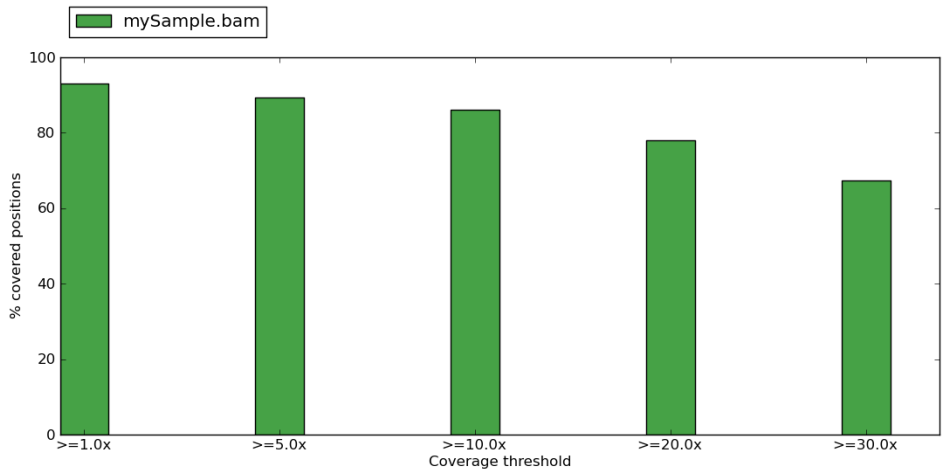

Figure 1: Percentage of target bases covered at different coverage thresholds

sents different coverage thresholds. Each bar represents the percentage of target bases covered at the given coverage threshold. Also, a table is given with the different percentage values of target bases covered. By default, the following coverage thresholds are run by *ngsCAT*: 1x, 5x, 10x, 20x and 30x, but can be customized when calling *ngsCAT* with the *–coveragethrs* argument. For example, *–coveragethrs 1,5,10,20,30,40,50,60* will plot the percentage of target bases covered at 1x, 5x, 10x, 20x, 30x, 40x, 50x and 60x.

What is expected?

A typical target-enrichment NGS experiment results in 90% of target-bases covered at coverage $\geq 1\times$. This value tends to decrease as the coverage threshold increases. How fast this percentage decreases with the coverage increment depends on the specific experimental design/results. A warning is issued if the percentage of bases with coverage $\geq 1\times$ is less than a 90%. This is the default percentage value, but can be changed in *config.py* file through *warnbasescovered* parameter to a custom percentage.

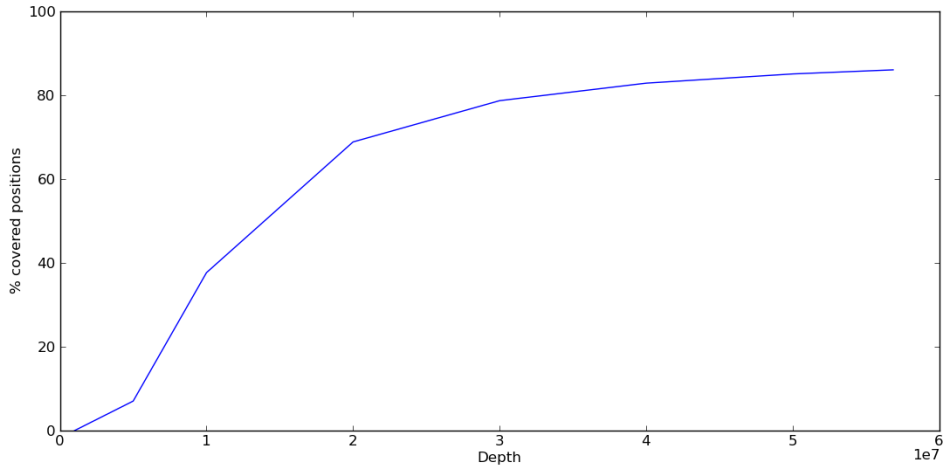

Figure 2: saturation curve

5.2.2. Saturation curve of the percentage of target bases covered as a function of the number of mapped reads

Through this plot, it is estimated if a higher sequence depth will translate into a higher percentage of covered positions. In a simulated procedure, it is measured the percentage of target-bases covered at coverage $\geq 10\times$ as a function of the number of mapped reads (see Fig 2).

By default, the saturation curve is not shown, but can be calculated through the *–saturation y* argument when calling *ngsCAT*. Sequencing depth simulations were carried out by randomly selecting different number of reads from the BAM file. If *–saturation y*, the number of reads to be selected are automatically calculated from the total number of reads in the BAM file, being 5 different sets of reads by default. However, a list of comma-separated values, that are automatically multiplied by 10^6, can be added through the *–depthlist* argument, denoting different set of reads. For each set, the percentage of target-covered positions at coverage $\geq 10\times$ are calculated. This graph aims to give an idea on how much one can improve the percentage of target-bases covered by rese-

quencing. A table is also given with the values plot in the graph.

IMPORTANT

- If *–saturation y* and *–depthlist* arguments are placed when calling ngsCAT and no saturation plot is provided in the report, please, check that the number of reads of your BAM file indicated in the summary section of the report is greater than the list of numbers provided in the *–depthlist* argument multiplied by 10^6. For example, if your BAM file has 1,000,000 reads, suitable values for *–depthlist* argument could be *0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1* since all these values multiplied by 10^6 are less or equal than 1,000,000.
- The calculation of the saturation plot may be computationally intensive, depending on the size of your experiment (BAM file) and the number of set of reads provided out in the *–depthlist* argument.

What is expected?

A curve saturated on the right part indicates that resequencing will not improve the number of target-bases covered at 10x. A warning is issued if the curve does not tend to saturation on the right side (slope $> 1/10^5$ by default). The slope threshold can be customized in the *config.py* file through *warnsaturation* parameter.

5.3. Evaluation in terms of Specificity

A set of graphs and data tables that allow to assess how much of the sequencing effort is wasted in sequencing off-target regions:

- Number and percentage of reads on/off target. **NOTE:** *ngsCAT* considers a read to be ON-target if it overlaps with any ROI, even if it is a one base overlap. Conversely, *ngsCAT* considers a read to be OFF-target if its alignment does not overlap at all with any ROI.
- Duplicated reads on/off target
- Off target regions with high coverage

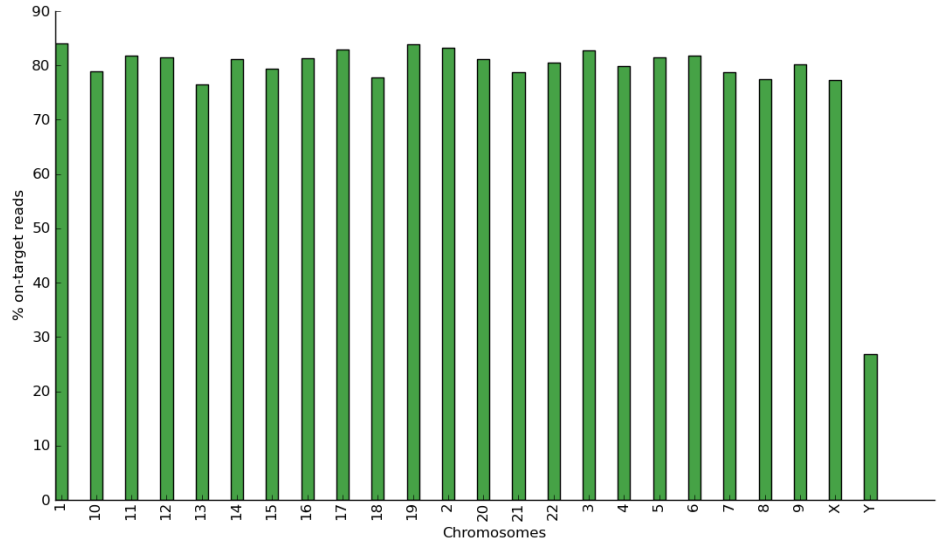

Figure 3: reads on target per chromosome

5.3.1. Number and percentage of reads on/off target

Through this metric, ngsCAT measures the number and percentage of reads on/off target (see Fig 3).

Bars represent the percentage of reads on-target per chromosome. Percentages for each bar were calculated relative to the total number of reads mapped in the corresponding chromosome. An enrichment value is also calculated as: (on-target reads per Kb)/(off-target reads per Kb).

Reads on/off target and % of reads on/off target per chromosome are also provided in a table.

What is expected?

In a typical experiment one may expect an overall 80% of reads mapping on-target. A warning is issued if the % of reads on-target is lower than 80%, which is the default percentage value. However, this percentage can be customized through the *warnontarget* parameter in *config.py* configuration file.

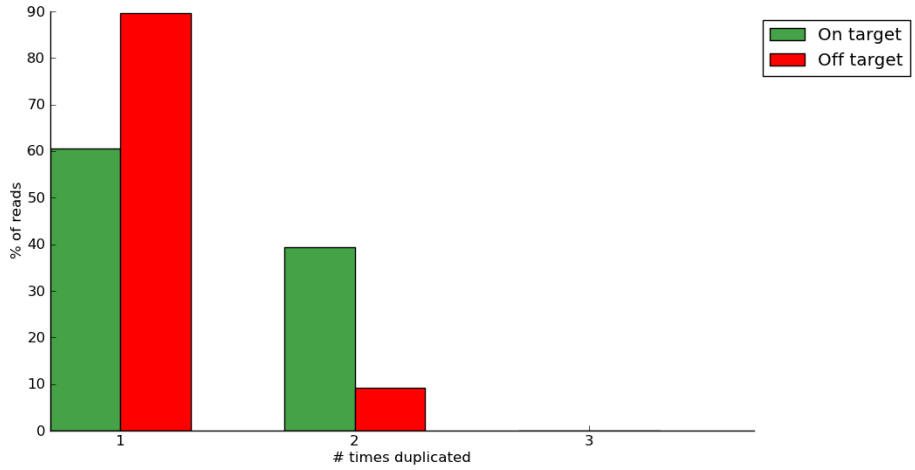

Figure 4: duplicated reads on/off target

5.3.2. Duplicated reads on/off target

Duplicated reads on/off target can indicate whether off-target reads can be due to experimental artifacts, for example bias in the PCR step. This metric measures the percentage of duplicated on/off-target reads, considering duplicated reads those mapping at exactly the same starting and ending position.

In Fig 4, X axis indicates de number of times the reads are duplicated. Green and red bars indicate the percentage of on-and off-target reads with respect to the total number of on-/off-target reads respectively. A table is also given with information of the number/percentage of duplicated reads on-/off-target at different levels (different number of times the reads are duplicated).

What is expected?

One may expect a greater proportion of duplicated reads on-target due to the enrichment process. Duplicated off-target reads should be due to some other experimental artifacts (e.g. PCR). Thus, a warning is issued if the total ratio of duplicated on-target reads is lower than the ratio of duplicated off-target reads.

5.3.3. Off target regions with high coverage

Through this metric, user can easily identify regions of off target reads, which may be indicative of low-specific probes in capture design.

Related to this feature, in *output_directory/data* directory a set of files (.off.bed) in BedGraph Track Format are generated (one file per cromosome/contig). Each file contains the coverage of off-target bases which are located far from ROIs ($>$ 1Kb and which present coverage $>$ 15$\times$ by default). These tracks allow the user to easily locate clusters of off target reads, which may be indicative of low-specific probes in capture design. Thresholds of coverage and minimum distance to the ROIs can be modified at the *config.py* file through *offtargetthreshold* and *offtargetoffset* respectively. Each file can be visualized in genome browsers such as the one of UCSC or IGV.

5.4. Evaluation in terms of uniformity

ngsCAT provides a set of four metrics to assess whether coverage is uniformly distributed among target regions or, in other words, to detect sequencing biases due to specific genomic locations or nucleotide composition:

- Distribution of the coverage per base position
- Distribution of the coverage along target bases per chromosome
- Distribution of the standard deviation of the coverage within Regions of Interest (ROIs)
- Distribution of the coverage as a function of the GC content.
- Coverage correlation per region of interest (ROI) (only when two BAM files are provided in *–bams* argument)

5.4.1. Coverage distribution per base position

Through this metric, it is studied the distribution of the coverage per target base. Two plots are given for this metric. The first one shows the distribution of

coverage per target base for only bases with coverage $\geq 1\times$ (Fig 5).

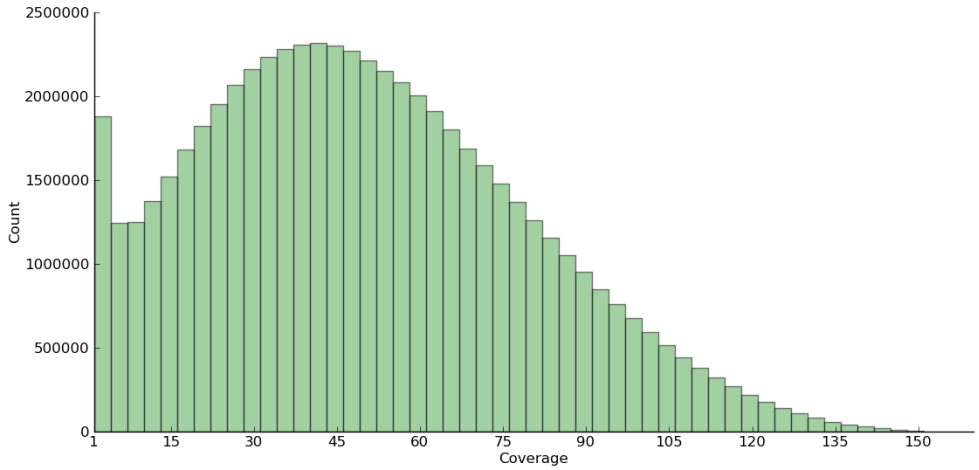

Figure 5: distribution of the coverage per base

The second graph is a representation of the distribution of the coverage per target base in a boxplot, being the star symbol the mean coverage value (Fig 6).

A table is also provided with the following information: number of bases with zero coverage, 1st quartile (Q1), 2nd quartile (Q2, median), 3rd quartile (Q3) and mean, maximum and minimum coverage values.

What is expected?

Low-medium coverage experiments may present a mean coverage of about 40x, although it depends on the experiment carried out. A warning is issued if mean coverage is below 40x, which is the default coverage threshold. This can be customized through *warnmeancoverage* parameter in *config.py* configuration file.

5.4.2. Distribution of coverage along target bases per chromosome

Through this metric, it is plot the coverage found at each target base of the ROIs to identify uncovered regions or unexpected coverage peaks that indicate cap-

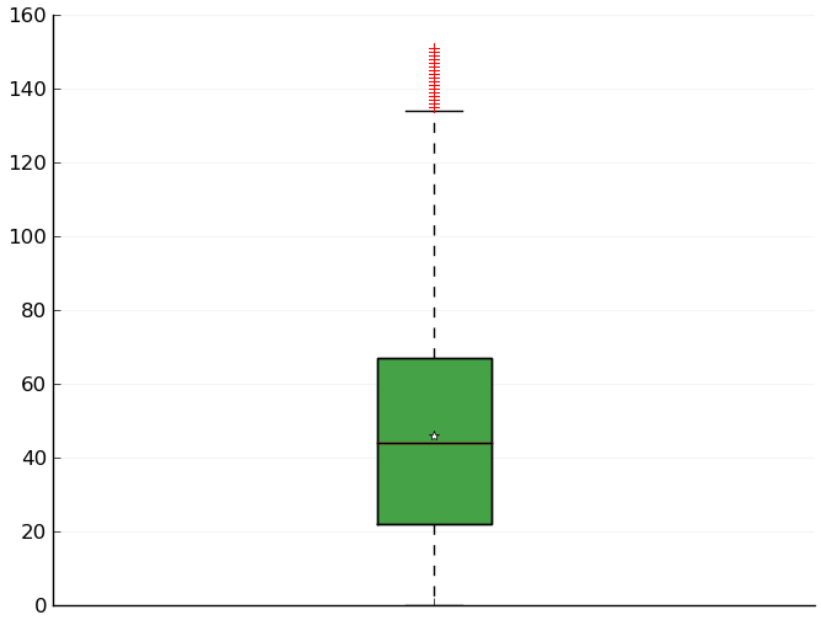

Figure 6: distribution of the coverage per base

ture/sequencing biases. A plot is provided for each chromosome (contig) contained in the target BED file and only ROIs positions for the chromosome are drawn by concatenating their coordinates to facilitate target inspection. For each graph (see an example in Fig 7 which corresponds to chromosome 18), target bases are represented in the X axis and Y axis represents coverage. The .txt file provided in this section lists all those target intervals with zero coverage.

What is expected?

No wide gaps or peaks are expected. A warning is issued if more than 100 consecutive bases in any of the graphs lie below than 6x. These default thresholds can be cutomized through *warncoverageregion* and *warncoveragethrehold* parameters respectively in *config.py* configuration file. In Fig 8, it can be observed a chromosome with several peaks.

Related to this metric, in *output_directory/data* directory, a set of files (.bed)

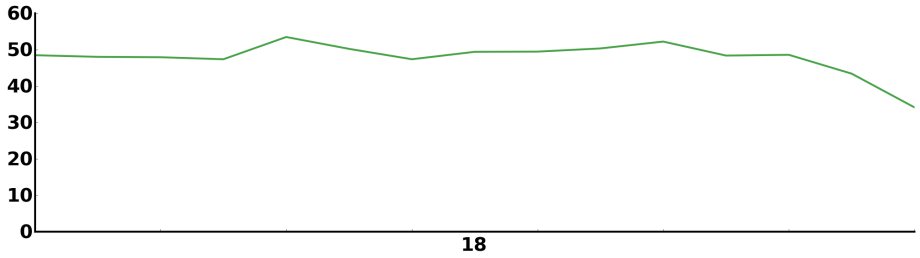

Figure 7: distribution of coverage along target bases

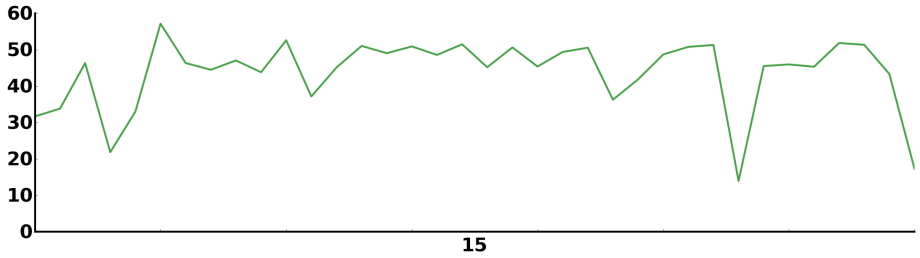

Figure 8: distribution of coverage along target bases

in BedGraph Track Format are generated (one file per chromosome). Each file can be used to visualize per-base coverage along chromosomal positions and can be visualized in genome browsers such as the one of UCSC or or IGV.

Owing to the fact that ngsCAT does not impose any limitation in the number of chromosomes and/or contigs present in the experiment, the more the number of chromosomes and contigs, the more the number of graphs provided through this metric.

5.4.3. Distribution of the standard deviation of the coverage within Regions of Interest (ROIs)

Through this metric, it is studied, the distribution of the standard deviation of the coverage within ROIs (target regions). In other words, for each target region, it is calculated both the mean and the standard deviation of the coverage of its

bases and a normalized standard deviation value (standard deviation / mean value) is obtained for that region. These normalized values for all regions are used to draw an histogram and a boxplot. A example of an histogram is in Fig 9.

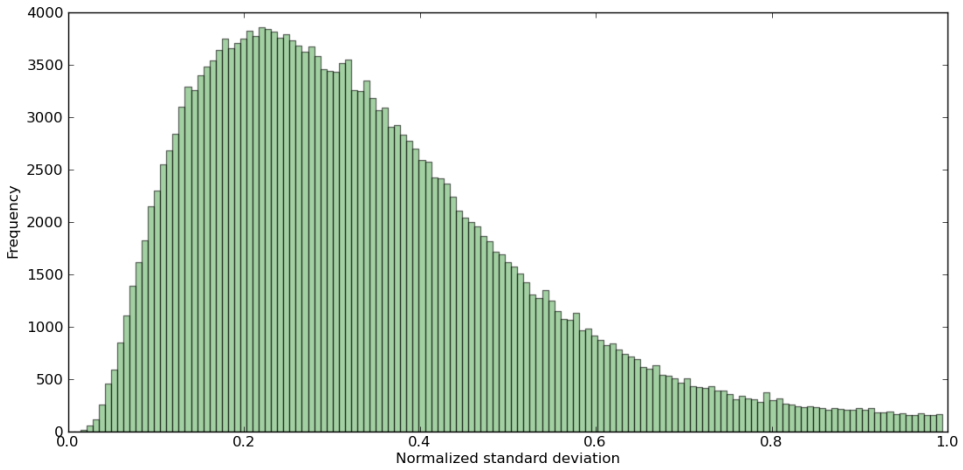

Figure 9: distribution of the standard deviation of the coverage within Regions of Interest (ROIs)

An example of the related boxplot is given Fig 10, showing Y axis in log-scale.

Values related to the boxplot such as the 1st quartile (Q1), the 2nd quartile (Q2 or median), the 3rd quartile (Q3) and the maximum, minimum and mean scores are also given in a table.

What is expected?

In general, given a target region one may observe two typical coverage profiles as shown in Fig 11.

Bases near the 5'/3' ends of target regions tend to be worse covered than bases located in the middle of target regions. The lower the mean of this distribution is, the more uniform the coverage is within target regions. A warning is issued if normalized mean standard value is greater than 0.3. This is the default threshold, but can be customized through *warnstd* parameter in *config.py* configuration file.

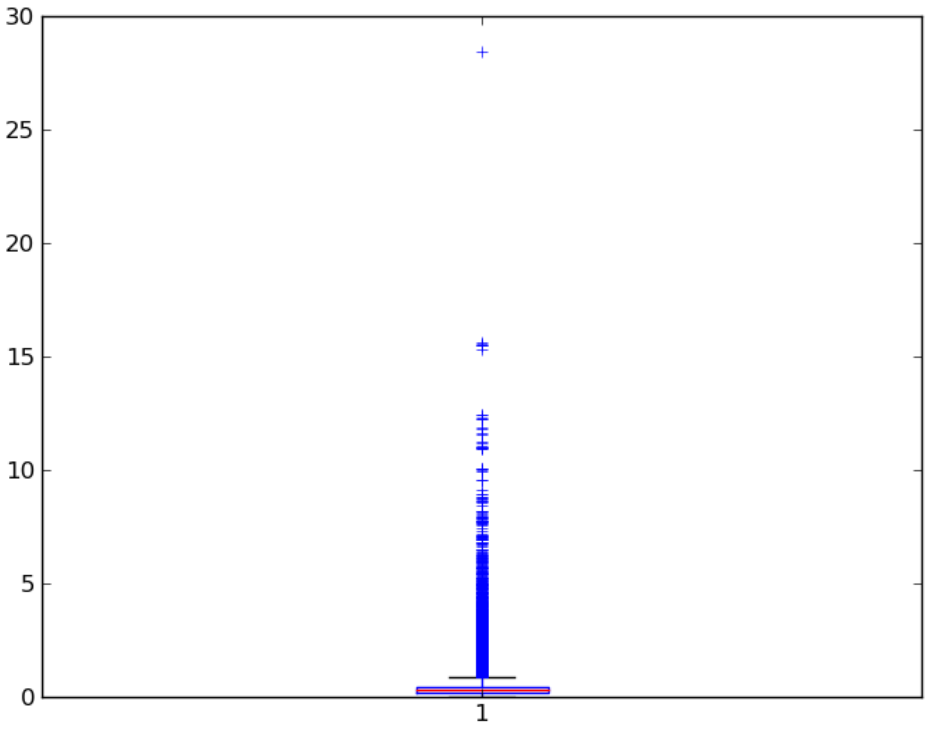

Figure 10: distribution of the standard deviation of the coverage within Regions of Interest (ROIs)

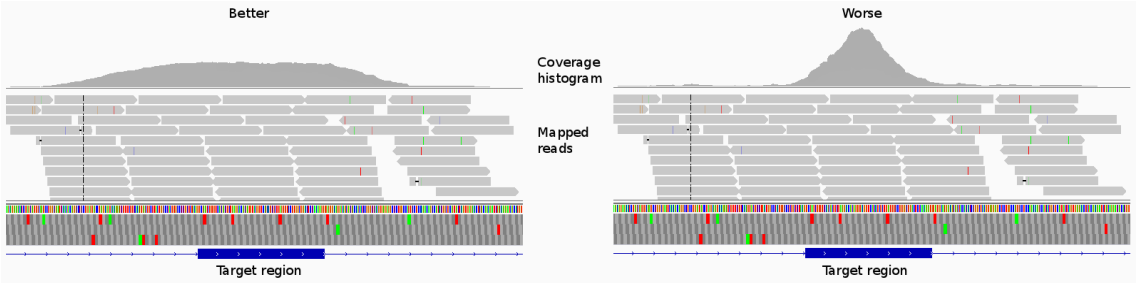

Figure 11: example of better (left) and worse (right) coverage profiles

5.4.4. Distribution of the coverage as a function of the GC content.

Through this metric and for each target region, the mean coverage of its bases is calculated as well as the percentaje of Gs and Cs it contains and plotted one against each other, thus allowing to measure uniformity in terms of nucleotide composition as shown in Fig 12:

This plot allows to observe sequencing biases which depend on the GC content of target regions. For example, it is usual to observe lower coverage in sequencing regions with high GC or high AT content. GC bias in sequencing studies is in large part due to early PCR steps during library generation where high and low GC content causes reduced amplification and therefore lower sequencing coverage.

In order to run this calculation, which is disabled by default, a reference genome in fasta format is required through the argument *–reference referenceGenome-file.fasta.*

What is expected?

It is expected to have most of the target regions with similar mean coverage around a GC content of 50%.

5.4.5. Coverage correlation per region of interest (ROI)

The *coverage correlation per Region Of Interest (ROI)* plot shows the correlation of coverage for all regions. It must be emphasized that this figure is only gen-

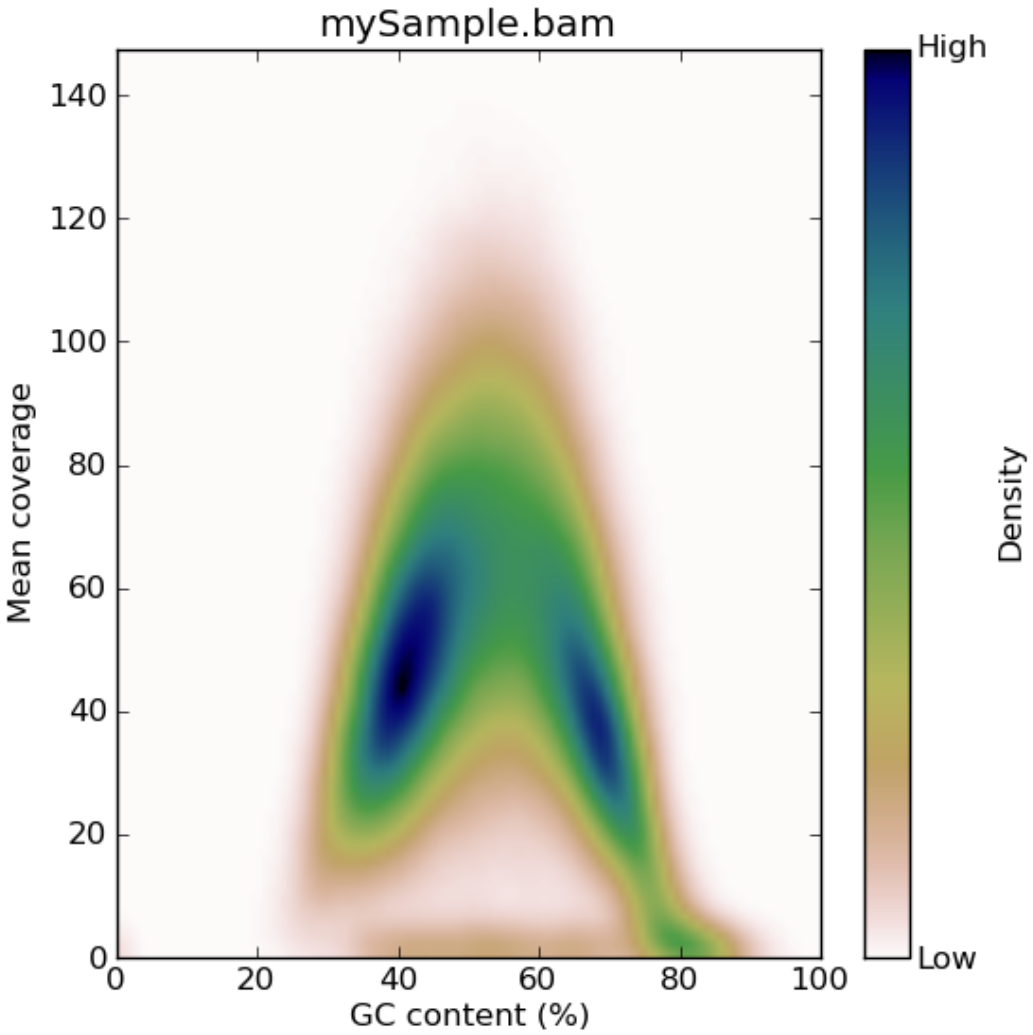

Figure 12: distribution of the coverage as a function of the GC content

erated when two BAM files are provided through the *–bams argument* when calling *ngsCAT*.

Each point in Fig 13 represents a target region (one interval in the bed file). X asis represents mean coverage found in one of the samples and Y axis represents mean coverage found in the other sample.

What is expected?

A strong linear correlation is expected between both samples in similar experiments. A warning is issued if correlation score R is lower than 0.9 by defult. However, this threshold can be changed through *warncoveragecorrelation* in *config.py* configuration file.

6. Case study

In order to show how ngsCAT can assess the performance of the capture process, an in-house dataset targeting about ∼5Mb of the human genome was used in this section. ngsCAT was used taking as input the BAM file and the BED file that describes the targeted regions. Thus, we could check the percentage of target bases covered at different coverage thresholds (See Fig. 14).

Interestingly, we found that 80% of targeted bases have at least at $10\times$ coverage, which is a relatively low percentage, and that the percentage of reads on target is 80%, which is a typical value for capture experiments, suggesting that reads were properly enriched. The $10\times$ coverage saturation curve tends to plateau at 80% of covered positions (see Fig. 15), indicating that no more target bases with a $10\times$ coverage will be obtained by increasing sequencing depth.

When ngsCAT was run with probe coordinates, instead of ROI coordinates, 95% of the bases were covered at $10\times$ (see Fig. 16). In addition, the inspection of the off-target information files showed the presence of a number of off-target regions with a coverage higher than 15. Thus, ngsCAT results allowed us to hypothesize that the low percentage of covered target bases could be due to the capture probe tiling and to an off-target effect.

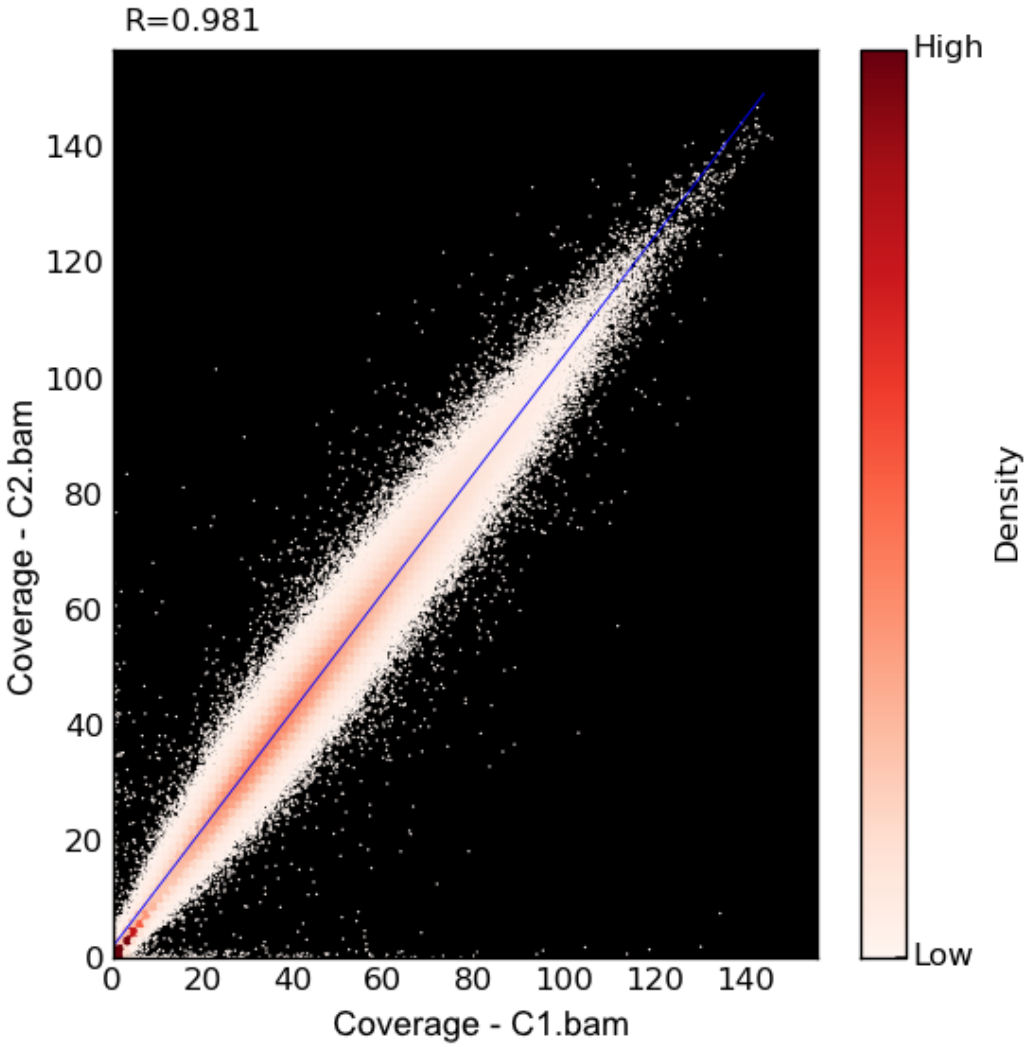

Figure 13: coverage correlation per region of interest (ROI)

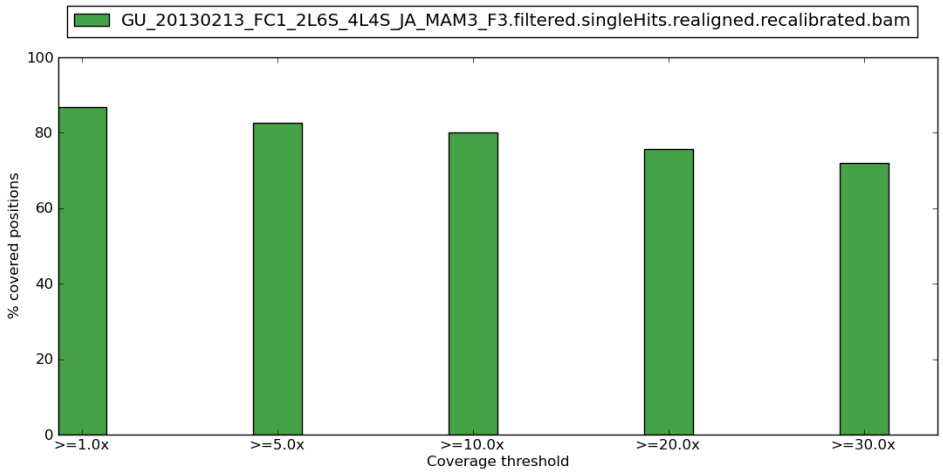

Figure 14: Percentage of target bases covered at different coverage thresholds for the case study

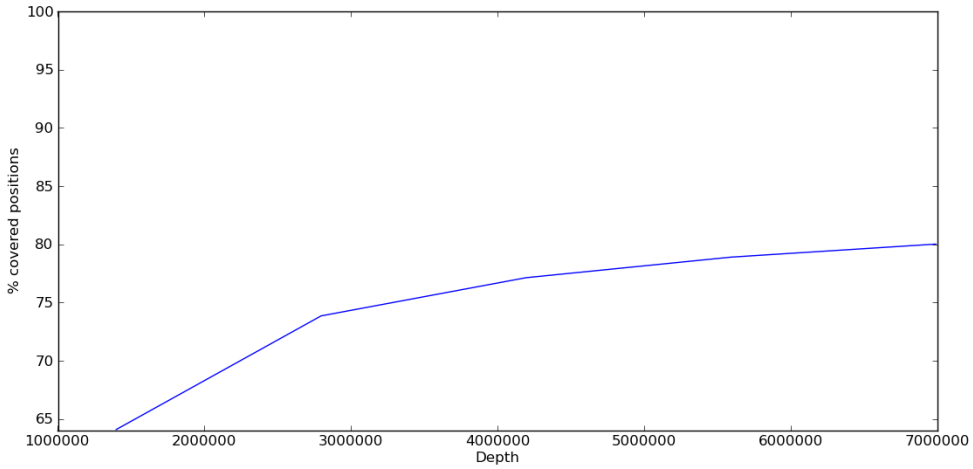

Figure 15: Saturation curve of the percentage of target bases covered as a function of the number of mapped reads. Case study example

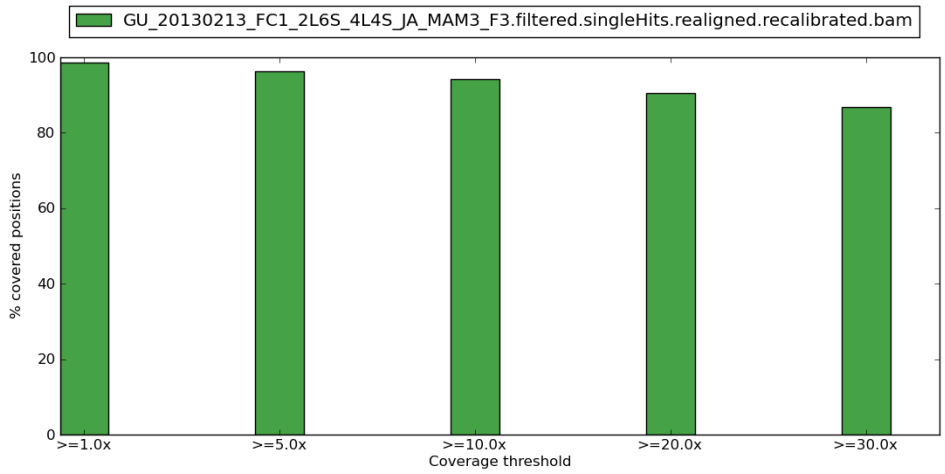

Figure 16: Percentage of target bases covered at different coverage thresholds for the case study when ngsCAT is run with probe coordinates instead of ROI coordinates

7. Running *ngsCAT* with two examples

In this section, it is shown how to run *ngsCAT* with two sintetic examples (BAM files) generated separately with dwgsim (sections 7.1. and 7.2.). Target regions are subsets from the *SeqCap EZ Library* from Niblegen . Furthermore, it will be shown how to run *ngsCAT* with two examples at the same time (section 7.3.).

7.1. Example 1

Once BAM and BED files are downloaded and dependencies have been installed (see section 3.3.), run *ngsCAT* with default parameters from the command line as follows:

```
ngscat.py --bams <path_to_example1>/example1.bam \
--bed <path_to_example1>/seqcap.example1.bed --out <output_directory_example1> \
```

The previous command line will run *ngsCAT* on *example1.bam*, using *SeqCap.example1.bed* as target file with default parameters. The report will be

BAM file	can be downloaded from here
Number of reads:	495,168
Length of reads:	250 bp
Platform:	Illumina
BED file:	can be downloaded from here
Size of target regions:	1,843,316 bases
Reference genome:	can be downloaded from here

Table 1: Results summary for in-house samples

generated in *output_directory_example1* and can be viewed here or downloaded here.

Previous report does not include the saturation curve of the coverage as a function of the number of reads and the distribution of the coverage as a function of GC content. To generate such plots, *–reference* and *–saturation* arguments has to be added:

```
ngscat.py --bams <path_to_example1>/example1.bam \
--bed <path_to_example1>seqcap.example1.bed --out <output_directory_example1> \
--reference <path_to_reference_genome/reference.fasta> --saturation y \
```

The previous command line will run *ngsCAT* on *example1.bam*, using *SeqCap.example1.bed* as target file. The optional arguments used are:

- *–reference path_to_reference_genome/reference.fasta*: will generate the distribution of the coverage as a function of the GC content since, for this plot, a reference genome in fasta format is required.
- *–saturation y*: will generate the saturation curve plot of the percentage of target bases covered as a function of the number of mapped reads. Since this BAM file has 495,168 reads (see summary section in the report) and *–depthlist* argument is not used, *ngsCAT* will generate five set of reads up

to 495,168 to generate this plot.

The report *captureQC.html*, which will be generated in ouput_directory_example1, can bew viewed here or downloaded from here.

7.2. Example 2

BAM file	can be downloaded from here
Number of reads:	237,535
Length of reads:	250 bp
Platform:	Illumina
BED file:	can be downloaded from here
Size of target regions:	1,257,257 bases
Reference genome:	can be downloaded from here

Table 2: Results summary for in-house samples

Once BAM, BED and reference genome are downloaded, run *ngsCAT* from the command line as follows:

```
ngscat.py --bams <path_to_example2>/example2.bam \
--bed <path_to_example2>seqcap.example2.bed --out <output_directory_example2> \
--reference <path_to_reference_genome/reference.fasta> --saturation y \
--depthlist 0.01,0.02,0.025,0.05,0.075,0.1,0.2,0.3,0.4,0.5 --threads 4 \
--tmp <path_to_tmp_dir>
```

The previous command line will run ngsCAT on *example2.bam*, using *SeqCap.example2.bed* as target file and *reference.fasta* as a genome reference. In this execution, the following optional arguments have been used:

- *–reference path_to_reference_genome/reference.fasta*: will generate the distribution of the coverage as a function of the GC content since, for this plot, a reference genome in fasta format is required.

- *–saturation y*: will generate the saturation curve plot of the percentage of target bases covered as a function of the number of mapped reads. In this case, the sets of reads to be selected are provided out in *–depthlist* argument: *–depthlist 0.01,0.02,0.025,0.05,0.075,0.1,0.2,0.3,0.4,0.5* will randomly select $0.01x10^6$,$0.02x10^6$,$0.025x10^6$,$0.05x10^6$,$0.075x10^6$,$0.1x10^6$,$0.2x10^6$,$0.3x10^6$,$0.4x10^6$ reads to plot the saturation curve.
- *–threads 4*: *ngsCAT* will run using 4 threads.
- *–tmp path_to_tmp_dir*: temporal directory where intermediate results are stored.

The report *captureQC.html* which will be generated in ouput_directory_example2, can be viewed here and downloaded from here.

7.3. Running with two examples at the same time

ngsCAT can also take as input information two samples (BAM files) allowing the comparison of two different experiments. The BED file (target regions) used must be the same. For this purpose, it is enough to include two BAM files separated by a comma in *–bams* argument. For example, to compare *example1* and *example2* BAM files using *seqcap.example2.bed* target file, the following command can be run:

```
ngscat.py --bams <path_to_example1>/example1.bam,\
<path_to_example2>/example2.bam --bed <path_to_example2>seqcap.example2.bed \
--out <output_directory_comparison> \
--reference <path_to_reference_genome/reference.fasta> --saturation y \
--depthlist 0.01,0.02,0.025,0.05,0.075,0.1,0.2,0.3,0.4,0.5 --threads 4 \
--tmp <path_to_tmp_dir>
```

The optional arguments are the same as in *example2*. The report generated by *ngsCAT* can be viewed here and also can be downloaded from here.

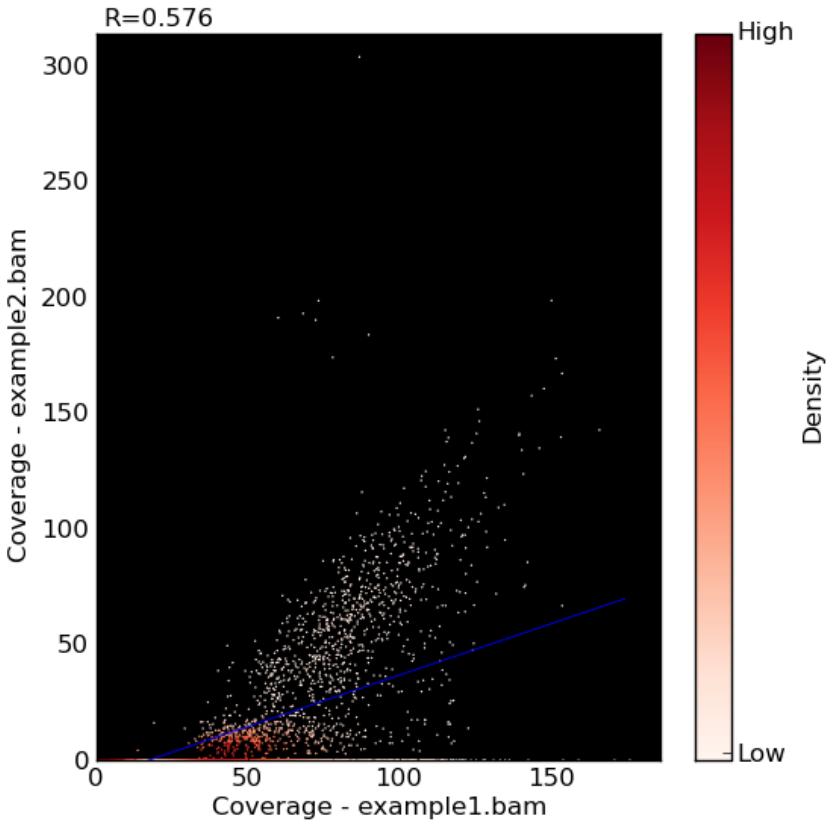

Figure 17: ngsCAT results for a given sample analyzed by two different pipelines

It must be noticed that in the report and for each metric, plots and tables reflect a comparison between the two samples. Since two BAM files are provided when calling *ngsCAT*, the *coverage correlation per Region Of Interest (ROI)* figure is generated as explained in section 5.4.. Fig 17 shows the correlation of coverage for all regions. Each point in the plot represents a target region (one interval in the bed file). X asis represents mean coverage found in one of the samples and Y axis represents mean coverage found in the other sample. Finally, in order to facilitate automatic comparisons among more than two samples, json files are created for each ngsCAT report at data/summary.json that may be easily imported for scripting purposes.

8. Examples of running *ngsCAT* on a whole-exome experiment

In this section, three whole-exome experiments taken from the 1000 genomes project (phase 3) are run with ngsCAT. These examples are available at the downloads section and are detailed in section 8.1. (HG00096), section 8.2. (HG00097) and section 8.3. (HG00099). Details of target regions for whole exome sequencing are provided here.

8.1. HG00096

BAM file	can be downloaded from here (original link)
Number of reads:	95,760,103
Length of reads:	100 bp
Platform:	Illumina
BED file:	can be downloaded from here (original link)
Size of target regions:	46,492,725 bases
Reference genome:	can be downloaded from here

Table 3: Summary for HG00096 sample

Once BAM and BED files are downloaded, run ngsCAT from the command line as follows:

```
ngscat.py --bams <path_to_HG00096>/hg00096.bam \
--bed <path_to_HG00096>exome.targets.g1k.bed \
--out <output_directory_HG00096> \
--reference <path_to_reference_genome>/hg19.wochr.fasta --saturation y \
```

The previous command line will run ngsCAT on HG00096 sample, using exome.targets.g1k.bed as target file with default parameters. The report will be

generated in *output_directory_HG00096* and can be downloaded here.

8.2. HG00097

BAM file	can be downloaded from here (original link)
Number of reads:	57,270,738
Length of reads:	100 bp
Platform:	Illumina
BED file:	can be downloaded from here (original link)
Size of target regions:	46,492,725 bases
Reference genome:	can be downloaded from here

Table 4: Summary for HG00097 sample

Once BAM and BED files are downloaded, run ngsCAT from the command line as follows:

```
ngscat.py --bams <path_to_HG00097>/hg00097.bam \
--bed <path_to_HG00097>exome.targets.g1k.bed \
--out <output_directory_HG00097> \
--reference <path_to_reference_genome>/hg19.wochr.fasta --saturation y \
```

The previous command line will run ngsCAT on HG00097 sample, using exome.targets.g1k.bed as target file with default parameters. The report will be generated in *output_directory_HG00097* and can be downloaded here.

8.3. HG00099

Once BAM and BED files are downloaded, run ngsCAT from the command line as follows:

BAM file	can be downloaded from here (original link)
Number of reads:	79,760,961
Length of reads:	100 bp
Platform:	Illumina
BED file:	can be downloaded from here (original link)
Size of target regions:	46,492,725 bases
Reference genome:	can be downloaded from here

Table 5: Summary for HG00099 sample

```
ngscat.py --bams <path_to_HG00099>/hg00099.bam \
--bed <path_to_HG00099>exome.targets.g1k.bed \
--out <output_directory_HG00099> \
--reference <path_to_reference_genome>/hg19.wochr.fasta --saturation y \
```

The previous command line will run ngsCAT on HG00099 sample, using exome.targets.g1k.bed as target file with default parameters. The report will be generated in *output_directory_HG00099* and can be downloaded here.