

README file for CONSERTING software distribution

(I) Software download options

1. Run CONSERTING on Amazon Web Services (AWS) without local installation.
 - a. Please refer to the AWS version instruction (<http://www.stjude.com/research/docs/consernting/consernting-ami-steps.pdf>).
 - b. You may stop here.
2. Download the CONSERTING package for local installation.
 - a. URL: <http://www.stjude.com/research/lab/zhang>

(II) Initial setup

1. Note: CONSERTING should be able to work with an alternative SV detection algorithm as long as the SV algorithm supports localized SV detection in a predefined genomic region. However, this would require modifying SV detection pipeline code and we are willing to assist users to make the necessary changes on a case by case basis.
2. Included module version information

The CONSERTING pipeline makes use of the following programs. The version numbers as of the publish date of this guide are listed in the table below:

Program	Version used
GNU Parallel	20140922
wigToBigWig	4
Bambino	2.0
Samtools	0.1.10
CREST	2.0
CAP3	10/15/07
BLAT tool	35x1
gfClient	35x1
gfServer	35x1
twoBitToFa	N/A
bigwigToWig	N/A
JAVA	1.7.0_65
Perl	5.16.3

R	3.1.1
Bash	4.2
Bio-SamTools	1.39
Picard	1.65

3. Install R (64-bit) software (<http://www.r-project.org/>) and the tree package.

4. Download the following script to the code directory:
http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bigWigToWig
http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/wigToBigWig

5. Construct the Mapability track (Once for each window size and assembly):
 - a. Download the Alignability track from UCSC Genome Browser. Choose the appropriate track according to your read length. For example, the wgEncodeCrgMapabilityAlign100mer.bw file is used for the example because the read length is 100 bp in this dataset.
 - i. Hg19: <http://genome.ucsc.edu/cgi-bin/hgFileUi?db=hg19&g=wgEncodeMapability>
 - ii. Hg18: <http://hgdownload.cse.ucsc.edu/goldenPath/hg18/encodeDCC/wgEncodeMapability/>
 - b. Extract the downloaded track using the bigWigToWig script downloaded at Step 4.
`code/bigWigToWig DOWNLOADED_MAP_TRACK MAP_WIGGLE_FILE`
Parameters:
`DOWNLOADED_MAP_TRACK`: The downloaded BigWig file from Step II-5a.
`MAP_WIGGLE_FILE`: Name of the extracted wiggle file

Example:
`code/bigWigToWig`
`Mapability/hg18/wgEncodeCrgMapabilityAlign100mer.bw`
`Mapability/hg18/full.wig`
 - c. Create the chromosome length file for the assembly (For hg18 and hg19 assembly, the file is included for convenience: Mapability/hg18/chromosome_lengths.txt and Mapability/hg19/chromosome_lengths.txt)

- d. Run the java code `ExtractMappability` to construct the mapability track for `CONSERTING`.

```
java -Xmx3g -cp code/ ExtractMappability DEST_DIR  
MAP_WIGGLE_FILE CHR_LENGTH_FILE SIZE
```

Parameters:

DEST_DIR: Destination mapability directory (make sure it exists)

MAP_WIGGLE_FILE: Name of the extracted wiggle file from Step II-5b

CHR_LENGTH_FILE: The file containing the chromosomal length information (from Step II-5c)

SIZE: window size

Example:

```
java -Xmx3g -cp code/ ExtractMappability Mapability/hg18  
Mapability/hg18/full.wig  
Mapability/hg18/chromosome_lengths.txt 100
```

6. Construct the GC content track (Once for each window size and assembly).

```
java -cp code/ FA2GC SIZE FA_DIR DEST_DIR
```

Parameters:

SIZE: window size

FA_DIR: FA_DIR: Directory for the FA files of the assembly. Expected files in the directory are 1.fa, 2.fa, ..., X.fa and Y.fa.

DEST_DIR: Destination GC directory (make sure it exists)

Example:

```
java -cp code/ FA2GC 100 HG18-FA-Location GC/hg18
```

7. Install the CREST software (<http://www.stjuderesearch.org/site/lab/zhang>)

Note: *CONSERTING should be able to work with an alternative SV detection algorithm as long as the SV algorithm supports localized SV detection in a predefined genomic region. However, this would require modifying SV detection pipeline code and we are willing to assist users to make the necessary changes on a case by case basis. Please contact us if you need support,*

8. Download Bambino package[1] to the code directory

(<http://ftp.stjude.org/pub/software/conserting/bambino-1.0.jar>)

9. Download sam-1.65.jar from PICARD to the code directory

(<http://ftp.stjude.org/pub/software/conserting/sam-1.65.jar>)

10. Download mysql-connector-java-5.1.10-bin.jar to the code directory

(<http://dev.mysql.com/downloads/connector/j>)

11. Install GNU Parallel utility.

12. Include the code directory in the PATH variable. Include the code directory and the jar files in the CLASSPATH variable (or specify the correct path in the command line)

Example:

```
CONSERTING_ROOT_DIR=`pwd`  
export PATH=$PATH:$CONSERTING_ROOT_DIR/code  
export  
CLASSPATH=$CLASSPATH:$CONSERTING_ROOT_DIR/code:$CONSERTING_ROOT_D  
IR/code/av.jar:$CONSERTING_ROOT_DIR/code/sam-1.27-  
SIGNED.jar:$CONSERTING_ROOT_DIR/code/mysql-connector-java-5.1.10-  
bin.jar
```

(III) For each sample pair:

1. Input overview:

Running CONSERTING requires the input of two BAM files: Tumor BAM, and matching Normal BAM. The table below briefly documents the input files:

File Name	Description
SAMPLE_D_TUMOR.bam	The “diagnosis” or tumor sample BAM file.
SAMPLE_G_NORMAL.bam	The “germline” or normal sample BAM file.

2. Run the coverage analysis for each BAM

- a. BAM2WIG: this substep calls a class (Ace2.SAMFinneyCoverage) from Bambino package[1].

Note: 1) this step requires 9 Gb of memory to ensure data integrity.

2) if the BAMs are in hg19, a warning message (the reference genome does not appear to be hg18) will be given by the Ace2.SAMFinneyCoverage procedure.

Please ignore this warning as the assembly information is NOT used in this step.

```
sh code/bam2wig.sh INPUT_BAM WIG_FILE_NAME DEST_DIR
```

Parameters:

INPUT_BAM: Input BAM file

WIG_FILE_NAME: Output filename (a gzipped file)

DEST_DIR: Destination directory

Example:

```
sh code/bam2wig.sh test_data/BAM/SJSIM001_D-CREST-2011-03-nq.chr22.bam SJSIM001_D.wig.gz test_data/Coverage/  
sh code/bam2wig.sh test_data/BAM/SJSIM918_G-WUHG18-nomut-nq.chr22.bam SJSIM001_G.wig.gz test_data/Coverage/
```

- b. WIG2BW: this substep uses UCSC "WigToBigWig" utility to convert the wiggle format coverage to bigWiggle.

Note: 1) This step requires 35 Gb of memory for human genome. The memory required could be different for other genomes. 2) Files hg18.size and hg19.size are provided in the distribution package for convenience,

```
wigToBigWig INPUT_WIG_GZ HG_SIZE BW_FILE
```

Parameters:

INPUT_WIG_GZ: The output file from Step III-2a

HG_SIZE: hg18.sizes or hg19.sizes (provided in the package)

BW_FILE: The output file name

Example:

```
wigToBigWig test_data/Coverage/SJSIM001_D.wig.gz hg18.sizes  
test_data/Coverage/SJSIM001_D.bw  
wigToBigWig test_data/Coverage/SJSIM001_G.wig.gz hg18.sizes  
test_data/Coverage/SJSIM001_G.bw
```

3. Run the process_coverage.sh script to create read depth input file.

```
sh code/process_coverage.sh BW SAMPLE DEST_DIR TAG SIZE
```

Parameters:

BW: the BigWig file from Step II-1.

SAMPLE: Sample name

DEST_DIR: Destination directory

TAG: Identification tag. Default tags are D for tumor and G for normal. Alternative tags are accepted for tumor sample.

SIZE: window size

Example:

```
sh code/process_coverage.sh test_data/Coverage/SJSIM001_D.bw  
SJSIM001 SJSIM001 D 100  
sh code/process_coverage.sh test_data/Coverage/SJSIM001_G.bw  
SJSIM001 SJSIM001 G 100
```

After these two steps, input files SJSIM001_D_chr*_100 and SJSIM001_G_chr*_100 files are generated in the SJSIM001 folder.

4. (RECOMMENDED but OPTIONAL, paired analysis only) Generate the LOH input file:

Note: This step is skipped for the test dataset included since there are no SNVs in this dataset.

- a. Run putative SNV detection in the paired sample by the variation detection module of Bambino [1]:

```
code/snv_conserring.sh NIB_DIR D_BAM G_BAM OUTPUT
```

Parameters:

NIB_DIR: NIB directory or indexed fasta file (i.e. samtools faidx) for the reference genome

D_BAM: Tumor BAM

G_BAM: Germline BAM

OUTPUT: Output file

Example:

```
code/snv_conserring.sh PATH/NIB BAMS/SAMPLE_D.bam  
BAMS/SAMPLE_G.bam SAMPLE_SNV.out
```

In this example, the code will read the reference genome information from **PATH/NIB** and the tumor BAM from **BAMS/SAMPLE_D.bam** and germline BAM from **BAMS/SAMPLE_G.bam**. The result file is saved as **SAMPLE_SNV.out**.

- b. Create the LOH input file

```
java -cp code/ AllelicImbalance -d DEST_DIR SAMPLE  
SNV_OUTPUT
```

Parameters:

DEST_DIR: Destination directory, same as the DEST_DIR folder used in Step III-3

SAMPLE: Sample name.

SNV_OUTPUT: The SNV detection output from Step III-4a

Example:

```
java -cp code/ AllelicImbalance -d SAMPLE SAMPLE  
SAMPLE_SNV.out
```

The code will create a **SAMPLE.ai** file in the **SAMPLE** folder.

5. (STRONGLY RECOMMENDED but OPTIONAL) WGS CREST analysis.

Please refer to the CREST README file for Steps II-4a – II-4c. This step could run in parallel with Step II-3.

Example:

- a. Extract soft-clipped reads:

```
perl extractSCLip.pl --ref_genome hg18.fa -i SJSIM001_D-
CREST-2011-03-nq.chr22.bam -p SJSIM001_D
perl extractSCLip.pl --ref_genome hg18.fa -i SJSIM918_G-
WUHG18-nomut-nq.chr22.bam -p SJSIM001_G
```

Two files are generated from this step:

SJSIM001_D.cover

SJSIM001_G.cover

- b. Identify somatic soft-clip position (Skipped for Single-BAM analysis):

```
perl countDiff.pl -d SJSIM001_D.cover -g SJSIM001_G.cover
```

SJSIM001_D.cover.somatic.cover is generated. Please copy this file to the directory with RD input files from Step III-3a.

- c. SV detection:

```
perl CREST.pl --ref_genome hg18.fa --rmdup --cap3 cap3 --
blatclient gfClient --blat blat --blatport 50000 -t
hg18.2bit -f SJSIM001_D.cover.somatic.cover -d SJSIM001_D-
CREST-2011-03-nq.chr22.bam -g SJSIM918_G-WUHG18-nomut-
nq.chr22.bam -max_rep_cover 2000 --max_germline_percent 0 -
min_hit_len 15 -m 2 --min_one_side_reads 3 -p SJSIM001_D
```

A file SJSIM001_D.predSV.txt is generated.

- d. Reformat the CREST result for CONsertING usage and save the final file in the directory with RD input files. Briefly, the optional header in CREST output is removed and the first 8 columns of the CREST result are retained, which contain breakpoint information used by CONsertING.

```
grep -v posA SJSIM001/SJSIM001_D.predSV.txt | cut -f1-8 >
SJSIM001/SJSIM001.SV.lst
```

6. Run CONsertING :

Note: 1) Please start from the code directory; 2) Please use the absolute path or relative path to the WD; 3) This step requires approximately 24G memory for a paired human sample analysis using a window size of 100 bp; 4) Please strictly follow the R syntax.

```
cd code
```

```
R CMD BATCH --no-save --no-restore '--args wd=WD sample=SAMPLE
gc.prefix=GC_DIR map.prefix=MAP_DIR crest.cmd=CREST_CMD
CREST.parallel=CREST_PARALLEL code.dir=CODE_DIR norm.chr=NORM_CHR
DCODE=D_CODE GCODE=G_CODE ai.file=AI_FILE hg18=F' CONsertING.R
R_logfile
```

Required Parameters:

- 1) **WD**: working directory, where the read depth input, LOH input and SV input files are stored. Please use quotation marks.
- 2) **SAMPLE**: sample name (such as "SJSIM001", please use quotation marks).
- 3) **GC_DIR**: directory for GC content files (from Step II-6). Please use quotation marks and include the trailing "/".
- 4) **MAP_DIR**: folder for mapability track files (from Step II-5). Please use quotation marks and include the trailing "/".

Selected optional parameters:

- 1) **CREST_CMD**: part of CREST command line with sensitive parameters (starting from beginning, up to the BAM setting with recommended parameters: `-min_percent_hq 40 -m 1 -min_one_side_reads 2`). Please use quotation marks. If omitted, CONSERING ignores any CREST input and runs a single iteration of segmentation only.
- 2) **CREST_PARALLEL**: TRUE (if GNU Parallel utility is installed AND local CREST run in parallel desired)/FALSE (otherwise). Default: FALSE
- 3) **CODE_DIR**: directory with downloaded scripts. Could be ignored if the code directory is the current directory or included in the CLASSPATH variable
- 4) **NORM_CHR**: a set of pre-determined reference chromosomes (not used in the sample) in R format (example: `c(1,3:22)`). If omitted and the LOH input file is provided, CONSERING will infer the reference. If both are omitted, CONSERING takes all autosomes as reference. Please refer to [Guideline_Manual_Ref_Selection.pdf](#) for useful information on manual selection of reference chromosomes based on the coverage summary information.
- 5) **D_CODE**: Tag for the tumor sample (from Step III-3). Please use quotation marks. Default: "D" for tumor and "G" for normal sample.
- 6) **G_CODE**: Tag for the germline sample (from Step III-3). Please use quotation marks. Default: "G"
- 7) **AI_FILE**: the LOH input file from Step III-4 (Please use quotation marks). If no reference selection is given, CONSERING will use the information from this file to infer the reference. This information is also used to refine the final result.
- 8) **Hg18**: FALSE (BAMs mapped in hg19). Default: TRUE

Example:

```
cd code
R CMD BATCH --no-save --no-restore '--args wd="../SJSIM001"
sample="SJSIM001" gc.prefix="../GC/hg18/"
map.prefix="../Mapability/hg18/" crest.cmd="perl CREST.pl --
ref_genome hg18.fa --rmdup --cap3 cap3 --blatclient gfClient --
blat blat --blatport 50000 -t hg18.2bit -f
SJSIM001_D.cover.somatic.cover -max_rep_cover 2000 -min_hit_len
```

```
15 -min_percent_hq 40 -m 1 -min_one_side_reads 2 -d SJSIM001_D-
CREST-2011-03-nq.chr22.bam -g SJSIM918_G-WUHG18-nomut-
nq.chr22.bam" code.dir=" ../code"' CONsertING-general.R
SJSIM001.out
```

7. CONsertING Output:

Output file is stored in the **WD** /Result folder.

The CONsertING analysis run can generate several output files depending on the data and the parameters used. These files are documented in the table below:

Note: Please note that the bolded part of the file name is the notable part for the description of that file.

File Name	Description
SAMPLE_D_G_CONsertING_NoCREST_Mapability_100.txt	No crest run
SAMPLE_D_G_CONsertING_NoCREST_Mapability_100_ potential_conflict_segment .txt	Potential conflict in reference selection
SAMPLE_D_G_CONsertING_Mapability_100.txt. QualityMerge	Final CONsertING segmentation result with CREST
SAMPLE_D_G_CRESt-map_final_report.txt. QualityMerge	Final CONsertING unbalanced SV result with coverage support. Columns are described in the next page.
SAMPLE_D_G_CONsertING_Mapability_100_ potential_library_issue .txt	Set up as CONsertING run with CREST. However, potential library artifacts detected and CREST run aborted. To force CREST run, enable “forced.CREST” parameter.
SAMPLE_D_G_CONsertING_Mapability_ 100 .png	This file contains the CONsertING Image generated (the number 100 indicates the window size). Please see the Guide to reading this image in the next page.

Column Descriptions in CONsertING output.

For Files: SAMPLE_D_G_CONsertING_NoCREST_Mapability_100.txt
SAMPLE_D_G_CONsertING_Mapability_100_**potential_library_issue**.txt, or
SAMPLE_D_G_CONsertING_Mapability_100.txt.QualityMerge

Column	Explanation
chrom	Chromosome (X=23 and Y=24)
loc.start	Starting coordinates of the segments
loc.end	Ending coordinates of the segments
num.mark	Number of windows in the segment (sequencing gaps and windows with low mapability score were excluded)
length.ratio	The ratio between the length of the used windows to the genomic length
seg.mean	The estimated GC corrected difference signal (2 copy gain will have a seg.mean of 1)
GMean	The mean coverage in the germline sample (a value of 1 represents diploid)
DMean	The mean coverage in the tumor sample
LogRatio	The log2ratio of the coverage signal between tumor and germline samples
QualityScore (Used in CONSERGING with CREST only)	A empirical score used in merging
SV_Matched (Used in CONSERGING with CREST only)	The SV support for the segment. A value of 0 shows no support on either end (the segment was called purely based on coverage changes); a value of 1 indicates support at the loc.start end (either starting of the chromosome or a supporting SV breakpoint); a value of 2 indicates support at the loc.end end and a value of 3 indicates supports at both ends.

For file: SAMPLE_D_G_CREST-map_final_report.txt.**QualityMerge** (contains 8 columns indicated below):

Column	Explanation
chrA	These 4 columns define the chromosome, position orientation and number of softclip reads support for the left side breakpoint. Chromosome (X=23 and Y=24)
posA	
oriA	
scA	
chrB	Same as above but these represent the data for right side breakpoint.
posB	
oriB	
scB	

For file: SAMPLE_D_G_CONSERTING_NoCREST_Mapability_100_potential_conflict_segment.txt
(contains 3 columns indicated below):

Column	Explanation
chr	These 3 columns define the chromosome, start position and end position of the segments with potential LOH and CNA conflict.
pos.start	
pos.end	

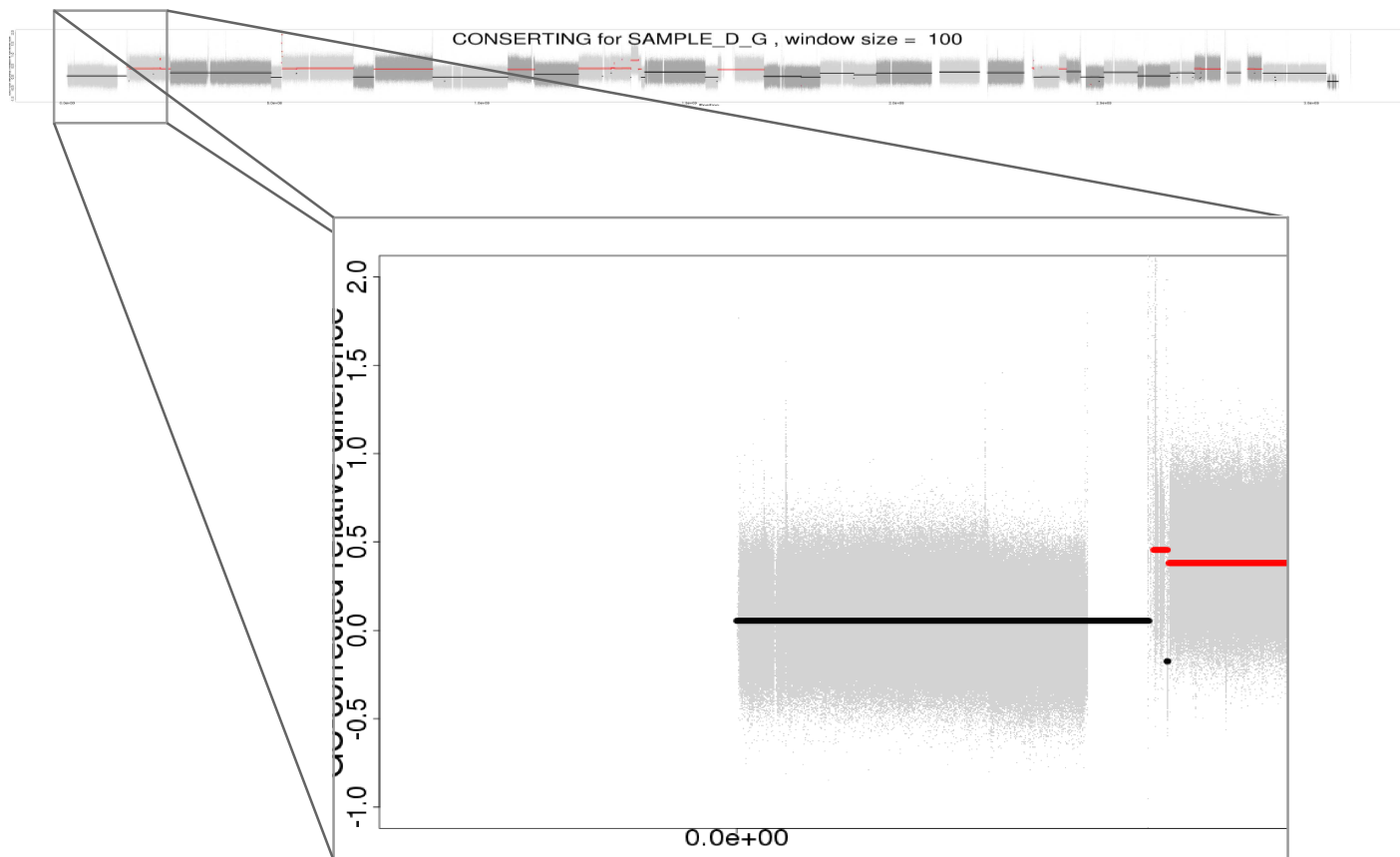
(IV) Guide to Reading CONSERTING Image



- 1 Karyotype View.** The top section of the output image consists of a zoomed out karyotype level view of the CONSERTING CNA analysis.
- 2 Chromosome Level Views.** For each chromosome, the output image will contain a graph obtained from the CONSERTING analysis. This graph is essentially a zoomed in version from the karyotype view with more details visible.
- 3 Centromere & Gap.** In the attached image, you can see the centromere and the gap as a dark blue line with no signal (In the attached example image, see Chromosome 1 window).
- 4 LOH View.** The cyan and purple graph represents the Loss of Heterozygosity. It is located in each of the chromosome windows and occupies the top fourth of the view (between 1.0 and 1.5 of the Y axis). The cyan portion represents intermedia level LOH, and purple represents Strong LOH.
- 5 Red Line.** In the gray graph, the red line represents the presence of the segment which has an absolute difference of greater than 0.25 ($\text{abs}(\text{diff}) > 0.25$). In other cases a black line is displayed if the absolute difference is less than 0.25 for that segment.
- 6 Green Line.** The green line represents the log ratio of the signal divided by 5. A bright green line represents copy number gain, whereas copy number loss is represented by a dark green line. An example of this can be seen in the lower right corner of the attached image (at Chromosome X and Y).

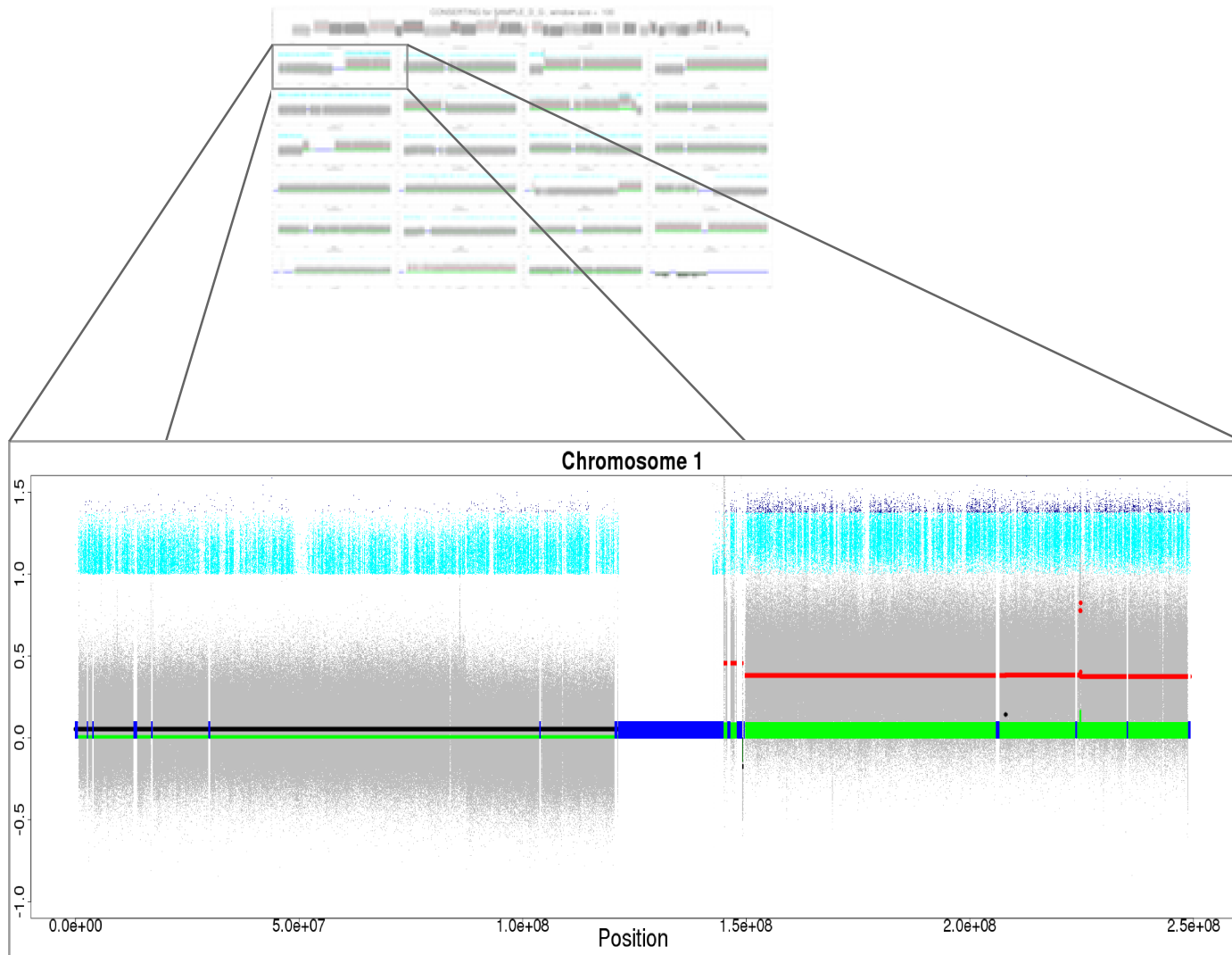
Karyotype View

- 1 Karyotype View.** The top section of the output image consists of a zoomed out karyotype level view of the CONSERING CNA analysis.



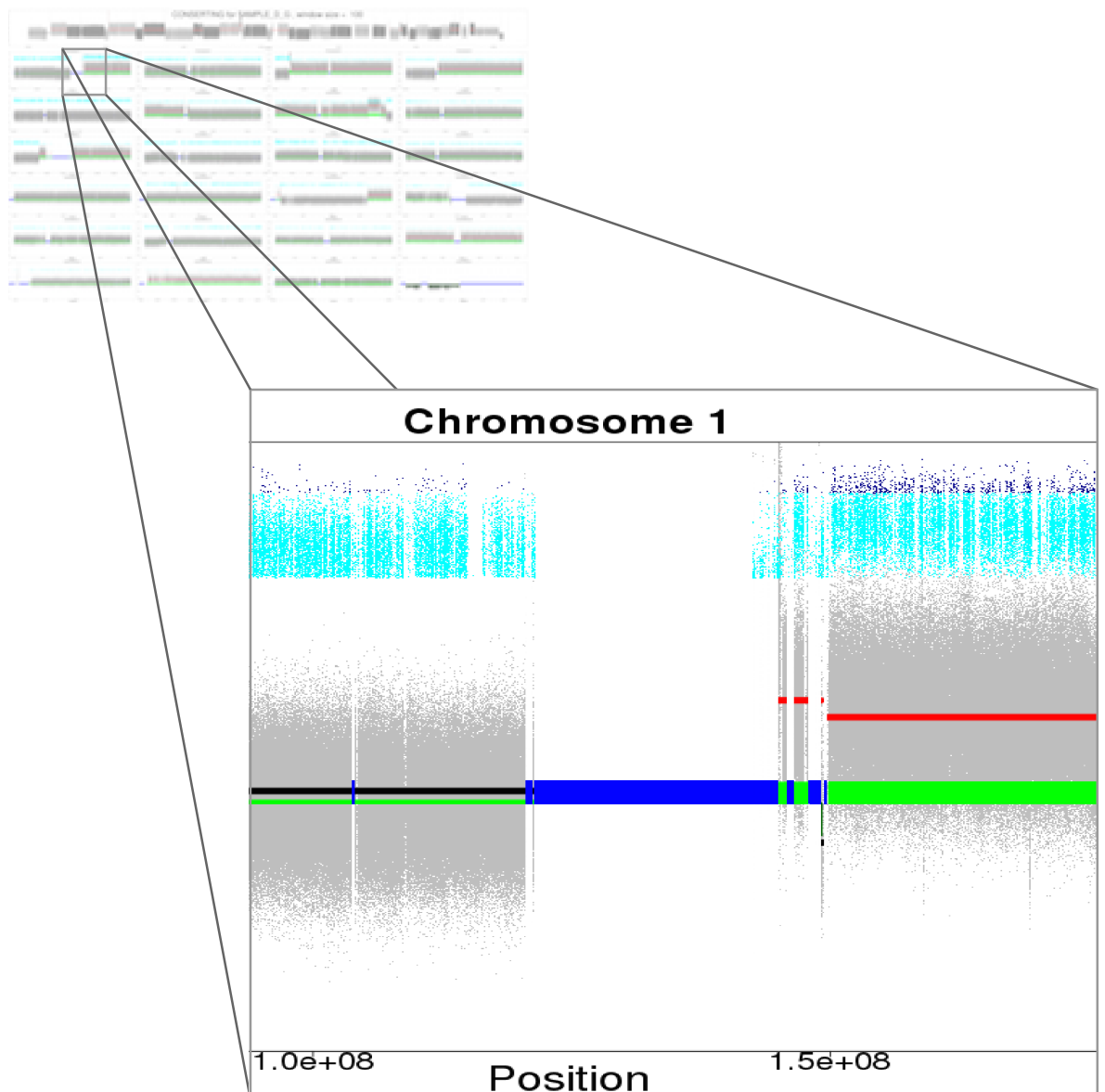
Chromosome Level Views

- 2 Chromosome Level Views.** For each chromosome, the output image will contain a graph obtained from the CONSERGING analysis. This graph is essentially a zoomed in version from the karyotype view with more details visible.



Centromere & Gap

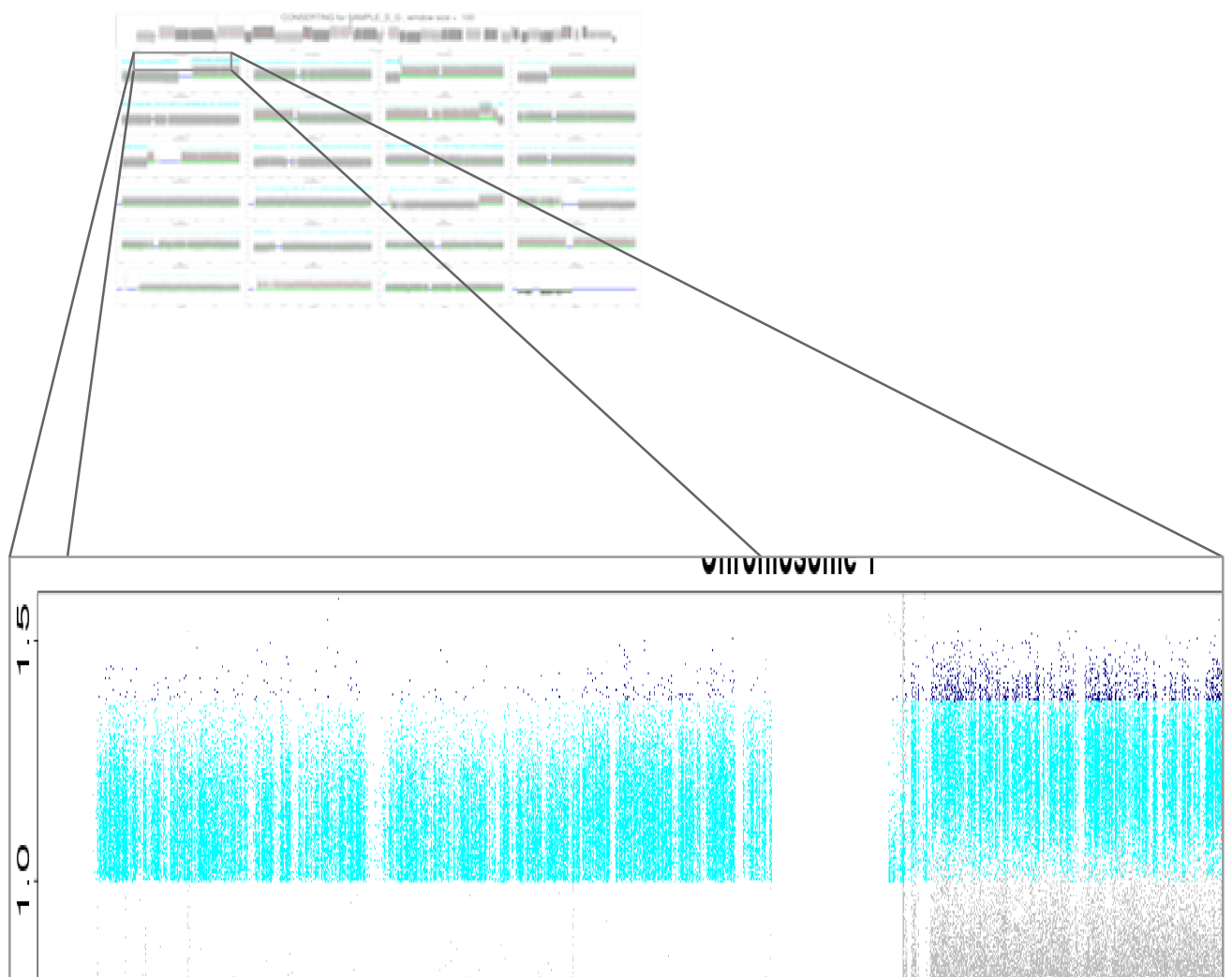
- 3 **Centromere & Gap.** In the attached image, you can see the centromere and the gap as a dark blue line with no signal (In the attached example image, see Chromosome 1 window).



LOH View

4

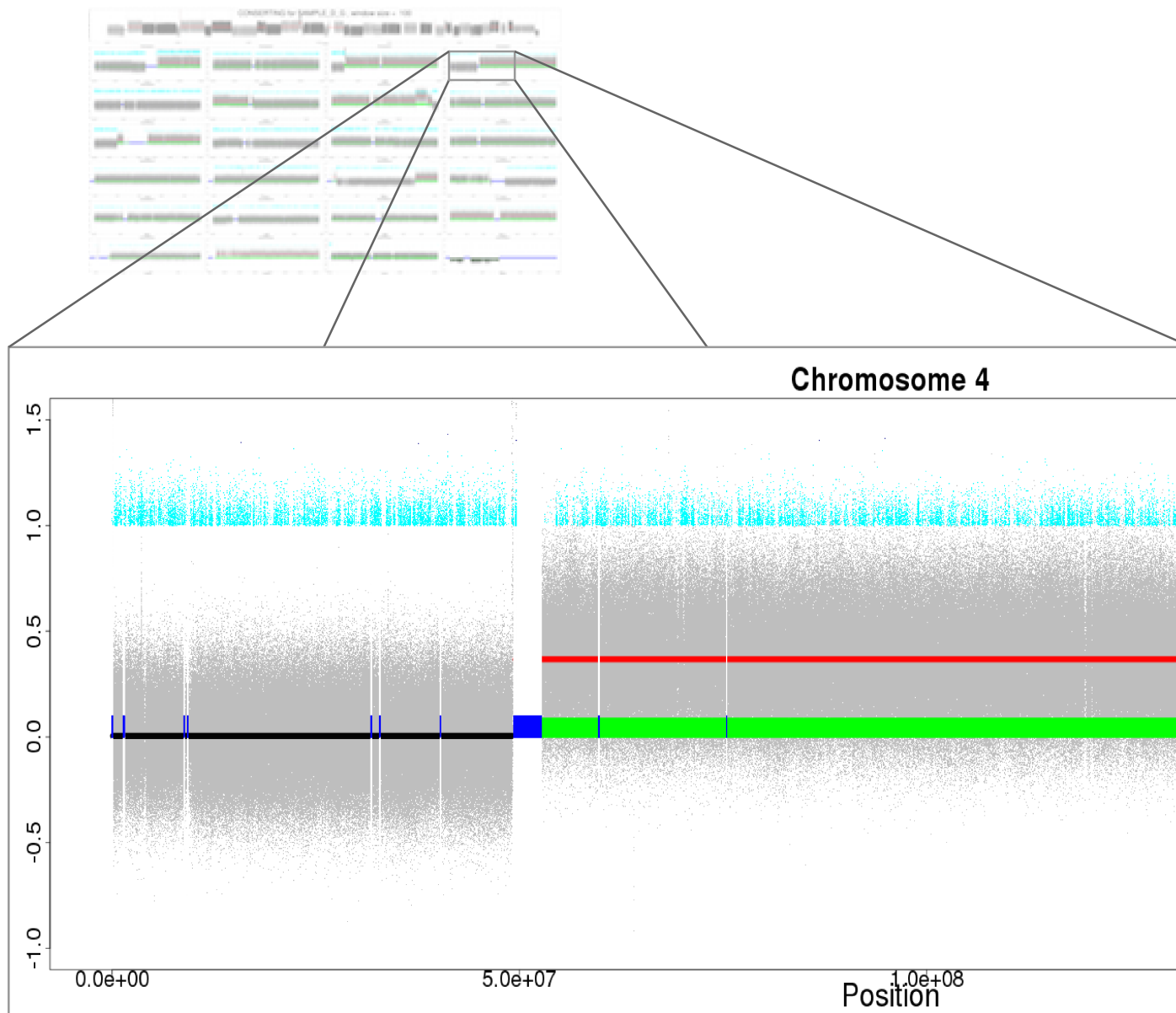
LOH View. The cyan and purple graph represents the Loss of Heterozygosity. It is located in each of the chromosome windows and occupies the top fourth of the view (between 1.0 and 1.5 of the Y axis). The cyan portion represents intermedia level LOH, and purple represents Strong LOH.



Red Line

5

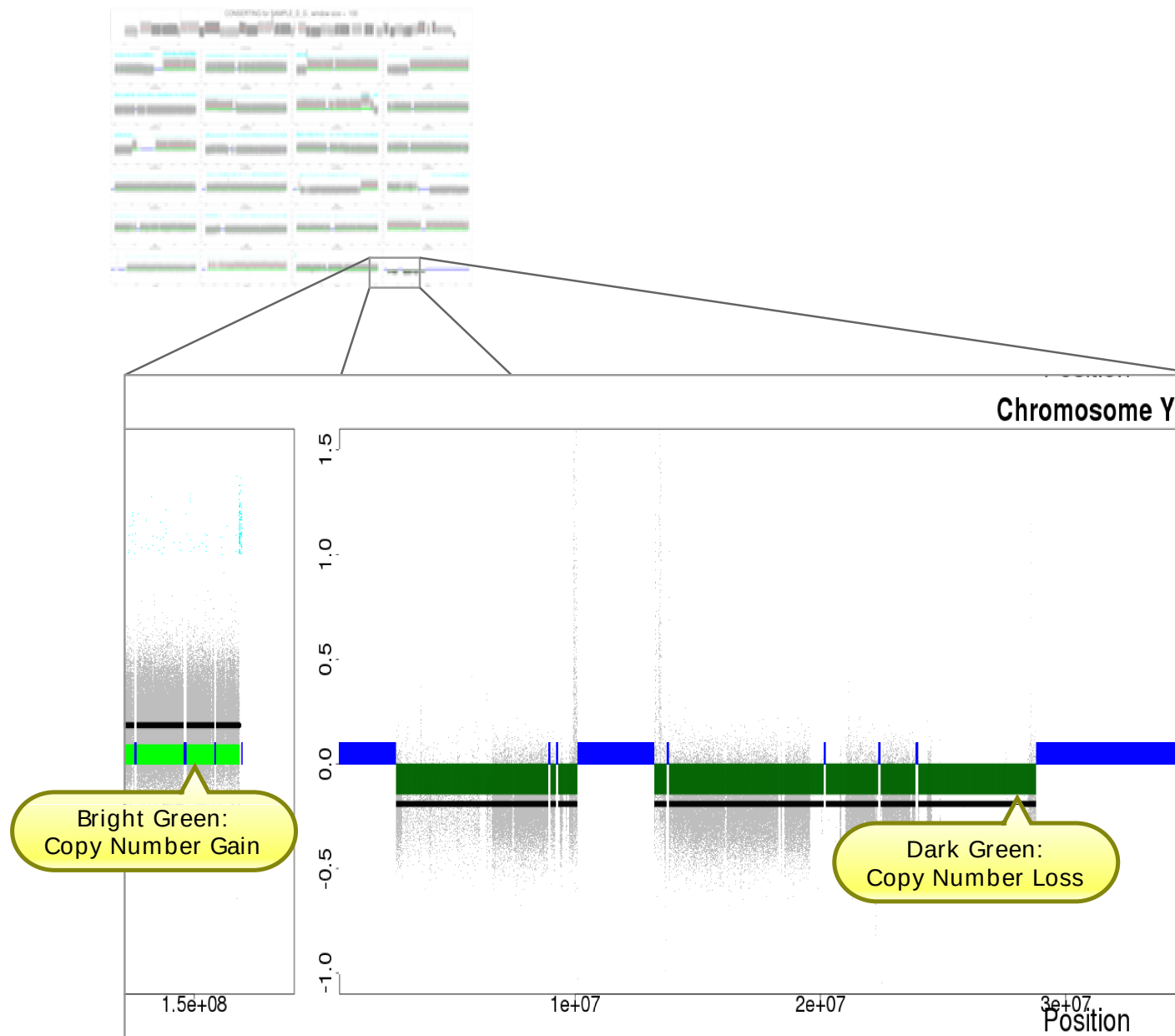
Red Line. In the gray graph, the red line represents the presence of the segment which has an absolute difference of greater than 0.25 ($\text{abs}(\text{diff}) > 0.25$). In other cases a black line is displayed if the absolute difference is less than 0.25 for that segment.



Green Line

6

Green Line. The green line represents the log ratio of the signal divided by 5. A bright green line represents copy number gain, whereas copy number loss is represented by a dark green line. An example of this can be seen in the lower right corner of the attached image (at Chromosome X and Y).



Reference:

1. Edmonson, M.N., et al., *Bambino: a variant detector and alignment viewer for next-generation sequencing data in the SAM/BAM format*. Bioinformatics, 2011. **27**(6): p. 865-6.