

iCopyDAV
**Integrated platform for Copy number variations –
Detection, Annotation and Visualization**

Tutorial

19 February 2018

PrashanthiDharanipragada, SriharshaVogeti and Nita Parekh
Center for Computational Natural Science and Bioinformatics
International Institute of Information Technology, Hyderabad, India

1. About *iCopyDAV*

Integrated platform for Copy number variation – Detection, Annotation and Visualization (*iCopyDAV*) enables the user to identify copy number variations (CNV), an important category of structural variants, in whole genome sequence data. The pipeline considers Depth-of-Coverage based approach for CNV detection due to its ability to predict absolute copy number of the variant. The pipeline doesn't require any matched-control sample and can be used for CNV detection in samples from species other than human (however, functional annotation is available only for human genome assembly hg18/hg19). *iCopyDAV* platform has a modular-framework with 7 main executables for CNV detection, annotation and visualization:

- a) ***calOptBinSize***: Calculates optimal bin size for a given sequencing coverage sample.
- b) ***prepareData***: Generates coordinate, GC and mappability score files according to the desired bin size
- c) ***pretreatment***: Normalizes GC content either by Loess regression or Median approach, and filters low mappable regions for a given mappability cut-off threshold. A GC profile before and after correction is also generated.
- d) ***runSegmentation***: Bins of similar read depth values are segmented together either by Circular Binary Segmentation (CBS) approach or Total Variation Minimization (TVM) approach or both.
- e) ***callCNV***: Calls segments with read depth values deviating from that of normal mean and assigns an absolute copy number to each variant predicted.
- f) ***annotate***: Information about genomic elements (protein coding genes, lincRNA, enhancers, miRNA target sites), structural annotations (segmental duplications, tandem repeats, heterochromatin and telomeric positions), clinical relevance (OMIM, ClinVar, DECIPHER and ExAC) and Known CNVs (DGV) overlapping the predicted CNVs are listed next to each predicted variant.
- g) ***plot***: CNVs distribution across the chromosome or for a given coordinates can be visualized using this function.

User may type '--help' next to any of the executable and syntax for each function can be viewed. Detailed step-by-step instructions for installation and execution are given below.

2. Installation

The pipeline is linux-based and works with any latest version. *iCopyDAV* can be used either by installing the complete package in Docker available at the *iCopyDAV* website (<http://bioinf.iit.ac.in/icopydav/>) or installing via source code, available at GitHub (<https://github.com/vogetihhrsh/icopydav>).

1. *iCopyDAV* via Docker

a. Install docker on your machine (Community version) from the link below:

<https://docs.docker.com/engine/installation/>

b. Download the tar file from the following link:

<http://bioinf.iit.ac.in/icopydav/>

c. Import the tar as a docker image using the docker import command:

sudo docker load--input iCopyDAV.tar

To verify the image has been imported use **docker images** command. You will see an image ID created for the docker.

d. Then create a container with the following command by using the image ID.

Sudo docker run -i -t <Image ID>

You are now in the container and ready to work. The source code and executables for *iCopyDAV* are available in `‘/root/’`. Change the directory by using following command:

cd /root/

e. In a separate window, query your docker container ID and transfer your desired files from the working folder using the following command:

sudo docker ps

The above command displays container ID, image ID along with the status of the docker.

dockercp<Desired file>containerID:.

The above command transfers the files into the root system (`‘/’`) of the docker. Files from this location can be directly accessed or can be moved (using `mv`) to *iCopyDAV* folder and pipeline can be run.

f. Once the CNVs are predicted, annotated and visualized using *iCopyDAV*, files can be transferred back to the desktop by using the following command:

```
docker cp containerID:/root/icopydav/files .
```

Please note files will not be saved for a given container ID. Users need to transfer the results back to the desktop, else, results will be erased upon using the Docker next time.

g. To close the docker, type 'exit'.

2. *iCopyDAV* from the source code

a. Download the following dependencies to run *iCopyDAV* from the source code.

- R Dependencies:
 - Bioconductor packages: DNACopy, GenomicAlignments, rtracklayer, quantsmoothIn R console, type the following:

```
>source("http://bioconductor.org/biocLite.R")
```

```
>biocLite("DNACopy", "GenomicAlignments", "rtracklayer", "quantsmooth")
```

- R package: ParDNACopy
(<https://cran.r-project.org/web/packages/ParDNACopy/index.html>)
In R console, type the following:
>install.packages("ParDNACopy")
- Samtools (<http://samtools.sourceforge.net/>)
- Bedtools (<http://bedtools.readthedocs.io/en/latest/>)
- OpenMPI (<https://www.open-mpi.org/>)

Samtools, Bedtools and OpenMPI must be added to the PATH variable by using the following command:

```
export PATH = $PATH:/usr/bin/samtools
```

Similarly Bedtools and OpenMPI must be added to PATH variation in user environment, so that these tools can be used in *iCopyDAV* directly.

b. Download the source code by clicking 'Download' button from <https://github.com/vogetihrsh/icopydav> and then extract the zip file as shown below

```
unzip iCopyDAV-master.zip  
cd iCopyDAV-master  
make
```

Download the annotations folder from the *iCopyDAV* website (<http://bioinf.iiit.ac.in/icopydav/>) and add the contents of the folder into 'annotations' folder available in *iCopyDAV*.

3. Workflow

We demonstrate the usage of our pipeline by considering a whole genome sequence sample (Chromosome 21, 6x sequencing depth) mapped to hg18 human genome reference assembly. The test file can be downloaded from the *iCopyDAV* website.

(i) Calculating optimal bin size

Files required: config file and alignment file (BAM)

Usage: calOptBinsize -c <config file> -i <input BAM>

Output: Optimal bin size number

In *iCopyDAV*, the user may set the bin size judiciously depending on the sequence coverage and the size of the CNVs to be detected (default 100 bp). For instance bin size ~ 500 bp may be appropriate for low sequencing depth (4-6×), ~100 bp for medium sequencing depth (20-30×) and ~ 30 bp bin size for very high coverage data (~ 100×). Alternately, in *iCopyDAV*, the user may use the function *calOptBinSize*. Function *calcOptBinSize* function which determines ideal bin size based on modeling the reads across the genome using negative binomial distribution. Previous studies detected CNVs under the assumption that all reads were randomly generated from the genome, following a Poisson distribution. However, from an extensive study carried by Miller et al [1], it was observed that the read distribution deviates from Poisson distribution with unequal mean and variance. The reads are instead modelled based on negative binomial distribution or mixture of Poisson distributions with mean value λ , which is equal to $n*b/g$, where n = total number of reads, b = bin size and g = genome size. The variation in λ accounts to excessive variance which has been observed. This variance can be independently controlled keeping the mean constant and is given as an overdispersion parameter = $\lambda/(d-1)$ in the config file ('-c') where d is overdispersion. The chromosome size specific to the reference assembly is added in config file. Distributions are generated using the expected number of reads with copy number of one, two, and three, and parameters 'percCNGain' and 'percCNLoss' (default: 0.05) are included in the config file, help in accurate modeling of the reads and minimizes the number of bins that are misclassified. The smallest bin size possible is progressively estimated for the given false-discovery rate (default: 0.01) in config file [1]. A sample config file is provided in the source package of the pipeline. Since negative binomial distribution overestimates the bin size, the user may consider this as an upper limit for bin size.

(ii) Data Preparation

Files required: map score file and gc score file, genome file

Usage: prepareData -m <map score file> -g <gc score file> --win <desired window size> --genome_file<Genome file> -o <Output filename prefix>

Output: '.bin', '.map', '.gc' for given window size

User may choose desired bin size (Default: 100 bp) or consider optimal bin size generated using the function *calOptBinSize*. Function *prepareData* assists in generating coordinate, gc and mappability score files that are essential for data pre-treatment.

Coordinate file: For Chromosome 21, with reads mapped to Human reference assembly hg18, a genome file must be generated as shown in the following format:

```
chr21 46944323
```

where chromosome number is tab-separated by size of chromosome corresponding to mapped assembly (here, hg18) are listed in a file (chr21.gen). For reference assemblies, hg18 and hg19, genome files are available on the *iCopyDAV* website.

Mappability file: Mappability scores for a genome must be supplied as an input. These scores are highly specific to the reference genome assembly to which reads are aligned and also, rely on length of reads generated in the NGS experiment. Mappability score of a bin range from 0 to 1, with regions of low mappability score (< 0.5) correspond to repetitive and low-complexity regions in the genome. Mappability scores for read length 100 bp and bin size 100 bp for human reference genome hg18/hg19 are available at the *iCopyDAV* portal. In case user wish to use bin size other than 100 bp, the mappability scores suitable to bin size of the interest can be converted using *prepareData* function. For different read length, user may download mappability annotation tracks from UCSC genome browser [2]. The user may also generate mappability scores for read length and genome of interest using GEM library [3].

GC content file: For correcting GC bias, user must supply a GC content score file in data pre-treatment step. The file must contain a column of GC content score per bin with values ranging 0-1. GC content scores for bin size 100 bp for human reference assembly hg18 and hg19 are available at the *iCopyDAV* website. For the demonstration, we consider a window size of 1000 bp and data files (coordinate, gc and mappability files) can be generated using following command:

```
./prepareData -m chr21.dat -g chr21.gc --win 1000 --genome_file chr21.gen -o chr21
```

Running this command results in three files: chr21_1000.bin, chr21_1000.map and chr21_1000.gc (shown in Figure 1) that are ready to use for data pre-treatment step.

1	chr21	0	1000	1	0	1	NA
2	chr21	1000	2000	2	0	2	NA
3	chr21	2000	3000	3	0	3	NA
4	chr21	3000	4000	4	0	4	NA
5	chr21	4000	5000	5	0	5	NA
6	chr21	5000	6000	6	0	6	NA
7	chr21	6000	7000	7	0	7	NA
8	chr21	7000	8000	8	0	8	NA
9	chr21	8000	9000	9	0	9	NA
10	chr21	9000	10000	10	0	10	NA
11	chr21	10000	11000	11	0	11	NA
12	chr21	11000	12000	12	0	12	NA
13	chr21	12000	13000	13	0	13	NA
14	chr21	13000	14000	14	0	14	NA
15	chr21	14000	15000	15	0	15	NA
16	chr21	15000	16000	16	0	16	NA
17	chr21	16000	17000	17	0	17	NA
18	chr21	17000	18000	18	0	18	NA
19	chr21	18000	19000	19	0	19	NA
20	chr21	19000	20000	20	0	20	NA

Figure 1: Snapshot of Coordinate, Mappability score and GC content score files calculated for Chromosome 21, read length 100 bp and bin size 1000 bp (Values for Mappability and GC content files are not available in the initial coordinates (above) due to acrocentric nature of Chromosome 21)

(iii) Data Pre-treatment

Files required: alignment file (BAM), coordinate file, mappability file and GC content score file

Usage: pretreatment -i <input BAM> -o <output Prefix> --mapfile<mappability file> -z <bin file> --mapThres<mappability cut-off value> --medianGC (or) --loessGC --gcfile<gc file>

Output: ‘_pCNVD.bincor’, ‘_pCNVD.input’

GC bias correction and filtering low mappable regions are carried out in *pretreatment* module. A user defined nomenclature (‘Test’ in our demonstration) is provided as output prefix (-o) and same output prefix must be used for subsequent steps (segmentation and variant calling). Along with the alignment file (-i), user needs to supply a coordinate (-z), GC content (--gcfile) and mappability score files (--mapfile or -m) generated in *prepareData* module. Mappability is corrected by considering those bins with score equal or higher than the cut-off threshold provided (default: 0.5). Mappability score cut-off can be changed according to the user by adding an optional parameter ‘--mapThres’ and appropriate cut-off value (0-1). We considered 0.6 as mappability threshold cutoff for the demonstration. The pipeline offers two different algorithms for GC content correction: (i) Loess regression [4] and (ii) Median approach [5].

./pretreatment -i Input.bam -o Test -z chr21_1000.bin --mapfile chr21_1000.map --mapThres 0.6 --medianGC --gcfile chr21_1000.gc

The above command carries out GC bias correction using Median approach. The pre-treatment step results in two intermediate files ‘Test_pCNVD.bincor’ that contain bin coordinates and ‘Test_pCNVD.input’ with corresponding normalized and filtered read depth values, as shown in Figure 2.

1	9720000	9721000	1	720.4497
2	9721000	9722000	2	112.0347
3	9722000	9723000	3	61
4	9723000	9724000	4	907.7164
5	9724000	9725000	5	117.9028
6	9725000	9726000	6	61
7	9726000	9727000	7	614.3571
8	9727000	9728000	8	673.5775
9	9728000	9729000	9	637.8642
10	9729000	9730000	10	61
11	9730000	9731000	11	61
12	9731000	9732000	12	847.5789
13	9732000	9733000	13	792.1286
14	9733000	9734000	14	915.4959
15	9734000	9735000	15	61
16	9735000	9736000	16	61
17	9736000	9737000	17	1163.067
18	9737000	9738000	18	61
19	9738000	9739000	19	812.4861
20	9739000	9740000	20	116.3279

Figure 2: A snapshot of (a) Test_pCNVD.bincor and (b) Test_pCNVD.input intermediate files generated as a result of *pretreatment* with Median-based GC correction and Mth =0.6

User may use the following command for GC correction using Loess regression and the results are different from Median-based GC correction approach, as seen from Figure 3.

```
./pretreatment -i Input.bam -o Test --mapfile chr21_1000.map --mapThres 0.6 -z chr21_1000.bin --loessGC --gcfile chr21_1000.gc
```

1	9720000	9721000	1	992.5837
2	9721000	9722000	2	1108.082
3	9722000	9723000	3	1157.858
4	9723000	9724000	4	991.5716
5	9724000	9725000	5	2181.756
6	9725000	9726000	6	620.5716
7	9726000	9727000	7	699.7888
8	9727000	9728000	8	778.5614
9	9728000	9729000	9	841.553
10	9729000	9730000	10	452.3505
11	9730000	9731000	11	465.5716
12	9731000	9732000	12	918.5716
13	9732000	9733000	13	903.5223
14	9733000	9734000	14	917.5239
15	9734000	9735000	15	838.5716
16	9735000	9736000	16	748.7649
17	9736000	9737000	17	1138.538
18	9737000	9738000	18	1045.532
19	9738000	9739000	19	953.8579
20	9739000	9740000	20	1164.021

Figure 3: A snapshot of (a) Test_pCNVD.bincor and (b) Test_pCNVD.input intermediate files generated as a result of *pretreatment* with Loess regression for GC correction and Mth =0.6

Please note that when a GC correction approach (--loessGC/medianGC) is not given in the command, GC correction of the reads do not take place and only reads in the regions with low mappability scores (<Mth) are filtered out.

(iv) Segmentation

Files require: ‘_pCNVD.bincor’, ‘_pCNVD.input’

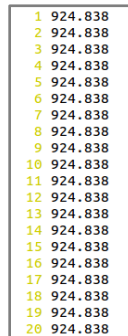
Usage: runSegmentation -o <output prefix> <one or both segmentation flags (-t, -d)> -p <no. of processors>

Output: ‘_pCNVD.output’

Bins with similar read depth values are identified and merged in segmentation step using *runSegmentation* function. Segmentation in *iCopyDAV* can be carried out using either (a) agglomerative approach, Total Variation Minimization (TVM) [6] or (b) Circular Binary Segmentation (CBS) [7], a divisive approach or both. Type of segmentation, ‘-t’ for TVM or ‘-d’ for CBS or both must be supplied to the function along with the same output prefix used in preprocess module. An optional parameter ‘-p’ may be set (default: 32) for parallelization of segmentation process, helps in faster segmentation in the presence of large number of processors.

`./runSegmentation -o Test -t`

The above command carry out segmentation using TVM approach (agglomerative) and results in Test_tvm_pCNVD.output file, that contains an average score of read depth across the bins which are segmented as shown in the Figure 4 below.



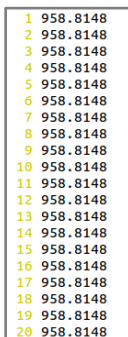
```
1 924.838
2 924.838
3 924.838
4 924.838
5 924.838
6 924.838
7 924.838
8 924.838
9 924.838
10 924.838
11 924.838
12 924.838
13 924.838
14 924.838
15 924.838
16 924.838
17 924.838
18 924.838
19 924.838
20 924.838
```

Figure 4: A snapshot of Test_tvm_pCNVD.output file generated as a result of segmentation using TVM approach

The following command is used when user wish to carry out segmentation using CBS (divisive) approach.

`./runSegmentation -o Test -d`

This generates an output file as Test_cbs_pCNVD.output, as shown in Figure 5.



```
1 958.8148
2 958.8148
3 958.8148
4 958.8148
5 958.8148
6 958.8148
7 958.8148
8 958.8148
9 958.8148
10 958.8148
11 958.8148
12 958.8148
13 958.8148
14 958.8148
15 958.8148
16 958.8148
17 958.8148
18 958.8148
19 958.8148
20 958.8148
```

Figure 5: A snapshot of Test_cbs_pCNVD.output file generated as a result of segmentation using CBS approach

The user may also run TVM as well as CBS simultaneously by using the following command.

```
./runSegmentation -o Test -t -d
```

As a result, files ‘Test_tvm_pCNVD.output’ as well as ‘Test_cbs_pCNVD.output’ are generated one after the other.

(v) Variant calling

Files required: ‘_pCNVD.output’

Usage: callCNV -o <output prefix> -z <bed file containing bins><genome flag (--hg18 or --hg19)>

Output: ‘_pCNVD.bed’

Segmented regions with abnormal copy number are identified and reported with their start/end coordinates, type of event (duplication as 1 or deletion as 0) and absolute copy number using *callCNV* function. The mean of read depth across the chromosome is calculated and any deviation (higher or lower than copy gain and copy loss thresholds respectively) from this mean are considered as CNVs. For both TVM and CBS approaches, CNVs are reported only when a minimum of two bins have similar read depth values, hence, reducing the number of false positives. For calling CNVs from the ‘_pCNVD.output’ file, coordinate file (z) generated in the beginning the output prefix (-o) and genome flag (--hg18 or --hg19) must be provided by the user. Adding genome assembly parameter helps in removing CNVs in chromosomal gaps, if at all falsely generated by TVM or CBS as a CNV.

```
./callCNV -o Test -z chr21_1000.bin --hg18
```

The command generates CNV calls in ‘_pCNVD.bed’ output file. A separate command for calling CNVs from TVM and CBS is not necessary. The function automatically identifies ‘.output’ files from both the methods and generates CNVs for TVM and CBS approaches in separate files. *callCNV* results in files similar to Figure 6.

1	chr21	9720000	9747000	1	10	
2	chr21	9747000	9789000	1	10	
3	chr21	9792000	10034000	1	3.10948	
4	chr21	10043000	10054000	1	6.35872	
5	chr21	10054000	10075000	1	10	
6	chr21	10075000	10103000	1	10	
7	chr21	10103000	10134000	1	10	
8	chr21	10134000	10146000	1	10	
9	chr21	10146000	10163000	1	10	
10	chr21	10163000	10174000	1	10	
11	chr21	10174000	10190000	1	9.07647	
12	chr21	10190000	10210000	1	7.1502	
13	chr21	13260000	13291000	1	7.1502	
14	chr21	13293000	13369000	1	3.25535	
15	chr21	13664000	13666000	1	3.61415	
16	chr21	14232000	14234000	1	3.89425	
17	chr21	20881000	20884000	0	0.575334	
18	chr21	46942000	46944000	1	4.64952	

Figure 6: A snapshot of CNV calls (Test_cbs_pCNVD.bed) made using CBS approach for segmentation and loess regression for GC correction and Mth=0.6 considered for data pre-treatment

(vi) CNV annotation

Files required: Any bed file (tab-separated)

Usage: ./annotate -i <input bed file> -o <output prefix> <genome flag (--hg18 or --hg19)>

Output: ‘_annotated.bed’

Once CNVs are generated from either TVM or CBS or both the methods (‘.bed’), the user may be interested to understand the structural and functional significance of the CNVs detected. The pipeline provides a basic CNV annotator (*annotate*) which gives information about four sets of annotations viz. (1) Functional elements that includes protein coding genes, lncRNA, regulatory elements such as enhancers, miRNA target sites spanning the CNV (with minimum of 1 bp overlap), (2) clinical features (ClinVar, OMIM, DECIPHER and ExAC), (3) structural annotations such as segmental duplications, tandem repeats, position of CNV (whether in telomeric or heterochromatin region) and finally, (4) list of already known CNVs reported in Database of Genomic Variants (DGV) spanning these CNV regions. These features help in filtering known CNVs and characterizing novel CNVs predicted from the pipeline. The function *annotateresults* in ‘_annotated.bed’ file.

./annotate -i Test_TVM_pCNVD.bed -o Test --hg18

This result in a tab-separated Test_annotated.bed file which contains predicted CNVs along with their annotations as shown in Figure 7. The tab-separated file can be visualized using any spreadsheet application including MS-Excel or Open Office Calc.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	#Added annotation columns in the order: GENE LOCATION Gene_Structural_Elements DGV_accessions Enhancers miRNA_target_sites Segmental_Duplications Interspersed_repeats Tandem_repeats ClinVar_pathoge																				
2	chr21	9720000	9747000	1	10	dgv2455e59,dgv2456
3	chr21	9747000	9789000	1	10	dgv2456e59,dgv2459
4	chr21	9792000	10034000	1	3	TPTE	.	.	.	exon,transcript	.	dgv2460e59,dgv2461
5	chr21	10043000	10054000	1	6	BAGE2,BAGE3,BAGE4	dgv2468e59,dgv4353	.	.	BAGE2:miR-183,BAG	chr21:10017102-1009
6	chr21	10054000	10075000	1	10	BAGE2,BAGE3,BAGE4	dgv2468e59,dgv4353	.	.	BAGE2:miR-122,BAG	chr21:10017102-1009
7	chr21	10075000	10103000	1	10	BAGE,BAGE2,BAGE3	dgv2468e59,dgv4354	.	.	BAGE4:miR-27,BAGE
8	chr21	10103000	10134000	1	10	BAGE,BAGE2,BAGE3	dgv2468e59,dgv4354
9	chr21	10134000	10146000	1	10	dgv23e197,dgv2468e
10	chr21	10146000	10163000	1	10	dgv2468e59,dgv4354	chr21:10138514-1016
11	chr21	10163000	10174000	1	10	dgv2468e59,dgv4354
12	chr21	10174000	10190000	1	9	dgv2468e59,dgv4354	chr21:10168389-1019
13	chr21	10190000	10210000	1	7	exon,transcript	.	dgv2468e59,dgv7729	chr21:10176355-1020
14	chr21	13260000	13291000	1	7	dgv100e19,dgv7751r
15	chr21	13293000	13369000	1	3	ANKRD30BP2	dgv4374n100,dgv437	chr21:13291335-1339

Figure 7: Snapshot of annotation file generated showing few of the annotations. Symbol ‘.’ indicates no annotation available for the given CNV

(vii) Plotting CNVs across the chromosome

Files required: ‘.bed’ with list of predicted CNVs

Usage: plot -i <input bed file> -o <output file><genome flag (--hg18 or --hg19)>

plot -i <input bed file> -d --start <start position> --end <end position> -o <output file><genome flag (--hg18 or --hg19)>

Output: A png file with CNVs plotted with copy gain in blue and copy loss in red

User can also use the pipeline for visualizing the CNVs predicted using *plot* function. Distribution of CNVs along the chromosome or a region of interest (user defined coordinates) and on corresponding ideogram is generated using the function.

./plot -i Test_cbs_pCNVD.bed -o CNV_distribution --hg18

The above command generates a CNV plot as shown below in Figure 8. Gain is represented in blue and deletion in red. Plot is generated in a .png file.

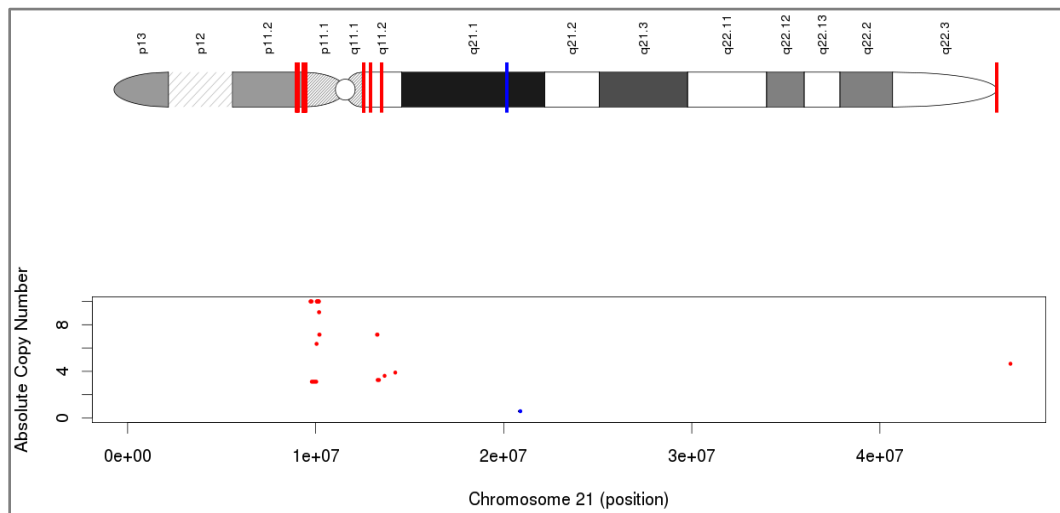


Figure 8: CNVs distribution predicted using *iCopyDAV*(Loess regression for GC correction + Mth=0.6 + CBS approach) across Chromosome 21 of Test sample generated as ‘CNV_distribution.png’

**./plot -i Test_cbs_pCNVD.bed -d --start 10000000 --end 20000000 -o
CNV_distribution_coordinates --hg18**

The above command generates a CNV plot for given set of coordinates as shown below in Figure 9. Gain is represented in blue and deletion in red.

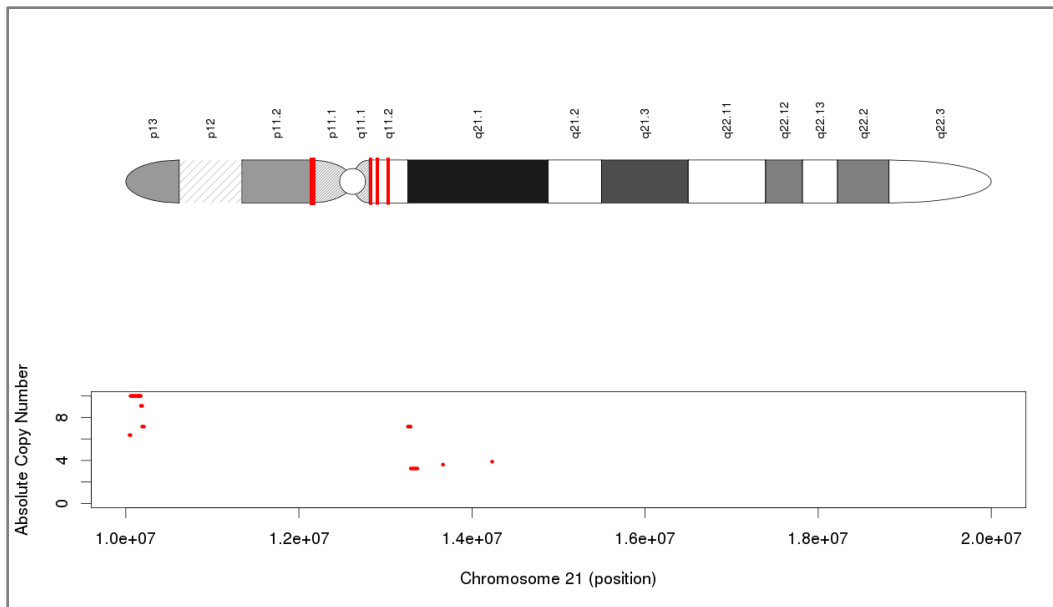


Figure 9: CNVs distribution predicted using *iCopyDAV* across Chromosome 21 of Test sample generated as ‘CNV_distribution_coordinates.png’

References

1. Miller CA, Hampton O, Coarfa C, Milosavljevic A. ReadDepth: a parallel R package for detecting copy number alterations from short sequencing reads. *PLoS One*. 2011;6: e16327. doi:10.1371/journal.pone.0016327
2. Karolchik D, Hinrichs AS, Kent WJ. The UCSC Genome Browser. *Curr Protoc Bioinforma Ed Board Andreas Baxevanis Al*. 2009;CHAPTER: Unit1.4. doi:10.1002/0471250953.bi0104s28
3. Derrien T, Estellé J, Sola SM, Knowles DG, Raineri E, Guigó R, et al. Fast Computation and Applications of Genome Mappability. *PLOS ONE*. 2012;7: e30377. doi:10.1371/journal.pone.0030377
4. Cleveland WS, Loader C. Smoothing by Local Regression: Principles and Methods. *Statistical Theory and Computational Aspects of Smoothing*. Physica-Verlag HD; 1996. pp. 10–49. doi:10.1007/978-3-642-48425-4_2
5. Yoon S, Xuan Z, Makarov V, Ye K, Sebat J. Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Res*. 2009;19: 1586–1592. doi:10.1101/gr.092981.109
6. Duan J, Zhang J-G, Deng H-W, Wang Y-P. CNV-TV: a robust method to discover copy number variation from short sequencing reads. *BMC Bioinformatics*. 2013;14: 150. doi:10.1186/1471-2105-14-150
7. Olshen A, Seshan V. DNACopy: DNA copy number data analysis. Available: <http://bioconductor.org/packages/DNACopy/>