Reference User Guide: CONVector 1.2 Package

Detection of Large-scale Structural Variations in PCR-enriched Target Sequencing Data

Document last modified on June 18, 2015

Software last modified on June 18, 2015

parseq
Lab

Contents

1 Disclaimer of warranties and limitation of liabilities

2 Authors

This set of tools was created by Bragin Anton (Parseq Lab) and Demidov German* (Parseq Lab, St Petersburg Academic University).
Corresponding author: gdemidov@parseq.pro

3 Software Description

This set of tools called *CONVector package* was created to detect CNVs in PCR-enriched target sequencing data and was tested on files produced using Life Technologies Ion^{TM} PGM machine.
The most important prerequisites of *CONVector package* are:

- It requires **a set of samples sequenced using the same panel of amplicons** as the test set of samples.

- This tool **requires at least one dataset with more than 25 samples** as the control group. Ensure that all of samples in the control group were sequenced very accurately under the same conditions.

- It is always better to use the **samples that were sequenced in one run of sequencing** as the both Test and Control datasets. **All measures of data's quality** suggested in this package **do not show the true differences** between datasets. Sometimes it is possible to merge two datasets consisted of samples sequenced in two runs of sequencing in case if the size of the Test Dataset is too small, but the results can be worse then results of the analysis of one run of sequencing.

- It works under the assumption that **each target region was affected by CNVs with the frequency no more than 1/5** in the dataset.

- In case **if the number of genes covered by the panel of amplicons is $n > 2$, then it is allowed no more than $\lfloor \frac{n}{2} \rfloor$ genes contain CNVs** in each sample for the correct work of the algorithm. We can not guarantee an accurate results with panels that covers less then 3 genes and with samples that contain larger number of CNVs.

3.1 Intuitive ideas of detection

We explain the main principles based on the part of real dataset. We have took the data about coverages of 7 amplicons $\{R_1, \ldots, R_7\}$ in 47 samples. 2 of these samples have CNVs in the same 4 amplicons $\{R_4, \ldots, R_7\}$.

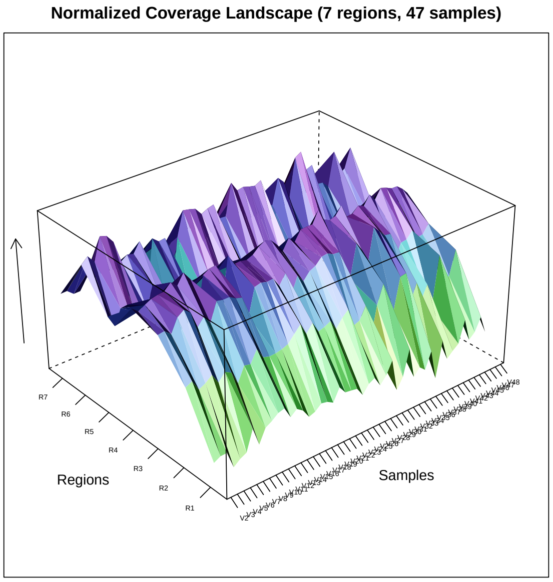


Figure 1: The landscape of normalized coverages.

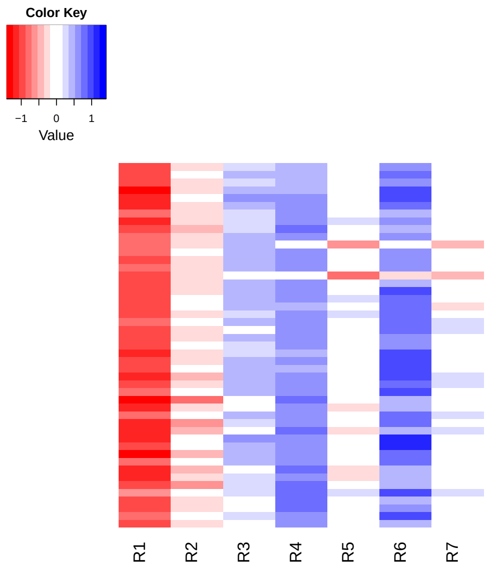


Figure 2: The heatmap of normalized coverages.

After some normalization of data we can see that each amplicon has its own approximate height. If we imagine a Traveller, who tries to explore this landscape, his possible travel journal could be:

- *Region 1*: the typical height is from 50 to 200 meters.
- *Region 2*: the typical height is from 200 to 250 meters.
- *Region 3*: the typical height is from 250 to 300 meters.
- *Region 4*: the typical height is from 150 to 250 meters. I have found 2 unusually low regions with coordinates $(V_{12}; R_4)$ and $(V_{16}; R_4)$.
- . . . and so on.

As you have already understood, the Traveller is the algorithm of the detection of CNVs and the unusually low cavities and high peaks located in some regions indicate the deletions and duplications of amplicons, respectively. His travel journal will be called the Matrix of CNVs.
Let us imagine the situation that the Traveller founds a lot of unusually low or high regions with coordinates $(V_1; R_i), i \in \{1, \ldots, 7\}$. It confuses Traveller and makes his estimations of height of regions inaccurate. Of course he thinks: "Hey, this does not give me any information. This part of the landscape is unpredictable, but I want my final results to be accurate. I will skip the regions with coordinates $(V_1; R_i)$." We call the sample V_1 *irregular* and the procedure of detection of Irregular samples will be described in the chapter Removing irregular samples. The same is true for some amplicons (they can be unpredictable), they will be indicated as "EX" and "NE" in the Matrix of CNVs.
We also can imagine that the earthquake changed the landscape and made it beyond recognition. The travel journal will contain the different heights, for example,

- *Region 1*: the typical height is from -500 to 400 meters.
- *Region 2*: the typical height is from 200 to 1200 meters.
- . . . and so on.

Just compare the intervals with the intervals from the first travel journal! Evidently, the Traveller will not recognize true cavities and peaks. We call such kind of landscapes *Irregular* and it is better not to work with datasets that produce such a landscape (more information). The reason of "earthquakes" in the target sequencing is usually wrong library preparation and other violations of the experimental protocol.

3.2 The scheme of the pipeline

At first let us make the following definitions:

Definition 1. Control Dataset *means the dataset that consists of more than 25 samples sequenced using the same panel of amplicons. All samples from the Control Dataset must be sequenced under the equal conditions. Ideally, Control set should not contain any CNVs, but the presence of small amount of CNVs in the Control set will not reduce the quality of the output greatly.*

Definition 2. Test Dataset *means the dataset that consists of some samples sequenced in one run of sequencing machine, using the same panel of amplicons as for the train set.*

Then, using these defintions, we can shortly describe the scheme of the pipeline step by step:

1. Count coverages. *Checkpoint:* an initial .xls file with the Matrix of Coverages.
2. Perform QC control and may be merge two Matrix of Coverages - Test and Control. *Checkpoint:* a .xls file with the Matrix of Coverages of samples that are suitable for the analysis.

3. Detect CNVs. *Checkpoint:* .xls file with the Matrix of CNVs.

4. Create final report. *Checkpoint:* report file, Sample: a region affected with variant.

This steps are performed with the following scripts from the pipeline:

1. Counting coverages using `chimeric_solver.py`. Re-alignment of chimeric parts. Output: `.xls`-file with coverages in folder result.

2. Quality control using `qc.py`. Output: `.xls`-file with coverages after QC in folder result and qc report `qc_control_log.txt`.

3. Unsupervised algorithm using `/deletionAnalysis/Main.java`. Output: file with CNVs in separate amplicons with name `outputYOUR_RUN_ID.xls`

4. Supervised algorithm using `/deletionAnalysis/Main.java`. Output: file with CNVs in CNV sites with name `outputYOUR_RUN_ID_after_LDA.xls`

5. Final output of results using `finalizer.py`. Output: files with CNVs in whole exons named `result_before_LDA_YOUR_RUN_ID.xls`, `result_after_LDA_YOUR_RUN_ID.xls`

All these steps will be peformed automatically when you run the `pipeline.py` script.

4 Input and Output files

4.1 Overall Description

Input files:

- the Matrix of Coverages of *Control* dataset (can be obtained using `chimeric_solver.py` and folder with .bam files from the *Control* dataset);
- the folder with .bam files from the *Test* dataset;
- .bed file with coordinates of regions.

Required format of .bed file (no additional tab characters `\t`, one region per line):

```
track name = Any comments \n
chrN \t start_coord \t end_coord \t region ID \t [gene, exon, etc.] \t comments \n
. . .
```

Input arguments:

- id of task, all output files will contain this string in their name, *string using only letters (lower and upper case), numbers and symbol ’_’, flags:* `-id, --id`;
- address of the .bed file, *absolute or relative path, flags:* `-b, --bed`;
- optionally: level of specificity, *string that is equal to one of the following: “hard”, “normal”, “soft”, they will be transformed to the quantiles of Student’s t-distribution with confidence level* $\alpha_h = 0.01$*,* $\alpha_n = 0.02$*,* $\alpha_s = 0.05$*, respectively, flags:* `-m, --mode`;
- optionally: if the coverages were not counted before - path to the folder with .bam files, *string with path, absolute or relative, flags:* `-d, --dir`;
- optionally: if there are too many non effective regions - the correlation threshold, *real number between 0.0 and 1.0 (recommended: 0.5 – 0.9, depending on uniformity of dataset), flags:* `-mc, --mincor`;
- optionally: the level of quality control filtration *(0.0 – 1.0, the bigger value means more strict filtration), flags:* `-qc, --qc`;
- optionally: the Control Dataset’s matrix of coverages, *string with path to .xls file, flags:* `-cs, --control`.

You can count coverages only once for the set of samples. After counting coverages you may just specify the id of task and not to specify the address of the folder with .bam files.
So the example of the launch using these parameters is

Listing 1: Example of launch with all parameters specified

```
./pipeline.py -d /Users/admin/Desktop/Run_78 -b 2.bed -id Run_78_fin --mode normal -mc 0.8 -cs Run_77 -qc 0.95
```

Output files:

- Files with coverages of each region
  1. of each sample:
     `/CONVector/result/Id_of_task.xls`;

  2. of samples that passes our QC procedures:
     `/CONVector/result/Id_of_task_qc.xls`;

- The raw files with the results of CNVs detection in each region
  1. of each sample after the unsupervised algorithm:
     `/CONVector/outputId_of_task.xls`;
  2. of samples with potentially affected CNV sites after the supervised algorithm:
     `/CONVector/outputId_of_task_after_LDA.xls`;
- Short reports about the CNV sites affected by CNVs:
  1. of each sample after the unsupervised algorithm:
     `/CONVector/result_before_LDA_Id_of_task.xls`;
  2. of samples with potentially affected CNV sites after the supervised algorithm:
     `/CONVector/result_after_LDA_Id_of_task.xls`;

4.2 Detailed description of several file formats

Matrix of Coverages

The key object is the Matrix of Coverages of the dataset.

Table 1: Matrix of Coverages

Gene	Target	Sample 1	Sample 2	Sample 3
N/A	Region 1	100	200	250
Gene 1	Region 2	200	300	320
Gene 2	Region 3	1000	1500	2000

We keep it inside the tab delimited file with extension .xls.
NOTE: It is important that the new line symbol is `\n` and the delimiter symbol is `\t`. Please, be accurate with editing of the input files with any spreadsheet applications.

Matrix of CNVs

The tab delimited file with the extension .xls that contains information about CNVs is organized as:

- At first, there is a block of techincal information about parameters that were used during the analysis: minimum of significance for correlation, quantiles of Student's t-distribution that were used, name of the dataset, etc..
- Then below this block there are a table separated with an empty new line, where
  1. EX means that this region was low covered in the dataset and it is better to EXclude it from the analysis;
  2. NE means that this region shows low correlation with other regions from the dataset and it is better to think that it amplifies Non Effectively and exclude from the analysis;
  3. empty cell means that the tool does not recognize the region as affected by CNV;
  4. DEL means that there probably are a heterozygous or homozygous deletion;
  5. ˆmeans that this region was probably duplicated.

Table 2: Example of the Matrix of CNVs

Chr	Startpos	Stoppos	Amplicon name	Description	Sample 1	Sample 2
chr1	100	200	AMPL1	Gene 1 Exon 1	NE	NE
chr2	1000	1100	AMPL2	Gene 2 Exon 1	EX	EX
chr2	2000	2150	AMPL3	Gene 2 Exon 2	DEL	
chr2	5000	5100	AMPL4	Gene 2 Exon 2		ˆ

5 Installation

The recommended requirements for the scripts from CONVector pipeline:

1. `Linux` or `MacOS`;
2. `Python 2.6.x - 2.7.x`;
3. `JRE 7` or newer;
4. `samtools` (its better to get the newest version);
5. `nano` or other text editor for examining the results. *Optionally, but recommended:* install any kind of spreadsheet application (like Excel, Numbers, OpenOffice, etc.).

Please, be sure that you have all the components listed above and all paths were added to `PATH` so the applications 2, 3, 4 can be launched from the Terminal.
Then you can simply clone the repository:

Listing 2: Installation of the package

```
cd your_installation_path
git clone http://gdemidov@10.5.20.11:7990/stash/scm/nnds/deled.git
```

to install the software. All the compilation steps will be performed before each run authomatically. The path `/CONVector/` from the examples listed below is now equal to:
`your_installation_path/deled/`

6 Simple launch: pipeline.py

```
Source code available in:
    /CONVector/pipeline.py
```

Let us assume that you have a folder `/home/folder/folder_with_bams` with .bam files (more than 25), a file `/home/bedfile.bed` and do not have any Control Datasets, but you want to test this tool and do not want to read this readme file. OK, the script `pipeline.py` can do it for you.

Listing 3: Simple run of the package

```
cd /CONVector
./pipeline.py -d /home/folder/folder_with_bams -b /home/bedfile.bed -id hello_world_run
```

This script will launch each element of the package step by step. Also you can use flags

```
   -d # folder with .bam files, you may skip this parameter if you already have
      # the Matrix of Coverages of the Test Dataset, then you should specify only
      # flag -id with the name of the file with the Matrix of Coverages (without .xls).
   -b # path to corresponding .bed file
  -cs # path to the Matrix of Coverages of the Control Dataset
  -mc # minimum corrleation threshold (0.7 by default)
   -m # mode - hard, normal or soft (normal by default)
  -qc # quality control level, default = 0.8
  -id # id of task
```

Is everything OK? What did you say? No?

7 The preparation of Matrix of Coverages depending on Test Dataset size

The usage of the pipeline depends on the number of samples included in the *Test* and *Train* datasets. Below we provide you recommendations how to use the package depending on your Test Dataset size.

7.1 Test dataset size: ≥ 40 samples

<table>
<tr><th>Use Case:</th><th>Test dataset size is ≥ 40 samples</th></tr>
<tr><td><i>Quick description</i></td><td>The size of Test dataset is quite big. The best strategy is to analyse Test dataset separately from Control dataset.</td></tr>
<tr><td><i>Property:</i></td><td>• Average Robust Variance of Test dataset with respect to Test dataset: ARV_T
• Average Robust Variance of Control dataset with respect to Control dataset: ARV_C</td></tr>
<tr><td><i>Quick hints</i></td><td>ARV can be calculated using <code>qc.py</code> (Quality Control tool).</td></tr>
<tr><td colspan="2"><i>Scenario Name:</i>
1. ARV_T is less or slightly bigger then $ARV_C \longrightarrow$ the comparative level of noise of the Test Dataset is quite low. You can perform the analysis.
2. ARV_T is significantly bigger then $ARV_C \longrightarrow$ the Test Dataset is noisy. The possible problems: 1) there are a lot of irregular samples; 2) the Test Dataset is irregular.</td></tr>
<tr><td colspan="2"><i>Solutions of problems:</i>
1. Run Quality Control tool with very strict filtration (percent of smallest outliers should be equal 100%) several times iteratively using generated files as the input files. Repeat the procedure with respect to the size of the new Test dataset until successful reduce of the ARV_T.
2. You should try to find a reason of irregularity in your experimental protocol and then repeat the analysis from the very beginning (library preparation, sequencing, etc.).</td></tr>
</table>

7.2 Test dataset size: ≥ 25, but < 40 samples

Use Case:	**Test dataset size is ≥ 25, but < 40 samples**

Quick description	The size of Test dataset is not very big, but the goal is to avoid the merging of Test and Control dataset due the fact that it can lead to the lower sensitivity of the CNV detection.
Property:	• Average Robust Variance of Test dataset with respect to Test dataset: ARV_T • Average Robust Variance of Control dataset with respect to Control dataset: ARV_C
Quick hints	ARV can be calculated using `qc.py` (Quality Control tool).

Scenario Name:

1. ARV_T is less then ARV_C $\longrightarrow$ the comparative level of noise of the Test Dataset is quite low. You can perform the analysis without merging Test and Control datasets.
2. $ARV_T \approx ARV_C$ $\longrightarrow$ we recommend to perform both analysis with and without merging datasets and to compare results. The possible problems: 1) a lot of samples from Test Dataset do not pass the quality control procedure.
3. ARV_T is significantly bigger then ARV_C $\longrightarrow$ the Test Dataset is noisy. The possible problems: 2) there are a lot of irregular samples; 3) the Test Dataset is irregular.

Solutions of problems:

1. If a lot of samples from Test Dataset do not pass the quality control procedure with respect to Control Dataset, we would recommend you not to merge datasets.
2. Run Quality Control tool with very strict filtration (percent of smallest outliers should be equal 100%) several times iteratively using generated files as the input files. Repeat the procedure with respect to the size of the new Test dataset.
3. You should try to find a reason of irregularity in your experimental protocol and then repeat the analysis from the very beginning (library preparation, sequencing, etc.).

7.3 Test dataset size: < 25 samples

Use Case:	**Test dataset size is < 25 samples**
Quick description	The size of Test dataset is comparatively small so you should try to merge the Control and Test datasets.

Scenario Name:

1. A lot of samples from Test dataset passed the qc procedure ⟶ merge two datasets.
2. A small number of samples from Test dataset passed the qc procedure ⟶ try to find more suitable Control dataset.

8 The analysis of prepared Matrix of Coverages

Once you have prepared the Matrix of Coverages based on the size of Test Dataset and saved it in file `hello_world_run.xls`, you can perform its analysis.

Listing 4: Run of the package using prepared Matrix of Coverages

```
cd /CONVector
./pipeline.py -b /home/bedfile.bed -id hello_world_run
```

At first, we recommend you to examine the file `/CONVector/result_before_LDA_Id_of_task.xls` and look at the amount of regions that has labels “NE” or “EX” in respective cells. If there are too many regions with labels “EX”, then your dataset contains too many low covered regions and you should re design your panel of regions and repeat the experiments. If there are too many regions with labels “NE”, you should repeat the analysis with the new specified parameter `-mc` (minimum correlation parameter, default value = 0.7), for example:

Listing 5: Run of the package using prepared Matrix of Coverages

```
./pipeline.py -b /home/bedfile.bed -id hello_world_run -mc 0.65
```

Usually, in case if the dataset has low ARV, the high amount of “NE” indicates that samples’ preparation and sequencing were accurate. If you have too many “NE” labels again, try to reduce the `-mc` value again:

Listing 6: Decreasing of correlation

```
./pipeline.py -b /home/bedfile.bed -id hello_world_run -mc 0.6
./pipeline.py -b /home/bedfile.bed -id hello_world_run -mc 0.55
```

...and so on. In case you do not have any “NE” labels, it is better to slightly increase the minimum correlation parameter:

Listing 7: Increasing of correlation

```
./pipeline.py -b /home/bedfile.bed -id hello_world_run -mc 0.75
./pipeline.py -b /home/bedfile.bed -id hello_world_run -mc 0.8
```

- until the first NE labels will appear.

9 Usage of Chimeric Solver

```
Source code available in:
    /CONVector/chimeric_solver.py
    /CONVector/parseq/chimeric_solver/Main.java
Additional files (will be created automatically):
    /CONVector/output_with_SAMs/*.sam
    /CONVector/tmp_chimeras.txt
    /CONVector/tmp_references.txt
    /CONVector/tmp_output.txt
```

Listing 8: Simple run of Chimeric Solver

```
./chimeric_solver.py -d path_to_folder_with_bam_files -b bed_file.bed -r id_of_task.xls --conv
```

Flag `--conv` is necessary for the converting the .bam files to .sam files. If you have already converted .bam files to .sam, you may skip this flag.
The arguments are:

```
   --dir, -d          # path to the folder with .bam files
   --bed, -b          # path to .bed file
   --mq, -m           # minimum allowed probability of wrong mapping
                      # of reads, default = 0.9
   --len, -l          # minimum allowed length threshold, default = 1
   --cutoff, -c       # minimum significant overlap of read and region, base pairs,
                      # default = 20
   --clipping, -clip  # minimum length of chimeric part of read, base pairs
                      # default = 30
   --numOfReads, -n   # minimal number of reads for the sample to be
                      # taken in analysis, default = 15.000
   --resFile, -r      # output file (Matrix of Coverages, extension = .xls)
```

10 Usage of QC Control

```
Source code available in:
    /CONVector/qc.py
Additional files (will be created automatically):
    /CONVector/qc_control_log.txt
```

10.1 Simple run

Command to run quality control of Matrix with Coverages X.xls (Test Dataset) against Matrix with Coverages Y.xls (Train Dataset):

Listing 9: Simple run of the QC control

```
cd /CONVector
./qc.py /home/bedfile.bed ./results/X.xls ./results/Y.xls output_file_name 0.8
```

The last number means the strictness of the filtration: 0.0 means “no quality filtration”, 1.0 means “very strict filtration”. 0.8 is a recommended value. Test and Control dataset will be automatically merged if X $\neq$ Y.
The example of test what samples from the Test Dataset are very similar to the whole Test Dataset (with strict filtration):

Listing 10: QC control to find samples from the Test Dataset that are similar to the Test Dataset

```
./qc.py /home/bedfile.bed ./results/X.xls ./results/X.xls id_of_task 0.99
```

10.2 Definitions

Definition 3. *The set of samples X is called irregular if its coverages shows high level of variance and statistical noise which makes the detection of CNVs in X spurious.*

Definition 4. *The sample x is called irregular with respect to control set of samples Y if its coverages are different from control set and that makes the CNV detection in x using the coverage structure of Y as the predictor spurious.*

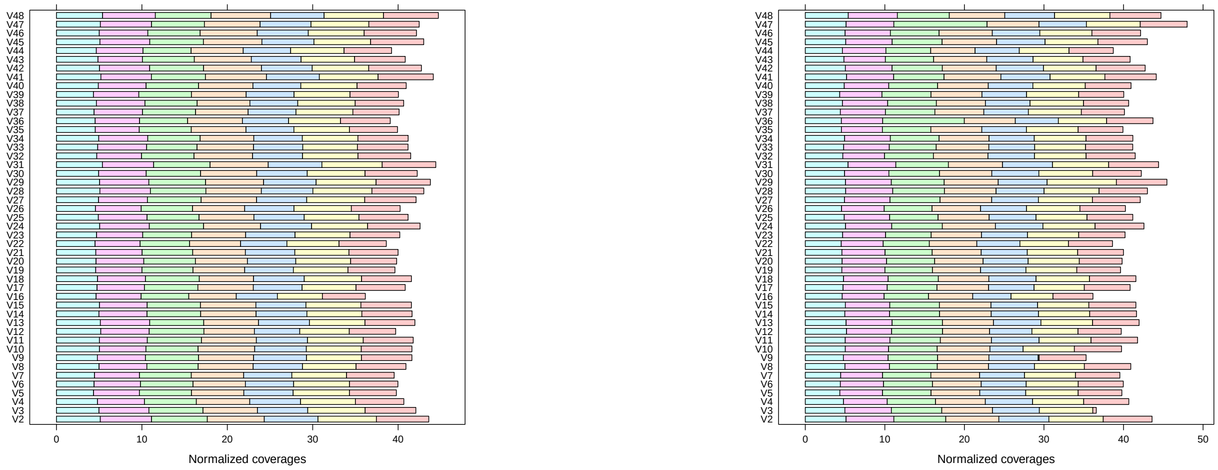
QC control algorithm detects

1. irregular sets of samples using *average robust variance*;

2. irregular samples using *robust regularized Mahalanobis distance* from the sample to control set of samples.

At first, informally speaking, we transform coverages of all amplicons as: how big is the coverage of a particular amplicon if we compare it to the average coverage of the gene in the sample. Then we calculate two types of distances: average robust variance and the distance of each sample to the control set in terms of variances (variation of Mahalanobis distance).

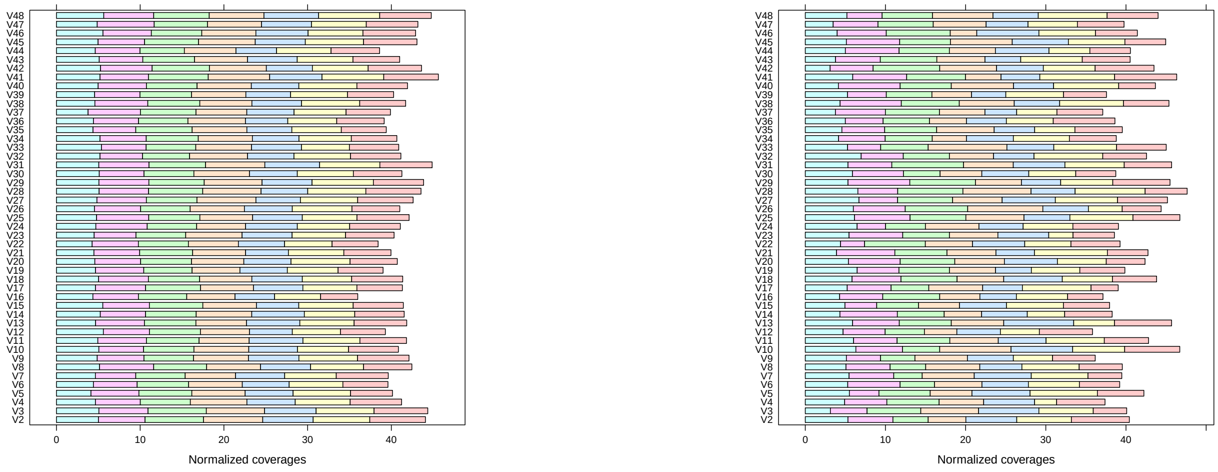
10.3 Average robust variance



A) Dataset with 8 regions affected with CNVs: $ARV = 0.08336864$

B) Dataset with 12 regions affected with CNVs: $ARV = 0.09010809$

Figure 3: .



C) Dataset with small amount of normal noise: $ARV = 0.2268741$

D) Dataset with large amount of normal noise: $ARV = 1.356552$

Figure 4: The coverages structure datasets with different Average Robust Variance.

The Average Robust Variance is calculated as

$$ARV = \frac{\sum_{j=1}^{N} \hat{\sigma}^2(\vec{c}(a_j))}{N},$$

where N is the number of regions, $\hat{\sigma}^2$ is the robust estimation of variance, $\vec{c}(a_j)$ means the vector of coverages of region a_j.

It is a *relative* measure. It is informative only if you know the ARV of Control Dataset so you can compare the new ARV with it.

This measure is useless if Test Dataset X is equal to Control Dataset Y because their Average Robust Variances will be equal.

Use Case:	**Check if the Test Dataset is noisy**
Quick description:	Measure if the Test Dataset is Irregular comparing to Control Dataset
Property:	• Average Robust Variance of Test dataset with respect to Test dataset: ARV_T • Average Robust Variance of Control dataset with respect to Control dataset: ARV_C

Scenario Name:

1. $ARV_T \approx ARV_C$: the overall level of noise in Control and Test datasets is approximately the same.
2. ARV_T is significantly bigger then ARV_C: the level of noise in the Test Dataset is high and you should try to perform the removing of irregular samples outside the Test Dataset.
3. $ARV_T < ARV_C$: the level of noise in the Test Dataset is low and may be it is better to use it as the new Control Dataset (after performing the analysis and removing noisy samples and samples with frequent CNVs (only CNV that is more frequent than 0.1 can spoil the results)).

The example is simple:

Listing 11: How to get the ARVs

```
cd /CONVector
./qc.py /home/bedfile.bed ./results/X.xls ./results/Y.xls id_of_task 0.9
nano /CONVector/qc_control_report.txt // look at the first two lines, you will find ARVs there
// if ARVt significantly larger than ARVc then - try to remove irregular samples
```

As you've understood, you can find two numbers, ARV_C and ARV_T, in the first lines of the file: `/CONVector/qc_control_log.txt`.

10.4 Removing irregular samples

Use Case:	**Remove irregular samples**
Quick description:	Remove all samples that are not similar to the whole dataset
Property:	• Percent of smallest values that will be taken into account: p, it is recommended to use $p = 0.99$ if the Test Dataset has large Average Robust Variance, and $p = 0.8$ otherwise.

Scenario Name:

1. A lot of samples from Test Dataset (almost all) passed the quality control: you can continue to work with the generated dataset.
2. A small amount of samples from Test Dataset passed the quality control: you should repeat the procedure of removing irregular samples using the generated file.

Let us assume that your Control Dataset has $ARV_C = 0.025$, but $ARV_T = 0.045$ which is significantly bigger. So we need to remove irregular samples and to try to make Test Dataset suitable for the analysis. We do not use Control Dataset for this procedure.

Listing 12: How to remove irregular samples

```
cd /CONVector
./qc.py /home/bedfile.bed ./results/X.xls ./results/X.xls id_of_task 0.9
./qc.py /home/bedfile.bed ./results/X_qc.xls ./results/X_qc.xls id_of_task 0.9
nano /qc_control_log.txt
```

Look at the ARV in the file `/CONVector/qc_control_log.txt`. Let us assume that 5 irregular samples were filtered out on the first step (second line of the Listing 11), and 3 samples were filtered on the second step (third line, Listing 11) and the number in the first line of `/CONVector/qc_control_log.txt` $ARV_C = 0.04$. It is not enough. We can repeat the procedure.

Listing 13: How to remove irregular samples iteratively

```
./qc.py /home/bedfile.bed ./results/X_qc_qc.xls ./results/X_qc_qc.xls id_of_task 0.9
nano /qc_control_log.txt
```

Horay - the number in the first line is $ARV_C = 0.027$, which is approximately 0.025! The regular dataset can be found in `/CONVector/results/X_qc_qc.xls`.
The list of samples that did not pass the quality control can be found in `/CONVector/qc_control_log.txt`.

10.5 Merging two datasets

Use Case:	**Form a Matrix of Coverages from Test and Control datasets**
Quick description:	Merge the regular samples (with respect to Control Dataset) from Test Dataset with Control Dataset
Property:	• Percent of smallest values that will be taken into account: p, I would recommend to use value $p \approx 0.9$.
Scenario Name:	

1. A lot of samples from Test Dataset passed the quality control: you can continue to work with the generated dataset.
2. Just a small number of samples from Test Dataset passed quality control: it is better to use another Control Dataset and repeat the procedure.

Listing 14: How merge Test Dataset X and Control Dataset Y

```
cd /CONVector
./qc.py /home/bedfile.bed ./results/X.xls ./results/Y.xls id_of_task 0.85
```

The resulting Matrix of Coverages can be found in `/CONVector/result/X_Y.xls`.
The list of samples that did not pass the quality control can be found in `/CONVector/qc_control_log.txt`.

11 Usage of CONVector

```
Source code available in:
    /CONVector/deletionAnalysis/*.java
Additional files (will be created automatically):
    /CONVector/tmpResultsCNV/*
```

Listing 15: An example of usage

```
java deletionsAnalysis/Main  -d ./result/MGC_25_apr_15_qc.xls \
-b /Users/german/Desktop/Parseq/DELETIONS/pipeline/2.bed \
-f outputMGC_25_apr_15 \
-mc 0.6 -mnm 5 -nne 4 -nod 4
-lb -2.425841 -ub 2.425841 \
-dist 1000000 -lcb 25000 -lca 50 \
-qc 4 -lss 0.01
```

You can vary parameters:

```
        -f    # name of the output file with the Matrix of CNVs (will have extension .xls)
        -d    # path to input file with the Matrix of Coverages (.xls)
        -b    # path to input .bed file
        -mc   # minimum correlation (default = 0.7)
        -mnm  # number of models with the highest correlations to take into account
              # (default = 5)
        -nne  # number of models in order to mark an amplicon as
              # uneffective (default = 4)
        -nod  # number of models for detection (default = 4)
        -lb   # lower bound (depending on confidence level of Student t-distribution
              # quantile and degrees of freedom (number of samples in dataset),
              # usually 1.7 - 3.5)
        -ub   # upper bound (depending on confidence level of Student t-distribution
              # quantile and degrees of freedom (number of samples in dataset),
              # usually 1.7 - 3.5)
        -dist # minimum distance between different amplicons
              # to compare their coveragers (base pairs, default = 1.000.000)
        -lca  # coverage of amplicons to be excluded from analysis (default = 50)
        -qc   # upper bound for quantiles for the 2nd stage, can be numbers from 0 to 9
        -lss  # learning sample sensitivity, can be a real number from 0.0001 to 0.1
              # approximately equal to the confidence level
        -lcb  # lower coverage bound, the tool will report about all samples that
              # will be covered lower than this bound
```

12 Usage of Finalizer

```
Source code available in:
    /CONVector/finalizer.py
```

The Finalizer script simply summarizes the Files with CNVs. It is necessary to use it if you have a lot of samples or a lot of regions.

Listing 16: Launch of Finalizer

```
cd /CONVector
./finalizer.py -i file_with_CNV.xls -o name_of_summary.xls
```

```
-i # input file (Matrix of CNVs, extension - .xls)
-o # output file (report, extension - .xls)
```

13 Troubleshooting

Please, do not try to run this tool on the empty datasets, other documents, etc.. I am really tired of dealing with all the possible cases, so - just be accurate and follow this instruction. I hope that you will not have any problems. If no...

Q: I have tried to run `pipeline.py`, but got an `-bash: ./pipeline.py ... : Permission denied` error!

A: Try to execute `chmod 755 pipeline.py`. The same can help for any other script.

14 Version history

Version 1.* - *Date of release.*

2. *(June, 2015)* Added the Average Robust Variance as the measure of irregularity. Added the automatic merging of Test and Control datasets after QC Control (if Test $\neq$ Control).

1. *(May, 2015)* The CNVs were determined not only as outliers, but also as the values that are more likely CNVs. The second algorithm worked with multi-dimensional random variables.

0. *(February, 2015)* The CNVs were determined as the outliers of robust models. The second algorithm worked only with one dimensional random variables.

15 References