CNVPanelizer: Reliable CNV detection in target sequencing applications

Oliveira, Cristiano
cristiano.oliveira@med.uni-heidelberg.de

Wolf, Thomas
thomas_Wolf71@gmx.de

October 24, 2023

Amplicon based targeted sequencing, over the last few years, has become a mainstay in the clinical use of next generation sequencing technologies. For the detection of somatic and germline SNPs this has been proven to be a highly robust methodology. One area of genomic analysis which is usually not covered by targeted sequencing, is the detection of copy number variations (CNVs). While a large number of available algorithms and software address the problem of CNV detection in whole genome or whole exome sequencing, there are no such established tools for amplicon based targeted sequencing. We introduced a novel algorithm for the reliable detection of CNVs from targeted sequencing.

1 Introduction

To assess if a region specific change in read counts correlates with the presence of CNVs, we implemented an algorithm that uses a subsampling strategy similar to Random Forest to reliable predict the presence of CNVs. We also introduce a novel method to correct for the background noise introduced by sequencing genes with a low number of amplicons. To make it available to the community we implemented the algorithm as an R package.

2 Using

This section provides an overview of the package functions.

2.1 Installing and Loading the package

The package is available through the Bioconductor repository and can be installed and loaded using the following R commands:

```
# To install from Bioconductor
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("CNVPanelizer")
```

```
# To load the package
library(CNVPanelizer)
```

2.2 Reading data

2.2.1 BED file

The BED file is required to obtain amplicon and gene name information associated with the panel.

```
# Bed file defining the amplicons
bedFilepath <- "/somePath/someFile.bed"

# The column number of the gene Names in the BED file.
amplColumnNumber <- 4

# Extract the information from a bed file
genomicRangesFromBed <- BedToGenomicRanges(bedFilepath,
                                           ampliconColumn = amplColumnNumber,
                                           split = "_")

metadataFromGenomicRanges <- elementMetadata(genomicRangesFromBed)
geneNames = metadataFromGenomicRanges["geneNames"][, 1]
ampliconNames = metadataFromGenomicRanges["ampliconNames"][, 1]
```

2.2.2 Selecting files

Two sets of data are required. The samples of interest and the set of reference bam files to compare against.

```
# Directory with the test data
sampleDirectory <- "/somePathToTestData"

# Directory with the reference data
referenceDirectory <- "/somePathToReferenceData"

# Vector with test filenames
sampleFilenames <- list.files(path = sampleDirectory,
                              pattern = ".bam$",
                              full.names = TRUE)

# Vector with reference filenames
referenceFilenames <- list.files(path = referenceDirectory,
                                 pattern = ".bam$",
                                 full.names = TRUE)
```

2.2.3 Counting reads

The reads were counted using a wrapper function around the **ExomeCopy** package from the Bioconductor project. All reads overlapping with the region of an amplicon were counted for this amplicon. Only reads with a mapping quality ≥ 20 were counted and the function allows to remove PCR Duplicates. For reads with the same start site, end site and chromosomal orientation only one is kept. PCR duplicates might cause a bias for the ratio between reference and the samples of interest. Thus this serves as an additional quality control step in the CNV detection pipeline. It is only recommended for Ion Torrent generated data. For Illumina data this step is not recommended.

```
# Should duplicated reads (same start, end site and strand) be removed
removePcrDuplicates <- FALSE # TRUE is only recommended for Ion Torrent data

# Read the Reference data set
referenceReadCounts <- ReadCountsFromBam(referenceFilenames,
                                         genomicRangesFromBed,
                                         sampleNames = referenceFilenames,
                                         ampliconNames = ampliconNames,
                                         removeDup = removePcrDuplicates)

# Read the sample of interest data set
sampleReadCounts <- ReadCountsFromBam(sampleFilenames,
                                      genomicRangesFromBed,
                                      sampleNames = sampleFilenames,
                                      ampliconNames = ampliconNames,
                                      removeDup = removePcrDuplicates)
```

2.2.4 Using Synthetic Data

We also make available synthetic data to test the functions. The following examples make use of two generated data sets, one for the reference and the other as a testing set

```
data(sampleReadCounts)
data(referenceReadCounts)

# Gene names should be same size as row columns
# For real data this is a vector of the genes associated
# with each row/amplicon. For example Gene1, Gene1, Gene2, Gene2, Gene3, ...
geneNames <- row.names(referenceReadCounts)

# Not defined for synthetic data
# For real data this gives a unique name to each amplicon.
# For example Gene1:Amplicon1, Gene1:Amplicon2, Gene2:Amplicon1,
# Gene2:Amplicon2, Gene3:Amplicon1 ...
ampliconNames <- NULL
```

2.3 Normalization

To account for sample and sequencing run specific variations the counts obtained for each sample were normalized using a wrapper function around the **tmm** normalization function from the **NOISeq** package.

```
normalizedReadCounts <- CombinedNormalizedCounts(sampleReadCounts,
                                                 referenceReadCounts,
                                                 ampliconNames = ampliconNames)

# After normalization data sets need to be splitted again to perform bootstrap
samplesNormalizedReadCounts = normalizedReadCounts["samples"][[1]]
referenceNormalizedReadCounts = normalizedReadCounts["reference"][[1]]
```

2.4 Bootstrap based CNV

This aproach is similar to the Random Forest method, which bootstraps the samples and subsamples the features. In our case features would be equivalent to amplicons. The subsampling procedure is repeated n times to generate a large set of randomized synthetic references $B = b_1, \ldots, b_n$ by selecting

with replacement (bootstrapping) from the set of reference samples. The ratio between the sample of interest and each randomized reference is calculated for each gene, using only a randomly selected subset of amplicons. To get an estimate of significance, the 95 percent confidence interval of the bootstrapping/subsampling ratio distribution was calculated. A significant change is considered as an amplification for a lower bound ratio > 1 and a deletion for an upper bound ratio < 1. The confidence interval was calculated using the 0.025 and 0.975 percent quantiles of the bootstrapping/subsampling ratio distribution.

```
# Number of bootstrap replicates to be used
replicates <- 10000
```

```
# Perform the bootstrap based analysis
bootList <- BootList(geneNames,
                     samplesNormalizedReadCounts,
                     referenceNormalizedReadCounts,
                     replicates = replicates)
```

2.5 Background Estimation

Not all genes have the same number of amplicons $|A_G|$ and it has been shown that sequencing genes with a higher number of amplicons yields better sensitivity and specifity when detecting putative copy number variations. Still genes sequenced with a low number of amplicons might still show significant changes in observed read counts. While normalization makes the comparison of read counts comparable between samples, genes with a small number of amplicons might still show a bias. To quantify the effect of a low number of amplicons on the calling of CNVs we introduced a background noise estimation procedure. Using the ratio between the mean reference and the sample used for calling we subsample for each unique number of amplicons. In the case of two amplicons we repeatedly sample two random amplicons from the set of all amplicons, and average the ratios. Amplicons that belong to genes showing significant copy number variations G_{sig} are not included in the subsampling pool. Each amplicon is weighted according to the number of amplicons the respective gene has $w_A = \frac{1}{|A_g|}$. Thus the probablity of sampling from a gene is the same regardless of the number amplicons. For each number of amplicons a background noise distribution is estimated. The reported background is defined by the lower noise $meanNoise + qnorm(0.025) * standardDeviationNoise$ and the upper noise $meanNoise + qnorm(0.975) * standardDeviationNoise$ of the respective distribution. This defines the 95 percent confidence interval. The significance level can also be passed as parameter. Less conservative detection can be achieved by setting the significanceLevel = 0.1. To calculate the mean and standard deviation (sd) of ratios the log ratios were used. If setting robust = TRUE, median is used instead of mean and mad (median absolute deviation) replaces sd. The upper bound ratio has to be below the lower noise (deletion) or the lower bound ratio above the upper noise (amplification) to be considered reliable.

```
# Estimate the background noise left after normalization
backgroundNoise <- Background(geneNames,
                              samplesNormalizedReadCounts,
                              referenceNormalizedReadCounts,
                              bootList,
                              replicates = replicates,
                              significanceLevel = 0.1,
                              robust = TRUE)
```

2.6 Results

To analyse the results we provide two outputs. A plot which shows the detected variations, and a report table with more detailed information about those variations.

2.6.1 Report

The report describes the bootstrap distribution for each gene. An example can be found in figure 1. The final report is genewise, and is based on the bootstrapping.

```
# Build report tables
reportTables <- ReportTables(geneNames,
                             samplesNormalizedReadCounts,
                             referenceNormalizedReadCounts,
                             bootList,
                             backgroundNoise)
```

At the figure 1 we can see an example of the report table for a single sample.

	LowerBoundBoot	MeanBoot	UpperBoundBoot	LowerNoise	MeanNoise	UpperNoise	Signif.	AboveNoise	Amplicons	PutativeStatus	ReliableStatus
Gene_01	0.18	0.69	2.73	0.53	0.85	1.37	FALSE	FALSE	4	Normal	Normal
Gene_02	0.46	0.87	1.7	0.56	0.82	1.19	FALSE	FALSE	7	Normal	Normal
Gene_03	0.6	0.98	1.59	0.53	0.85	1.37	FALSE	FALSE	4	Normal	Normal
Gene_04	0.22	0.7	2	0.58	0.82	1.16	FALSE	FALSE	8	Normal	Normal
Gene_05	0.21	0.49	1.09	0.59	0.82	1.14	FALSE	FALSE	9	Normal	Normal
Gene_06	0.87	1.32	2.08	0.48	0.91	1.70	FALSE	FALSE	2	Normal	Normal
Gene_07	0.37	0.6	1.04	0.55	0.84	1.29	FALSE	FALSE	5	Normal	Normal
Gene_08	0.11	0.5	1.46	0.58	0.82	1.16	FALSE	FALSE	8	Normal	Normal
Gene_09	0.6	1.02	1.69	0.55	0.84	1.29	FALSE	FALSE	5	Normal	Normal
Gene_10	0.27	0.66	1.34	0.55	0.84	1.29	FALSE	FALSE	5	Normal	Normal
Gene_11	0.36	0.79	1.73	0.59	0.82	1.14	FALSE	FALSE	9	Normal	Normal
Gene_12	1	1.4	2	0.55	0.84	1.29	FALSE	FALSE	5	Normal	Normal
Gene_13	0.1	0.57	1.61	0.56	0.83	1.24	FALSE	FALSE	6	Normal	Normal
Gene_14	0.06	0.28	0.98	0.55	0.84	1.29	TRUE	FALSE	5	Deletion	Normal
Gene_15	0.26	0.47	0.87	0.48	0.91	1.70	TRUE	FALSE	2	Deletion	Normal
Gene_16	0.61	1.24	2.21	0.58	0.82	1.16	FALSE	FALSE	8	Normal	Normal
Gene_17	2	2.35	2.87	0.51	0.87	1.48	TRUE	TRUE	3	Amplification	Amplification
Gene_18	0.33	0.53	0.9	0.48	0.91	1.70	TRUE	FALSE	2	Deletion	Normal
Gene_19	0.29	0.54	1.05	0.53	0.85	1.37	FALSE	FALSE	4	Normal	Normal
Gene_20	0.64	1.15	1.9	0.59	0.82	1.14	FALSE	FALSE	9	Normal	Normal

Figure 1: Sample report table.

LowerBoundBoot | MeanBoot | UpperBoundBoot	The confidence interval of the Bootstrap distribution
Lower | Mean | Upper Noise	The Lower/Mean/Upper background noise bounds
Signif.	Boolean value representing if read counts differences are Significant
AboveNoise	Boolean value representing if LowerBoundBoot is above UpperNoise (UpperBoundBoot below LowerNoise)
Amplicons	Number of Amplicons associated with the gene
PutativeStatus	The detected level of change
Passed	if both Signif. and AboveNoise are TRUE then 2, if only one is TRUE then 1 and if both are FALSE then 0

Table 1: Report table Column Description

2.6.2 Plots

The generated plots (one per test sample) show the bootstrap distribution for each gene. The function **PlotBootstrapDistributions** generates a list of plots for all test samples (A plot per sample). At the figure 2 we can see an example of the plot for a single sample.

```
PlotBootstrapDistributions(bootList, reportTables)
```

2.7 Shiny App

To allow a more user friendly interaction and quick testing of CNVPanelizer, a shiny app is made available through the following command:

```
# default port is 8100
RunCNVPanelizerShiny(port = 8080)
```

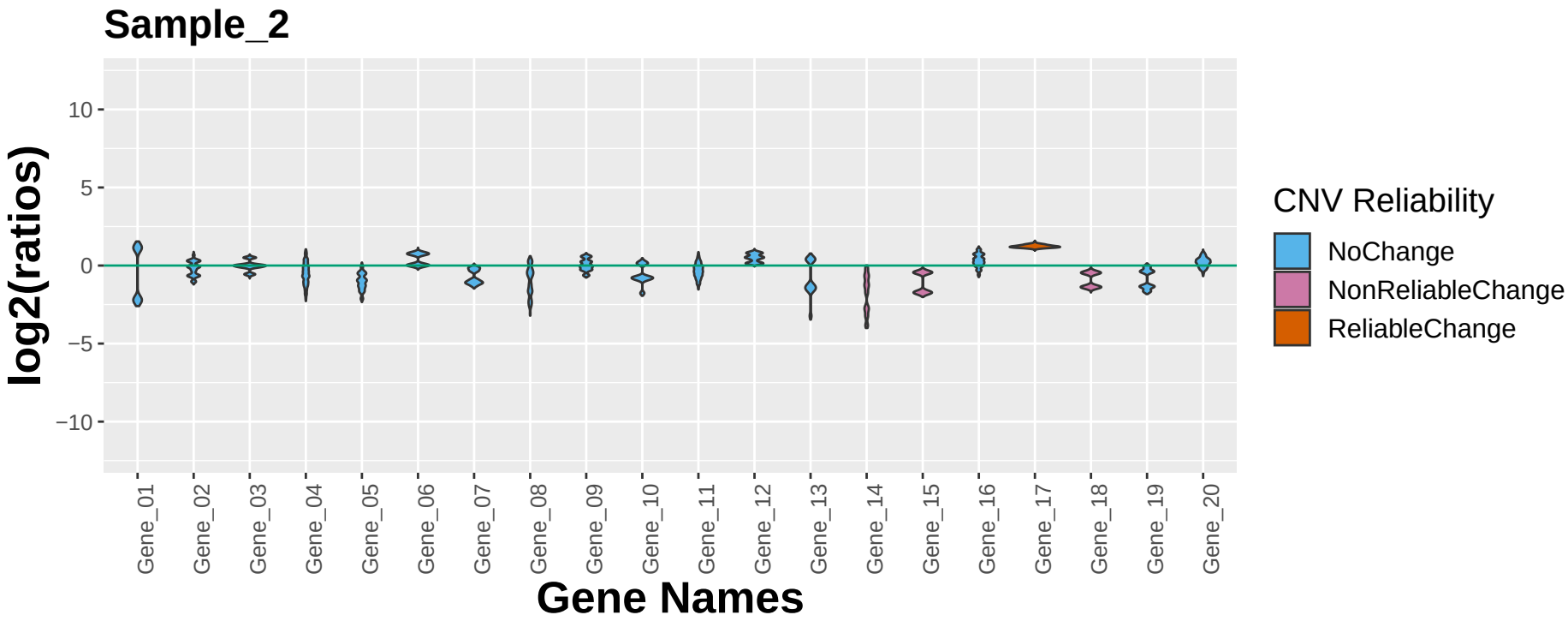

Figure 2: Plot for a single test sample

No Change	There is no change in the read count
Non Reliable Change	The change is significant but below the noise level for the number of amplicons associated with this gene
Reliable Change	The change is above the noise level and significant

Table 2: Description of the CNV detection levels