Software Manual

Institute of Bioinformatics, Johannes Kepler University Linz

panelcn.MOPS - CNV detection tool for targeted NGS panel sequencing

Verena Haunschmid and Gundula Povysil

Institute of Bioinformatics, Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
povysil@bioinf.jku.at

Version 0.1.3, August 30, 2016

Contents

1 Introduction

The `panelcn.mops` package is based on the `cn.mops` package and allows to detect copy number variations (CNVs) from targeted NGS panel data. Please visit `http://www.bioinf.jku.at/software/panelcnmops/index.html` for additional information.

2 Getting started and quick start

To load the package, enter the following in your R session:

```
library(panelcn.mops)
```

The whole pipeline will only take a few steps, if BAM files are available (for read count matrices directly go to step 2):

1. Getting count windows from the BED file (also see Section 3).

```
bed <- "Genes_part.bed"
countWindows <- getWindows(bed)
```

2. Getting read counts from BAM file (also see Section 3).

```
testfile <- "SAMPLE1.bam"
test <- countBamListInGRanges(countWindows = countWindows,
                              bam.files = testfile, read.width = 150)
```

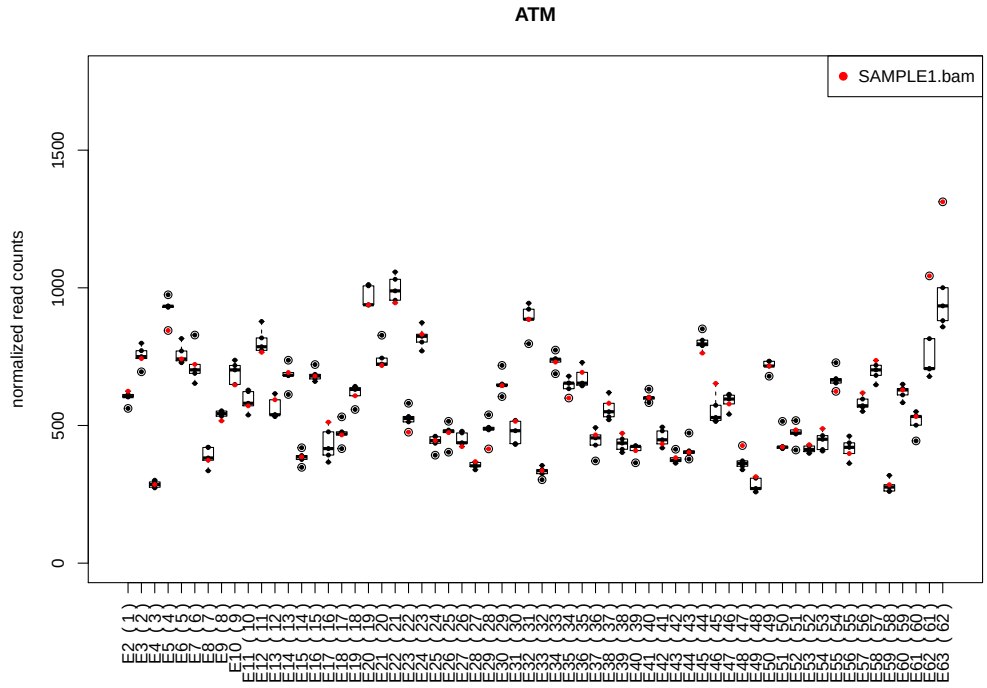
3. Running the algorithm (also see Section 4).

```
data(panelcn.mops)

selectedGenes <- c("ATM")

XandCB <- test
XandCB@elementMetadata <- cbind(XandCB@elementMetadata,
                                control@elementMetadata)

resultlist <- runPanelcnMops(XandCB,
                             testiv = 1:ncol(test@elementMetadata),
                             countWindows = countWindows,
                             selectedGenes = selectedGenes)
```

4. Visualization of the detected CNV regions. For more information about the result objects and visualization see Section 5 and Section 6.

```
sampleNames <- colnames(test@elementMetadata)
resulttable <- createResultTable(result = resultlist, XandCB = XandCB,
                                 countWindows = countWindows,
                                 selectedGenes = selectedGenes,
                                 sampleNames = sampleNames)

## Calculating results for sample(s) SAMPLE1.bam
## Building table...
## Finished

(tail(resulttable[[1]]))

##         Sample Chr Gene                              Exon
## 57 SAMPLE1.bam  11  ATM ATM.E58.chr11.108216439.108216666
## 58 SAMPLE1.bam  11  ATM ATM.E59.chr11.108217975.108218123
## 59 SAMPLE1.bam  11  ATM ATM.E60.chr11.108224462.108224638
## 60 SAMPLE1.bam  11  ATM ATM.E61.chr11.108225507.108225632
## 61 SAMPLE1.bam  11  ATM ATM.E62.chr11.108235778.108235976
## 62 SAMPLE1.bam  11  ATM ATM.E63.chr11.108236021.108236266
##        Start       End   RC  medRC RC.norm medRC.norm lowQual
## 57 108216439 108216666  969  768.5     736        702
## 58 108217975 108218123  375  301.0     285        276
## 59 108224462 108224638  828  692.0     629        629
## 60 108225507 108225632  701  550.0     533        533
## 61 108235778 108235976 1373  793.0    1043        707
## 62 108236021 108236266 1727 1018.5    1312        934
##     CN
## 57 CN2
## 58 CN2
## 59 CN2
## 60 CN2
## 61 CN3
## 62 CN3
```

```
plotBoxplot(result = resultlist[[1]], sampleName = sampleNames[1],
            countWindows = countWindows,
            selectedGenes = selectedGenes, showGene = 1)
```



3 Input

The most widely used file format for aligned short reads is the Sequence Alignment Map (SAM) format or in the compressed form the Binary Alignment Map (BAM). `panelcn.mops` modifies the read count function `countBamInGRanges` from the R package `exomeCopy` to extract read counts for a list of BAM files. The result object of the function can directly be used as input for `panelcn.mops`.

The first step is to extract all ROIs that define the count windows from a BED file with the function `getWindows`.

```
bed <- list.files(system.file("extdata", package = "panelcn.mops"),
                          pattern = ".bed$", full.names = TRUE)
countWindows <- getWindows(bed)

## naming without chr prefix chosen, but BED contains chr -> removing chr
```

The BED file should have the following structure:

```
## chr1 45794947 45795140 MUTYH.E16.chr1.45794947.45795140
## chr1 45796157 45796260 MUTYH.E15.chr1.45796157.45796260
## chr1 45796823 45797037 MUTYH.E14.chr1.45796823.45797037
## chr1 45797061 45797259 MUTYH.E13.chr1.45797061.45797259
```

```
## chr1 45797302 45797552 MUTYH.E12.chr1.45797302.45797552
## chr1 45797664 45797789 MUTYH.E11.chr1.45797664.45797789
```

While the first 3 columns list chromosome name, start and end position, the fourth column needs to start with the gene name. Additional information in the fourth column needs to be seperated with a dot and may include the exon number and a combination of the first 3 columns. By default the "chr" prefix of the chromosome name is removed if present. This can be changed by setting the `chr` parameter to TRUE. If a mismatch of chromosome naming between the `countWindows` object and the BAM files is detected, the naming convention of the BAM file is chosen.

In the second step RCs are generated from the BAM files. The read.width parameter reflects the typical length of the reads that should be counted.

```
testbam <- list.files(system.file("extdata", package = "panelcn.mops"),
                      pattern = ".bam$", full.names = TRUE)
test <- countBamListInGRanges(countWindows = countWindows,
                              bam.files = testbam, read.width = 150)

## Processing SAMPLE1.bam ...  1 / 1
## finished processing samples
```

In `test` you have now stored the genomic segments (left of the |'s) and the read counts (right of the |'s):

```
(test)

## GRanges object with 370 ranges and 1 metadata column:
##         seqnames               ranges strand   | SAMPLE1.bam
##            <Rle>            <IRanges>  <Rle>   |   <integer>
##     [1]        1 [45794947, 45795140]      *   |         637
##     [2]        1 [45796157, 45796260]      *   |         384
##     [3]        1 [45796823, 45797037]      *   |         414
##     [4]        1 [45797061, 45797259]      *   |         361
##     [5]        1 [45797302, 45797552]      *   |         482
##     ...      ...                  ...    ... ...         ...
##   [366]        2 [48032018, 48032197]      *   |         618
##   [367]        2 [48032726, 48032877]      *   |         206
##   [368]        2 [48033312, 48033528]      *   |         572
##   [369]        2 [48033560, 48033821]      *   |         735
##   [370]        2 [48033887, 48034030]      *   |         678
##   -------
##   seqinfo: 11 sequences from an unspecified genome; no seqlengths
```

4 runPanelcnMops

The actual copy number analysis is done with the function `runPanelcnMops`. The function requires a `GRanges` object of the RCs of test and control samples as well as the `countWindows` object used to extract these RCs. Optional parameters include a vector that indicates which samples to regard as test samples (default = c(1)), a vector of the names of the genes of interest (by default all genes are of interest), parameters for normalizing the RCs, a vector of expected fold changes for the copy number classes and a minimal median RC over all samples to exclude low coverage ROIs.

```
data(panelcn.mops)

XandCB <- test
XandCB@elementMetadata <- cbind(XandCB@elementMetadata,
                                control@elementMetadata)

resultlist <- runPanelcnMops(XandCB, countWindows = countWindows)
```

5 Results

The function `runPanelcnMops` resturns a list of objects of the S4 class CNVDetectionResult, one CNVDetectionResult object per test sample. The structure of the CNVDetectionResult object can be listed by calling

```
(str(resultlist[[1]]))
```

To get detailed information on which data are stored in such objects, enter

```
help(CNVDetectionResult)
```

The CNVs per individual are stored in the slot `integerCopyNumber`:

```
integerCopyNumber(resultlist[[1]])[1:5]

## GRanges object with 5 ranges and 5 metadata columns:
##       seqnames              ranges strand | SAMPLE1.bam
##          <Rle>           <IRanges>  <Rle> |    <factor>
##   [1]        1 [45794947, 45795140]      * |         CN2
##   [2]        1 [45796157, 45796260]      * |         CN2
##   [3]        1 [45796823, 45797037]      * |         CN2
##   [4]        1 [45797061, 45797259]      * |         CN2
##   [5]        1 [45797302, 45797552]      * |         CN2
##       SAMPLE4.bam SAMPLE3.bam SAMPLE6.bam SAMPLE2.bam
```

```
##          <factor>    <factor>    <factor>    <factor>
##   [1]         CN2         CN2         CN2         CN2
##   [2]         CN2         CN2         CN2         CN2
##   [3]         CN2         CN2         CN2         CN2
##   [4]         CN2         CN2         CN2         CN2
##   [5]         CN2         CN2         CN2         CN2
##   -------
##   seqinfo: 11 sequences from an unspecified genome; no seqlengths
```

The function `createResultTable` summerizes all relevant information for user selected genes of interest in a list of tables with one table per test sample:

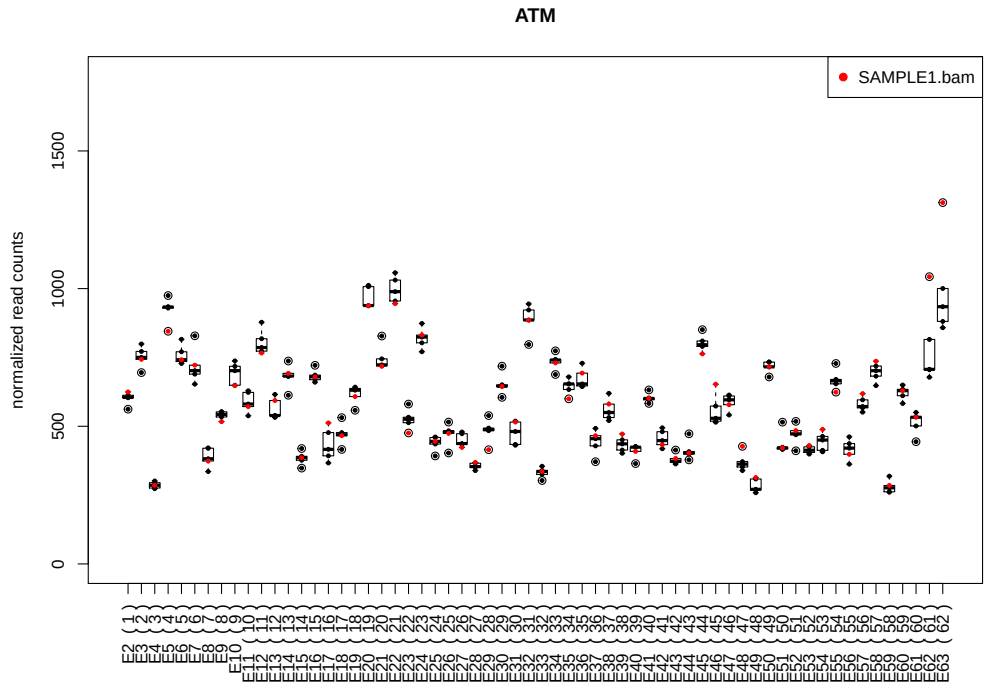
```
sampleNames <- colnames(test@elementMetadata)
resulttable <- createResultTable(result = resultlist, XandCB = XandCB,
                                 countWindows = countWindows,
                                 selectedGenes = selectedGenes,
                                 sampleNames = sampleNames)

## Calculating results for sample(s) SAMPLE1.bam

## Building table...

## Finished

(tail(resulttable[[1]]))

##         Sample Chr Gene                              Exon
## 57 SAMPLE1.bam  11  ATM ATM.E58.chr11.108216439.108216666
## 58 SAMPLE1.bam  11  ATM ATM.E59.chr11.108217975.108218123
## 59 SAMPLE1.bam  11  ATM ATM.E60.chr11.108224462.108224638
## 60 SAMPLE1.bam  11  ATM ATM.E61.chr11.108225507.108225632
## 61 SAMPLE1.bam  11  ATM ATM.E62.chr11.108235778.108235976
## 62 SAMPLE1.bam  11  ATM ATM.E63.chr11.108236021.108236266
##        Start       End   RC  medRC RC.norm medRC.norm lowQual
## 57 108216439 108216666  969  768.5     736        702
## 58 108217975 108218123  375  301.0     285        276
## 59 108224462 108224638  828  692.0     629        629
## 60 108225507 108225632  701  550.0     533        533
## 61 108235778 108235976 1373  793.0    1043        707
## 62 108236021 108236266 1727 1018.5    1312        934
##     CN
## 57 CN2
## 58 CN2
## 59 CN2
## 60 CN2
## 61 CN3
## 62 CN3
```

The table contains one line per Region Of Interest (ROI) with information about the RCs of the test sample ("RC"), the median RCs of all control samples ("medRC"), the normalized RCs of the test sample ("RC.norm"), the median of the normalized RCs of all control samples ("medRC.norm"), as well as the estimated CN ("CN"). Additionally, in the column "lowQual" low quality ROIs are flagged.

6 Visualization of results

`panelcn.mops` contains a plotting function that visualizes the normalized RCs of the samples analyzed as boxplots:

```
plotBoxplot(result = resultlist[[1]], sampleName = sampleNames[1],
            countWindows = countWindows,
            selectedGenes = selectedGenes, showGene = 1)
```



The function expects a single CNVDetectionResult object as input together with the name of the test sample, the countWindows used, as well as a vector with the names of the genes of interest and an integer specifying which of the genes of interest to plot.

7 Adjusting sensitivity and specificity

The default parameters of the `panelcn.mops` algorithm were optimized on a data set of targeted NGS panel data with the aim of detecting CNVs ranging in size from part of a ROI to whole genes. However, you might want to adjust sensitivity and specificity to your specific needs.

The parameter that influences sensitivity and specificity the most is `I`, the vector of expected fold changes of the copy number classes. The default for `panelcn.mops` c(0.025, 0.57, 1, 1.46, 2), leads to a higher sensitivity compared to the default of `cn.mops` which is c(0.025, 0.5, 1, 1.5, 2). Increasing the values for CN0 and CN1 further and decreasing the values for CN3 and CN4 may help to improve the sensitivity, a change in the other direction may increase the specificity.

Additional parameters that can be tuned to improve the results are the different normalization parameters: `normType`, `sizeFactor`, `qu`, `quSizeFactor`, and `norm`.

8 How to cite this package

If you use this package for research that is published later, you are kindly asked to cite it as follows: (Klambauer *et al.*, 2012).

To obtain BibTEX entries of the reference, you can enter the following into your R session:

```
toBibtex(citation("cn.mops"))
```

References

Klambauer, G., Schwarzbauer, K., Mitterecker, A., Mayr, A., Clevert, D.-A., Bodenhofer, U., and Hochreiter, S. (2012). cn.MOPS: Mixture of Poissons for Discovering Copy Number Variations in Next Generation Sequencing Data with a Low False Discovery Rate. *Nucleic Acids Research*, **40**(9), e69.