

# Package ‘CNVPanelizer’

November 20, 2024

**Type** Package

**Title** Reliable CNV detection in targeted sequencing applications

**Version** 1.38.0

**Date** 2023-03-28

**Description** A method that allows for the use of a collection of non-matched normal tissue samples. Our approach uses a non-parametric bootstrap subsampling of the available reference samples to estimate the distribution of read counts from targeted sequencing. As inspired by random forest, this is combined with a procedure that subsamples the amplicons associated with each of the targeted genes. The obtained information allows us to reliably classify the copy number aberrations on the gene level.

**Depends** R ( $\geq 3.2.0$ ), GenomicRanges

**Suggests** knitr, RUnit

**Imports** BiocGenerics, S4Vectors, grDevices, stats, utils, NOISeq, IRanges, Rsamtools, foreach, ggplot2, plyr, GenomeInfoDb, gplots, reshape2, stringr, testthat, graphics, methods, shiny, shinyFiles, shinyjs, grid, openxlsx

**License** GPL-3

**LazyData** false

**biocViews** Classification, Sequencing, Normalization, CopyNumberVariation, Coverage

**VignetteBuilder** knitr

**NeedsCompilation** no

**git\_url** <https://git.bioconductor.org/packages/CNVPanelizer>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** bff6031

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-11-19

**Author** Cristiano Oliveira [aut],  
Thomas Wolf [aut, cre],  
Albrecht Stenzinger [ctb],  
Volker Endris [ctb],  
Nicole Pfarr [ctb],

Benedikt Brors [ths],  
 Wilko Weichert [ths]

**Maintainer** Thomas Wolf <thomas\_wolf71@gmx.de>

## Contents

CNVPanelizer-package . . . . .	2
Background . . . . .	3
BedToGenomicRanges . . . . .	4
BootList . . . . .	5
CNVPanelizerFromReadCounts . . . . .	6
CNVPanelizerFromReadCountsHELPER . . . . .	7
CollectColumnFromAllReportTables . . . . .	9
CombinedNormalizedCounts . . . . .	9
IndexMultipleBams . . . . .	10
NormalizeCounts . . . . .	11
PlotBootstrapDistributions . . . . .	12
ReadCountsFromBam . . . . .	13
ReadXLSXToList . . . . .	14
referenceReadCounts . . . . .	15
ReportTables . . . . .	15
RunCNVPanelizerShiny . . . . .	16
sampleReadCounts . . . . .	17
SelectReferenceSetByInterquartileRange . . . . .	18
SelectReferenceSetByKmeans . . . . .	18
SelectReferenceSetByPercentil . . . . .	19
SelectReferenceSetFromReadCounts . . . . .	20
StatusHeatmap . . . . .	21
WriteListToXLSX . . . . .	22
<b>Index</b>	<b>23</b>

---

CNVPanelizer-package *Reliable CNV detection in targeted sequencing applications*

---

## Description

This package implements an algorithm that uses a collection of non-matched normal tissue samples as a reference set to detect CNV aberrations in data generated from amplicon based targeted sequencing.

## Details

Our approach uses a non-parametric bootstrap subsampling of the available reference samples, to estimate the distribution

For a complete list of functions, use `library(help = "CNVPanelizer")`.

Package: CNVPanelizer  
 Type: Package  
 License: GPL-3

**Author(s)**

Thomas Wolf <thomas\_wolf71f@gmx.de>  
Cristiano Oliveira <cristiano.oliveira@med.uni-heidelberg.de>

---

Background

*Background*

---

**Description**

Makes use of a subsampling approach to estimate the background noise when sequencing a gene with a specific number of amplicons. The 95 percent confidence interval is returned for each unique number of amplicons in the experiment.

**Usage**

```
Background(geneNames,  
           samplesNormalizedReadCounts,  
           referenceNormalizedReadCounts,  
           bootList,  
           replicates = 1000,  
           significanceLevel = 0.05,  
           robust = FALSE)
```

**Arguments**

geneNames	A vector of gene names, with one entry for each sequenced amplicon.
samplesNormalizedReadCounts	A matrix with the normalized read counts of the samples of interest
referenceNormalizedReadCounts	A matrix with the normalized reference read counts
bootList	A list as returned by BootList
replicates	an integer number of how many replicates should be performed
significanceLevel	The significance level for the calculated confidence interval
robust	If set to true the confidence interval is calculated replacing mean with median and sd with mad.

**Value**

Returns a list of data frames. One data frame for each sample of interest. The data frames report the 95 percent confidence interval of the background noise for each number of amplicons and sample combination.

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```

data(sampleReadCounts)
data(referenceReadCounts)
## Gene names should be same size as row columns
geneNames <- row.names(referenceReadCounts)

ampliconNames <- NULL

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]]

#Values above 10000 should be used
replicates <- 10

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

background <- Background(geneNames,
                        samplesNormalizedReadCounts,
                        referenceNormalizedReadCounts,
                        bootList,
                        replicates = replicates,
                        significanceLevel = 0.1)

```

---

BedToGenomicRanges      *BedToGenomicRanges*

---

**Description**

It generates a GenomicRanges object from a bed file. Needs to be passed the correct number of the gene name column. If the strings contain more information than just the gene name, a splitting character (split) has to be defined. I.e GeneName1;Amplicon2

**Usage**

```

BedToGenomicRanges(panelBedFilepath,
                  ampliconColumn,
                  split,
                  doReduce,
                  rangeExtend,
                  dropChromossomes,
                  skip)

```

**Arguments**

panelBedFilepath	Filepath of the bed file.
ampliconColumn	Number of the column that identifies the gene name in the bed file passed through panelBedFilepath.
split	The character used as separator in the ampliconColumn. It is ";" by default.
doReduce	Should overlapping ranges be merged.
rangeExtend	Should the defined ranges be extended left and right by the given value. Affects the merging of overlapping regions and also read counting.
dropChromossomes	Drop chromossomes.
skip	How many lines should be skipped from the top of the bed file. The function assumes a bed file with column names. Thus default is skip = 1.

**Value**

A GenomicRanges object containing information about the amplicons described in the bed file.

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
bedFilepath <- file.path("someFile.bed")
ampliconColumn <- 4
genomicRangesFromBed <- BedToGenomicRanges(bedFilepath, ampliconColumn)
```

---

BootList

*BootList*

---

**Description**

Performs a hybrid bootstrapping subsampling procedure similar to random forest. It bootstraps the reference samples and subsamples the amplicons associated with each gene. Returns a distribution of sample/reference ratios for each gene and sample of interest combination.

**Usage**

```
BootList(geneNames, sampleMatrix, refmat, replicates)
```

**Arguments**

geneNames	A vector of gene names, with one entry for each sequenced amplicon.
sampleMatrix	A vector or matrix of the read counts from the sample of interest. In the case of a matrix columns represent samples and rows amplicons.
refmat	A matrix of the read counts obtained from the reference samples. Columns represent reference samples and rows amplicons.
replicates	How many bootstrap replicates should be performed.

**Value**

Returns a list of numeric matrices: For each matrix a row represent a gene while each column represents a bootstrapping/subsampling iteration.

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
data(sampleReadCounts)
data(referenceReadCounts)
## Gene names should be same size as row columns
geneNames <- row.names(referenceReadCounts)

ampliconNames <- NULL

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]]

# Should be used values above 10000
replicates <- 10

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

---

CNVPanelizerFromReadCounts

*CNVPanelizerFromReadCounts*

---

**Description**

Performs the workflow analysis with CNVPanelizer from the read counts and splitting the batch of samples analyzed

**Usage**

```
CNVPanelizerFromReadCounts(sampleReadCounts,
                            referenceReadCounts,
                            genomicRangesFromBed,
                            numberOfBootstrapReplicates = 10000,
                            normalizationMethod = "tmm",
                            robust = TRUE,
                            backgroundSignificanceLevel = 0.05,
                            outputDir = file.path(getwd(), "CNVPanelizer"))
```

**Arguments**

sampleReadCounts  
samples read counts matrix

referenceReadCounts  
reference read counts matrix

genomicRangesFromBed  
genomic ranges from bed

numberOfBootstrapReplicates  
number of bootstrap replicates

normalizationMethod  
Normalization method ("tmm" or "tss")

robust  
if TRUE, the median is used instead of mean

backgroundSignificanceLevel  
The background Significance Level

outputDir  
Output directory

**Value**

Returns a list with the results of each samples analyzed

**Author(s)**

Cristiano Oliveira

**Examples**

```
CNVPanelizerFromReadCounts(sampleReadCounts,  
                             referenceReadCounts,  
                             genomicRangesFromBed,  
                             numberOfBootstrapReplicates = 10000,  
                             normalizationMethod = "tmm",  
                             robust = TRUE,  
                             backgroundSignificanceLevel = 0.05,  
                             outputDir = file.path(getwd(), "CNVPanelizer"))
```

---

CNVPanelizerFromReadCountsHELPER

*CNVPanelizerFromReadCountsHELPER*

---

**Description**

Helper to performs the workflow analysis with CNVPanelizer from the read counts and splitting the batch of samples analyzed

**Usage**

```
CNVPanelizerFromReadCountsHELPER(sampleReadCounts,  
                                  referenceReadCounts,  
                                  genomicRangesFromBed,  
                                  numberOfBootstrapReplicates = 10000,  
                                  normalizationMethod = "tmm",  
                                  robust = TRUE,  
                                  backgroundSignificanceLevel = 0.05,  
                                  outputDir = file.path(getwd(), "CNVPanelizer"),  
                                  splitSize = 5)
```

**Arguments**

sampleReadCounts	samples read counts matrix
referenceReadCounts	reference read counts matrix
genomicRangesFromBed	genomic ranges from bed
numberOfBootstrapReplicates	number of bootstrap replicates
normalizationMethod	Normalization method ("tmm" or "tss")
robust	if TRUE, the median is used instead of mean
backgroundSignificanceLevel	The background Significance Level
outputDir	Output directory
splitSize	Split size of the batches analyzed

**Value**

Returns a list with the results of each samples analyzed

**Author(s)**

Cristiano Oliveira

**Examples**

```
CNVPanelizerFromReadCountsHELPER(sampleReadCounts,  
                                  referenceReadCounts,  
                                  genomicRangesFromBed,  
                                  numberOfBootstrapReplicates = 10000,  
                                  normalizationMethod = "tmm",  
                                  robust = TRUE,  
                                  backgroundSignificanceLevel = 0.05,  
                                  outputDir = file.path(getwd(), "CNVPanelizer"),  
                                  splitSize = 5)
```



---

CollectColumnFromAllReportTables  
*CollectColumnFromAllReportTables*

---

**Description**

Collect a single column from all report tables at the list

**Usage**

```
CollectColumnFromAllReportTables(reportTables, columnName)
```

**Arguments**

reportTables	A list of report tables
columnName	The column name

**Value**

Returns a data frame with where the columns were collected from the entire list of report tables

**Author(s)**

Cristiano Oliveira

**Examples**

```
CollectColumnFromAllReportTables(reportTables, columnName)
```

---

CombinedNormalizedCounts  
*CombinedNormalizedCounts*

---

**Description**

This function makes use of Total sum scaling or NOISeq::tmm to normalize the read counts of all samples and references to the same median read count

**Usage**

```
CombinedNormalizedCounts(sampleCounts,  
                          referenceCounts,  
                          method,  
                          ampliconNames = NULL)
```

**Arguments**

sampleCounts Matrix or vector with sample read counts (rows: amplicons, columns: samples)  
 referenceCounts Matrix with reference read counts (rows: amplicons, columns: samples)  
 method either "tmm" (trimmed mean of m values) or "tss"(total sum scaling)  
 ampliconNames A vector with amplicon defining names for the reference and sample matrices

**Value**

A list object with two matrices

samples The samples matrix normalized  
 reference The reference matrix normalized

**Author(s)**

Cristiano Oliveira, Thomas Wolf

**Examples**

```
data(sampleReadCounts)
data(referenceReadCounts)

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

---

IndexMultipleBams      *IndexMultipleBams*

---

**Description**

Index a list of bam files if there is no index exists for the file entries in the list.

**Usage**

```
IndexMultipleBams(bams, index_type = ".bam.bai")
```

**Arguments**

bams A character vector of bam files to be indexed  
 index\_type The index file type extension

**Value**

Not returning any value

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
files = c("file1.bam", "file2.bam", "file3.bam")
IndexMultipleBams(bams = files)
```

---

NormalizeCounts	<i>NormalizeCounts</i>
-----------------	------------------------

---

**Description**

This function normalize counts use of Total sum scaling or NOISeq::tmm to normalize the read counts

**Usage**

```
NormalizeCounts(allCounts,
                method)
```

**Arguments**

allCounts	Matrix or vector with sample read counts (rows: amplicons, columns: samples)
method	either "tmm" (trimmed mean of m values) or "tss"(total sum scaling)

**Value**

A matrice	
samples	The samples matrix normalized

**Author(s)**

Cristiano Oliveira, Thomas Wolf

**Examples**

```
data(sampleReadCounts)
normalizedReadCounts <- NormalizeCounts(sampleReadCounts)
```

---

PlotBootstrapDistributions

*PlotBootstrapDistributions*

---

## Description

Plots the generated bootstrap distribution as violin plots. Genes showing significant values are marked in a different color.

## Usage

```
PlotBootstrapDistributions(bootList,  
                           reportTables,  
                           outputFolder = getwd(),  
                           sampleNames = NULL,  
                           save = FALSE,  
                           scale = 10)
```

## Arguments

bootList	List of bootstrapped read counts for each sample data
reportTables	List of report tables for each sample data
outputFolder	Path to the folder where the data plots will be created
sampleNames	List with sample names
save	Boolean to save the plots to the output folder
scale	Numeric scale factor

## Value

A list with ggplot2 objects.

## Author(s)

Thomas Wolf, Cristiano Oliveira

## Examples

```
data(sampleReadCounts)  
data(referenceReadCounts)  
## Gene names should be same size as row columns  
geneNames <- row.names(referenceReadCounts)  
  
ampliconNames <- NULL  
  
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]]
```

```

# Should be used values above 10000
replicates <- 10

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

backgroundNoise <- Background(geneNames,
                              samplesNormalizedReadCounts,
                              referenceNormalizedReadCounts,
                              bootList,
                              replicates = replicates)

reportTables <- ReportTables(geneNames,
                             samplesNormalizedReadCounts,
                             referenceNormalizedReadCounts,
                             bootList,
                             backgroundNoise)

PlotBootstrapDistributions(bootList, reportTables, save = FALSE)

```

---

ReadCountsFromBam      *ReadCountsFromBam*

---

## Description

Returns a matrix with the read counts from a set of bam files.

## Usage

```

ReadCountsFromBam(bamFileNames,
                 sampleNames,
                 gr,
                 ampliconNames,
                 minimumMappingQuality,
                 removeDup = FALSE)

```

## Arguments

bamFileNames	Vector of bamfile filepaths
sampleNames	Vector of sample names to be used as columns names instead of bam filepaths
gr	Genomic Range object as created by BedToGenomicRanges
ampliconNames	List of amplicon defining names
minimumMappingQuality	Minimum mapping quality
removeDup	Boolean value to remove duplicates. For reads with the same start site, end site and orientation only one is kept. For IonTorrent data this can be used to as an additional quality control. For Illumina data too many reads are being removed.

**Value**

A matrix with read counts where the rows represents the Amplicons and the columns represents the samples.

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
ReadCountsFromBam(bamFileNames,  
                  sampleNames,  
                  gF,  
                  ampliconNames,  
                  removeDup)
```

---

ReadXLSXToList

*ReadXLSXToList*

---

**Description**

Reads a list of read count matrices from a xlsx as generated by WriteReadCountsToXLSX

**Usage**

```
ReadXLSXToList(filepath, rowNames = TRUE, colNames = TRUE)
```

**Arguments**

filepath	filepath
rowNames	if row names should be included
colNames	if col names should be included

**Value**

A list of read count matrices

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
ReadXLSXToList(filepath)
```

---

referenceReadCounts	<i>Reference sample data</i>
---------------------	------------------------------

---

**Description**

Synthetic reference data set of simulated read counts. Only to be used for code examples.

**Usage**

```
referenceSamples
```

**Format**

A matrix with columns identifying the sample names and columns the gene names

**Value**

A matrix with columns identifying the sample names and columns the gene names

**Source**

Artificially generated data

---

ReportTables	<i>ReportTables</i>
--------------	---------------------

---

**Description**

This function generates the final report of the CNV detection procedure. One data frame is generated for each sample of interest.

**Usage**

```
ReportTables(geneNames,
             samplesNormalizedReadCounts,
             referenceNormalizedReadCounts,
             bootList,
             backgroundNoise)
```

**Arguments**

geneNames	Describe geneNames here
samplesNormalizedReadCounts	Describe samplesNormalizedReadCounts here
referenceNormalizedReadCounts	Describe referenceNormalizedReadCounts here
bootList	A list as returned by the BootList function
backgroundNoise	A list of background noise as returned by the Background function

**Value**

Returns a list of tables, one for each sample of interest. Each of these tables contains numerical information of the aberration status of each gene. For a detailed description see the Vignette.

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```

data(sampleReadCounts)
data(referenceReadCounts)
## Gene names should be same size as row columns
geneNames <- row.names(referenceReadCounts)

ampliconNames <- NULL

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]]

# Should be used values above 10000
replicates <- 10

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

backgroundNoise = Background(geneNames,
                             samplesNormalizedReadCounts,
                             referenceNormalizedReadCounts,
                             bootList,
                             replicates = replicates)

reportTables <- ReportTables(geneNames,
                             samplesNormalizedReadCounts,
                             referenceNormalizedReadCounts,
                             bootList,
                             backgroundNoise)

```

---

RunCNVPanelizerShiny *RunCNVPanelizerShiny*

---

**Description**

Run CNVPanelizer as a shiny app



**Usage**

```
RunCNVPanelizerShiny(port = 8100)
```

**Arguments**

port                    Port where the app will be listening

**Value**

Not returning any value

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
RunCNVPanelizerShiny(port=8080)
```

---

`sampleReadCounts`            *Test sample data*

---

**Description**

Synthetic data set of simulated read counts. Only to be used for running the code examples.

**Usage**

```
testSamples
```

**Format**

A matrix with columns identifying the sample names and columns the gene names

**Value**

A matrix with columns identifying the sample names and columns the gene names

**Source**

Artificially generated data

SelectReferenceSetByInterquartileRange  
*SelectReferenceSetByInterquartileRange*

---

**Description**

Select a reference set using a factor of the Interquartile Range

**Usage**

```
SelectReferenceSetByInterquartileRange(allSamplesReadCounts,  
                                       normalizationMethod = "tmm",  
                                       iqrFactor = 1)
```

**Arguments**

allSamplesReadCounts	All samples read counts matrix
normalizationMethod	tmm (trimmed mean of m values) or tss (total sum scaling)
iqrFactor	Interquantile range factor

**Value**

Returns a list of sample identifiers to be used as reference

**Author(s)**

Cristiano Oliveira

**Examples**

```
SelectReferenceSetByPercentil(allSamplesReadCounts,  
                              normalizationMethod = "tmm",  
                              iqrFactor = 1)
```

---

SelectReferenceSetByKmeans  
*SelectReferenceSetByKmeans*

---

**Description**

Select a reference set using Kmeans

**Usage**

```
SelectReferenceSetByKmeans(allSamplesReadCounts,  
                           normalizationMethod = "tmm",  
                           referenceNumberOfElements)
```

**Arguments**

`allSamplesReadCounts`  
All samples read counts matrix

`normalizationMethod`  
tmm (trimmed mean of m values) or tss (total sum scaling)

`referenceNumberOfElements`  
Number of elements to select for the reference set

**Value**

Returns a list of sample identifiers to be used as reference

**Author(s)**

Cristiano Oliveira

**Examples**

```
SelectReferenceSetByKmeans(allSamplesReadCounts,  
                           normalizationMethod = "tmm",  
                           referenceNumberOfElements)
```

---

`SelectReferenceSetByPercentil`  
*SelectReferenceSetByPercentil*

---

**Description**

Select a reference set using percentiles

**Usage**

```
SelectReferenceSetByPercentil(allSamplesReadCounts,  
                              normalizationMethod = "tmm",  
                              lowerBoundPercentage = 1,  
                              upperBoundPercentage = 99)
```

**Arguments**

`allSamplesReadCounts`  
All samples read counts matrix

`normalizationMethod`  
tmm (trimmed mean of m values) or tss (total sum scaling)

`lowerBoundPercentage`  
Lower bound percentage

`upperBoundPercentage`  
Upper bound percentage

**Value**

Returns a list of sample identifiers to be used as reference

**Author(s)**

Cristiano Oliveira

**Examples**

```
SelectReferenceSetByPercentil(allSamplesReadCounts,
                             normalizationMethod = "tmm",
                             lowerBoundPercentage = 1,
                             upperBoundPercentage = 99)
```

---

SelectReferenceSetFromReadCounts

*SelectReferenceSetFromReadCounts*

---

**Description**

Select a reference set from read counts

**Usage**

```
SelectReferenceSetFromReadCounts(allSamplesReadCounts,
                                 normalizationMethod = "tmm",
                                 referenceMaximumNumberOfElements = 30,
                                 referenceSelectionMethod = "kmeans",
                                 lowerBoundPercentage = 1,
                                 upperBoundPercentage = 99)
```

**Arguments**

`allSamplesReadCounts`  
All samples read counts matrix

`normalizationMethod`  
tmm (trimmed mean of m values) or tss (total sum scaling)

`referenceMaximumNumberOfElements`  
Maximum number of elements to consider as reference (only to be used in case interquantile reference selection method)

`referenceSelectionMethod`  
Reference selection method ("kmeans", ...)

`lowerBoundPercentage`  
Lower bound percentage (only to be used in case interquantile reference selection method)

`upperBoundPercentage`  
Upper bound percentage (only to be used in case interquantile reference selection method)

**Value**

Returns a list of sample identifiers to be used as reference

**Author(s)**

Cristiano Oliveira

**Examples**

```
SelectReferenceSetFromReadCounts(allSamplesReadCounts,  
                                normalizationMethod = "tmm",  
                                referenceMaximumNumberOfElements = 30,  
                                referenceSelectionMethod = "kmeans")
```

---

StatusHeatmap

*StatusHeatmap*

---

**Description**

Generates a status heatmap for all samples analyzed

**Usage**

```
StatusHeatmap(dfData,  
              statusColors = c("Deletion" = "blue",  
                              "Normal" = "green",  
                              "Amplification" = "red"),  
              header = "Status Heatmap",  
              filepath = "CNVPanelizerHeatMap.png")
```

**Arguments**

dfData	data frame with the "Amplification", "Deletion" and "Normal" status
statusColors	A named vector with the colors associated with each level
header	Header text at the plot
filepath	Filepath where the generated heatmap is saved

**Value**

Returns the filepath of the saved Heatmap

**Author(s)**

Cristiano Oliveira

**Examples**

```
StatusHeatmap(dfData,
               statusColors = c("Deletion" = "blue",
                                "Normal" = "green",
                                "Amplification" = "red"),
               header = "Status Heatmap",
               filepath = "CNVPanelizerHeatMap.png")
```

---

WriteListToXLSX

*WriteListToXLSX*


---

**Description**

Writes list of data frames to an xlsx file

**Usage**

```
WriteListToXLSX(listOfDataFrames,
                 multipleFiles = FALSE,
                 outputFolder = file.path(getwd(), "xlsx"),
                 filepath = "list.xlsx")
```

**Arguments**

listOfDataFrames	list of dataframes
multipleFiles	If should be generated on single file with all results or multiple files
outputFolder	Output folder
filepath	filepath

**Value**

Not returning any value

**Author(s)**

Thomas Wolf, Cristiano Oliveira

**Examples**

```
WriteListToXLSX(listOfDataFrames = exampleList, filepath = "list.xlsx")
```

# Index

- \* **R, Random Forest, CNV, Bootstrapping, Panel Sequencing, Ion Torrent**
  - CNVPanelizer-package, [2](#)
- \* **datasets**
  - referenceReadCounts, [15](#)
  - sampleReadCounts, [17](#)
- \*
  - referenceReadCounts, [15](#)
  
- Background, [3](#)
- BedToGenomicRanges, [4](#)
- BootList, [5](#)
  
- CNVPanelizer (CNVPanelizer-package), [2](#)
- CNVPanelizer-package, [2](#)
- CNVPanelizerFromReadCounts, [6](#)
- CNVPanelizerFromReadCountsHELPER, [7](#)
- CollectColumnFromAllReportTables, [9](#)
- CombinedNormalizedCounts, [9](#)
  
- IndexMultipleBams, [10](#)
  
- NormalizeCounts, [11](#)
  
- PlotBootstrapDistributions, [12](#)
  
- ReadCountsFromBam, [13](#)
- ReadXLSXToList, [14](#)
- referenceReadCounts, [15](#)
- ReportTables, [15](#)
- RunCNVPanelizerShiny, [16](#)
  
- sampleReadCounts, [17](#)
- SelectReferenceSetByInterquartileRange, [18](#)
- SelectReferenceSetByKmeans, [18](#)
- SelectReferenceSetByPercentil, [19](#)
- SelectReferenceSetFromReadCounts, [20](#)
- StatusHeatmap, [21](#)
  
- WriteListToXLSX, [22](#)