

Package ‘CNVfilterR’

October 17, 2020

Type Package

Title Identifies false positives of CNV calling tools by using SNV calls

Version 1.2.0

Author Jose Marcos Moreno-Cabrera <jmoreno@igtp.cat> and Bernat Gel <bgel@igtp.cat>

Maintainer Jose Marcos Moreno-Cabrera <jmoreno@igtp.cat>

Description CNVfilterR identifies those CNVs that can be discarded by using the single nucleotide variant (SNV) calls that are usually obtained in common NGS pipelines.

License Artistic-2.0

URL <https://github.com/jpuntomarcos/CNVfilterR>

Encoding UTF-8

RoxygenNote 7.0.0

Depends R (>= 3.6)

Imports IRanges, GenomicRanges, SummarizedExperiment, pracma, regioneR, assertthat, karyoploteR, CopyNumberPlots, graphics, utils, VariantAnnotation, Rsamtools, GenomeInfoDb, Biostrings, methods

Suggests knitr, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg19.masked

biocViews CopyNumberVariation, Sequencing, DNASeq, Visualization, DataImport

VignetteBuilder knitr

BugReports <https://github.com/jpuntomarcos/CNVfilterR/issues>

git_url <https://git.bioconductor.org/packages/CNVfilterR>

git_branch RELEASE_3_11

git_last_commit 5b693f0

git_last_commit_date 2020-04-27

Date/Publication 2020-10-16

R topics documented:

auxAddCNcolumn	2
auxGetVcfSource	3
auxProcessVariants	3
filterCNVs	4
getVariantScore	6
loadCNVcalls	7
loadSNPsFromVCF	9
loadVCFs	10
plotAllCNVs	12
plotScoringModel	13
plotVariantsForCNV	14
Index	16

auxAddCNcolumn	<i>auxAddCNcolumn</i>
----------------	-----------------------

Description

Adds a 'cn' column to the `cnvs.gr` data.frame or GRanges.

Usage

```
auxAddCNcolumn(cnvs.gr)
```

Arguments

<code>cnvs.gr</code>	data.frame or GRanges containing the column 'cnv' with "deletion" or "duplication" as values
----------------------	--

Details

For each row, cn column is filled with 1 if cnv is "deletion", 3 if cnv is "duplication"

Value

input `cnvs.gr` with the new column 'cn'

auxGetVcfSource	<i>auxGetVcfSource</i>
-----------------	------------------------

Description

Obtains VCF source from a given VCF file path. Auxiliar function used by loadSNPsFromVCF.

Usage

```
auxGetVcfSource(vcf.source = NULL, vcf.file)
```

Arguments

vcf.source	VCF source. Leave NULL to allow the function to recognize it. Otherwise, the function will not try to recognize the source. (Defaults to NULL)
vcf.file	VCF file path

Value

VCF source

auxProcessVariants	<i>auxProcessVariants</i>
--------------------	---------------------------

Description

Auxiliar function called by loadVCFs to process variants

Usage

```
auxProcessVariants(
  vars,
  cnvGR,
  heterozygous.range,
  homozygous.range,
  min.total.depth,
  exclude.indels,
  regions.to.exclude
)
```

Arguments

vars	GRanges object containing variants for a certain sample.
cnvGR	GRanges object containing CNV calls for a certain sample.
heterozygous.range	Heterozygous range. Variants not in the homozygous/heterozygous intervals will be excluded.
homozygous.range	Homozygous range. Variants not in the homozygous/heterozygous intervals will be excluded.

min.total.depth Minimum total depth. Variants under this value will be excluded.

exclude.indels Whether to exclude indels when loading the variants. TRUE is the recommended value given that indels frequency varies in a different way than SNVs.

regions.to.exclude A GRanges object defining the regions for which the variants should be excluded.

Value

Processed vars

filterCNVs	<i>filterCNVs</i>
------------	-------------------

Description

Identifies those copy number calls that can be filtered out

Usage

```
filterCNVs(
  cnvs.gr,
  vcfs,
  expected.ht.mean = 50,
  expected.dup.ht.mean1 = 33.3,
  expected.dup.ht.mean2 = 66.6,
  sigmoid.c1 = 2,
  sigmoid.c2.vector = c(28, 38.3, 44.7, 55.3, 61.3, 71.3),
  dup.threshold.score = 0.5,
  ht.deletions.threshold = 30,
  verbose = FALSE,
  margin.pct = 10
)
```

Arguments

cnvs.gr GRanges containing CNVs to be filtered out. Use loadCNVcalls to load them.

vcfs List of GRanges containing all variants (SNV/indel) obtained with the loadVCFs function.

expected.ht.mean Expected heterozygous SNV/indel allele frequency (defaults to 50)

expected.dup.ht.mean1 Expected heterozygous SNV/indel allele frequency when the variant IS NOT in the same allele than the CNV duplication call. (defaults to 33.3)

expected.dup.ht.mean2 Expected heterozygous SNV/indel allele frequency when the variant IS in the same allele than the CNV duplication call. (defaults to 66.6)

sigmoid.c1 Sigmoid c1 parameter. (defaults to 2)

<code>sigmoid.c2.vector</code>	Vector containing sigmoid c2 parameters for the six sigmoids functions. (defaults to c(28, 38.3, 44.7, 55.3, 61.3, 71.3))
<code>dup.threshold.score</code>	Limit value to decide if a CNV duplication can be filtered out or not. A CNV duplication can be filtered out if the total score computed from heterozygous variants in the CNV is equal or greater than <code>dup.threshold.score</code> . (defaults to 0.5)
<code>ht.deletions.threshold</code>	Minimum percentage of heterozygous variants falling in a CNV deletion to filter that CNV. (defaults to 30)
<code>verbose</code>	Whether to show information messages. (defaults to TRUE)
<code>margin.pct</code>	Variants in the CNV but close to the ends of the CNV will be ignored. <code>margin.pct</code> defines the percentage of CNV length, located at each CNV limit, where variants will be ignored. For example, for a CNV <code>chr1:1000-2000</code> and a <code>margin.pct</code> value of 10, variants within <code>chr1:1000-1100</code> and <code>chr1:1900-2000</code> will be ignored.

Details

Checks all the variants (SNV and optionally INDELs) in each CNV present in `cnvs.gr` to decide whether a CNV can be filtered out or not. It returns an S3 object with 3 elements: `cnvs`, `variantsForEachCNV` and `filterParameters`. See return section for further details.

A CNV deletion can be filtered out if there is at least `ht.deletions.threshold` A CNV duplication can be filtered out if the score is \geq `dup.threshold.score` after computing all heterozygous variants falling in the CNV.

If a CNV can be filtered out, then the value TRUE is set in the `filter` column of the `cnvs` element.

Value

A S3 object with 3 elements:

- `cnvs`: GRanges with the input CNVs and the meta-columns added during the call:
 - `cnv.id`: CNV id
 - `filter`: Set to TRUE if the CNV can be filtered out
 - `n.total.variants`: Number of variants in the CNV
 - `n.hm.variants`: Number of homozygous variants. They do not give any evidenced for confirming or discarding the CNV.
 - `n.ht.discard.CNV`: For a CNV duplication, number of heterozygous variants in that discard the CNV (those with a positive score)
 - `n.ht.confirm.CNV`: For a CNV duplication, number of heterozygous variants that confirm the CNV (those with a negative score)
 - `ht.pct`: Percentage of heterozygous variants for deletion CNVs
 - `score`: total score when computing all the variants scores
- `variantsForEachCNV`: named list where each name correspond to a CNV id and the value is a `data.frame` with all variants falling in that CNV
- `filterParameters`: input parameters used for filtering

Examples

```

# Load CNVs data
cnvs.file <- system.file("extdata", "DECoN.CNVcalls.csv", package = "CNVfilter", mustWork = TRUE)
cnvs.gr <- loadCNVcalls(cnvs.file = cnvs.file, chr.column = "Chromosome", start.column = "Start", end.column = "End")

# Load VCFs data
vcf.files <- c(system.file("extdata", "variants.sample1.vcf.gz", package = "CNVfilter", mustWork = TRUE),
               system.file("extdata", "variants.sample2.vcf.gz", package = "CNVfilter", mustWork = TRUE))
vcfs <- loadVCFs(vcf.files, cnvs.gr = cnvs.gr)

# Filter CNVs
results <- filterCNVs(cnvs.gr, vcfs)

# Check CNVs that can be filtered out
as.data.frame(results$cnvs[results$cnvs$filter == TRUE])

```

<code>getVariantScore</code>	<i>getVariantScore</i>
------------------------------	------------------------

Description

Returns score for a given allele frequency

Usage

```

getVariantScore(
  freq,
  expected.ht.mean,
  expected.dup.ht.mean1,
  expected.dup.ht.mean2,
  sigmoid.c1,
  sigmoid.c2.vector,
  sigmoid.int1,
  sigmoid.int2
)

```

Arguments

<code>freq</code>	Variant allele frequency
<code>expected.ht.mean</code>	Expected heterozygous SNV/indel allele frequency
<code>expected.dup.ht.mean1</code>	Expected heterozygous SNV/indel allele frequency when the variant IS NOT in the same allele than the CNV duplication call
<code>expected.dup.ht.mean2</code>	Expected heterozygous SNV/indel allele frequency when the variant IS in the same allele than the CNV duplication call
<code>sigmoid.c1</code>	Sigmoid c1 parameter

```

sigmoid.c2.vector      Vector containing sigmoid c2 parameters for the six sigmoids functions
sigmoid.int1          Sigmoid int 1
sigmoid.int2          Sigmoid int 2

```

Details

Returns a value between -1 and 1. If the allele frequency increases the evidence of discarding a CNV, then the score is positive. If the allele frequency decreases the evidence for discarding a CNV, the score is negative.

The model is based on the fuzzy logic and the score is calculated using sigmoids. See the vignette to get more details.

Value

variant score in the [-1, 1] range

loadCNVcalls	<i>loadCNVcalls</i>
--------------	---------------------

Description

Loads CNV calls from a csv/tsv file

Usage

```

loadCNVcalls(
  cnvs.file,
  chr.column,
  start.column,
  end.column,
  coord.column = NULL,
  cnv.column,
  sample.column,
  sample.name = NULL,
  gene.column = NULL,
  deletion = "deletion",
  duplication = "duplication",
  ignore.unexpected.rows = FALSE,
  sep = "\t",
  skip = 0,
  genome = "hg19",
  exclude.non.canonical.chrs = TRUE
)

```

Arguments

```

cnvs.file      Path to csv/tsv file containing the CNV calls.
chr.column     Which column stores the chr location of the CNV.
start.column   Which column stores the start location of the CNV.

```

<code>end.column</code>	Which column stores the end location of the CNV.
<code>coord.column</code>	CNV location in the chr:start-end format. Example: "1:538001-540000". If NULL, <code>chr.column</code> , <code>start.column</code> and <code>end.column</code> columns will be used. (Defaults to NULL)
<code>cnv.column</code>	Which column stores the type of CNV (deletion or duplication).
<code>sample.column</code>	Which column stores the sample name.
<code>sample.name</code>	Sample name for all CNVs defined in <code>cnvs.file</code> . If set, <code>sample.column</code> is ignored (Defaults to NULL)
<code>gene.column</code>	Which columns store the gene or genes affected (optional). (Defaults to NULL)
<code>deletion</code>	Text used in the <code>cnv.column</code> to represent deletion CNVs. Multiple values are also allowed, for example: <code>c("CN0", "CN1")</code> . (Defaults to "deletion")
<code>duplication</code>	Text used in the <code>cnv.column</code> to represent duplication CNVs. Multiple values are also allowed, for example: <code>c("CN3", "CN4")</code> (Defaults to "duplication")
<code>ignore.unexpected.rows</code>	Whether to ignore the rows which CNV <code>cnv.column</code> value is different to <code>deletion</code> or <code>duplication</code> values (Defaults to FALSE). It is useful for processing output from callers like LUMPY or Manta (they call also events that are not CNVs)
<code>sep</code>	Separator symbol to load the csv/tsv file. (Defaults to "\t")
<code>skip</code>	Number of rows that should be skipped when reading the csv/tsv file. (Defaults to 0)
<code>genome</code>	The name of the genome. (Defaults to "hg19")
<code>exclude.non.canonical.chrs</code>	Whether to exclude non canonical chromosomes (Defaults to TRUE)

Details

Loads a csv/tsv file containing CNV calls, and transform it into a GRanges with `cnv` and `sample` metadata columns.

Value

A GRanges with a range per each CNV and the metadata columns:

- `cnv`: type of CNV, "duplication" or "deletion"
- `sample`: sample name

Returns NULL if `cnvs.file` has no CNVs

Examples

```
# Load CNVs data
cnvs.file <- system.file("extdata", "DECoN.CNVcalls.csv", package = "CNVfilter", mustWork = TRUE)
cnvs.gr <- loadCNVcalls(cnvs.file = cnvs.file, chr.column = "Chromosome", start.column = "Start", end.column =
```

loadSNPsFromVCF	<i>loadSNPsFromVCF</i>
-----------------	------------------------

Description

Loads SNPs (SNVs/indels) from a VCF file

Usage

```
loadSNPsFromVCF(
  vcf.file,
  vcf.source = NULL,
  ref.support.field = NULL,
  alt.support.field = NULL,
  list.support.field = NULL,
  regions.to.filter = NULL,
  genome = "hg19",
  exclude.non.canonical.chrs = TRUE,
  verbose = TRUE
)
```

Arguments

<code>vcf.file</code>	VCF file path
<code>vcf.source</code>	VCF source, i.e., the variant caller used to generate the VCF file. If set, the function will not try to recognize the source. (Defaults to NULL)
<code>ref.support.field</code>	Reference allele depth field. (Defaults to NULL)
<code>alt.support.field</code>	Alternative allele depth field. (Defaults to NULL)
<code>list.support.field</code>	Allele support field in a list format: reference allele, alternative allele. (Defaults to NULL)
<code>regions.to.filter</code>	The regions to which limit the VCF import. It can be used to speed up the import process. (Defaults to NULL)
<code>genome</code>	The name of the genome (Defaults to "hg19")
<code>exclude.non.canonical.chrs</code>	Whether to exclude non canonical chromosomes (Defaults to TRUE)
<code>verbose</code>	Whether to show information messages. (Defaults to TRUE)

Details

Given a VCF file path, the function recognizes the variant caller source to decide which fields should be used to calculate ref/alt support and allelic frequency (see return). Current supported variant callers are VarScan2, Strelka/Strelka2, freebayes, HaplotypeCaller and UnifiedGenotyper.

Optionally, the fields where the data is stored can be manually set by using the parameters `ref.support.field`, `alt.support.field` and `list.support.field`

Requirement: a TabixFile (.tbi) should exist in the same directory of the VCF file.

Value

A list where names are sample names, and values are GRanges objects containing the variants for each sample, including the following metadata columns:

- `ref.support`: Reference allele depth field
- `alt.support`: Alternative allele depth field
- `alt.freq`: allelic frequency
- `total.depth`: total depth

Examples

```
vcf.file <- system.file("extdata", "variants.sample1.vcf.gz", package = "CNVfilter", mustWork = TRUE)
vcf <- loadSNPsFromVCF(vcf.file)
```

loadVCFs

*loadVCFs***Description**

Loads VCFs files

Usage

```
loadVCFs(
  vcf.files,
  sample.names = NULL,
  cnvs.gr,
  min.total.depth = 10,
  regions.to.exclude = NULL,
  vcf.source = NULL,
  ref.support.field = NULL,
  alt.support.field = NULL,
  list.support.field = NULL,
  homozygous.range = c(90, 100),
  heterozygous.range = c(28, 72),
  exclude.indels = TRUE,
  genome = "hg19",
  exclude.non.canonical.chrs = TRUE,
  verbose = TRUE
)
```

Arguments

<code>vcf.files</code>	vector of VCFs paths. Both <code>.vcf</code> and <code>.vcf.gz</code> extensions are allowed.
<code>sample.names</code>	Sample names vector containing sample names for each <code>vcf.files</code> . If <code>NULL</code> , sample name will be obtained from the VCF sample column. (Defaults to <code>NULL</code>)
<code>cnvs.gr</code>	GRanges object containing CNV calls. Call <code>loadCNVcalls</code> to obtain it. Only those variants in regions affected by CNVs will be loaded to speed up the load.

<code>min.total.depth</code>	Minimum total depth. Variants under this value will be excluded. (Defaults to 10)
<code>regions.to.exclude</code>	A GRanges object defining the regions for which the variants should be excluded. Useful for defining known difficult regions like pseudogenes where the allele frequency is not trustable. (Defaults to NULL)
<code>vcf.source</code>	VCF source, i.e., the variant caller used to generate the VCF file. If set, the <code>loadSNPsFromVCF</code> function will not try to recognize the source. (Defaults to NULL)
<code>ref.support.field</code>	Reference allele depth field. (Defaults to NULL)
<code>alt.support.field</code>	Alternative allele depth field. (Defaults to NULL)
<code>list.support.field</code>	Allele support field in a list format: reference allele, alternative allele. (Defaults to NULL)
<code>homozygous.range</code>	Homozygous range. Variants not in the homozygous/heterozygous intervals will be excluded. (Defaults to <code>c(90, 100)</code>)
<code>heterozygous.range</code>	Heterozygous range. Variants not in the homozygous/heterozygous intervals will be excluded. (Defaults to <code>c(28, 72)</code>)
<code>exclude.indels</code>	Whether to exclude indels when loading the variants. TRUE is the recommended value given that indels frequency varies in a different way than SNVs. (Defaults to TRUE)
<code>genome</code>	The name of the genome. (Defaults to "hg19")
<code>exclude.non.canonical.chrs</code>	Whether to exclude non canonical chromosomes (Defaults to TRUE)
<code>verbose</code>	Whether to show information messages. (Defaults to TRUE)

Details

Loads VCF files and computes alt allele frequency for each variant. It uses `loadSNPsFromVCF` function load the data and identify the correct VCF format for allele frequency computation.

If `sample.names` is not provided, the sample names included in the VCF itself will be used. Both single-sample and multi-sample VCFs are accepted, but when multi-sample VCFs are used, `sample.names` parameter must be NULL.

If `vcf` is not compressed with `bgzip`, the function compresses it and generates the `.gz` file. If `.tbi` file does not exist for a given VCF file, the function also generates it. All files are generated in a temporary folder.

Value

A list where names are the sample names, and values are the GRanges objects for each sample.

Note

Important: Compressed VCF must be compressed with `[bgzip ("block gzip")]` from Samtools `htslib` (<http://www.htslib.org/doc/bgzip.html>) and not using the standard `Gzip` utility.

Examples

```
# Load CNVs data (required by loadVCFs to speed up the load process)
cnvs.file <- system.file("extdata", "DECoN.CNVcalls.csv", package = "CNVfilter", mustWork = TRUE)
cnvs.gr <- loadCNVcalls(cnvs.file = cnvs.file, chr.column = "Chromosome", start.column = "Start", end.column = "End")

# Load VCFs data
vcf.files <- c(system.file("extdata", "variants.sample1.vcf.gz", package = "CNVfilter", mustWork = TRUE),
               system.file("extdata", "variants.sample2.vcf.gz", package = "CNVfilter", mustWork = TRUE))
vcfs <- loadVCFs(vcf.files, cnvs.gr = cnvs.gr)
```

plotAllCNVs

plotAllCNVs

Description

Plots all CNVs on chromosome ideograms

Usage

```
plotAllCNVs(cnvs.gr)
```

Arguments

cnvs.gr	GRanges containing all CNV definitions returned by filterCNVs or loadCNVcalls functions.
---------	--

Details

Plots all CNVs defined at cnvs.gr on a view of horizontal ideograms representing all chromosomes.

Value

invisibly returns a karyoplot object

Examples

```
cnvs.file <- system.file("extdata", "DECoN.CNVcalls.2.csv", package = "CNVfilter", mustWork = TRUE)
cnvs.gr <- loadCNVcalls(cnvs.file = cnvs.file, chr.column = "Chromosome", start.column = "Start", end.column = "End")

# Plot all CNVs
plotAllCNVs(cnvs.gr)
```

plotScoringModel	<i>plotVariantsForCNV</i>
------------------	---------------------------

Description

Plots scoring model used for CNV duplications

Usage

```
plotScoringModel(
  expected.ht.mean,
  expected.dup.ht.mean1,
  expected.dup.ht.mean2,
  sigmoid.c1,
  sigmoid.c2.vector
)
```

Arguments

```
expected.ht.mean           Expected heterozygous SNV/indel allele frequency
expected.dup.ht.mean1     Expected heterozygous SNV/indel allele frequency when the variant IS NOT in
                           the same allele than the CNV duplication call
expected.dup.ht.mean2     Expected heterozygous SNV/indel allele frequency when the variant IS in the
                           same allele than the CNV duplication call
sigmoid.c1                 Sigmoid c1 parameter
sigmoid.c2.vector         Vector containing sigmoid c2 parameters for the six sigmoids functions
```

Value

nothing

Examples

```
# Load CNVs data
cnvs.file <- system.file("extdata", "DECoN.CNVcalls.csv", package = "CNVfilter", mustWork = TRUE)
cnvs.gr <- loadCNVcalls(cnvs.file = cnvs.file, chr.column = "Chromosome", start.column = "Start", end.column = "End")

# Load VCFs data
vcf.files <- c(system.file("extdata", "variants.sample1.vcf.gz", package = "CNVfilter", mustWork = TRUE),
               system.file("extdata", "variants.sample2.vcf.gz", package = "CNVfilter", mustWork = TRUE))
vcfs <- loadVCFs(vcf.files, cnvs.gr = cnvs.gr)

# Filter CNVs
results <- filterCNVs(cnvs.gr, vcfs)

# Plot scoring model for duplication CNVs
p <- results$filterParameters
plotScoringModel(expected.ht.mean = p$expected.ht.mean, expected.dup.ht.mean1 = p$expected.dup.ht.mean1,
```

```
expected.dup.ht.mean2 = p$expected.dup.ht.mean2, sigmoid.c1 = p$sigmoid.c1, sigmoid.c2.vector = p
```

```
plotVariantsForCNV      plotVariantsForCNV
```

Description

Plots a CNV with all the variants in it

Usage

```
plotVariantsForCNV(
  cnvfilter.results,
  cnv.id,
  points.cex = 1,
  points.pch = 19,
  legend.x.pos = 0.08,
  legend.y.pos = 0.25,
  legend.cex = 0.8,
  karyotype.cex = 1,
  cnv.label.cex = 1,
  x.axis.bases.cex = 0.7,
  x.axis.bases.digits = 5,
  y.axis.title.cex = 0.8,
  y.axis.label.cex = 0.8,
  cnv.zoom.margin = TRUE
)
```

Arguments

cnvfilter.results	S3 object returned by filterCNVs function
cnv.id	CNV id for which to plot variants
points.cex	Points cex (size). (Defaults to 1)
points.pch	Points pch (symbol). (Defaults to 19)
legend.x.pos	Legend x position. (Defaults to 0.08)
legend.y.pos	Legend y position. (Defaults to 0.25)
legend.cex	Legend cex. (Defaults to 0.8)
karyotype.cex	karyotype cex: affects top title and chromosome text (at bottom). (Defaults to 1)
cnv.label.cex	"CNV" text cex. (Defaults to 1)
x.axis.bases.cex	X-axis bases position cex. (Defaults to 0.7)
x.axis.bases.digits	X-axis bases position number of digits. (Defaults to 5)
y.axis.title.cex	Y-axis title cex. (Defaults to 0.8)

`y.axis.label.cex`
Y-axis labels cex. (Defaults to 0.8)

`cnv.zoom.margin`
If TRUE, the zoom leaves an small margin at both sides of the CNV. False otherwise. (Defaults to TRUE)

Value

invisibly returns a karyoplot object

Examples

```
# Load CNVs data
cnvs.file <- system.file("extdata", "DECoN.CNVcalls.csv", package = "CNVfilter", mustWork = TRUE)
cnvs.gr <- loadCNVcalls(cnvs.file = cnvs.file, chr.column = "Chromosome", start.column = "Start", end.column = "End")

# Load VCFs data
vcf.files <- c(system.file("extdata", "variants.sample1.vcf.gz", package = "CNVfilter", mustWork = TRUE),
              system.file("extdata", "variants.sample2.vcf.gz", package = "CNVfilter", mustWork = TRUE))
vcfs <- loadVCFs(vcf.files, cnvs.gr = cnvs.gr)

# Filter CNVs
results <- filterCNVs(cnvs.gr, vcfs)

# Check CNVs that can be filtered out
as.data.frame(results$cnvs[results$cnvs$filter == TRUE])

# Plot one of them
plotVariantsForCNV(results, "3")
```

Index

[auxAddCNcolumn](#), [2](#)
[auxGetVcfSource](#), [3](#)
[auxProcessVariants](#), [3](#)

[filterCNVs](#), [4](#)

[getVariantScore](#), [6](#)

[loadCNVcalls](#), [7](#)
[loadSNPsFromVCF](#), [9](#), [11](#)
[loadVCFs](#), [10](#)

[plotAllCNVs](#), [12](#)
[plotScoringModel](#), [13](#)
[plotVariantsForCNV](#), [14](#)