Package ‘CNVRanger’

May 27, 2024

Version 1.20.0

Title Summarization and expression/phenotype association of CNV ranges

Author Ludwig Geistlinger [aut, cre], Vinicius Henrique da Silva [aut], Marcel Ramos [ctb], Levi Waldron [ctb]

Maintainer Ludwig Geistlinger <ludwig.geistlinger@gmail.com>

Depends GenomicRanges, RaggedExperiment

Imports BiocGenerics, BiocParallel, GDSArray, GenomeInfoDb, IRanges, S4Vectors, SNPRelate, SummarizedExperiment, data.table, edgeR, gdsfmt, grDevices, lattice, limma, methods, plyr, qqman, rappdirs, reshape2, stats, utils


Description The CNVRanger package implements a comprehensive tool suite for CNV analysis. This includes functionality for summarizing individual CNV calls across a population, assessing overlap with functional genomic regions, and association analysis with gene expression and quantitative phenotypes.

License Artistic-2.0

BugReports https://github.com/waldronlab/CNVRanger/issues

Encoding UTF-8

VignetteBuilder knitr

biocViews CopyNumberVariation, DifferentialExpression, GeneExpression, GenomeWideAssociation, GenomicVariation, Microarray, RNASeq, SNP

RoxygenNote 7.2.3

git_url https://git.bioconductor.org/packages/CNVRanger

git_branch RELEASE_3_19

git_last_commit dca5773
cnvEQTL

CNV-expression association analysis

Description

Testing CNV regions for effects on the expression level of genes in defined genomic windows.

Usage

cnvEQTL(
  cnvrs,
  calls,
  rcounts,
  data,
  window = "1Mbp",
  multi.calls = .largest,
  min.samples = 10,
  min.state.freq = 3,
  de.method = c("edgeR", "limma"),
  padj.method = "BH",
  filter.by.expr = TRUE,
  verbose = FALSE
)
Arguments

cnvrs
A GRanges or character object containing the summarized CNV regions as e.g. obtained with populationRanges. Alternatively, the assay name if the `data` argument is provided.

calls
Either a GRangesList or RaggedExperiment storing the individual CNV calls for each sample. Alternatively, the assay name if `data` is provided.

rcounts
A RangedSummarizedExperiment or character name storing either the raw RNA-seq read counts in a rectangular fashion (genes x samples). Alternatively, the assay name if `data` is provided.

data
(optional) A MultiAssayExperiment object with `cnvrs`, `calls`, and `rcounts` arguments corresponding to assay names.

window
Numeric or Character. Size of the genomic window in base pairs by which each CNV region is extended up- and downstream. This determines which genes are tested for each CNV region. Character notation is supported for convenience such as "100kbp" (same as 100000) or "1Mbp" (same as 1000000). Defaults to "1Mbp". Can also be set to NULL to test against all genes included in the analysis.

multi.calls
A function. Determines how to summarize the CN state in a CNV region when there are multiple (potentially conflicting) calls for one sample in that region. Defaults to .largest, which assigns the CN state of the call that covers the largest part of the CNV region tested. A user-defined function that is passed on to qreduceAssay can also be provided for customized behavior.

min.samples
Integer. Minimum number of samples with at least one call overlapping the CNV region tested. Defaults to 10. See details.

min.state.freq
Integer. Minimum number of samples in each CNV state being tested. Defaults to 3.

de.method
Character. Differential expression method. Defaults to "edgeR".

padj.method
Character. Method for adjusting p-values to multiple testing. For available methods see the man page of the function p.adjust. Defaults to "BH".

filter.by.expr
Logical. Include only genes with sufficiently large counts in the DE analysis? If TRUE, excludes genes not satisfying a minimum number of read counts across samples using the filterByExpr function from the edgeR package. Defaults to TRUE.

verbose
Logical. Display progress messages? Defaults to FALSE.

Details

Association testing between CNV regions and RNA-seq read counts is carried out using edgeR, which applies generalized linear models (GLMs) based on the negative-binomial distribution while incorporating normalization factors for different library sizes.

In the case of only one CN state deviating from 2n for a CNV region under investigation, this reduces to the classical 2-group comparison. For more than two states (e.g. 0n, 1n, 2n), edgeR’s ANOVA-like test is applied to test all deviating groups for significant expression differences relative to 2n.

To avoid artificial effects due to low expression of a gene or insufficient sample size in deviating groups, it is typically recommended to exclude from the analysis (i) genes with fewer than r reads
per million reads mapped (cpm, counts per million) in the maximally expressed sample group, and
(ii) CNV regions with fewer than $s$ samples in a group deviating from 2n. Use the min.cpm and
min.samples arguments, respectively.

When testing local effects (adjacent or coinciding genes of a CNV region), suitable thresholds for
candidate discovery are $r = 3$, $s = 4$, and a nominal significance level of 0.05; as such effects have a
clear biological indication and the number of genes tested is typically small.

For distal effects (i.e. when testing genes far away from a CNV region) more stringent thresholds
such as $r = 20$ and $s = 10$ for distal effects in conjunction with multiple testing correction using
a conservative adjusted significance level such as 0.01 is typically recommended (due to power
considerations and to avoid detection of spurious effects).

Value

A DataFrame containing measures of association for each CNV region and each gene tested in the
genomic window around the CNV region.

Author(s)

Ludwig Geistlinger

References

Geistlinger et al. (2018) Widespread modulation of gene expression by copy number variation in

See Also

findOverlaps to find overlaps between sets of genomic regions,
qreduceAssay to summarize ragged genomic location data in defined genomic regions,
glmQLFit and glmQLFTest to conduct negative binomial generalized linear models for RNA-seq
read count data.

Examples

# (1) CNV calls
states <- sample(c(0,1,3,4), 17, replace=TRUE)
calls <- GRangesList(
  sample1 = GRanges( c("chr1:1-10", "chr2:15-18", "chr2:25-34"), state=states[1:3]),
  sample3 = GRanges( c("chr1:2-11", "chr2:14-18", "chr2:26-36"), state=states[7:9] ),
  sample4 = GRanges( c("chr1:1-12", "chr2:18-35" ), state=states[10:11] ),
  sample5 = GRanges( c("chr1:1-12", "chr2:11-17", "chr2:26-34"), state=states[12:14] ),
)

# (2) summarized CNV regions
cnvr <- populationRanges(calls, density=0.1)

# (3) RNA-seq read counts
genes <- GRanges(c("chr1:2-9", "chr1:100-150", "chr1:200-300", "chr1:1000-1500"),
  state=states, density=0.1)
y <- matrix(rnbinom(42, size=1, mu=10), 7, 6)

names(genes) <- rownames(y) <- paste0("gene", 1:7)
colnames(y) <- paste0("sample", 1:6)

library(SummarizedExperiment)
rse <- SummarizedExperiment(assays=list(counts=y), rowRanges=granges(genes))

# (4) perform the association analysis
res <- cnvEQTL(cnvr, calls, rse,
               min.samples = 1, min.state.freq = 1, filter.by.expr = FALSE)

---

cnvGWAS

Run the CNV-GWAS

**Description**

Wraps all the necessary functions to run a CNV-GWAS using the output of `setupCnvGWAS` function.

(i) Produces the GDS file containing the genotype information (if `produce.gds` == TRUE), (ii) Produces the requested inputs for a PLINK analysis, (iii) run a CNV-GWAS analysis using a linear model (i.e. `lm` function), and (iv) export a QQ-plot displaying the adjusted p-values. In this release only the p-value for the copy number is available (i.e. 'P(CNP)').

**Usage**

```r
cnvGWAS(
  phen.info,
  n.cor = 1,
  min.sim = 0.95,
  freq.cn = 0.01,
  snp.matrix = FALSE,
  method.m.test = "fdr",
  lo.phe = 1,
  chr.code.name = NULL,
  genotype.nodes = "CNVGenotype",
  coding.translate = "all",
  path.files = NULL,
  list.of.files = NULL,
  produce.gds = TRUE,
  run.lrr = FALSE,
  assign.probe = "min.pvalue",
  correct.inflation = FALSE,
  both.up.down = FALSE,
  verbose = FALSE
)
```

Arguments

phen.info
- Returned by `setupCnvGWAS`

n.cor
- Number of cores to be used

min.sim
- Minimum CNV genotype distribution similarity among subsequent probes. Default is 0.95 (i.e. 95%)

freq.cn
- Minimum CNV frequency where 1 (i.e. 100%), or all samples deviating from diploid state. Default 0.01 (i.e. 1%)

snp.matrix
- Only FALSE implemented - If TRUE B allele frequencies (BAF) would be used to reconstruct CNV-SNP genotypes

method.m.test
- Correction for multiple tests to be used. FDR is default, see `p.adjust` for other methods.

lo.phe
- The phenotype to be analyzed in the PhenInfo$phenotypesSam data-frame

chr.code.name
- A data-frame with the integer name in the first column and the original name for each chromosome

genotype.nodes
- Expression data type. Nodes with CNV genotypes to be produced in the gds file.

coding.translate
- For 'CNVgenotypeSNPlike'. If NULL or unrecognized string use only biallelic CNVs. If 'all' code multiallelic CNVs as 0 for loss; 1 for 2n and 2 for gain.

path.files
- Folder containing the input CNV files used for the CNV calling (i.e. one text file with 5 columns for each sample). Columns should contain (i) probe name, (ii) Chromosome, (iii) Position, (iv) LRR, and (v) BAF.

list.of.files
- Data-frame with two columns where the (i) is the file name with signals and (ii) is the correspondent name of the sample in the gds file

produce.gds
- logical. If TRUE produce a new gds, if FALSE use gds previously created

run.lrr
- If TRUE use LRR values instead absolute copy numbers in the association

assign.probe
- ‘min.pvalue’ or ‘high.freq’ to represent the CNV segment

correct.inflation
- logical. Estimate lambda from raw p-values and correct for genomic inflation. Use with argument `method.m.test` to generate strict p-values.

both.up.down
- Check for CNV genotype similarity in both directions. Default is FALSE (i.e. only downstream)

verbose
- Show progress in the analysis

Value

The CNV segments and the representative probes and their respective p-value

Author(s)

Vinicius Henrique da Silva

References

# Load phenotype-CNV information
data.dir <- system.file("extdata", package="CNVRanger")

phen.loc <- file.path(data.dir, "Pheno.txt")
cnv.out.loc <- file.path(data.dir, "CNVOut.txt")
map.loc <- file.path(data.dir, "MapPenn.txt")

phen.info <- setupCnvGWAS('Example', phen.loc, cnv.out.loc, map.loc)

# Define chr correspondence to numeric, if necessary
df <- '
16 1A
25 4A
29 25LG1
30 25LG2
31 LGE22'

chr.code.name <- read.table(text=df, header=FALSE)
segs.pvalue.gr <- cnvGWAS(phen.info, chr.code.name=chr.code.name)

---

cnvOncoPrint

OncoPrint plot for CNV regions

Description

Illustrates overlaps between CNV calls and genomic features across a sample population.

Usage

cnvOncoPrint(
calls,
features,
multi.calls = .largest,
top.features = 25,
top.samples = 100,
...)

Arguments

calls Either a GRangesList or RaggedExperiment storing the individual CNV calls for each sample.
features A GRanges object containing the genomic features of interest, typically genes. Feature names are either expected as a meta-column symbol or as the names of the object.

multi.calls A function. Determines how to summarize the CN state in a CNV region when there are multiple (potentially conflicting) calls for one sample in that region. Defaults to .largest, which assigns the CN state of the call that covers the largest part of the CNV region tested. A user-defined function that is passed on to qreduceAssay can also be provided for customized behavior.

top.features integer. Restricts the number of features for plotting to features experiencing highest alteration frequency. Defaults to 25. Use -1 to display all features.

top.samples integer. Restricts the number of samples for plotting to samples experiencing highest alteration frequency. Defaults to 100. Use -1 to display all samples.

... Additional arguments passed on to ComplexHeatmap::oncoPrint

Value
None. Plots to a graphics device.

Author(s)
Ludwig Geistlinger

See Also
ComplexHeatmap::oncoPrint

Examples

# read in example CNV calls
data.dir <- system.file("extdata", package="CNVRanger")
call.file <- file.path(data.dir, "Silva16_PONE_CNV_calls.csv")
calls <- read.csv(call.file, as.is=TRUE)

# store in a GRangesList
calls <- makeGRangesListFromDataFrame(calls, split.field="NE_id", keep.extra.columns=TRUE)

# three example genes
genes <- c("chr1:140368053-140522639:-", "chr2:97843887-97988140:+", "chr2:135418586-135422028:-")
names(genes) <- c("ATP2C1", "MAP2", "ACTL8")
gen <- GRanges(genes)

# plot
cnvOncoPrint(calls, genes, top.samples = 25)
generateGDS

Produce CNV-GDS for the phenotyped samples

Description

Function to produce the GDS file in a probe-wise fashion for CNV genotypes. The GDS file which is produced also incorporates one phenotype to be analyzed. If several phenotypes are enclosed in the 'phen.info' object, the user may specify the phenotype to be analyzed with the 'lo.phe' parameter. Only diploid chromosomes should be included.

Usage

```r
generateGDS(
  phen.info,
  freq.cn = 0.01,
  snp.matrix = FALSE,
  lo.phe = 1,
  chr.code.name = NULL,
  genotype.nodes = c("CNVGenotype", "CNVgenotypeSNPlike"),
  coding.translate = NULL,
  n.cor = 1
)
```

Arguments

- `phen.info`: Returned by setupCnvGWAS
- `freq.cn`: Minimum frequency. Default is 0.01 (i.e. 1%)
- `snp.matrix`: Only FALSE implemented. If TRUE, B allele frequencies (BAF) and SNP genotypes would be used to reconstruct CNV-SNP genotypes - under development
- `lo.phe`: The phenotype to be analyzed in the PhenInfo$phenotypesSam dataframe
- `chr.code.name`: A data-frame with the integer name in the first column and the original name in the second for each chromosome previously converted to numeric
- `genotype.nodes`: Nodes with CNV genotypes to be produced in the gds file. Use 'CNVGenotype' for dosage-like genotypes (i.e. from 0 to Inf). Use 'CNVgenotypeSNPlike' alongside for SNP-like CNV genotype in a separated node (i.e. '0, 1, 2, 3, 4' as '0/0, 0/1, 1/1, 1/2, 2/2').
- `coding.translate`: For 'CNVgenotypeSNPlike'. If NULL or unrecognized string use only biallelic CNVs. If 'all' code multiallelic CNVs as 0 for loss; 1 for 2n and 2 for gain.
- `n.cor`: Number of cores

Value

probes.cnv.gr Object with information about all probes to be used in the downstream CNV-GWAS. Only numeric chromosomes
importLrrBaf

**Author(s)**

Vinicius Henrique da Silva

**Examples**

```r
# Load phenotype-CNV information
data.dir <- system.file("extdata", package="CNVRanger")

phen.loc <- file.path(data.dir, "Pheno.txt")
cnv.out.loc <- file.path(data.dir, "CNVOut.txt")
map.loc <- file.path(data.dir, "MapPenn.txt")

phen.info <- setupCnvGWAS('Example', phen.loc, cnv.out.loc, map.loc)

# Construct the data-frame with integer and original chromosome names
# Define chr correspondence to numeric, if necessary
df <- '16 1A
  25 4A
  29 25LG1
  30 25LG2
  31 LGE22'

chr.code.name <- read.table(text=df, header=FALSE)
probes.cnv.gr <- generateGDS(phen.info, chr.code.name=chr.code.name)
```

**Description**

This function imports the LRR/BAF values and create a node for each one in the GDS file at the working folder 'Inputs' created by the `setupCnvGWAS` function. Once imported, the LRR values can be used to perform a GWAS directly as an alternative to copy number dosage.

**Usage**

```r
importLrrBaf(
  all.paths,  # path to the LRR/BAF files
  path.files,  # list of path files
  list.of.files,  # list of file names
  gds.file = NULL,  # GDS file for the result
  verbose = TRUE  # verbose output
)
```
importLrrBaf

Arguments

all.paths  
Folder containing the input CNV files used for the CNV calling (i.e. one text file with 5 columns for each sample). Columns should contain (i) probe name, (ii) Chromosome, (iii) Position, (iv) LRR, and (v) BAF.

path.files  
Object returned from CreateFolderTree function with the working folder tree

list.of.files  
Data-frame with two columns where the (i) is the file name with signals and (ii) is the correspondent name of the sample in the gds file

gds.file  
Path to the GDS file which contains nodes harboring respective LRR and BAF values. The 'snp.rs.id', 'sample.id', 'LRR' and 'BAF' nodes are mandatory. Both the SNPs and samples should follow the order and length in the CNV.gds (located at all.paths["Inputs"] folder). 'path.files' and ‘list.of.files’ will be ignored if ‘gds.file’ is not NULL.

verbose  
Print the samples while importing

Value

Writes to the specified GDS file by side effect.

Author(s)

Vinicius Henrique da Silva

Examples

# Load phenotype-CNV information
data.dir <- system.file("extdata", package="CNVRanger")

phen.loc <- file.path(data.dir, "Pheno.txt")
cnv.out.loc <- file.path(data.dir, "CNVOut.txt")
map.loc <- file.path(data.dir, "MapPenn.txt")

phen.info <- setupCnvGWAS('Example', phen.loc, cnv.out.loc, map.loc)

# Extract path names
all.paths <- phen.info$all.paths

# List files to import LRR/BAF
list.of.files <- list.files(path=data.dir, pattern="cnv.txt.adjusted$")
list.of.files <- as.data.frame(list.of.files)
colnames(list.of.files)[1] <- "file.names"
list.of.files$sample.names <- sub(".*cnv.txt.adjusted", ",", list.of.files$file.names)

# All missing samples will have LRR = '0' and BAF = '0.5' in all SNPs listed in the GDS file
importLrrBaf(all.paths, data.dir, list.of.files)

# Read the GDS to check if the LRR/BAF nodes were added
cnv.gds <- file.path(all.paths["Inputs"], 'CNV.gds')
genofile <- SNPRelate::snpgdsOpen(cnv.gds, allow.fork=TRUE, readonly=FALSE)
SNPRelate::snpgdsClose(genofile)
plotEQTL

Plot EQTL region

Description
Illustrates differential expression of genes in the neighborhood of a CNV.

Usage
plotEQTL(cnvr, genes, genome, cn = "CN1", cex = 0.8)

Arguments
cnvr
A GRanges of length 1, containing the genomic coordinates of the CNV region
of interest.
genes
GRanges containing genes in the neighborhood of the CNV region of interest.
genome
Character. A valid UCSC genome assembly ID such as 'hg19' or 'bosTau6'.
cn
Character. Copy number state of interest.
cex
A numerical value giving the amount by which gene names should be magnified.
Default is 0.8. Use smaller values to decrease font size.

Value
None. Plots to a graphics device.

Author(s)
Ludwig Geistlinger

See Also
Gviz::plotTracks

Examples
# CNV region of interest
cnvr <- GRanges("chr1:7908902-8336254")

# Two genes in the neighborhood
genes <- c("chr1:8021714-8045342:+", "chr1:8412464-8877699:-")
names(genes) <- c("PARK7", "RERE")
genes <- GRanges(genes)

# Annotate differential expression for 1-copy loss
genes$logFC.CN1 <- c(-0.635, -0.728)
genes$AdjPValue <- c(8.29e-09, 1.76e-08)

# plot
plotManhattan

plotEQTLEQTL(cnvr, genes, genome="hg19", cn="CN1")

plotManhattan Manhattan Plot

Description
Manhattan plot for p-values of a CNV-GWAS

Usage
plotManhattan(all.paths, regions, chr.size.order, plot.pdf = FALSE)

Arguments
all.paths Object returned from CreateFolderTree function with the working folder tree
regions GRanges as returned by cnvGWAS
chr.size.order data.frame with two columns: (i) 'chr': chromosome names (character), and
(ii) 'size': length of the chromosomes in bp (integer). A GRanges containing
one chromosome per range can be used instead (the chromosomes should be in
the expected order).
plot.pdf Logical plot a to pdf file

Value
Plots to graphics device.

Author(s)
Vinicius Henrique da Silva

Examples
# Load phenotype-CNV information
data.dir <- system.file("extdata", package="CNVRanger")

phen.loc <- file.path(data.dir, "Pheno.txt")
cnv.out.loc <- file.path(data.dir, "CNVOut.txt")
map.loc <- file.path(data.dir, "MapPenn.txt")

phen.info <- setupCnvGWAS('Example', phen.loc, cnv.out.loc, map.loc)
all.paths <- phen.info$all.paths
segs.pvalue.gr <- cnvGWAS(phen.info)

# Define the chromosome order in the plot
order.chrs <- c(1:24, "25LG1", "25LG2", 27:28, "LGE22", "1A", "4A")
# Chromosome sizes
chr.size.file <- file.path(data.dir, "Parus_major_chr_sizes.txt")
chr.sizes <- scan(chr.size.file)
chr.size.order <- data.frame(chr=order.chrs, sizes=chr.sizes, stringsAsFactors=FALSE)

# Plot Manhatthan to a pdf within the 'Results' workfolder
plotManhattan(all.paths, segs.pvalue.gr, chr.size.order)

---

**plotRecurrentRegions**  
*Plot recurrent CNV regions*

**Description**
Illustrates summarized CNV regions along a chromosome.

**Usage**
plotRecurrentRegions(regs, genome, chr, pthresh = 0.05)

**Arguments**
- **regs** A GRanges. Typically the result of `populationRanges` with `est.recur=TRUE`.
- **genome** Character. A valid UCSC genome assembly ID such as 'hg19' or 'bosTau6'.
- **chr** Character. A UCSC-style chromosome name such as 'chr1'.
- **pthresh** Numeric. Significance threshold for recurrence. Defaults to 0.05.

**Value**
None. Plots to a graphics device.

**Author(s)**
Ludwig Geistlinger

**See Also**
Gviz::plotTracks

**Examples**

```r
# read in example CNV calls
data.dir <- system.file("extdata", package="CNVRanger")
call.file <- file.path(data.dir, "Silva16_PONE_CNV_calls.csv")
calls <- read.csv(call.file, as.is=TRUE)

# store in a GRangesList
grl <- GenomicRanges::makeGRangesListFromDataFrame(calls,
```
populationRanges

Summarizing CNV ranges across a population

Description

In CNV analysis, it is often of interest to summarize individual calls across the population, (i.e. to define CNV regions), for subsequent association analysis with e.g. phenotype data.

Usage

populationRanges(
  grl,
  mode = c("density", "RO"),
  density = 0.1,
  ro.thresh = 0.5,
  multi.assign = FALSE,
  verbose = FALSE,
  min.size = 2,
  classify.ranges = TRUE,
  type.thresh = 0.1,
  est.recur = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grl</td>
<td>A GRangesList.</td>
</tr>
<tr>
<td>mode</td>
<td>Character. Should population ranges be computed based on regional density (&quot;density&quot;) or reciprocal overlap (&quot;RO&quot;). See Details.</td>
</tr>
<tr>
<td>density</td>
<td>Numeric. Defaults to 0.1.</td>
</tr>
<tr>
<td>ro.thresh</td>
<td>Numeric. Threshold for reciprocal overlap required for merging two overlapping regions. Defaults to 0.5.</td>
</tr>
<tr>
<td>multi.assign</td>
<td>Logical. Allow regions to be assigned to several region clusters? Defaults to FALSE.</td>
</tr>
<tr>
<td>verbose</td>
<td>Logical. Report progress messages? Defaults to FALSE.</td>
</tr>
<tr>
<td>min.size</td>
<td>Numeric. Minimum size of a summarized region to be included. Defaults to 2 bp.</td>
</tr>
</tbody>
</table>

split.field="NE_id", keep.extra.columns=TRUE)

# summarize CNV regions
cnvsr <- populationRanges(grl, density=0.1, est.recur=TRUE)

# plot
plotRecurrentRegions(cnvsrs, genome="bosTau6", chr="chr1")
classify.ranges Logical. Should CNV frequency (number of samples overlapping the region) and CNV type (gain, loss, or both) be annotated? Defaults to TRUE.

type.thresh Numeric. Required minimum relative frequency of each CNV type (gain / loss) to be taken into account when assigning CNV type to a region. Defaults to 0.1. That means for a region overlapped by individual gain and loss calls that both types must be present in >10 in order to be typed as 'both'. If gain or loss calls are present below the threshold they are ignored.

est.recur Logical. Should recurrence of regions be assessed via a permutation test? Defaults to FALSE. See Details.

Details

• CNVRuler procedure that trims region margins based on regional density
  Trims low-density areas (usually <10% of the total contributing individual calls within a summarized region).
  An illustration of the concept can be found here: https://www.ncbi.nlm.nih.gov/pubmed/22539667 (Figure 1)

• Reciprocal overlap (RO) approach (e.g. Conrad et al., Nature, 2010)
  Reciprocal overlap of 0.51 between two genomic regions A and B:
  requires that B overlaps at least 51% of A, *and* that A also overlaps at least 51% of B
  Approach:
  At the top level of the hierarchy, all contiguous bases overlapping at least 1bp of individual calls are merged into one region. Within each region, we further define reciprocally overlapping regions with the following algorithm:
    – Calculate reciprocal overlap (RO) between all remaining calls.
    – Identify pair of calls with greatest RO. If RO > threshold, merge and create a new CNV. If not, exit.
    – Continue adding unclustered calls to the region, in order of best overlap. In order to add a call, the new call must have > threshold to all calls within the region to be added. When no additional calls may be added, move to next step.
    – If calls remain, return to 1. Otherwise exit.

• GISTIC procedure (Beroukhim et al., PNAS, 2007) to identify recurrent CNV regions
  GISTIC scores each CNV region with a G-score that is proportional to the total magnitude of CNV calls in each CNV region. In addition, by permuting the locations in each sample, GISTIC determines the frequency with which a given score would be attained if the events were due to chance and therefore randomly distributed. A significance threshold can then be used to determine scores / regions that are unlikely to occur by chance alone.

Value

A GRanges object containing the summarized CNV ranges.

Author(s)

Ludwig Geistlinger, Martin Morgan
setupCnvGWAS

References


See Also

findOverlaps

Examples

grl <- GRangesList(
    sample1 = GRanges( c(“chr1:1-10”, “chr2:15-18”, “chr2:25-34” ) ),
    sample2 = GRanges( c(“chr1:1-10”, “chr2:11-18”, “chr2:25-36” ) ),
    sample3 = GRanges( c(“chr1:2-11”, “chr2:14-18”, “chr2:26-36” ) ),
    sample4 = GRanges( c(“chr1:1-12”, “chr2:18-35” ) ),
    sample5 = GRanges( c(“chr1:1-12”, “chr2:11-17”, “chr2:26-34” ) ),
    sample6 = GRanges( c(“chr1:1-12”, “chr2:12-18”, “chr2:25-35” ) )
)

# default as chosen in the original CNVRuler procedure
populationRanges(grl, density=0.1, classify.ranges=FALSE)

# density = 0 merges all overlapping regions,
# equivalent to: reduce(unlist(grl))
populationRanges(grl, density=0, classify.ranges=FALSE)

# density = 1 disjoins all overlapping regions,
# equivalent to: disjoin(unlist(grl))
populationRanges(grl, density=1, classify.ranges=FALSE)

# RO procedure
populationRanges(grl, mode=”RO”, ro.thresh=0.5, classify.ranges=FALSE)

setupCnvGWAS

Setup the folders and files to run CNV-GWAS analysis

Description

This function creates the (i) necessary folders in disk to perform downstream analysis on CNV genome-wide association and (ii) import the necessary input files (i.e. phenotypes, probe map and CNV list) from other locations in disk.
setupCnvGWAS

Usage

setupCnvGWAS(
  name,
  phen.loc,
  cnv.out.loc,
  map.loc = NULL,
  folder = NULL,
  pops.names = NULL,
  n.cor = 1
)

Arguments

name          String with a project code or name (e.g. 'Project1')
phen.loc      Path/paths to the tab separated text file containing phenotype and sample info. When using more than one population, for populations without phenotypes include the string 'INEXISTENT' instead the path for a file.
cnv.out.loc   Path(s) to the CNV analysis output (i.e. PennCNV output, SNP-chip general format or sequencing general format). It is also possible to use a RaggedExperiment or a GRangesList object instead if the run includes only one population.
map.loc       Path to the probe map (e.g. used in PennCNV analysis). Column names containing probe name, chromosome and coordinate must be named as: Name, Chr and Position. Tab delimited. If NULL, artificial probes will be generated based on the CNV breakpoints.
folder        Choose manually the project folder (i.e. path as the root folder). Otherwise, user-specific data dir will be used automatically.
pops.names    Indicate the name of the populations, if using more than one.
n.cor         Number of cores

Details

The user can import several phenotypes at once. All information will be stored in the list returned by this function. The user should be aware although several phenotypes can be imported, the cnvGWAS or generateGDS functions will handle only one phenotype per run.

Value

List 'phen.info' with 'samplesPhen', 'phenotypes', 'phenotypesdf', 'phenotypesSam', 'FamID', 'SexIds', 'pops.names' (if more than one population) and 'all.paths'

Author(s)

Vinicius Henrique da Silva
Examples

```r
data.dir <- system.file("extdata", package="CNVRanger")

phen.loc <- file.path(data.dir, "Pheno.txt")
cnv.out.loc <- file.path(data.dir, "CNVOut.txt")
map.loc <- file.path(data.dir, "MapPenn.txt")

phen.info <- setupCnvGWAS("Example", phen.loc, cnv.out.loc, map.loc)
```
Index

cnvEQTl, 2
cnvGWAS, 5, 13, 18
 cnvOncoPrint, 7

data.frame, 13
 DataFrame, 4

filterByExpr, 3
findOverlaps, 4, 17

generateGDS, 9, 18
glmQLFit, 4
glmQLFTest, 4
GRanges, 3, 8, 12–14, 16
GRangesList, 3, 7, 15, 18

importLrrBaf, 10

lm, 5

p.adjust, 3, 6
plotEQTl, 12
plotManhattan, 13
plotRecurrentRegions, 14
populationRanges, 3, 14, 15

qreduceAssay, 3, 4, 8

RaggedExperiment, 3, 7, 18
RangedSummarizedExperiment, 3

setupCnvGWAS, 5, 6, 10, 17