Package 'SARC'

April 4, 2025

Type Package

Title Statistical Analysis of Regions with CNVs

Version 1.5.0

Description Imports a cov/coverage file (normalised read coverages from BAM files) and a cnv file (list of CNVs - similiar to a BED file) from WES/ WGS CNV (copy number variation) detection pipelines and utilises several metrics to weigh the likelihood of a sample containing a detected CNV being a true CNV or a false positive. Highly useful for diagnostic testing to filter out false positives to provide clinicians with fewer variants to interpret. SARC uniquely only used cov and csv (similiar to BED file) files which are the common CNV pipeline calling filetypes, and can be used as to supplement the Interactive Genome Browser (IGV) to generate many figures automatedly, which can be especially helpful in large cohorts with 100s-1000s of patients.

Encoding UTF-8

RoxygenNote 7.2.3

License GPL-3

Imports tidyverse, utils, reshape2, DescTools, metap, multtest, plyranges, data.table, scales, RColorBrewer, grid, gtable, gridExtra, GenomicFeatures, stats, ggplot2, plotly, IRanges

Depends R (>= 4.3), RaggedExperiment, GenomicRanges

Suggests knitr, kableExtra, testthat, TxDb.Hsapiens.UCSC.hg38.knownGene, Homo.sapiens, TxDb.Mmusculus.UCSC.mm10.knownGene, Mus.musculus, GenomicAlignments

biocViews Software, CopyNumberVariation, Visualization, DNASeq, Sequencing

VignetteBuilder knitr

URL https://github.com/Krutik6/SARC/

BugReports https://github.com/Krutik6/SARC/issues

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

git_branch devel

git_last_commit 4be56bd

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-04-03

Author Krutik Patel [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6806-8675>)

Maintainer Krutik Patel <nkp68@newcastle.ac.uk>

Contents

addExonsGenes 2
anovaOnCNV 3
cnvConfidence 4
pasteExonsGenes 6
phDunnetonCNV 7
plotCNV 8
plotCovPrep 10
prepAnova 11
regionGrangeMake 12
regionMean 13
regionQuantiles 14
regionSet 15
regionSplit 16
seeDist 16
setQDplot 17
setupCNVplot 18
test_cnv 19
test_cnv2 21
test_cov 22

Index **24**

`addExonsGenes` *addExonsGenes*

Description

For the length of the CNV (+padding, if this was done), exons and gene symbol information will be attached in each Grange object.

Usage

```
addExonsGenes(RE, covgranges, txdb, txgene)
```

Arguments

RE	RaggedExperiment object used to store all information.
covgranges	List of grange objects created by the SARC::regionGrangeMake function. This should be found in the metadata of the RE object.
txdb	Loaded txDB object e.g. TxDb.Hsapiens.UCSC.hg38.knownGene. Standard usage from the GenomicFeatures tutorial https://kasperdanielhansen.github.io/genbioconductor/html/Gen
txgene	List of genes and exons associated with each genomic region of interest. Requires specific species package to make e.g. Homo.sapiens for human samples.

Value

A list of grange objects with exons and genes added. One grange object for each detected CNV. Will be stored in the RE object provided as metadata.

Examples

```
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
require("TxDb.Hsapiens.UCSC.hg38.knownGene")
} else {}

if (requireNamespace("Homo.sapiens", quietly = TRUE)) {
require("Homo.sapiens")
} else {}

data("test_cnv")
test_cnv <- test_cnv[c(1),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- plotCovPrep(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionGrangeMake(RE = SARC, covprepped = metadata(SARC)[[4]])

TxDb(Homo.sapiens) <- TxDb.Hsapiens.UCSC.hg38.knownGene
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
tx <- transcriptsBy(Homo.sapiens, columns = "SYMBOL")
txgene <- tx@unlistData
SARC <- addExonsGenes(RE = SARC, covgranges = metadata(SARC)[[5]],
                      txdb = txdb, txgene = txgene)
```

---

anovaOnCNV *anovaOnCNV*

Description

Apply anova on the CNV data. For each CNV listed in the input cnv file, the read-depths will be used to calculate if this region of the DNA, in this sample, is significantly different from other samples in the cohort of samples, at this same region.

Usage

```
anovaOnCNV(RE, cnv, anovacov, nameofnewdf = "CNVanova")
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines.
anovacov	List of dataframes which contain the required information to perform anova for each region with a detected CNV. Generated from prepAnova, and stored as metadata.
nameofnewdf	Name of new dataframe to be saved in metadata(RE)[['CNVlist']]. Default is CNVanova.

Value

A new cnv dataframe with an additional column for the p-value from anova. Will be stored as an experiment in the RE object.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- prepAnova(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                  start = metadata(SARC)[[2]], end = metadata(SARC)[[3]])
SARC <- anovaOnCNV(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                   anovacov = metadata(SARC)[[4]])
```

---

cnvConfidence *cnvConfidence*

Description

Flags the confidence of the CNV being a true CNV. The flags range from Very Unconfident - Very Confident. Our clinicians preferred variants to be flagged rather than filtered out - so we simply do this. Confidence scores will be added to the cnv file given. The cnv file should be one which contains all the statistical results from other functions of the SARC package.

Usage

```
cnvConfidence(
  RE,
  cnv,
  ph = FALSE,
  m1 = 0.2,
  m2 = 0.8,
  m3 = 1.2,
  m4 = 1.8,
  nameofnewdf = "CNVrank"
)
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. Must use one which contains at least MeanScore, Qlow, Qhigh and anova p-values. If Post-hoc tests have also been performed, these results can also be used by this function.
ph	If Dunnet tests was performed, the p-value from the tests can be taken into account. Default is FALSE.
m1	Value used to cut-off the highest mean score allowed for HOMOZYGOUS DELECTIONS, and lowest cut-off for HETEROZYGOUS DELETIONS. Default is 0.2.
m2	Value used to cut-off the highest mean score allowed for HETEROZYGOUS DELETINS. Default is 0.8.
m3	Value used to cut-off the lowest mean score allowed for HETEROZYGOUS DUPLICATIONS. Default is 1.2.
m4	Value used to cut-off the lowest mean score allowed for HETEROZYGOUS DUPLICATIONS. Default is 1.8.
nameofnewdf	Name of new dataframe to be saved in metadata(RE)[['CNVlist']]. Default is CNVrank.

Value

A new cnv file with additional columns which describe our level of confidence of the detected CNV being a true CNV.

Examples

```
data("test_cnv")
test_cnv <- test_cnv[c(1:3),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                     endlist = metadata(SARC)[[3]])
SARC <- regionMean(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
```

```
                  splitcov = metadata(SARC)[[4]])
SARC <- regionQuantiles(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[2]],
                        meancov = metadata(SARC)[[3]], q1 =.1, q2 = .9)
SARC <- prepAnova(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[3]],
                 start = metadata(SARC)[[2]], end=metadata(SARC)[[3]])
SARC <- anovaOnCNV(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[3]],
                  anovacov = metadata(SARC)[[8]])
SARC <- cnvConfidence(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[4]])
```

---

pasteExonsGenes *pasteExonsGenes*

Description

Add Genes and Exons to the cnv file in order to show the range the detected CNV extends. This is useful information for clinicians/ diagnostic scientists when interpreting the variants. Note - if padding was performed during plotCovPrep step, the CNV range would be extended by the padding so extra exons may be included in these cases.

Usage

```
pasteExonsGenes(RE, setup, cnv, nameofnewdf = "CNVexonsgenes")
```

Arguments

RE	RaggedExperiment object used to store all information.
setup	List of dataframes which have been processed for plotting. This will be stored as metadata in the RE object after SARC::setupCNVPlot.
cnv	cnv file containing CNVs which the user wishes to generate plots for. It is recommended that the most recently created cnv file is used. Check print(RE) to see more cnv files created by SARC.
nameofnewdf	Name of new dataframe to be saved in metadata(RE)[['CNVlist']]. Default is CNVexonsgenes.

Value

A new cnv file with an additional column which lists the genes and exons which the detected variant ranges.

Examples

```
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
require("TxDb.Hsapiens.UCSC.hg38.knownGene")
} else {}

if (requireNamespace("Homo.sapiens", quietly = TRUE)) {
require("Homo.sapiens")
} else {}
```

```
data("test_cnv")
test_cnv <- test_cnv[c(1),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv =  metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- plotCovPrep(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionGrangeMake(RE = SARC, covprepped = metadata(SARC)[[4]])

TxDb(Homo.sapiens) <- TxDb.Hsapiens.UCSC.hg38.knownGene
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
tx <- transcriptsBy(Homo.sapiens, columns = "SYMBOL")
txgene <- tx@unlistData

SARC <- addExonsGenes(RE = SARC, covgranges = metadata(SARC)[[6]],
                      txdb = txdb, txgene = txgene)
SARC <- setupCNVplot(RE = SARC, namedgranges =  metadata(SARC)[[7]],
                  covprepped = metadata(SARC)[[4]])
SARC <- pasteExonsGenes(RE = SARC, setup =  metadata(SARC)[[8]],
                        cnv = metadata(SARC)[['CNVlist']][[1]])
```

---

phDunnetonCNV *phDunnetonCNV*

Description

Applies post-hoc test Dunnet test on each CNV in the cnv file. Performs a pair-wise test between the sample where the CNV was detected and each other sample at the same genomic region. WARNING - very slow, only use if a small number of CNVs / samples are available. Other tests e.g. Quantile Distribution are more effective with larger cohorts.

Usage

```
phDunnetonCNV(RE, cnv, anovacov, nameofnewdf = "CNVdunnet")
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recommended to use the most recently created cnv file. Use print(RE) to check.
anovacov	List of dataframes which contain the required information to perform anova for each region with a detected CNV. Generated from prepAnova.
nameofnewdf	Name of new dataframe to be saved in metadata(RE)[['CNVlist']]. Default is CNVdunnet.

Value

A new cnv dataframe with an additional column of the sumlog p-value from each dunnet test performed for the CNV entry.

---

plotCNV *plotCNV*

Description

Plots of the region of the DNA from several samples where a CNV is detected. The sample with the detected CNV, from the cnv file, will be highlighted in purple. Additional specificity such as gene of interest, or batch of WES/ WGS can be specified. This function is made to be easily looped for multiple CNVs. It is also made as a quick and more visually preferable alternative to looking for true CNVs visually.

Usage

```
plotCNV(
  cnv,
  setup,
  FilteredCNV = 1,
  batch = NULL,
  nSamples = NULL,
  gene = NULL,
  log = NULL,
  logbase = 2
)
```

Arguments

cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recommended to use the most recently generated cnv file. Check print(RE) to see if cnv files from other analyses have been generated.
setup	ist of dataframes which have been processed for plotting. This will be stored as metadata in the RE object after SARC::setupCNVPlot.
FilteredCNV	Which CNV from the list of dataframes (setup) should be plotted. Can be automated within a for-loop. Default is 1.
batch	If WES/ WGS was performed in batches, users may wish to plot samples from different batches separately. In this case, a column called 'BATCH' should be found in the cnv file. Different batches should be labelled with differentiating strings, and this string should be added to the batch parameter. Default is NULL.
nSamples	Numbers of samples to be printed in the plots. Default is 10. Add an integer to represent how many subplots should be made. The higher the number, the more difficult the plot will be for viewing.

gene	String of a gene of interest. Only this gene will be printed. It must match standard gene symbols e.g. FCGR3A, P53. Only one string should be inputted at a time. The default is null.
log	Applies log normalisation to reads. This can be helpful in WES data, where reads can be quite disperse across a short region of DNA. Recommended is 2, and default is NULL.
logbase	Base to log read depth by - default is 2.

Value

A grid plot showing the read-depths for a specific regions of the the genome. This region will contain one samples which will have had a CNV detected. Heat-plot like colouring will be used to visually show if the sample has a significant change at this region, in contrast to several other samples.

Examples

```
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
require("TxDb.Hsapiens.UCSC.hg38.knownGene")
} else {}

if (requireNamespace("Homo.sapiens", quietly = TRUE)) {
require("Homo.sapiens")
} else {}

data("test_cnv")
test_cnv <- test_cnv[c(1),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv =  metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- plotCovPrep(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionGrangeMake(RE = SARC, covprepped = metadata(SARC)[[4]])

TxDb(Homo.sapiens) <- TxDb.Hsapiens.UCSC.hg38.knownGene
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
tx <- transcriptsBy(Homo.sapiens, columns = "SYMBOL")
txgene <- tx@unlistData

SARC <- addExonsGenes(RE = SARC, covgranges = metadata(SARC)[[6]],
                      txdb = txdb, txgene = txgene)

SARC <- setupCNVplot(RE = SARC, namedgranges =  metadata(SARC)[[7]],
                  covprepped = metadata(SARC)[[4]])
p <- plotCNV(cnv =  metadata(SARC)[['CNVlist']][[1]],
             setup = metadata(SARC)[[8]], FilteredCNV=1)
```

plotCovPrep *plotCovPrep*

Description

Prepares plotting of coverage at specific regions of the DNA - where CNVs have been detected.

Usage

```
plotCovPrep(RE, cnv, n1 = 0, n2 = 0, startlist, endlist)
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. Must use one which contains at least MeanScore, Qlow, Qhigh and anova p-values. If Post-hoc tests have also been performed, these results can also be used by this function.
n1	How many rows to pad the start site of each CNV - default is 0.
n2	How many rows to pad the end site of each CNV - default is 0.
startlist	List of start sites created from SARC::regionSet and stored as metadata.
endlist	List of end sites created from SARC::regionSet and stored as metadata.

Value

A new list of dataframes, each df is a region where a CNV was detected.

Examples

```
data("test_cnv")
test_cnv <- test_cnv[c(1:3),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- plotCovPrep(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
```

prepAnova *prepAnova*

Description

Function for annovaOnCNV.This function sets up samples for anova analysis. Can also be used as input for Dunnet analysis.

Usage

```
prepAnova(RE, cnv, startlist, endlist)
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recommended that the most recently created cnv file is used. Check print(RE) to see more cnv files created by SARC.
startlist	List of start sites created from SARC::regionSet and stored as metadata.
endlist	List of end sites created from SARC::regionSet and stored as metadata.

Value

Additional list of dataframes which will be needed to calculate anova for each region with a detected CNV.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionMean(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                   splitcov = metadata(SARC)[[4]])
SARC <- regionQuantiles(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[2]],
                      meancov = metadata(SARC)[[5]], q1=.1, q2=.9)
SARC <- prepAnova(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[3]],
                  startlist = metadata(SARC)[[2]],
                  endlist = metadata(SARC)[[3]])
```

regionGrangeMake *regionGrangeMake*

Description

Makes Grange objects for each region with a detected CNV.

Usage

```
regionGrangeMake(RE, covprepped, range = 10, gap = 10)
```

Arguments

RE	RaggedExperiment object used to store all information.
covprepped	List of dataframes, each dataframe is small cov file which ranges the detected CNV. This should be the padded cov file created with SARC::plotCovPrep and will be in the matadata.
range	Number of rows (genomic ranges) to group together as one entry for grange. As some WES platforms will have very short genomic start-end coordinates, this is useful to extract the exon/ gene data. Default is 10.
gap	The gap between the start of one grange entry to the next entry. Default is 10.

Value

List of Grange Objects, one for each CNV detected.

Examples

```
data("test_cnv")
test_cnv <- test_cnv[c(1:3),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv =  metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- plotCovPrep(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionGrangeMake(RE = SARC, covprepped = metadata(SARC)[[4]])
```

regionMean *regionMean*

Description

Function to calculate MeanScores for each region where a CNV has been detected. Dependent of sequencing quality and WES/WGS input, the estimated MeanScores for CNV types are generally: HOMOZYGOUS DELETION 0-0.3, HETEROZYGOUS DELETION 0.3-0.7, HETEROZYGOUS DUPLICATION 1.3-1.8 HOMOZYGOUS DUPLICATION >1.8.

Usage

```
regionMean(RE, cnv, splitcov, nameofnewdf = "CNVmeans")
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recomended to use the most recently generated cnv file. Check print(RE) to see if cnv files from other anlyses have been generated.
splitcov	List of small .cov dataframes. Would be stored as metadata after SARC::regionSplit.
nameofnewdf	Name of new dataframe to be saved in metadata(RE)[['CNVlist']]. Default is CNVmeans.

Value

A new cnv file with an additional column representing the meanScores. A new list of dataframes which contains the meanscores.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionMean(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                   splitcov = metadata(SARC)[[4]])
```

regionQuantiles *regionQuantiles*

Description

Calculates quantile distribution from each sample, within a region with a suspected CNV. We assume true deletions would be in the lower end of the quantile distribution and true duplications to be in the upper end of the quantile distribution. Many CNVs captured by detection pipelines will be false positives, this this is a good metric to remove obvious false positives. Lower quantile (q1) and upper quantile (q2) should be adjusted based on the number of patient samples available.

Usage

```
regionQuantiles(
  RE,
  cnv,
  meancov,
  q1 = 0.05,
  q2 = 0.95,
  nameofnewdf = "CNVquantiles"
)
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recommended to use the most recently generated cnv file. Check print(RE) to see if cnv files from other analyses have been generated.
meancov	Small cov files with meanscores included. This will be stored as metadata after being generated by SARC::regionMean.
q1	Lower quantile cut-off. Default is .05. This should be adjusted based on the number of samples, i.e. less samples > higher the lower threshold.
q2	Upper quantile cut-off. Default is .95. This should be adjusted based on the number of samples, i.e. less samples > lower the higher threshold.
nameofnewdf	Name of new dataframe to be saved in metadata(RE)[['CNVlist']]. Default is CNVquantiles.

Value

A new cnv file with two new columns. For deletions the Qlow columns will be 0 or 1, and for duplications the Qhigh columns will be 0 or 1.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionMean(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                   splitcov = metadata(SARC)[[4]])
SARC <- regionQuantiles(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[2]],
                      meancov = metadata(SARC)[[5]], q1=.1, q2=.9)
```

`regionSet` *regionSet*

Description

Stores .cov and .cnv file (BED inspired file) into a RaggedExperiment object. Will use genomic locations from the .cnv file to find the start and end points of the CNVs from the .cov file and store these as two lists. The assay of the RE object will store the cov file and a list of nested dataframes called "CNVlist" within the metadata will store all the cnv dataframes.

Usage

```
regionSet(cnv, cov, col = "experiment")
```

Arguments

`cnv`	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. Additional columns such as BATCH and GENE can also be used for better plotting.
`cov`	cov file from WES platform/ sequencer kit or if WGS regular intervals. Stored as a dataframe - genomic locations as rows and samples as columns. It is recommended to normalize Read Depth by library size for fairer comparisons.
`col`	colData - A string which is used in RaggedExperiment creation. Default is "experiment".

Value

RaggedExperiment object with two lists: Start sites and End sites.The cov file will be stored as the main assay of the RE object, and a new list will appear in the metadata to contain all cnv dataframes.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
```

regionSplit *regionSplit*

Description

Splits the large cov file into multiple smaller cov files. Each one is specific for each CNV entry in the bam file.

Usage

```
regionSplit(RE, cnv, startlist, endlist)
```

Arguments

RE	RaggedExperiment object used to store all information.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recommended that the most recently created cnv file is used. Check print(RE) to see more cnv files created by SARC.
startlist	List of start sites created from SARC::regionSet and stored as metadata.
endlist	List of end sites created from SARC::regionSet and stored as metadata.

Value

A list of dataframes - each one a smaller cov file.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv =  metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
```

seeDist *seeDist*

Description

Displays the distribution of mean scores from read depths from all the samples. This is a quick method of checking why some detected CNVs might be false positives. We expect true duplications to have very high read-depths and true deletions to have very low read-depths.

Usage

```
seeDist(meanList, cnv, sample, plotly = FALSE, colourCNV = "red", size = 2)
```

Arguments

meanList	List of dataframes which show the ranked mean scores for each CNV. Stored in metadata after SARC::setDPlot.
cnv	List of CNVs in a dataframe containing CNVs from detection algorithms/ pipelines. It is recommended that the most recently created cnv file is used. Check print(RE) to see more cnv files created by SARC.
sample	Which CNV/ row from the cnv file should be checked. Default is 1.
plotly	Should plotly be used - this could be useful when interested in seeing the samples at each point.
colourCNV	Colour of the sample which had the CNV detected. Default is red.
size	Size if the dots. Default is 2.

Value

A scatter graph which shows the mean score of read-depths for a particular region of DNA where a CNV was detected. The sample which had the CNV detected is coloured as red/ used selected colour.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[[1]][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionMean(RE = SARC, cnv = metadata(SARC)[[1]][[1]],
                   splitcov = metadata(SARC)[[4]])
SARC <- setQDplot(RE = SARC, meancov = metadata(SARC)[[5]])
p <- seeDist(meanList = metadata(SARC)[[6]], cnv = metadata(SARC)[[1]][[2]],
 plotly=FALSE, sample=1)
```

---

setQDplot *setQDPlot*

Description

Quality checking plot to see the Distribution of mean scores at a given region of the genome. The sample with the suspected CNV is highlighted. Useful way of checking why a certain CNV may be a false positive.

Usage

```
setQDplot(RE, meancov)
```

Arguments

RE	RaggedExperiment object used to store all information.
meancov	Small cov files with mean scores included. This will be stored as metadata after being generated by SARC::regionMean.

Value

A list of dataframes. Each dataframe contains ranked meanscores for each sample. One dataframe is made for each CNV in the cnv file.

Examples

```
data("test_cnv")
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv = metadata(SARC)[[1]][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionMean(RE = SARC, cnv = metadata(SARC)[[1]][[1]],
                   splitcov = metadata(SARC)[[4]])
SARC <- setQDplot(RE = SARC, meancov = metadata(SARC)[[5]])
```

---

setupCNVplot *setupCNVplot*

Description

Sets up dataframes for plotting the CNVs. If there are grange objects with no genes/ exons, it may be the CNVs are too small. These CNVs should be removed to not lead to errors.

Usage

```
setupCNVplot(RE, namedgranges, covprepped)
```

Arguments

RE	RaggedExperiment object used to store all information.
namedgranges	List of grange objects, one for each CNV. These objects should have exons and genes, and will be in metadata after being created by SARC::addExonsGenes.
covprepped	List of dataframes, one for each CNV. This list should be in metadata after bring created by SARC::plotCovPrep.

Value

List of dataframes, each dataframe can be plotted using plotting functions from this package.

Examples

```
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
require("TxDb.Hsapiens.UCSC.hg38.knownGene")
} else {}

if (requireNamespace("Homo.sapiens", quietly = TRUE)) {
require("Homo.sapiens")
} else {}

data("test_cnv")
test_cnv <- test_cnv[c(1),]
data("test_cov")
SARC <- regionSet(cnv = test_cnv, cov = test_cov)
SARC <- regionSplit(RE = SARC, cnv =  metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- plotCovPrep(RE = SARC, cnv = metadata(SARC)[['CNVlist']][[1]],
                    startlist = metadata(SARC)[[2]],
                    endlist = metadata(SARC)[[3]])
SARC <- regionGrangeMake(RE = SARC, covprepped = metadata(SARC)[[4]])

TxDb(Homo.sapiens) <- TxDb.Hsapiens.UCSC.hg38.knownGene
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
tx <- transcriptsBy(Homo.sapiens, columns = "SYMBOL")
txgene <- tx@unlistData

SARC <- addExonsGenes(RE = SARC, covgranges = metadata(SARC)[[6]],
                      txdb = txdb, txgene = txgene)
SARC <- setupCNVplot(RE = SARC, namedgranges =  metadata(SARC)[[7]],
                 covprepped = metadata(SARC)[[4]])
```

---

test_cnv *test cnv*

Description

This is not a traditional bed file, but shares the purpose of a bed file during CNV analysis - which is to store a list of CNVs detected from CNV detection algorithms/ pipelines. This files shares the first three columns of a traditional bed file (chrom, chromstart, chromend), and in addition has several more columns e.g. sample, type,value/ score and tool. The column names must match: SAMPLE, CHROM, START, END, TYPE, VALUE - but only the order of the first three columns must remain the same.

Usage

```
data("test_cnv")
```

Format

A data frame with 15 observations on the following 7 variables.

`SAMPLE` a character vector

`CHROM` a factor with levels `chr1` `chr10` `chr11` `chr12` `chr14` `chr15` `chr16` `chr17` `chr18` `chr19` `chr2` `chr20` `chr22` `chr3` `chr4` `chr5` `chr6` `chr7` `chr8` `chr9` `chrX` `chrY`

`START` a numeric vector

`END` a numeric vector

`TYPE` a factor with levels `DEL` `DUP`

`VALUE` a numeric vector

`TOOL` a factor with levels `clinCNV` `cn.mops` `exomeDepth`

Details

Several annonymised whole exome sequencing samples which have been put through a Copy Number Variaiton pipleine and had their CNVs recorded in a way which resembles a bed file. This file has additional columns too.

Value

Dataframe listing CNVs from a WES CNV detection pipeline.

Source

This is annonmysed patient data from the Newcastle Mitochondrial Research group, from the Wellcome Center for Mitochondrial research, Newcastle Universiry, UK.

References

Taylor, Robert W., et al. "Use of whole-exome sequencing to determine the genetic basis of multiple mitochondrial respiratory chain complex deficiencies." Jama 312.1 (2014): 68-77.

Examples

```
data(test_cnv)
## maybe str(test_cnv) ; plot(test_cnv) ...
```

test_cnv2 *test cnv 2*

Description

This is not a traditional bed file, but shares the purpose of a bed file during CNV analysis - which is to store a list of CNVs detected from CNV detection algorithms/ pipelines. This files shares the first three columns of a traditional bed file (chrom, chromstart, chromend), and in addition has several more columns e.g. sample, type,value/ score and tool. The column names must match: SAMPLE, CHROM, START, END, TYPE, VALUE - but only the order of the first three columns must remain the same. In addition, columns for the batch and gene are included to improve visualisation.

Usage

```
data("test_cnv2")
```

Format

A data frame with 15 observations on the following 10 variables.

`SAMPLE` a character vector

`CHROM` a factor with levels `chr1` `chr10` `chr11` `chr12` `chr14` `chr15` `chr16` `chr17` `chr18` `chr19` `chr2` `chr20` `chr22` `chr3` `chr4` `chr5` `chr6` `chr7` `chr8` `chr9` `chrX` `chrY`

`START` a numeric vector

`END` a numeric vector

`TYPE` a factor with levels `DEL` `DUP`

`VALUE` a numeric vector

`TOOL` a factor with levels `clinCNV` `cn.mops` `exomeDepth`

`BATCH` a character vector

`GENE` a character vector

Details

Several annonymised whole exome sequencing samples which have been put through a Copy Number Variaiton pipleine and had their CNVs recorded in a way which resembles a bed file. This file has additional columns too.

Value

Dataframe listing CNVs from a WES CNV detection pipeline. Contains extra columns for plot specificity.

Source

This is annonmysed patient data from the Newcastle Mitochondrial Research group, from the Wellcome Center for Mitochondrial research, Newcastle Universiry, UK.

References

Taylor, Robert W., et al. "Use of whole-exome sequencing to determine the genetic basis of multiple mitochondrial respiratory chain complex deficiencies." Jama 312.1 (2014): 68-77.

Examples

```
data(test_cnv2)
## maybe str(test_cnv2) ; plot(test_cnv2) ...
```

`test_cov` *test coverage file*

Description

This file is a large matrix of raw read depth from whole exome sequencing BAM files. The first four columns are: ID, chromosome, start position and end position.Following on, each sample has its own column which gives an integer for each position of DNA found in the whole exome experiment. Samples are annoymised. Read depth has been normalised by total library sizes via RPM (reads per million) calculations.

Usage

```
data("test_cov")
```

Format

A data frame with 34987 observations on the following 24 variables.

`ID` a numeric vector

`CHROM` a factor with levels `chr1` `chr10` `chr11` `chr12` `chr13` `chr14` `chr15` `chr16` `chr17` `chr18` `chr19` `chr2` `chr20` `chr21` `chr22` `chr3` `chr4` `chr5` `chr6` `chr7` `chr8` `chr9` `chrM` `chrX` `chrY`

`START` a numeric vector

`END` a numeric vector

`SampleA` a numeric vector

`SampleB` a numeric vector

`SampleC` a numeric vector

`SampleD` a numeric vector

`SampleE` a numeric vector

`SampleF` a numeric vector

`SampleG` a numeric vector

`SampleH` a numeric vector

`SampleI` a numeric vector

`SampleJ` a numeric vector

SampleK a numeric vector

SampleL a numeric vector

SampleM a numeric vector

SampleN a numeric vector

SampleO a numeric vector

SampleP a numeric vector

SampleQ a numeric vector

SampleR a numeric vector

SampleS a numeric vector

SampleT a numeric vector

Details

Several annonymised whole exome sequencing samples had been converted to a bam file and had a read counting tool applied.Only values from chromosome 1 are present to save space and time in examples.

Value

Dataframe of normalised coverage from WES samples.

Source

This is annonmysed patient data from the Newcastle Mitochondrial Research group, from the Wellcome Center for Mitochondrial research, Newcastle Universiry, UK.

References

Taylor, Robert W., et al. "Use of whole-exome sequencing to determine the genetic basis of multiple mitochondrial respiratory chain complex deficiencies." Jama 312.1 (2014): 68-77.

Examples

```
data(test_cov)
## maybe str(test_cov) ; plot(test_cov) ...
```

Index

∗ **datasets**
test_cnv, 19
test_cnv2, 21
test_cov, 22

addExonsGenes, 2
anovaOnCNV, 3

cnvConfidence, 4

pasteExonsGenes, 6
phDunnetonCNV, 7
plotCNV, 8
plotCovPrep, 10
prepAnova, 11

regionGrangeMake, 12
regionMean, 13
regionQuantiles, 14
regionSet, 15
regionSplit, 16

seeDist, 16
setQDplot, 17
setupCNVplot, 18

test_cnv, 19
test_cnv2, 21
test_cov, 22