Package 'sveval'

July 19, 2023

Title SV evaluation

Version 2.3.0

Description Evaluate SV in a call set against a truth set using overlap-based approaches and sequence comparison for insertions.

Depends R (>= 3.4.4)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

LinkingTo Rcpp

Imports VariantAnnotation,
GenomicRanges,
IRanges,
magrittr,
dplyr,
rlang,
DelayedArray,
Biostrings,
GenomeInfoDb,
parallel,
testthat,
tidyr,
ggplot2,
shiny,
DiagrammeR,
S4Vectors,
DT,
igraph,
Rcpp,
logging,
Matrix

R topics documented:

`sveval-package` *SV evaluation*

Description

Evaluate SV in a call set against a truth set using overlap-based approaches and sequence comparison for insertions.

Details

Package:	sveval
Type:	Package
Version:	2.2.0
Date:	2023-04-07
License:	MIT

Author(s)

Jean Monlong <jmonlong@ucsc.edu>

See Also

http://www.github.com/jmonlong/sveval

Examples

```
## Not run:
eval = svevalOl('calls.vcf', 'truth.vcf')
plot_prcurve(eval$curve)

# Comparing multiple methods
eval.1 = svevalOl('calls1.vcf', 'truth.vcf')
eval.2 = svevalOl('calls2.vcf', 'truth.vcf')
plot_prcurve(list(eval.1$curve, eval.2$curve), labels=c('method1', 'method2'))

## End(Not run)
```

---

clusterSVs *Cluster SVs based on overlap/similarity*

Description

Cluster SVs based on overlap/similarity

Usage

```
clusterSVs(
  svs.gr,
  min.rol = 0.8,
  max.ins.dist = 20,
  range.seq.comp = FALSE,
  ins.seq.comp = FALSE,
  simprep = NULL,
  nb.cores = 1,
  batch.maxsize = 5000,
  log.level = c("CRITICAL", "WARNING", "INFO")
)
```

Arguments

svs.gr	A GRanges with SVs (for example read by readSVvcf or readSVvcf.multisamps)
min.rol	minimum reciprocal overlap for deletions and other "ranges" SVs. Default is 0.1
max.ins.dist	maximum distance for insertions to be clustered.
range.seq.comp	compare sequence instead of overlapping deletions/inversion/etc. Default is FALSE.
ins.seq.comp	compare sequence instead of insertion sizes. Default is FALSE.

simprep	optional simple repeat annotation. Default is NULL. If non-NULL, GRanges to be used to
nb.cores	number of processors to use. Default is 1.
batch.maxsize	batch size to aim. To reduce memory usage, see Details. Default is 5000.
log.level	the level of information in the log. Default is "CRITICAL" (basically no log).

Details

SVs are overlapped with each other. A graph is then built where nodes (SVs) are connected is they overlap/match. A cluster is a component in this graph.

To reduce the memory usage, the SVs are first grossly clustered into batches. The actual clustering (and graph construction) is then performed separately for each batch, potentially in parallel.

Value

the svs.gr object annotated with two new columns:

svsite	the ID of the cluster (or SV site)
clique	is this cluster a clique, i.e. all SVs overlapping/matching all other SVs in the cluster

Author(s)

Jean Monlong

Examples

```
## Not run:

svs = readSVvcf('svs.vcf.gz', keep.ids=TRUE)
svs = clusterSVs(svs)


## End(Not run)
```

---

countAlleles *Count alleles in SV sites across samples*

Description

Read a VCF file and count alleles for SVs grouped by SV site. The `sv.sites` parameter is a list defining which SVs are assigned to which SV site. The SV ids are those created by `readSVvcf.multisamps` which is expected to be run first.

Usage

```
countAlleles(vcf.file, sv.sites, gq.instead = FALSE)
```

Arguments

vcf.file	the path to the VCF file
sv.sites	a list defining the SV sites (each element contains a vector with SV ids)
gq.instead	return a matrix with the minimum genotype quality for each site instead of the allele counts. Default is FALSE.

Value

a matrix with allele counts for each site (rows) and samples (columns)

Author(s)

Jean Monlong

---

explore_eval_svs *Interactive exploration of FP/FN/TP SVs*

Description

Opens a Shiny app with a dynamic table that contains FP, FN and TP SVs. Clicking on a SV (row in the table) generates a simplified representation of the variants in the region. Useful to explore the calls, the truth and what was called TP/FN/FP.

Usage

```
explore_eval_svs(
  eval.o,
  ucsc.genome = "hg38",
  graph.height = 400,
  eval.o.2 = NULL,
  run.names = c("run1", "run2")
)
```

Arguments

eval.o	output from svevalOl.R
ucsc.genome	the genome version for the UCSC Genome Browser automated link.
graph.height	height of the graph in the shiny app, in pixel. Default is 400
eval.o.2	(optional) output from another run. If provided the graph will show a second panel for this run (change names with run.names=.
run.names	names to use for the panels when a second input, gr.l.2, is provided. Vector of two characters.

Value

Starts a Shiny app in a web browser.

Author(s)

Jean Monlong

`filterSVs` *Filter SVs for size and regions of interest*

Description

Filter SVs for size and regions of interest

Usage

```
filterSVs(
  sv.gr,
  regions.gr = NULL,
  ol.prop = 0.5,
  min.size = 0,
  max.size = Inf,
  accepted.filters = NULL,
  mark.pass = FALSE
)
```

Arguments

`sv.gr`	the input SVs (e.g. read from `readSVvcf`)
`regions.gr`	the regions of interest. Ignored if NULL (default).
`ol.prop`	minimum proportion of sv.gr that must overlap regions.gr. Default is 0.5
`min.size`	the minimum SV size to be considered. Default 0.
`max.size`	the maximum SV size to be considered. Default is Inf.
`accepted.filters`	vector of the values of the FILTER field to keep. If NULL (default), all values are accepted
`mark.pass`	don't actually filter variants but mark the variants with a 'pass' columns. Default is FALSE.

Value

`if mark.pass=FALSE (default)`
: a subset of sv.gr that overlaps regions.gr or in the specified size range.

`if mark.pass=TRUE`
: the input sv.gr with a new column 'pass' to mark the variants to keep.

Author(s)

Jean Monlong

`findNocalls` *Find no-calls variants*

Description

Compare calls with a truth set and identifies which variants from the truth set specifically not called (genotype ./.).

Usage

```
findNocalls(
  calls.gr,
  truth.gr,
  max.ins.dist = 20,
  min.ol = 0.5,
  min.del.rol = 0.1,
  range.seq.comp = FALSE,
  ins.seq.comp = FALSE,
  nb.cores = 1,
  sample.name = NULL,
  check.inv = FALSE,
  method = c("coverage", "bipartite")
)
```

Arguments

`calls.gr`	call set. A GRanges or the path to a VCF file.
`truth.gr`	truth set. A GRanges or the path to a VCF file.
`max.ins.dist`	maximum distance for insertions to be clustered. Default is 20.
`min.ol`	the minimum overlap/coverage to be considered a match. Default is 0.5
`min.del.rol`	minimum reciprocal overlap for deletions. Default is 0.1
`range.seq.comp`	compare sequence instead of only overlapping deletions/inversions/etc. Default is FALSE.
`ins.seq.comp`	compare sequence instead of insertion sizes. Default is FALSE.
`nb.cores`	number of processors to use. Default is 1.
`sample.name`	the name of the sample to use if VCF files given as input. If NULL (default), use first sample.
`check.inv`	should the sequence of MNV be compared to identify inversions.
`method`	the method to annotate the overlap. Either 'coverage' (default) for the cumulative coverage (e.g. to deal with fragmented calls); or 'bipartite' for a 1-to-1 matching of variants in the calls and truth sets.

Details

Same overlapping strategy as in `svevalOl` although here no-calls are kept and there is no splitting by genotype.

Value

a data.frame with coordinates and variant ids from the truth set corresponding to no-calls.

Author(s)

Jean Monlong

`freqAnnotate` *Annotate SVs with frequency in catalog*

Description

Input SVs are matched to SVs in the catalog. Each SV is then annotated with the maximum frequency across all the variants in the catalog that match. Although the different overlap approaches could be used here, the 'reciprocal' overlap makes the more sense because we want to list all possible matches without penalizing over-matching (like 'bipartite' would) or include fragmented calls (like 'coverage' would) as they could be very variants with uncomparable frequencies.

Usage

```
freqAnnotate(
  svs,
  cat,
  min.ol = 0.5,
  min.del.rol = 0.1,
  max.ins.dist = 20,
  check.inv = FALSE,
  range.seq.comp = FALSE,
  ins.seq.comp = FALSE,
  out.vcf = NULL,
  freq.field = "AF",
  out.freq.field = "AFMAX",
  method = c("reciprocal", "coverage", "bipartite"),
  nb.cores = 1,
  log.level = c("CRITICAL", "WARNING", "INFO")
)
```

Arguments

`svs`	a VCF object with SVs to annotate.
`cat`	a VCF object with the SV catalog with frequency estimates.
`min.ol`	the minimum overlap/coverage to be considered a match. Default is 0.5
`min.del.rol`	minimum reciprocal overlap for deletions. Default is 0.1
`max.ins.dist`	maximum distance for insertions to be clustered. Default is 20.
`check.inv`	should the sequence of MNV be compared to identify inversions.

range.seq.comp	compare sequence instead of only overlapping deletions/inversions/etc. Default is FALSE.
ins.seq.comp	compare sequence instead of insertion sizes. Default is FALSE.
out.vcf	If non-NULL, write output to this VCF file.
freq.field	the field with the frequency estimate in the 'cat' input. Default is 'AF'.
out.freq.field	the new field's name. Default is 'AFMAX'
method	the method to annotate the overlap. Recommended is 'reciprocal' (default). See details.
nb.cores	number of processors to use. Default is 1.
log.level	the level of information in the log. Default is "CRITICAL" (basically no log).

Value

a GRanges object.

Author(s)

Jean Monlong

Examples

```
## Not run:
## From VCF files with output written to VCF file
freqAnnotate('calls.vcf', 'gnomad.vcf', out.vcf='calls.withFreq.vcf')

## Within R
calls.vcf = readSVvcf('calls.vcf', out.fmt="vcf")
cat.vcf = readSVvcf('gnomad.vcf', out.fmt="vcf")
calls.freq.vcf = freqAnnotate(calls.vcf, cat.vcf)

## End(Not run)
```

---

ivg_sv *Interactive exploration of SVs in a variation graph*

Description

Opens a Shiny app with a dynamic table that contains input SVs. Clicking on a SV (row in the table) generates a simplified representation of the variation graph around this SV. The number of flanking nodes (context) can be increased if necessary, e.g. for large insertions. vg needs to be installed (https://github.com/vgteam/vg).

Usage

```
ivg_sv(svs, xg, ucsc.genome = "hg38")
```

Arguments

svs	either a GRanges with SVs (e.g. from readSVvcf) or the path to a VCF file.
xg	the path to the xg object of the variation graph.
ucsc.genome	the genome version for the UCSC Genome Browser automated link.

Value

Starts a Shiny app in a web browser.

Author(s)

Jean Monlong

---

plot_perregion *Recall, precision, F1 per region*

Description

Recall, precision, F1 per region

Usage

```
plot_perregion(eval, regions.gr, min.region.ol = 0.5, plot = TRUE)
```

Arguments

eval	the output of svevalOl.
regions.gr	GRanges object with regions of interest
min.region.ol	minimum proportion of variant that must overlap regions.gr. Default is 0.5
plot	should the function return the plot list. Default is TRUE. If FALSE, returns a data.frame.

Value

a list of ggplot objects if plot=TRUE (default); a data.frame otherwise.

Author(s)

Jean Monlong

plot_persize *Recall, precision, F1 per SV size*

Description

Recall, precision, F1 per SV size

Usage

```
plot_persize(
  eval,
  size.breaks = c(50, 100, 500, 1000, 10000, Inf),
  plot = TRUE
)
```

Arguments

eval	the output of sveval0l.
size.breaks	a vector for breaking the sizes into classes.
plot	should the function return the plot list. Default is TRUE. If FALSE, returns a data.frame.

Value

a list of ggplot objects if plot=TRUE (default); a data.frame otherwise.

Author(s)

Jean Monlong

plot_prcurve *Create precision-recall graphs*

Description

Create a precision/recall curve using metrics computed by the svevalOl function. The svevalOl function returns a list containing a "curve" data.frame with the evaluation metrics for different quality thresholds.

Usage

```
plot_prcurve(eval, labels = NULL)
```

Arguments

eval	a data.frame, a list of data.frames, or a vector with one or several paths to files with "curve" information.
labels	the labels to use for each input (when multiple inputs are used). Ignored is NULL (default).

Details

If the input is a data.frame (or list of data.frames) it should be the "curve" element of the list returned by the `svevalOl` function. If the input is a character (or a vector of characters), they are considered to be file names and the data will be read from these files.

If multiple inputs are given, either using a list of data.frames or a vectors with several filenames, one curve per input will be created. This is to be used to quickly compare several methods. The "labels" parameters can be used to specify a label for each input to use for the graphs.

Value

list of ggplot graph objects

Author(s)

Jean Monlong

Examples

```
## Not run:
eval = svevalOl('calls.vcf', 'truth.vcf')
plot_prcurve(eval$curve)

# Comparing multiple methods
eval.1 = svevalOl('calls1.vcf', 'truth.vcf')
eval.2 = svevalOl('calls2.vcf', 'truth.vcf')
plot_prcurve(list(eval.1$curve, eval.2$curve), labels=c('method1', 'method2'))

# Or if the results were previously written in files
plot_prcurve(c('methods1-prcurve.tsv', 'methods2-prcurve.tsv'), labels=c('method1', 'method2'))

## End(Not run)
```

`plot_ranges` *Plot variants in a region*

Description

A simple ggplot2 representation of variants in a region. The beginning of the variant is represented as a point (shape=SV type). The point is annotated with the variant size. A line outlines the range (e.g. for deletions or inversions).

Usage

```
plot_ranges(
  gr.l,
  region.gr = NULL,
  pt.size = 2,
  lab.size = 4,
  maxgap = 20,
  scale.legend = "auto",
  show.svids = TRUE,
  gr.l.2 = NULL,
  run.names = c("run1", "run2")
)
```

Arguments

gr.l	a list of GRanges. If named, the names are used as colors in the graph.
region.gr	the region of interest. If NULL (default), all variants are displayed.
pt.size	the point (and line) sizes. Default is 2.
lab.size	the label size. Default is 4
maxgap	the maximum gap allowed when filtering variants in regions. Default is 20.
scale.legend	the size of the scale legend at the bottom. 0 to switch off. Default is 'auto'.
show.svids	should the SV ids be shown on the y-axis. Default is TRUE
gr.l.2	(Optional) a second list of GRanges, like gr.l but from another run for example. If provided the graph will show a second panel for this run (change names with run.names=.
run.names	names to use for the panels when a second input, gr.l.2, is provided. Vector of two characters.

Value

a ggplot2 object

Author(s)

Jean Monlong

---

prf *Compute precision, recall and F1 score*

Description

Compute the precision, recall and F1 score using the TP, TP.baseline, FP and FN columns.

Usage

```
prf(eval.df, use.calls.tp.everywhere = TRUE)
```

Arguments

eval.df	a data.frame with columns TP, TP.baseline, FP, and FN.
use.calls.tp.everywhere	use the TP count as number of calls matched even for the recall computation

Value

the input data.frame with 3 new columns precision, recall and F1.

Author(s)

Jean Monlong

---

readSVvcf *Read SVs from a VCF file*

Description

Read a VCF file that contains SVs and create a GRanges with relevant information, e.g. SV size or genotype quality.

Usage

```
readSVvcf(
  vcf.file,
  keep.ins.seq = FALSE,
  keep.ref.seq = FALSE,
  sample.name = "",
  qual.field = c("GQ", "QUAL"),
  other.field = NULL,
  check.inv = FALSE,
  keep.ids = FALSE,
  nocalls = FALSE,
  out.fmt = c("gr", "df", "vcf"),
  min.sv.size = 10
)
```

Arguments

vcf.file	the path to the VCF file
keep.ins.seq	should it keep the inserted sequence? Default is FALSE.
keep.ref.seq	should it keep the reference allele sequence? Default is FALSE.

sample.name	the name of the sample to use. If "" (default) or sample names not in the VCF, select the first sample. If NULL, don't select particular sample.
qual.field	field to use as quality. Can be in INFO (e.g. default GQ) or FORMAT (e.g. DP). If not found in INFO/FORMAT, QUAL field is used.
other.field	name of other fields to extract from the INFO (e.g. AF). Default is NULL
check.inv	should the sequence of MNV be compared to identify inversions.
keep.ids	keep variant ids? Default is FALSE.
nocalls	if TRUE returns no-calls only (genotype ./.). Default FALSE.
out.fmt	output format. Default is 'gr' for GRanges. Other options: 'df' for data.frame and 'vcf' for the VCF object from the VariantAnnotation package.
min.sv.size	the minimum size of the variant to extract from the VCF. Default is 10

Details

By default, the quality information is taken from the GQ field. If GQ (or the desired field) is missing from both FORMAT or INFO, QUAL will be used.

The 'sample.name' argument can be used to select genotypes for specific sample from the VCF. In addition, variants that are homozygous reference in this sample will be filtered. If 'sample.name' is not in the VCF, the first sample will be selected (default). To force the entire VCF to be read no matter the genotypes of samples, use 'sample.name=NULL'.

Alleles are split and, for each, column 'ac' reports the allele count. Notable cases incude 'ac=-1' for no/missing calls (e.g. './.'), and 'ac=0' on the first allele to report hom ref, variants. These cases are often filtered later with 'ac>0' to keep only non-ref calls. If the VCF contains no samples or if no sample selection if forced (sample.name=NULL), 'ac' will contain '-1' for all variants in the VCF.

Value

depending on 'out.fmt', a GRanges, data.frame, or VCF object with relevant information.

Author(s)

Jean Monlong

Examples

```
## Not run:
calls.gr = readSVvcf('calls.vcf')

## End(Not run)
```

`readSVvcf.multisamps` *Read SVs from a VCF file*

Description

Read a VCF file that contains SVs for multiple samples and create a GRanges with population estimates.

Usage

```
readSVvcf.multisamps(
  vcf.file,
  keep.ins.seq = FALSE,
  keep.ref.seq = FALSE,
  check.inv = FALSE,
  keep.ids = FALSE,
  out.fmt = c("gr", "df"),
  min.sv.size = 10
)
```

Arguments

`vcf.file`	the path to the VCF file
`keep.ins.seq`	should it keep the inserted sequence? Default is FALSE.
`keep.ref.seq`	should it keep the reference allele sequence? Default is FALSE.
`check.inv`	should the sequence of MNV be compared to identify inversions.
`keep.ids`	keep variant ids? Default is FALSE.
`out.fmt`	output format. Default is 'gr' for GRanges. Other options: 'df' for data.frame.
`min.sv.size`	the minimum size of the variant to extract from the VCF. Default is 10

Details

Alleles are split and, for each, column 'ac' reports the total allele count. The number of "ref" genotypes, i.e. homozygous refs e.g. '0/0', and the number of called samples (not missing './.') are also counted. The 'af' column contains an estimate of the allele frequency.

Value

depending on 'out.fmt', a GRanges or a data.frame with relevant information.

Author(s)

Jean Monlong

Examples

```
## Not run:
svs.gr = readSVvcf.multisamps('svs.vcf')

## End(Not run)
```

`rmskAnnotate` *Annotate REF/ALT sequence with RepeatMasker*

Description

Extracts REF/ALT sequence from a VCF, runs RepeatMasker to annotate transposable elements and simple repeats, and annotates the original variants.

Usage

```
rmskAnnotate(svs.gr, nb.cores = 1, species = "human", docker.image = NULL)
```

Arguments

`svs.gr`	SVs. A GRanges or the path to a VCF file.
`nb.cores`	the number of cores that RepeatMasker should use. Default:1.
`species`	the species to use in RepeatMasker. Default: human.
`docker.image`	docker image with RepeatMasker. Default is NULL.

Details

This is a simple annotation where only the main repeat class is retrieved for each variant (covering most of the sequence).

RepeatMasker must be installed.

Value

an updated GRanges with new columns 'rmsk.name', 'rmsk.classfam' and 'rmsk.cov'.

Author(s)

Jean Monlong

Examples

```
## Not run:
svs.gr = readSVvcf('calls.vcf', keep.ins.seq=TRUE, keep.ref.seq=TRUE)
svs.gr = rmskAnnotate(svs.gr)

## End(Not run)
```

`stitchMergeHets` *Stitch and merge heterozygous SVs*

Description

Fragmented calls often can't be matched as is to another call set. Calls that are fragmented in multiple pieces could be stitched together if they are close by and with the same genotype and SV type. An homozygous call can also be fragmented in two heterozygous calls. To counter this, we can merge pairs of heterozygous SVs (from the same type) that overlap substantially.

Usage

```
stitchMergeHets(
  svs.gr,
  do.stitch = TRUE,
  do.merge = TRUE,
  stitch.dist = 20,
  min.rol = 0.8,
  max.ins.dist = 20,
  range.seq.comp = FALSE,
  ins.seq.comp = FALSE
)
```

Arguments

`svs.gr`	input GRanges with SVs, e.g. read from `readSVvcf`.
`do.stitch`	should nearby het SVs of the same type be stitched? Default is TRUE.
`do.merge`	should similar het SVs of the same type "merged" into one homozygous variant? Default is TRUE.
`stitch.dist`	the maximum distance allowed for two SVs to be stitched.
`min.rol`	minimum reciprocal overlap to merge two hets into one hom
`max.ins.dist`	maximum distance for insertions to be clustered when merging hets.
`range.seq.comp`	compare sequence instead of overlapping deletions/inversion/etc. Default is FALSE.
`ins.seq.comp`	compare sequence instead of insertion sizes. Default is FALSE.

Value

a GRanges with het SVs from input svs.gr stitched and/or merged.

Author(s)

Jean Monlong

`subset_eval` *Compute evaluation results on a subset of the calls*

Description

Compute evaluation results on a subset of the calls

Usage

```
subset_eval(
  eval,
  regions.gr = NULL,
  accepted.filters = NULL,
  min.region.ol = 0.5,
  nb.cores = 1
)
```

Arguments

`eval`	the output of `svevalOl`.
`regions.gr`	GRanges object with regions of interest. NULL (default) means all SVs.
`accepted.filters`	vector of the values of the FILTER field to keep. If NULL (default), all values are accepted which means all values are kept
`min.region.ol`	minimum proportion of variant that must overlap regions.gr. Default is 0.5
`nb.cores`	number of processors to use. Default is 1.

Value

a list like for `svevalOl`

Author(s)

Jean Monlong

`svevalOl` *SV evaluation based on overlap and variant size*

Description

Compares SVs from a call-set to SVs from a truth-set.

Usage

```
svevalOl(
  calls.gr,
  truth.gr,
  max.ins.dist = 20,
  min.ol = 0.5,
  min.del.rol = 0.1,
  range.seq.comp = FALSE,
  ins.seq.comp = FALSE,
  nb.cores = 1,
  min.size = 50,
  max.size = Inf,
  bed.regions = NULL,
  bed.regions.ol = 0.5,
  qual.field = c("GQ", "QUAL"),
  sample.name = NULL,
  outfile = NULL,
  out.bed.prefix = NULL,
 qual.ths = c(0, 2, 3, 4, 5, 7, 10, 12, 14, 21, 27, 35, 45, 50, 60, 75, 90, 99, 110,
    133, 167, 180, 250, 350, 450, 550, 650),
  qual.quantiles = seq(0, 1, 0.1),
  check.inv = FALSE,
  geno.eval = FALSE,
  stitch.hets = FALSE,
  stitch.dist = 20,
  merge.hets = FALSE,
  merge.rol = 0.8,
  method = c("coverage", "bipartite"),
  simprep = NULL,
  log.level = c("CRITICAL", "WARNING", "INFO")
)
```

Arguments

calls.gr	call set. A GRanges or the path to a VCF file.
truth.gr	truth set. A GRanges or the path to a VCF file.
max.ins.dist	maximum distance for insertions to be clustered. Default is 20.
min.ol	the minimum overlap/coverage to be considered a match. Default is 0.5
min.del.rol	minimum reciprocal overlap for deletions. Default is 0.1
range.seq.comp	compare sequence instead of only overlapping deletions/inversion/etc. Default is FALSE.
ins.seq.comp	compare sequence instead of insertion sizes. Default is FALSE.
nb.cores	number of processors to use. Default is 1.
min.size	the minimum SV size to be considered. Default 50.
max.size	the maximum SV size to be considered. Default is Inf.

bed.regions	If non-NULL, a GRanges object or path to a BED file (no headers) with regions of interest.
bed.regions.ol	minimum proportion of sv.gr that must overlap regions.gr. Default is 0.5
qual.field	fields to use as quality.
sample.name	the name of the sample to use if VCF files given as input. If NULL (default), use first sample.
outfile	the TSV file to output the results. If NULL (default), returns a data.frame.
out.bed.prefix	prefix for the output BED files. If NULL (default), no BED output.
qual.ths	the quality thresholds for the PR curve. If NULL, will use quantiles. see the qual.quantiles parameter below.
qual.quantiles	the quality quantiles for the PR curve, if qual.ths is NULL. Default is (0, .1, ..., .9, 1).
check.inv	should the sequence of MNV be compared to identify inversions.
geno.eval	should het/hom be evaluated separately (genotype evaluation). Default FALSE.
stitch.hets	should clustered hets be stitched together before genotype evatuation. Default is FALSE.
stitch.dist	the maximum distance to stitch hets during genotype evaluation.
merge.hets	should similar hets be merged into homs before genotype evaluation. Default is FALSE.
merge.rol	the minimum reciprocal overlap to merge hets before genotype evaluation.
method	the method to annotate the overlap. Either 'coverage' (default) for the cumulative coverage (e.g. to deal with fragmented calls); or 'bipartite' for a 1-to-1 matching of variants in the calls and truth sets.
simprep	optional simple repeat annotation. Default is NULL. If non-NULL, GRanges to be used to extend variants when overlapping/clustering
log.level	the level of information in the log. Default is "CRITICAL" (basically no log).

Details

Different overlapping approaches are available. See svOverlap for more details. We recommend using the default method='coverage' when evaluating the calls (absence/presence of SVs) and method='bipartite' when evaluating genotypes. The latter will match a variant in the call-set with at most one variant in the truth-set which will penalyze. one-to-many configurations as is usually preferred when comparing exact genotypes. To further switch on the 'genotype evaluation' mode, use geno.eval=TRUE. It can also help to stitch fragmented calls and merge heterozygotes into homozygotes using merge.hets=TRUE and stitch.hets=TRUE (see example below).

The evaluation will be performed for different quality thresholds on the call-set in order to make a precision-recall curve. If the VCF are to be read it will look for the field specified in qual.field (GQ by default, and QUAL if not found). If you don't want the PR curve, the evaluation can be sped up by running with for example qual.ths=0.

Equivalent SVs are sometimes recorded as quite different variants because placed at different locations of a short tandem repeat. For example, imagine a large 100 bp tandem repeat in the reference genome. An expansion of 50 bp might be represented as a 50 bp insertion at the beginning of the

repeat in the callset but at the end of the repeat in the truth set. Because they are distant by 100 bp they might not match. Instead of increasing the distance threshold too much, passing an annotation of known simple repeats in the `simprep=` parameter provides a more flexible way of matching variants by first extending them with nearby simple repeats. In this example, because we know of this tandem repeat, both insertions will be extended to span the full annotated reference repeat, hence ensuring that they are matched and compared (e.g. by reciprocal size or sequence alignment distance) short tandem repeat.

Value

a list with

`eval`	a data.frame with TP, FP and FN for each SV type when including all variants
`curve`	a data.frame with TP, FP and FN for each SV type when using different quality thesholds
`svs`	a list of GRanges object with FP, TP and FN for each SV type (using quality threshold with best F1).
`mqual.bestf1`	the quality threshold that produces best F1 (and corresponding to 'svs' GRanges).

Author(s)

Jean Monlong

Examples

```
## Not run:
## From VCF files
eval = svevalOl('calls.vcf', 'truth.vcf')

## From GRanges
calls.gr = readSVvcf('calls.vcf')
truth.gr = readSVvcf('truth.vcf')
eval = svevalOl(calls.gr, truth.gr)

## Genotype evaluation
eval = svevalOl(calls.gr, truth.gr, geno.eval=TRUE, merge.hets=TRUE,
                stitch.hets=TRUE, method='bipartite')

## End(Not run)
```

`svOverlap` *Overlap SVs*

Description

Overlap SVs by SV type using one of the overlap approaches (see below). Variants covering a genomic range (e.g. deletions, duplications, inversions) are overlapped while insertions are clustered (using `max.ins.dist`) and their size or sequence (if `ins.seq.comp`) are compared.

Usage

```
svOverlap(
  query,
  subject,
  min.ol = 0.5,
  method = c("reciprocal", "coverage", "bipartite"),
  max.ins.dist = 20,
  min.del.rol = 0.1,
  range.seq.comp = FALSE,
  ins.seq.comp = FALSE,
  simprep = NULL,
  nb.cores = 1,
  log.level = c("CRITICAL", "WARNING", "INFO")
)
```

Arguments

query	a GRanges object with SVs
subject	another GRanges object with SVs
min.ol	the minimum overlap/coverage to be considered a match. Default is 0.5
method	the method to annotate the overlap. Either 'coverage' (default) for the cumulative coverage (e.g. to deal with fragmented calls); or 'bipartite' for a 1-to-1 matching of variants in the calls and truth sets.
max.ins.dist	maximum distance for insertions to be clustered. Default is 20.
min.del.rol	minimum reciprocal overlap for deletions. Default is 0.1
range.seq.comp	compare sequence instead of overlapping deletions/inversions/etc. Default is FALSE.
ins.seq.comp	compare sequence instead of insertion sizes. Default is FALSE.
simprep	optional simple repeat annotation. Default is NULL. If non-NULL, GRanges to be used to extend variants when overlapping/clustering
nb.cores	number of processors to use. Default is 1.
log.level	the level of information in the log. Default is "CRITICAL" (basically no log).

Details

Available overlap approaches, passed with `method=`, include: reciprocal, coverage, bipartite. If you are using this function directly, you might be interested in the 'reciprocal' method (default). When evaluating SVs versus a truthset, `svevalOl` uses the 'coverage' method to compare calls (absence/presence) and 'bipartite' to compare genotypes (when run with the recommended settings).

The "reciprocal" method corresponds to the simple reciprocal overlap for the variants covering a genomic range (e.g. deletions, duplications, inversions), or the reciprocal size/sequence similarity for insertions.

With the "coverage" approach, a variant needs to be covered enough by variants from the other set to be counted "matched" or "overlapped". Here again, the ranges are overlapped for SV spanning a genomic region while for insertions, the size or aligned sequences are summed.

With the "bipartite" approach, the variants are first matched using the reciprocal overlap method (see "reciprocal"), and then matched one-to-one using bipartite clustering. This ensures that a variant in one set is only matched to one variant in the other set. Useful when comparing genotypes for example when redundancy should be penalized.

Equivalent SVs are sometimes recorded as quite different variants because placed at different locations of a short tandem repeat. For example, imagine a large 100 bp tandem repeat in the reference genome. An expansion of 50 bp might be represented as a 50 bp insertion at the beginning of the repeat in the callset but at the end of the repeat in the truth set. Because they are distant by 100 bp they might not match. Instead of increasing the distance threshold too much, passing an annotation of known simple repeats in the `simprep=` parameter provides a more flexible way of matching variants by first extending them with nearby simple repeats. In this example, because we know of this tandem repeat, both insertions will be extended to span the full annotated reference repeat, hence ensuring that they are matched and compared (e.g. by reciprocal size or sequence alignment distance) short tandem repeat.

Value

a GRanges with information about pairs of SVs in query and subject that overlap

`GRange`	intersected ranges (informative for "ranges" SVs)
`queryHits`	the id of the input query
`subjectHits`	the id of the input subject
`querSize`	the size of the input query
`subjectSize`	the size of the input subject
`interSize`	the size of the intersection (e.g. range, ins size, ins seq alignment)
`olScore`	the overlap score (usually the value of the reciprocal overlap)
`type`	the SV type of the pair

Author(s)

Jean Monlong

`wrapper` *Wrapper function to use sveval as a command line tool*

Description

Wrapper functions that reads arguments and runs sveval.

Usage

```
wrapper(subcmd = c("", "sveval", "mergetsvs"), args = commandArgs(TRUE))
```

Arguments

subcmd	which subcommand to use. Default is '' which prints the list of available subcommands
args	vector of arguments. By default, reads them from the command line.

Value

return code (0: success, 1: error)

Author(s)

Jean Monlong

Index