Package 'appreci8R'

April 15, 2019

Type Package

Title appreci8R: an R/Bioconductor package for filtering SNVs and short indels with high sensitivity and high PPV

Version 1.0.0

Author Sarah Sandmann

Maintainer Sarah Sandmann <sarah.sandmann@uni-muenster.de>

Description The appreci8R is an R version of our appreci8-algorithm - A Pipeline for PREcise variant Calling Integrating 8 tools. Variant calling results of our standard appreci8-tools (GATK, Platypus, VarScan, FreeBayes, LoFreq, SNVer, samtools and VarDict), as well as up to 5 additional tools is combined, evaluated and filtered.

License LGPL-3

Encoding UTF-8

LazyData true

Imports shiny, shinyjs, DT, VariantAnnotation, BSgenome, BSgenome.Hsapiens.UCSC.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene, Homo.sapiens, SNPlocs.Hsapiens.dbSNP144.GRCh37, XtraSNPlocs.Hsapiens.dbSNP144.GRCh37, rsnps, Biostrings, MafDb.1Kgenomes.phase3.hs37d5, MafDb.ExAC.r1.0.hs37d5, MafDb.ESP6500SI.V2.SSA137.hs37d5, MafDb.gnomADex.r2.0.1.hs37d5, COSMIC.67, rentrez, PolyPhen.Hsapiens.dbSNP131, SIFT.Hsapiens.dbSNP137, seqinr, openxlsx, Rsamtools, stringr, utils, stats, GenomicRanges, S4Vectors, GenomicFeatures, IRanges, GenomicScores, SummarizedExperiment

Suggests GO.db, org.Hs.eg.db

biocViews VariantDetection, GeneticVariability, SNP, VariantAnnotation, Sequencing,

RoxygenNote 6.0.1

git_url https://git.bioconductor.org/packages/appreci8R

git_branch RELEASE_3_8

git_last_commit addad71

git_last_commit_date 2018-10-30

Date/Publication 2019-04-15

R topics documented:

`appreci8R-package` *appreci8R: an R/Bioconductor package for filtering SNVs and short indels with high sensitivity and high PPV*

Description

The appreci8R is an R version of our appreci8-algorithm - A Pipeline for PREcise variant Calling Integrating 8 tools. Variant calling results of our standard appreci8-tools (GATK, Platypus, VarScan, FreeBayes, LoFreq, SNVer, samtools and VarDict), as well as up to 5 additional tools is combined, evaluated and filtered.

Details

This package was not yet installed at build time.
For the use of next-generation sequencing in clinical routine valid variant calling results are crucial. However, numerous variant calling tools are available. These tools usually differ in the variant calling algorithsms, the characteristics reported along with the varaint calls, the recommended filtration strategies for the raw calls and thus, also in the output. Especially when calling variants with a low variant allele frequency (VAF), perfect results are hard to obtain. High sensitivity is usually accompanied by low positive predictive value (PPV).

appreci8R is a package for combining and filterating the output of differen variant calling tools according to the 'appreci8'-algorithm. vcf as well as txt files containing variant calls can be evaluated. The number of variant calling tools to consider is unlimited (for the user interface version it is limited to 13). The final output contains a list of variant calls, classified as "probably true", "polymophism" or "artifact".

Important note: Currently, only hg19 is supported.

Index: This package was not yet installed at build time.
The package contains a function performing the whole analysis using a shiny user interface - `appreci8Rshiny`.

Additionally, seven individual functions for performing the seven analysis steps are available:

1) `filterTarget`: Exclude all off-target calls from further analysis.

2) `normalize`: Normalize calls with respect to reporting indels, MNVs, reporting of several alternate alleles and reporting of complex indels.

3) `annotate`: Annotate calls (using VariantAnnotation), and filter the output according to the locations and consequences of interest.

4) `combineOutput`: Combine output of the different variant calling tools.

5) `evaluateCovAndBQ`: Evaluate coverage and base quality (using Rsamtools), and filter calls with insufficient coverage and/or base quality.

6) `determineCharacteristics`: Determine characteristics of the calls, including database check-ups and impact prediction on protein level.

7) `finalFiltration`: Perform final filtration according to the appreci8-algorithm.

Author(s)

Sarah Sandmann

Maintainer: Sarah Sandmann <sarah.sandmann@uni-muenster.de>

References

More information on appreci8 can be found in our Bioinformatics paper: appreci8: A Pipeline for Precise Variant Calling Integrating 8 Tools https://doi.org/10.1093/bioinformatics/bty518.

More information on the performance of eight commonly used variant calling tools can be found in our Scientific Reports paper: Evaluating Variant Calling Tools for Non-Matched Next-Generation Sequencing Data https://www.nature.com/articles/srep43169

See Also

appreci8Rshiny, filterTarget, normalize, annotate, combineOutput, evaluateCovAndBQ, determineCharacteristics, finalFiltration

Examples

```
output_folder<-""

target<-bedFileWithTargetRegions
targetFiltered<-list()

caller_folder<-"/test/gatk/"
targetFiltered[[1]]<-filterTarget(output_folder, "GATK", caller_folder,
                                  ".rawMutations", ".vcf", TRUE, "", "")

caller_folder<-"/test/varscan/"
targetFiltered[[2]]<-filterTarget(output_folder, "VarScan", caller_folder,
                                  "", ".txt", FALSE, "_snvs", "_indels", 1 ,
                                  2 , 3, 4)

normalized<-list()
normalized[[1]]<-normalize(output_folder, "GATK", targetFiltered[[1]], FALSE,
                           FALSE)
normalized[[2]]<-normalize(output_folder, "VarScan", targetFiltered[[2]], TRUE,
                           FALSE)

annotated<-list()
annotated[[1]]<-annotate(output_folder, "GATK", normalized[[1]],
                         locations = c("coding","spliceSite"),
                         consequences = c("nonsynonymous","frameshift","nonsense"))
annotated[[2]]<-annotate(output_folder, "VarScan", normalized[[2]],
                         locations = c("coding","spliceSite"),
```

```
                        consequences = c("nonsynonymous","frameshift","nonsense"))

combined<-combineOutput(output_folder, c("GATK","VarScan"), annotated)

bam_folder<-"/test/alignment/"
filtered<-evaluateCovAndBQ(output_folder, combined, bam_folder)

databases<-determineCharacteristics(output_folder, filtered,
                                    predict = "Provean")

final<-finalFiltration(output_folder, frequency_calls = filtered,
                       database_calls = databases, combined_calls = combined,
                       damaging_safe = -3, tolerated_safe = -1.5, primer = NA,
                       hotspots = NA, overlapTools = c("VarScan"))
```

`annotate` *Annotate and filter calls*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 3rd analysis step, all calls are annotated (using VariantAnnotation) and filtered according to the selected locations and consequences of interest. A GRanges object with all annotated calls is returned.

Usage

```
annotate(output_folder, caller_name, normalized_calls_g, locations,
                      consequences)
```

Arguments

`output_folder`	The folder to write the output files into. If an empty string is provided, no files are written out.
`caller_name`	Name of the variant calling tool (only necessary if an output folder is provided).
`normalized_calls_g`	A GRanges object. One normalized variant call per line. Necessary metadata columns are: SampleID, Ref, Alt. Normalize()-output can directly be taken as input.
`locations`	A vector containing the locations of interest. Accepted values are: coding, intron, threeUTR, fiveUTR, intergenic, spliceSite, promoter.
`consequences`	A vector containing the consequences of interest. Accepted values are: synonymous, nonsynonymous, frameshift, nonsense, not translated.

Details

The function `annotate` performs annotations of all variants in the input GRanges object (according to VariantAnnotation) and filteres the annotated calls with respect to the selected locations and consequences of interest (based on the functions `locateVariants` and `predictCoding` of the package VariantAnnotation). At least one location of interest has to be chosen. If - among others - "coding" is selected, at least one consequence of interest has to be chosen as well.

All possible transcripts are investigated. If "coding" is chosen and a variant is annotated as "coding" in only 1 out of many transcripts, the matching transcript is, together with the annotation information, reported. All the other non-matching transcripts are not reported.

Value

A GRanges object is returned containing only the calls at the selected locations of interest and with the selected consequences of interest. Reported metadata columns are: SampleID, Ref, Alt, Location, c. (position of variant on cDNA level), p. (position of variant on protein level), AA_ref, AA_alt, Codon_ref, Codon_alt, Consequence, Gene, GeneID, TranscriptID. TxDb.Hsapiens.UCSC.hg19.knownGene is used for annotation. Whenever there is more than one transcriptID, all of them are reported, separated by commas. The first transcriptID matches the first entry under Location, c., p. etc.

If an output folder is provided, the output is saved as <caller_name>.annotated.txt.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

References

VariantAnnotation: https://www.bioconductor.org/packages/release/bioc/html/VariantAnnotation.html

See Also

appreci8R, appreci8Rshiny, filterTarget, normalize, combineOutput, evaluateCovAndBQ, determineCharacteristics, finalFiltration

Examples

```
library("GenomicRanges")
input <- GRanges(seqnames = c("2","X","4","21"),
                 ranges = IRanges(start = c (25469504,15838366,106196951,36164405),
                                  end = c (25469504,15838366,106196951,36164405)),
                 SampleID = c("Sample1","Sample1","Sample2","Sample2"),
                 Ref = c("G","C","A","G"),
                 Alt = c("T","A","G","T"))

annotated<-annotate("", "", input,
                    locations = c("coding","spliceSite"),
                    consequences = c("nonsynonymous","frameshift","nonsense"))
```

---

appreci8Rshiny *A user interface to perform the whole appreci8-analysis*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. The whole analysis-pipeline consists of seven steps. Individual functions for executing these steps are available. However, an additional user-interface is available that enables easy execution of the analysis-pipeline, restarting parts of the pipeline from checkpoints, changing parameters, exporting and importing the configuration and evaluation of the results. The user-interface is opend by executing the function `appreci8Rshiny`.

Usage

```
appreci8Rshiny()
```

Arguments

No arguments have to be provided.

Details

The user-interface is structured by tabs: **Analysis**, **Overview results** and **Detailed results**.

Analysis: The tab is essential for performing the whole analysis. It is further separated into 10 additional tabs:

* `Important notes`: General information on prerequisites for performing the analysis and current limitations (only hg19).

* `0. Preparations`: Defining the general output folder and the varaint caller input. GATK, Platypus, VarScan, FreeBayes, LoFreq, SNVer, SamTools and VarDict are available as these make up the classical appreci8-pipeline. The tools can be `added` or `removed`. For every tool the following parameters have to be defined: the folder containing the variant calling results, the output file type (vcf or txt; as vcf is a standardized format, the columns containing chr, pos, ref and alt are assumed to be fixed; when selecting txt, the columns containing chr, pos, ref and alt have to be defined), a possible suffix for naming the output, if MNVs are reported (AG>CT instead of A>C and G>T), if SNVs and indels are reported in one file (if not, what is the suffix for the SNV files and what is the suffix for the indel files?), and how are indels reported. In addition to the eight tools, up to 5 additional tools can be added (select the number and click `Go`). In addition to the previous parameters, a name can be defined for each additionally added tool.

* `1. Target filtration`: Upload your target region (bed file; chromosome definition without 'chr').

* `2. Normalization`

* `3. Annotation`: Annotates the variants and filters them according to the selected locations (coding, intron, threeUTR, fiveUTR, intergenic, spliceSite, promoter) and consequences (synonymous, nonsynonymous, frameshift, nonsense, not translated) of interest.

* `4. Combine output`

* `5. Evaluate Coverage and BQ`: Evaluate coverage and base quality by analyzing the bam files. Calls with insufficient coverage and or base quality are filtered. The following parameters have to be defined: the folder containing the bam- and bai files, minimum coverage, minimum number of reads carrying the alternate allele, minimum VAF, minimum base quality and the maximum difference in base quality between reference and alternative.

* `6. Extended Set of Characteristics`: Select the databases you would like to query (dbSNP, 1000Genomes, ExAC, ESP6500, Genome Aggregation Database, COSMIC, ClinVar) and the source you would like to use for impact prediction (SIFT, Provean, PolyPhen2).

* `7. Final Filtration`: Adjust parameters and define input for the final filtration. A bed file containing the primer positions can be uploaded (optional). A txt file containing expected/hotspot mutations (defined via gene, mutation on AA level, and an optional minimum VAF) can be uploaded (optional). Stricter thresholds for coverage and base quality can be defined (see parameters for `5. Evaluate Coverage and BQ`). The number of samples making a call striking can be defined. Dependent on the previously selected source for impact prediction (SIFT, Provean, PolyPhen2), a threshold can be defined to identify reliable damaging predictions and reliable tolerated prediction. Furthermore, the default scoring for the artifact- and the polymorphism score can be changed (just recommended for experts).

* Action: The complete analysis can be executed (Start complete analysis). The availability of checkpoints can be checked (Check for possible checkpoints). If checkpoints are available, one can be selected and the analysis can be started from the selected checkpoint. The defined configuration can be exported (Export current configuration) as well as imported (Import configuration). Press Done if you are done with your analysis.

Overview results: The tab provides an overview of the results. It is further separated into 5 additional tabs:

* Log: The log of the analysis is displayed here. In case of errors, e.g. input not being available, these messages are also reported in the log.

* Raw Calls: The raw number of calls per sample and per caller are reported.

* Calls On Target: The number of calls on target per sample and per caller are reported.

* Annotaed Calls: The number of annotated (and filtered according to annotation) calls per sample and per caller are reported.

* Filtered Calls: The number of coverage and base quality filtered calls per sample is reported.

Detailed results: The tab provides detailed information on the results. It is further separated into 3 additional tabs:

* Mutations: Variant calls classified as likely true mutations.

* Polymorphisms: Variant calls classified as likely polymorphisms.

* Artifacts: Variant calls classified as likely artifacts.

Value

No value is returned.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

appreci8R, filterTarget, normalize, annotate, combineOutput, evaluateCovAndBQ, determineCharacteristics
finalFiltration

Examples

```
appreci8Rshiny()
```

---

combineOutput *Combine output of different variant calling tools*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 4th analysis step, all normalized and annotated calls of the different variant calling tools are combined. A GRangesObject with all combined calls is returned.

Usage

```
combineOutput(output_folder, caller_names, annotated_calls_g)
```

Arguments

output_folder	The folder to write the output files into. If an empty string is provided, no files are written out.
caller_names	Vector containing the name of the variant calling tools.
annotated_calls_g	A GRangesList object. Every list element contains the variant calls of one variant calling tool. Annotate()-output can directly be taken as input. Providing a list with only one list element, i.e. evaluating only one variant calling tool, is possible, but not assumed to be useful. The point of the appreci8-algorithm lies in the combined evaluation and filtration of the output of several variant calling tools.

Details

The function `combineOutput` performs a combination of the normalized and annotated variant calls from different variant calling tools. The results are sorted according to Chr, Pos, Ref, Alt and SampleID. If two callers - caller1 and caller2 - report the same variant for the same sample, it is only reported once in the output file with a "1" in the column "caller1" and another "1" in the column "caller2".

Value

A GRanges object is returned containing all combined calls. Reported metadata columns are: SampleID, Ref, Alt, Location, c. (position of variant on cDNA level), p. (position of variant on protein level), AA_ref, AA_alt, Codon_ref, Codon_alt, Consequence, Gene, GeneID, TranscriptID. In addition, one column for every variant calling tool is reported, containing a "1" for every call detected by that tool and "NA" for every call not detected by that tool.

If an output folder is provided, the output is saved as Results_Raw.txt.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

appreci8R, appreci8Rshiny, filterTarget, normalize, annotate, evaluateCovAndBQ, determineCharacteristic
finalFiltration

Examples

```
library("GenomicRanges")
gatk<-GRanges(seqnames = c("4","X"),
              ranges = IRanges(start = c(106196951,15838366),
                               end = c(106196951,15838366)),
              SampleID = c("Sample2","Sample1"),
              Ref = c("A","C"),
              Alt = c("G","A"),
              Location = c("coding,coding","coding"),
              c. = c("5284,5347","864"),
```

```
                p. = c("1762,1783","288"),
                AA_ref = c("I,I","N"),
                AA_alt = c("V,V","K"),
                Codon_ref = c("ATA,ATA","AAC"),
                Codon_alt = c("GTA,GTA","AAA"),
                Consequence = c("nonsynonymous,nonsynonymous","nonsynonymous"),
                Gene = c("TET2,TET2","ZRSR2"),
                GeneID = c("54790,54790","8233"),
                TranscriptID = c("18308,18309","75467"))
varscan<-gatk[2,]
annotated<-GRangesList()
annotated[[1]]<-gatk
annotated[[2]]<-varscan

combined<-combineOutput("", c("GATK","VarScan"), annotated)
```

`determineCharacteristics` *Determine characteristics of the calls*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 6th analysis step, an extended set of characteristics for all calls is determined. This includes database check-ups and impact prediction on protein level. A GRanges object with all calls, their database information and their predicted impact is reported.

Usage

```
determineCharacteristics(output_folder, frequency_calls_g, predict,
                         dbSNP = TRUE, `1kgenomes` = TRUE, exacDB = TRUE,
                         espDB = TRUE, gadDB = TRUE, cosmicDB = TRUE,
                         clinvarDB = TRUE)
```

Arguments

`output_folder`	The folder to write the output files into. If an empty string is provided, no files are written out.
`frequency_calls_g`	A GRanges object. The GRanges object contains one call per line. EvaluateCovAndBQ()-output can directly be taken as input.
`predict`	String defining the prediction tool to use (accepted strings are: SIFT, Provean, PolyPhen2)
`dbSNP`	Query dbSNP (144.GRCh37) for the presence of your variants (`TRUE` or `FALSE`; default `TRUE`).
`1kgenomes`	Query 1000 Genomes (phase3) for the presence of your variants (`TRUE` or `FALSE`; default `TRUE`).
`exacDB`	Query ExAC (r1.0) for the presence of your variants (`TRUE` or `FALSE`; default `TRUE`).
`espDB`	Query ESP6500 (V2) for the presence of your variants (`TRUE` or `FALSE`; default `TRUE`).

gadDB	Query Genome Aggregation Database (r2.0.1) for the presence of your variants (TRUE or FALSE; default TRUE).
cosmicDB	Query COSMIC (67) for the presence of your variants (TRUE or FALSE; default TRUE).
clinvarDB	Query ClinVar (via rentrez) for the presence of your variants (TRUE or FALSE; default TRUE).

Details

The function evaluateCharacteristics determines an extended set of characteristics for the normalized, annotated, coverage and base quality filtered calls. This includes database check-ups and impact prediction on protein level:

First, the database check-up is performed. At a maximum, dbSNP, 1000Genomes, ExAC, ESP6500, Genome Aggregation Database, COSMIC and ClinVar can be checked. If all the variables are set to FALSE, no database is checked and none will be reported. Thus, the check-up can be disabled.

Second, the impact perdiction on protein level is performed. Impact can be predicted using either SIFT, Provean or PolyPhen2. In any case, the prediction is performed on the basis of rs-IDs. For every call, the predicted effect as well as the score is reported.

Value

A GRanges Object is returned containing all calls, information on their presence in the selected databases and information on the predicted effect. Reported metadata columns are (if all databases are selected): SampleID, Ref, Alt, dbSNP (containing the rs-ID), dbSNP_MAF, G1000_AF, ExAC_AF, ESP6500_AF, GAD_AF, CosmicID, Cosmic_Counts (number of Cosmic entries), ClinVar, Prediction (damaging or neutral), Score (on the basis of the selected prediction tool), c. (variant on cDNA level containing position and variant), c.complement (complement of the variant; if c. is c.5284A>G, then c.complement is c.5284T>C), p. (variant on protein level containing position and amino acids).

If an output folder is provided, the output is saved as Results_Databases.txt.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

appreci8R, appreci8Rshiny, filterTarget, normalize, annotate, combineOutput, evaluateCovAndBQ, finalFiltration

Examples

```
library("GenomicRanges")
filtered<-GRanges(seqnames = c("4"),
                  ranges = IRanges(start = c (106196951),
                                   end = c (106196951)), SampleID = c("Sample2"),
                  Ref = c("A"), Alt = c("G"), Location = c("coding,coding"),
                  c. = c("5284,5347"), p. = c("1762,1783"),
                  AA_ref = c("I,I"), AA_alt = c("V,V"),
                  Codon_ref = c("ATA,ATA"), Codon_alt = c("GTA,GTA"),
                  Consequence = c("nonsynonymous,nonsynonymous"),
                  Gene = c("TET2,TET2"), GeneID = c("54790,54790"),
                  TranscriptID = c("18308,18309"), GATK = c(1),
```

```
                    VarScan = c(NA), Nr_Ref = c(1268), Nr_Alt = c(1283),
                    DP = c(2551), VAF = c(0.50294), BQ_REF = c(38.66798),
                    BQ_ALT = c(38.8145), Nr_Ref_fwd = c(428),
                    Nr_Alt_fwd = c(469), DP_fwd = c(897),
                    VAF_fwd = c(0.522854), Nr_Ref_rev = c(840),
                    Nr_Alt_rev = c(814), DP_rev = c(1654),
                    VAF_rev = c(0.4921403))

characteristics<-determineCharacteristics("", filtered, predict = "Provean")
```

`evaluateCovAndBQ` *Evaluate coverage and base quality*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 5th analysis step, all calls are evaluated with respect to coverage and basequality (using Rsamtools). Calls with insufficient coverage and/or basequality are filtered from further analysis. A GRanges object with all calls featuring sufficient coverage and basequality is returned.

Usage

```
evaluateCovAndBQ(output_folder, combined_calls_g, bam_folder,
                 dp = 50, nr_alt = 20, vaf = 0.01, bq = 15, bq_diff = 7)
```

Arguments

`output_folder`	The folder to write the output files into. If an empty string is provided, no files are written out.
`combined_calls_g`	A GRanges object; necessary input object. The GRanges object contains one call per line. CombineOutput()-output can directly be taken as input.
`bam_folder`	The folder containing the bam- and bai-files for every sample. The file names have to match the sample names, i.e. Sample1.bam and Sample1.bai if Sample1.vcf (without any defined suffixes) was analyzed for caller1.
`dp`	Minimum depth that is required for a call to pass the filter (default: 50).
`nr_alt`	Minimum number of reads carrying the alternate allele that is required for a call to pass the filter (default: 20).
`vaf`	Minimum VAF that is required for a call to pass the filter (default: 0.01).
`bq`	Minimum base quality that is required for a call to pass the filter (default: 15; values above 44 not recommended).
`bq_diff`	Maximum difference in basequality between reference and alternative that is allowed for a call to pass the filter (default: 7).

Details

The function `evaluateCovAndBQ` evaluates coverage and base quality of all calls and filteres them according to user-definable thresholds. Only those calls that feature sufficient coverage and base quality are reported. Filtration is performed in a 2-step procedure:

First, coverage is evaluated using Rsamtools `pileup` and a threshold of `min_base_quality=0`. If depth is below the defined threshold `dp` and/or the number of reads carrying the alternate allele is below the defined threshold `nr_alt` and/or VAF is below the defined threshold `VAF`, a call is excluded. However, if coverage is sufficient, base quality - which is more time-consuming - is evaluated.

To evaluate base quality, the threshold for `min_base_quality` is successively increased from 0 to 44 (thus, no values above 44 are recommended for `bq`). The average base quality for all reads carrying the reference and the alternate allele is calculated. For indels, which are always summed up as "+" by Rsamtools, no base quality can be determined. If the average base quality of the alternate allele is lower than `bq`, the call is filtered. If "average base quality reference allele" - "average base quality alternate allele" is higher than `bq_diff`, the call is filtered.

Coverage is reported with respect to the forward- and reverse reads separately. It is not yet evaluated. This is done in the 7th analysis step (`finalFiltration`).

Value

A GRanges object is returned containing all calls with sufficient coverage and base quality. Reported metadata columns are: SampleID, Ref, Alt, Location, c. (position of variant on cDNA level), p. (position of variant on protein level), AA_ref, AA_alt, Codon_ref, Codon_alt, Consequence, Gene, GeneID, TranscriptID, Caller1 to CallerX (dependent on the number of callers that is evaluated), Nr_Ref, Nr_Alt, DP, VAF, BQ_REF, BQ_ALT, Nr_Ref_fwd, Nr_Alt_fwd, DP_fwd, VAF_rev, Nr_Ref_rev, Nr_Alt_rev, DP_rev, VAF_rev.

If an output folder is provided, the output is saved as Results_Frequency.txt.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

References

Rsamtools: http://bioconductor.org/packages/release/bioc/html/Rsamtools.html

See Also

appreci8R, appreci8Rshiny, filterTarget, normalize, annotate, combineOutput, determineCharacteristics, finalFiltration

Examples

```
library("GenomicRanges")
combined<-GRanges(seqnames = c("4","X"),
                  ranges = IRanges(start = c (106196951,15838366),
                                   end = c (106196951,15838366)),
                  SampleID = c("Sample2","Sample1"), Ref = c("A","C"),
                  Alt = c("G","A"), Location = c("coding,coding","coding"),
                  c. = c("5284,5347","864"), p. = c("1762,1783","288"),
                  AA_ref = c("I,I","N"), AA_alt = c("V,V","K"),
                  Codon_ref = c("ATA,ATA","AAC"),
                  Codon_alt = c("GTA,GTA","AAA"),
```

```
                    Consequence = c("nonsynonymous,nonsynonymous","nonsynonymous"),
                    Gene = c("TET2,TET2","ZRSR2"),
                    GeneID = c("54790,54790","8233"),
                    TranscriptID = c("18308,18309","75467"),
                    GATK = c(1,1), VarScan = c(NA,1))
bam_folder <- system.file("extdata", package = "appreci8R")
bam_folder <- paste(bam_folder, "/", sep="")

filtered<-evaluateCovAndBQ("", combined, bam_folder)
```

`filterTarget` *Excludes all off-target calls from further analysis.*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 1st analysis step, all off-target calls are excluded from further analysis. A list of data.frames (one list per sample) with all on-target calls is returned.

Usage

```
filterTarget (output_folder, caller_name, caller_folder, caller_file_names_add,
              caller_file_type, caller_snv_indel, caller_snv_names_add,
              caller_indel_names_add, caller_chr = 1, caller_pos = 2,
              caller_ref = 4, caller_alt = 5, targetRegions)
```

Arguments

`output_folder`	The folder to write the output files into. If an empty string is provided, no files are written out.
`caller_name`	Name of the variant calling tool (only necessary if an output folder is provided).
`caller_folder`	Folder containing the variant calling results.
`caller_file_names_add`	Suffix for naming the variant calling files. If an empty string is provided, it is assumed that the files only contain the sample name, e.g. "Sample1.vcf".
`caller_file_type`	File type of the variant calling results (".vcf" or ".txt").
`caller_snv_indel`	SNVs and indels are reported in the same file (`TRUE` or FALSE).
`caller_snv_names_add`	Suffix for naming the variant calling files containing SNVs (only evaluated if `caller_snv_indel==TRUE`).
`caller_indel_names_add`	Suffix for naming the variant calling files containing indels (only evaluated if `caller_snv_indel==TRUE`).
`caller_chr`	Column of the variant calling input containing information on chr (default: 1).
`caller_pos`	Column of the variant calling input containing information on pos (default: 2).
`caller_ref`	Column of the variant calling input containing information on ref (default: 4).

caller_alt	Column of the variant calling input containing information on alt (default: 5).
targetRegions	Data.frame object containing the target regions to be analyzed (bed-format: 1st column chr, 2nd column 0-based start pos, 3rd column 1-based end pos). Or: GRanges object containing the target regions to be analyzed.

Details

The function filterTarget covers two steps: reading input and target filtration.

First, all files in caller_folder of the file type caller_file_type with the suffix caller_file_names_add are read. Sample names are automatically derived from the file names (e.g. a sample name would be called "Sample1" if a file was called "Sample1.txt" and no suffix was defined; a sample would be called "Sample1.mutations" if a file was called "Sample1.mutations.vcf" and no suffix was defined, but "Sample1" if the suffix ".mutations" was defined).

If SNVs and indels are reported in separated files (in the same folder), caller_snv_indel==TRUE and caller_snv_names_add and caller_indel_names_add are defined, input from two files per sample is read and automatically combined (e.g. a sample would be called "Sample1" if files "Sample1.SNV.vcf" and "Sample1.indel.vcf" are read and caller_snv_indel==TRUE, caller_snv_names_add was defined as ".SNV" and caller_indel_names_add was defined as ".indel").

Subsequently, the read variant calling results are filtered according to the defined target region. All off-target calls are excluded from further analysis.

Value

A list of data.frames is returned. Every list element contains the information on one sample. Every data.frame contains the columns: the SampleID (taken from the input file names), Chr, Pos, Ref and Alt.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

appreci8R, appreci8Rshiny, normalize, annotate, combineOutput, evaluateCovAndBQ, determineCharacteristi
finalFiltration

Examples

```
output_folder<-""
target<-data.frame(chr = c("2","4","12","17","21","X"),
                   start = c(25469500,106196950,12046280,7579470,36164400,15838363),
                   end = c(25469510,106196960,12046350,7579475,36164410,15838366))
caller_folder <- system.file("extdata", package = "appreci8R")

targetFiltered<-filterTarget(output_folder, "GATK", caller_folder,
                             ".rawMutations", ".vcf", TRUE, "", "",
                             targetRegions = target)
```

finalFiltration *Perform final filtration according to the appreci8-algorithm*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 7th analysis step, the final filtration according to the appreci8-algorithm is performed. A GRanges object with all calls and their categorization as "Probably true", "Polymorphism" or "Artifact" is reported.

Usage

```
finalFiltration(output_folder, frequency_calls_g, database_calls_g, combined_calls_g,
                damaging_safe, tolerated_safe, primer = NA, hotspots = NA,
                overlapTools, nrsamples = 3, dp = 50, nr_alt = 20, vaf = 0.01,
                bq = 15, bq_diff = 7, detectedLow = 2, detectedHigh = 2,
                isIndel = 1, isIndelVAF = 1, detectedLowVAF = 2, noPrimerP = 1,
                primerPAlt = -1, noPrimerPFwd = 1, primerPFwd = -1,
                noPrimerPRev = 1, primerPRev = -1, primerLocation = -1,
                vafLow = 2, databaseVAF = 1, databaseHigh = 1,
                predictionSafe = -1, predictionVAF = 1, nrcaller4 = 4,
                reward4 = -1, nrcaller5 = 5, reward5 = -1, nrcaller6 = 6,
                reward6 = -1, oneCaller = 1, BQ_AltMean = 4, knownHotspot = -3,
                overlapReward = -3, artifactThreshold = 0, polyDetected = 1,
                polyDetectedOnce = -1, polyDatabasesPolyLow = 2,
                polyDatabasesPolyLowReward = 1, polyDatabasesPolyHigh = 4,
                polyDatabasesPolyHighReward = 1, polyDatabasesMut = 2,
                polyDatabasesMutReward = -1, polyNoDatabase = -1,
                polyDatabases = 6, polyDatabasesReward = 1, polyEffect = 1,
                polyVAF = 1, polyPrediction = 1, polyPredictionEffect = -1,
                polyCosmic = 100, polyThresholdCritical = 2, polyThreshold = 3,
                PolymorphismVAF10 = 5, PolymorphismVAF20 = 2,
                PolymorphismFrame = 2)
```

Arguments

output_folder	The folder to write the output files into. If an empty string is provided, no files are written out.
frequency_calls_g	A GRanges object. The GRanges object contains one call per line. EvaluateCovAndBQ()-output can directly be taken as input.
database_calls_g	A GRanges object. The GRanges object contains one call per line. EvaluateCharacteristics()-output can directly be taken as input.
combined_calls_g	A GRanges object. The GRanges object contains one call per line. CombineOutput()-output can directly be taken as input.
damaging_safe	Threshold for the impact prediction score, identifying reliably damaging calls. For example, if Provean has been selected as the impact prediction tool, the internal threshold of the tool is -2.5. However, calls with a score of -2.51 are

less likely to have a correct "Damaging" prediction compared to calls with a score of -12.0. A threshold of -5.0 for `damaging_safe` marks all calls with a score below -5.0 as reliably damaging.

`tolerated_safe` Threshold for the impact prediction score, identifying reliably tolerated calls. For example, if Provean has been selected as the impact prediction tool, the internal threshold of the tool is -2.5. However, calls with a score of -2.49 are less likely to have a correct "Neutral" prediction compared to calls with a score of +7.0. A threshold of -1.0 for `tolerated_safe` marks all calls with a score above -1.0 as reliably tolerated.

`primer` Optional: Data.frame containing primer positions in bed-format, i.e. 1st column chromosome, 2nd column 0-based start position, 3rd column 1-based end position (default: NA).

`hotspots` Optional: Data.frame containing expected/hotspot mutations: 1st column gene name (e.g. ASXL1), 2nd column mutation on AA level (e.g. G12S or E628fs or I836del or G12 if any AA change at this position is considered an expected/hotspot mutation), 3rd column minimum VAF or NA (default: NA).

`overlapTools` Vector of strings containing the names of variant calling tools that, should a call be overlappingly reported by these tools, will be rewarded.

`nrsamples` Threshold for the number of samples. Shold a variant be detected in more than the threshold defined by `nrsamples`, it will be interpreted as possible evidence for an artifact (default: 3).

`dp` Minimum depth that is required for a call (default: 50).

`nr_alt` Minimum number of reads carrying the alternate allele that is required for a call (default: 20).

`vaf` Minimum VAF that is required for a call (default: 0.01).

`bq` Minimum base quality that is required for a call (default: 15; values above 44 not recommended).

`bq_diff` Maximum difference in basequality between reference and alternative that is allowed for a call (default: 7).

`detectedLow` Value added to the artifact score if more than `nrsamples` number of samples feature the same call (default: 2).

`detectedHigh` Value added to the artifact score if more than 50% of the samples analyzed (if more than 1 sample is analyzed) feature the same call, which is not an expected/hotspot mutation (default: 2).

`isIndel` Value added to the artifact score if the call is an indel (default: 1).

`isIndelVAF` Value added to the artifact score if the call is an indel and the VAF is below 0.05 (default: 1).

`detectedLowVAF` Value added to the artifact score if more than `nrsamples` number of samples feature the same call and the VAF is above 0.85 (default: 2).

`noPrimerP` Value added to the artifact score if no primer information is provided or no primer is located at this position and the p value (Fisher's Exact Test evaluating forward and reverse reads to determine strandbias) is below 0.001 (default: 1).

`primerPAlt` Value added to the artifact score if no primer information is provided or no primer is located at this position and the p value (Fisher's Exact Test evaluating forward and reverse reads to determine strandbias) is below 0.001 and Nr_Alt_fwd is at least `Nr_Alt/2` and Nr_Alt_rev is at least `Nr_Alt/1` (default: -1).

noPrimerPFwd Value added to the artifact score if no primer information is provided or no primer is located at this position and the p value (Fisher's Exact Test evaluating forward and reverse reads to determine strandbias) is at least 0.001 and Nr_Ref_fwd is at least (DP-Nr_Alt)/2 and Nr_Alt_fwd is below 3 (default: 1).

primerPFwd Value added to the artifact score if no primer information is provided or no primer is located at this position and the p value (Fisher's Exact Test evaluating forward and reverse reads to determine strandbias) is below 0.001 and Nr_Ref_fwd is below (DP-Nr_Alt)/2 and Nr_Alt_fwd is below 3 (default: -1).

noPrimerPRev Value added to the artifact score if no primer information is provided or no primer is located at this position and the p value (Fisher's Exact Test evaluating forward and reverse reads to determine strandbias) is at least 0.001 and Nr_Ref_rev is at least (DP-Nr_Alt)/2 and Nr_Alt_rev is below 3 (default: 1).

primerPRev Value added to the artifact score if no primer information is provided or no primer is located at this position and the p value (Fisher's Exact Test evaluating forward and reverse reads to determine strandbias) is below 0.001 and Nr_Ref_rev is below (DP-Nr_Alt)/2 and Nr_Alt_rev is below 3 (default: -1).

primerLocation Value added to the artifact score if a call is located at a position where a primer is located (default: -1).

vafLow Value added to the artifact score if the VAF is below 0.02 (default: 2).

databaseVAF Value added to the artifact score if the VAF is below 0.10 and the variant was not found in any database (default: 1).

databaseHigh Value added to the artifact score if the variant was detected in at least 50% of all samples, but not found in any database (default: 1).

predictionSafe Value added to the artifact score if the variant has a reliable damaging prediction (default: -1).

predictionVAF Value added to the artifact score if the variant has a reliable tolerated prediction and a VAF below 0.35 or between 0.65 and 0.85 (default: 1).

nrcaller4 Intermediate number of callers that overlappingly reported a variant (default: 4).

reward4 Value added to the artifact score if a variant is overlappingly reported by nrcaller4 callers (default: -1).

nrcaller5 High number of callers that overlappingly reported a variant (default: 5).

reward5 Value added to the artifact score if a variant is overlappingly reported by nrcaller5 callers (default: -1).

nrcaller6 Very high number of callers that overlappingly reported a variant (default: 6).

reward6 Value added to the artifact score if a variant is overlappingly reported by nrcaller6 callers (default: -1).

oneCaller Value added to the artifact score if a variant is only reported by one caller (default: 1).

BQ_AltMean Value added to the artifact score if the base quality of a variant is below mean(BQ_Alt)-3*(BQ_Alt) over all variants (default: 4).

knownHotspot Value added to the artifact score if a variant is an expected/hotspot mutation (default: -3).

overlapReward Value added to the artifact score if a variant is overlappingly reported by the tools defined in overlapTools (default: -3).

artifactThreshold
Threshold for the artifact score, i.e. variants with a score below this threshold will be categorized as "probably true" (default: 0).

`polyDetected` Value added to the polymorphism score if more than `nrsamples` number of samples feature the same call (default: 1).

`polyDetectedOnce`
Value added to the polymorphism score if a variant is only detected in one sample (default: -1).

`polyDatabasesPolyLow`
Intermediate number of polymorphism databases that have information on a variant (default: 2).

`polyDatabasesPolyLowReward`
Value added to the polymorphism score if a variant is present in at least `polyDatabasesPolyLow` databases (default: 1).

`polyDatabasesPolyHigh`
High number of polymorphism databases that have information on a variant (default: 4).

`polyDatabasesPolyHighReward`
Value added to the polymorphism score if a variant is present in at least `polyDatabasesPolyHigh` databases (default: 1).

`polyDatabasesMut`
Critical number of mutation databases that have information on a variant (default: 2).

`polyDatabasesMutReward`
Value added to the polymorphism score if a variant is present in at least `polyDatabasesMut` databases (default: -1).

`polyNoDatabase` Value added to the polymorphism score if a variant is not present in any polymorphism database (default: -1).

`polyDatabases` High number of databases that have information on a variant (default: 6).

`polyDatabasesReward`
Value added to the polymorphism score if a variant is present in at least `polyDatabases` databases (default: 1).

`polyEffect` Value added to the polymorphism score if a variant is not a frameshift variant, not a stop gain variant and not a stop lost variant (default: 1).

`polyVAF` Value added to the polymorphism score if the VAF of a variant is between 0.65 and 0.35 or above 0.85 (default: 1).

`polyPrediction` Value added to the polymorphism score if the variant has a reliable tolerated prediction (default: 1).

`polyPredictionEffect`
Value added to the polymorphism score if a variant has a reliable damaging prediction or is a stop gain variant or is a stop lost variant (default: -1).

`polyCosmic` Critical number of COSMIC entries (default: 100).

`polyThresholdCritical`
Threshold for the polymorphism score if the number of COSMIC entries is not critical, i.e. variants with at least this score will be categorized as "polymorphism" (default: 2).

`polyThreshold` Threshold for the polymorphism score if the number of COSMIC entries is critical, i.e. variants with at least this score will be categorized as "polymorphism" (default: 3).

`PolymorphismVAF10`
Value added to the artifact score if a variant is a "polymorphism" based on the polymorphism score, but the VAF is below 0.10 (default: 5).

PolymorphismVAF20
: Value added to the artifact score if a variant is a "polymorphism" based on the polymorphism score, but the VAF is below 0.20 (default: 2).

PolymorphismFrame
: Value added to the artifact score if a variant is a "polymorphism" based on the polymorphism score, but it is a frameshift variant (default: 2).

Details

The function finalFiltration performs the final filtration according to the appreci8-algorithm. The previously determined characteristics of the calls are evaluated and an automatic categorization of the calls is performed. Possible categories are: Probably true or Hotspot, Polymorphism and Artifact. Final filtration consists of several steps:

1) Frequency and base quality are re-considered. Stricter thresholds compared to evaluateCovAndBQ can be defined.

2) Samples with the same call are considered. Counting is based on the normalized and annotated calls, not on the coverage- and base quality filtered calls.

3) Samples with a call at the same position are considered (e.g. A>AG at pos 1 in sample 1 and A>AGG at pos 1 in sample 2 are reported as 2 calls at the same position). Counting is based on the normalized and annotated calls, not on the coverage- and base quality filtered calls.

4) Background information is considered. The number of samples with the same variant in the coverage- and base quality filtered data are considered.

5) Number of databases is considered. Dependent on the previously selected number of databases. The presence of a variant in mutation- and polymorphism databases is evaluated.

6) VAF in relation to a predicted effect is considered. Variants are marked if their VAF is typical of polymorphisms and their predicted effect is "tolerated".

7) VAF in relation to the number of samples is considered. Variants are marked if they are detected in more than nrsamples samples and the VAF is at least 0.85 in more than 90% of these samples.

8) Strandbias is considered. Fisher's Exact Test is performed to analyze the relation between forward-reverse and reference-alternate allele.

9) Hotspot list - if any is provided - is considered. Variants are marked if they are present on the hotspot list.

10) Final filtration is performed. The artifact- and the polymorphism score are calculated. On the basis of these two scores, a call is either classified as a probably true/hotspot call, polymorphism or artifact.

Value

A GRanges object is returned containing all calls with a predicted category. Categories can be: probably true, hotspot, polymorphism or artifact. Reported metadata columns are: SampleID, Ref, Alt, Gene, GeneID, TranscriptID, Location, Consequence, c. (variant on cDNA level containing position and variant), c.complement (complement of the variant; if c. is c.5284A>G, then c.complement is c.5284T>C), p. (variant on protein level containing position and amino acids), Codon_ref, Codon_alt, Nr_Ref, Nr_Alt, DP, VAF, Caller1 to CallerX (dependent on the number of callers that is evaluated), dbSNP (containing the rs-ID), dbSNP_MAF, G1000_AF, ExAC_AF, ESP6500_AF, GAD_AF, CosmicID, Cosmic_Counts (number of Cosmic entries), ClinVar, Prediction (damaging or neutral), Score (on the basis of the selected prediction tool), BQ_REF, BQ_ALT, Nr_Ref_fwd, Nr_Alt_fwd, DP_fwd, VAF_rev, Nr_Ref_rev, Nr_Alt_rev, DP_rev, VAF_rev, strandbias, nr_samples (number of samples with the same variant), nr_samples_similar (number of samples with a variant at the same position), Category.

If an output folder is provided, the output is saved as Results_Final.txt. Additionally, Results_Final.xlsx is saved, containing three sheets: Mutations, Polymorphisms and Artifacts.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

appreci8R, appreci8Rshiny, filterTarget, normalize, annotate, combineOutput, evaluateCovAndBQ, determineCharacteristics

Examples

```
library("GenomicRanges")
filtered<-GRanges(seqnames = c("4","X"),
                  ranges = IRanges(start = c (106196951,15838366),
                                   end = c (106196951,15838366)),
                  SampleID = c("Sample2","Sample1"), Ref = c("A","C"),
                  Alt = c("G","A"), Location = c("coding,coding","coding"),
                  c. = c("5284,5347","864"), p. = c("1762,1783","288"),
                  AA_ref = c("I,I","N"), AA_alt = c("V,V","K"),
                  Codon_ref = c("ATA,ATA","AAC"),
                  Codon_alt = c("GTA,GTA","AAA"),
                  Consequence = c("nonsynonymous,nonsynonymous","nonsynonymous"),
                  Gene = c("TET2,TET2","ZRSR2"),
                  GeneID = c("54790,54790","8233"),
                  TranscriptID = c("18308,18309","75467"),
                  GATK = c(NA,1), VarScan = c(1,NA), Nr_Ref = c(1268,1991),
                  Nr_Alt = c(1283,31), DP = c(2551,3630),
                  VAF = c(0.5029,0.0085), BQ_REF = c(38.67,37.46),
                  BQ_ALT = c(38.81,18.00), Nr_Ref_fwd = c(428,839),
                  Nr_Alt_fwd = c(469,0), DP_fwd = c(897,1507),
                  VAF_fwd = c(0.5229,0.0000), Nr_Ref_rev = c(840,1152),
                  Nr_Alt_rev = c(814,31), DP_rev = c(1654,2123),
                  VAF_rev = c(0.4921,0.0146))
databases<-GRanges(seqnames = c("4","X"),
                  ranges = IRanges(start = c (106196951,15838366),
                                   end = c (106196951,15838366)),
                  SampleID = c("Sample2","Sample1"), Ref = c("A","C"),
                  Alt = c("G","A"),dbSNP = c("rs2454206",NA),
                  dbSNP_MAF = c(0.2304,NA), G1000_AF = c(0.23,0.44),
                  ExAC_AF = c(0.27,0.47), ESP6500_AF = c(0.29,0.49),
                  GAD_AF = c(0.30,0.47), CosmicID = c(NA,NA),
                  Cosmic_Counts = c(NA,NA), ClinVar = c(NA,NA),
                  Prediction = c("Neutral",NA), Score = c(-0.061,NA),
                  c. = c("c.5284A>G,c.5347A>G","c.864C>A"),
                  c.complement = c("c.5284T>C,c.5347T>C","c.864G>T"),
                  p. = c("p.I1762V,p.I1783V","p.N288K"))
filtered<-GRanges(seqnames = c("4","X"),
                  ranges = IRanges(start = c (106196951,15838366),
                                   end = c (106196951,15838366)),
                  SampleID = c("Sample2","Sample1"), Ref = c("A","C"),
                  Alt = c("G","A"), Location = c("coding,coding","coding"),
                  c. = c("5284,5347","864"), p. = c("1762,1783","288"),
                  AA_ref = c("I,I","N"), AA_alt = c("V,V","K"),
                  Codon_ref = c("ATA,ATA","AAC"),
```

```
                    Codon_alt = c("GTA,GTA","AAA"),
                    Consequence = c("nonsynonymous,nonsynonymous","nonsynonymous"),
                    Gene = c("TET2,TET2","ZRSR2"),
                    GeneID = c("54790,54790","8233"),
                    TranscriptID = c("18308,18309","75467"),
                    GATK = c(NA,1), VarScan = c(1,NA))


final<-finalFiltration("", frequency_calls = filtered,
                       database_calls = databases, combined_calls = combined,
                       damaging_safe = -3, tolerated_safe = -1.5,
                       overlapTools = c("VarScan"), bq_diff = 20,vaf = 0.001)
```

`normalize` *Normalize calls*

Description

appreci8R combines and filters the output of different variant calling tools according to the 'appreci8'-algorithm. In the 2nd analysis step, all calls are normalized with respect to reporting indels, MNVs, reporting of several alternate alleles and reporting of complex indels. A GRanges object with all normalized calls is returned.

Usage

```
normalize(output_folder, caller_name, target_calls, caller_indels_pm,
          caller_mnvs)
```

Arguments

`output_folder`	The folder to write the output files into. If an empty string is provided, no files are written out.
`caller_name`	Name of the variant calling tool (only necessary if an output folder is provided).
`target_calls`	List of data.frames. One list element per sample. FilterTarget()-output can directly be taken as input.
`caller_indels_pm`	Deletions are reported with a "minus" (e.g. C > -A), insertions are reported with a "plus" (e.g. C > +A) (`TRUE` or FALSE).
`caller_mnvs`	MNVs are reported (e.g. CA > GT instead of C > G and A > T; or CGAG > TGAT instead of C > T and G > T) (`TRUE` or FALSE).

Details

The function `normalize` covers two to four normalization steps:

1) Check alternative bases: Calls containing a "comma" are split up. A call like C > A,G is converted to C > A and an additional C > G call. This enables evaluation of the output of different callers while caller1 reports C > A,G, caller2 only reports C > A and caller3 only reports C > G. This normalization step is always performed.

2) Find string differences: Calls are checked for un-mutated bases. The smallest option of reporting a variant at the left-most position is chosen. For example, CAAAC > CAAC is converted to CA > C. This normalization step is always performed.

3) Convert indels: If deletions are reported with a "minus" and insertions are reported with a "plus", these are converted. An deletion like C > -G is converted to CG > C, while an insertion like C > +G is converted to C > CG. This normalization step is only performed if `caller_indels_pm` is `TRUE`.

4) Convert MNVs: If MNVs are reported, these are converted. This enables evaluation of the output of different callers if not all callers report all mutations being part of an MNV. A call like CA > GT is split up to a C > G and an A > T variant. But also a call like CGAG > TGAT is split up to C > T and G > T (G > G and A > A are not reported as they do not pass the normalization step "Find string differences"). This normalization step is only performed if `caller_mnvs` is `TRUE`.

Value

A GRanges object is returned (metadata columns: SampleID, Ref, Alt).

If an output folder is provided, the output is saved as <caller_name>.normalized.txt.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

appreci8R, appreci8Rshiny, filterTarget, annotate, combineOutput, evaluateCovAndBQ, determineCharacteristics, finalFiltration

Examples

```
sample1<-data.frame(SampleID = c("Sample1","Sample1","Sample1"),
                    Chr = c("2","17","X"),
                    Pos = c(25469502,7579472,15838366),
                    Ref = c("CAG","G","C"),
                    Alt = c("TAT","C","T,A"))
sample2<-data.frame(SampleID = c("Sample2","Sample2","Sample2","sample2"),
                    Chr = c("4","12","12","21"),
                    Pos = c(106196951,12046289,12046341,36164405),
                    Ref = c("A","C","A","GGG"),
                    Alt = c("G","+AAAG","G","TGG"))
input<-list(sample1, sample2)

normalized<-normalize("", "", input, TRUE, TRUE)
```

Index