# Package 'ICAMS'

September 29, 2021

**Type** Package

**Title** In-Depth Characterization and Analysis of Mutational Signatures ('ICAMS')

**Version** 3.0.1.9004

**Author** Steve Rozen, Nanhai Jiang, Arnoud Boot, Mo Liu, Yang Wu

**Maintainer** Steve Rozen <steverozen@gmail.com>

**Description** Analysis and visualization of experimentally elucidated mutational
signatures -- the kind of analysis and visualization in Boot et al.,
``In-depth characterization of the cisplatin mutational signature in
human cell lines and in esophageal and liver tumors'', Genome Research 2018,
<doi:10.1101/gr.230219.117> and
``Characterization of colibactin-associated mutational signature in an
Asian oral squamous cell carcinoma and in other mucosal tumor types'',
Genome Research 2020 <doi:10.1101/gr.255620.119>.
'ICAMS' stands for In-depth Characterization
and Analysis of Mutational Signatures. 'ICAMS' has functions to read in
variant call files (VCFs) and to collate the corresponding catalogs of
mutational spectra and to analyze and plot catalogs of mutational spectra
and signatures. Handles both ``counts-based'' and ``density-based'' (i.e.
representation as mutations per megabase) mutational spectra or signatures.

**License** GPL-3 | file LICENSE

**URL** https://github.com/steverozen/ICAMS

**BugReports** https://github.com/steverozen/ICAMS/issues

**Encoding** UTF-8

**LazyData** true

**Language** en-US

**biocViews**

**Imports** Biostrings,
BSgenome,
data.table,
dplyr,
fuzzyjoin,
GenomeInfoDb,
GenomicRanges,
graphics,
grDevices,

IRanges,
lifecycle,
RColorBrewer,
stats,
stringi,
utils,
zip

**Depends**  R (>= 3.5),

**RoxygenNote**  7.1.1

**Suggests**  BSgenome.Hsapiens.1000genomes.hs37d5,
BSgenome.Hsapiens.UCSC.hg38,
BSgenome.Mmusculus.UCSC.mm10,
ggplot2,
reshape2,
rlang,
testthat

# R **topics documented:**

---

all.abundance *K-mer abundances*

---

### Description

An R list with one element each for `BSgenome.Hsapiens.1000genomes.hs37d5`, `BSgenome.Hsapiens.UCSC.hg38` and `BSgenome.Mmusculus.UCSC.mm10`. Each element is in turn a sub-list keyed by `exome`, `transcript`, and `genome`. Each element of the sub list is keyed by the number of rows in the catalog class (as a string, e.g. `"78"`, not `78`). The keys are: 78 (`DBS78Catalog`), 96 (`SBS96Catalog`), 136 (`DBS136Catalog`), 144 (`DBS144Catalog`), 192 (`SBS192Catalog`), and 1536 (`SBS1536Catalog`). So, for example to get the exome abundances for SBS96 catalogs for `BSgenome.Hsapiens.UCSC.hg38` exomes one would reference
`all.abundance[["BSgenome.Hsapiens.UCSC.hg38"]][["exome"]][["96"]]`
or `all.abundance$BSgenome.Hsapiens.UCSC.hg38$exome$"96"`. The value of the abundance is an integer vector with the K-mers as names and each value being the count of that K-mer.

### Usage

```
all.abundance
```

### Format

See Description.

### Examples

```
all.abundance$BSgenome.Hsapiens.UCSC.hg38$transcript$`144`
#        AA        AC        AG        AT        CA        CC ...
#  90769160  57156295  85738416  87552737  83479655  63267896 ...
# There are 90769160 AAs on the sense strands of transcripts in
# this genome.
```

---

AnnotateDBSVCF                  *Add sequence context and transcript information to an in-memory DBS VCF*

---

### Description

Add sequence context and transcript information to an in-memory DBS VCF

### Usage

```
AnnotateDBSVCF(DBS.vcf, ref.genome, trans.ranges = NULL, name.of.VCF = NULL)
```

### Arguments

DBS.vcf        An in-memory DBS VCF as a `data.frame`.

ref.genome     A `ref.genome` argument as described in `ICAMS`.

trans.ranges   Optional. If `ref.genome` specifies one of the `BSgenome` object

1. BSgenome.Hsapiens.1000genomes.hs37d5
2. BSgenome.Hsapiens.UCSC.hg38
3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to `TranscriptRanges` for more details. If `is.null(trans.ranges)` do not add transcript range information.

name.of.VCF    Name of the VCF file.

### Value

An in-memory DBS VCF as a `data.table`. This has been annotated with the sequence context (column name `seq.21bases`) and with transcript information in the form of a gene symbol (e.g. `"TP53"`) and transcript strand. This information is in the columns `trans.start.pos`, `trans.end.pos`, `trans.strand`, `trans.Ensembl.gene.ID` and `trans.gene.symbol` in the output. These columns are not added if `is.null(trans.ranges)`.

### Examples

```
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitVCFs(file, variant.caller = "strelka")
DBS.vcf <- list.of.vcfs$DBS[[1]]
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.DBS.vcf <- AnnotateDBSVCF(DBS.vcf, ref.genome = "hg19",
                                      trans.ranges = trans.ranges.GRCh37)}
```

---

AnnotateIDVCF *Add sequence context to an in-memory ID (insertion/deletion) VCF, and confirm that they match the given reference genome*

---

## Description

Add sequence context to an in-memory ID (insertion/deletion) VCF, and confirm that they match the given reference genome

## Usage

```
AnnotateIDVCF(
  ID.vcf,
  ref.genome,
  flag.mismatches = 0,
  name.of.VCF = NULL,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

ID.vcf
: An in-memory ID (insertion/deletion) VCF as a `data.frame`. This function expects that there is a "context base" to the left, for example REF = ACG, ALT = A (deletion of CG) or REF = A, ALT = ACC (insertion of CC).

ref.genome
: A `ref.genome` argument as described in [ICAMS](#).

flag.mismatches
: Deprecated. If there are ID variants whose REF do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants. See element `discarded.variants` in the return value for more details.

name.of.VCF
: Name of the VCF file.

suppress.discarded.variants.warnings
: Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Value

A list of elements:

- `annotated.vcf`: The original VCF data frame with two new columns added to the input data frame:

  - `seq.context`: The sequence embedding the variant.
  - `seq.context.width`: The width of `seq.context` to the left.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

## Examples

```
file <- c(system.file("extdata/Strelka-ID-vcf/",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMS"))
ID.vcf <- ReadAndSplitVCFs(file, variant.caller = "strelka")$ID[[1]]
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  list <- AnnotateIDVCF(ID.vcf, ref.genome = "hg19")
  annotated.ID.vcf <- list$annotated.vcf}
```

---

| AnnotateSBSVCF | *Add sequence context and transcript information to an in-memory SBS VCF* |
|---|---|

---

## Description

Add sequence context and transcript information to an in-memory SBS VCF

## Usage

```
AnnotateSBSVCF(SBS.vcf, ref.genome, trans.ranges = NULL, name.of.VCF = NULL)
```

## Arguments

| | |
|---|---|
| SBS.vcf | An in-memory SBS VCF as a data.frame. |
| ref.genome | A ref.genome argument as described in ICAMS. |
| trans.ranges | Optional. If ref.genome specifies one of the BSgenome object |

        1. BSgenome.Hsapiens.1000genomes.hs37d5

        2. BSgenome.Hsapiens.UCSC.hg38

        3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer trans.ranges automatically. Otherwise, user will need to provide the necessary trans.ranges. Please refer to TranscriptRanges for more details. If is.null(trans.ranges) do not add transcript range information.

| | |
|---|---|
| name.of.VCF | Name of the VCF file. |

## Value

An in-memory SBS VCF as a data.table. This has been annotated with the sequence context (column name seq.21bases) and with transcript information in the form of a gene symbol (e.g. "TP53") and transcript strand. This information is in the columns trans.start.pos, trans.end.pos, trans.strand, trans.Ensembl.gene.ID and trans.gene.symbol in the output. These columns are not added if is.null(trans.ranges).

## Examples

```
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitVCFs(file, variant.caller = "strelka")
SBS.vcf <- list.of.vcfs$SBS[[1]]
```

```
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.SBS.vcf <- AnnotateSBSVCF(SBS.vcf, ref.genome = "hg19",
                                      trans.ranges = trans.ranges.GRCh37)}
```

---

as.catalog        *Create a catalog from a* matrix, data.frame, *or* vector

---

### Description

Create a catalog from a matrix, data.frame, or vector

### Usage

```
as.catalog(
  object,
  ref.genome = NULL,
  region = "unknown",
  catalog.type = "counts",
  abundance = NULL,
  infer.rownames = FALSE
)
```

### Arguments

object
: A numeric matrix, numeric data.frame, or vector. If a vector, converted to a 1-column matrix with rownames taken from the element names of the vector and with column name "Unknown". If argument infer.rownames is FALSE then this argument must have rownames to denote the mutation types. See CatalogRowOrder for more details.

ref.genome
: A ref.genome argument as described in ICAMS.

region
: A character string designating a region, one of genome, transcript, exome, unknown; see ICAMS. If the catalog type is a stranded catalog type (SBS192 or DBS144), region = "genome" will be silently converted to "transcript".

catalog.type
: One of "counts", "density", "counts.signature", "density.signature".

abundance
: If NULL, then inferred if ref.genome is one of the reference genomes known to ICAMS and region is not unknown. See ICAMS. The argument abundance should contain the counts of different source sequences for mutations in the same format as the numeric vectors in all.abundance.

infer.rownames
: If TRUE, and object has no rownames, then assume the rows of object are in the correct order and add the rownames implied by the number of rows in object (e.g. rownames for SBS 192 if there are 192 rows). If TRUE, **be sure the order of rows is correct.**

### Value

A catalog as described in ICAMS.

## Examples

```
# Create an SBS96 catalog with all mutation counts equal to 1.
object <- matrix(1, nrow = 96, ncol = 1,
                 dimnames = list(catalog.row.order$SBS96))
catSBS96 <- as.catalog(object)
```

---

Canonicalize1Del          *Given a deletion and its sequence context, categorize it*

---

## Description

This function is primarily for internal use, but we export it to document the underlying logic.

## Usage

```
Canonicalize1Del(context, del.seq, pos, trace = 0)
```

## Arguments

| | |
|---|---|
| context | The deleted sequence plus ample surrounding sequence on each side (at least as long as del.seq). |
| del.seq | The deleted sequence in context. |
| pos | The position of del.sequence in context. |
| trace | If > 0, then generate messages tracing how the computation is carried out. |

## Details

See https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_
indel_classification_2017_12_08.xlsx for additional information on deletion muta-
tion classification.

This function first handles deletions in homopolymers, then handles deletions in simple repeats with
longer repeat units, (e.g. CACACACA, see FindMaxRepeatDel), and if the deletion is not in a
simple repeat, looks for microhomology (see FindDelMH).

See the code for unexported function CanonicalizeID and the functions it calls for handling of
insertions.

## Value

A string that is the canonical representation of the given deletion type. Return NA and raise a warn-
ing if there is an un-normalized representation of the deletion of a repeat unit. See FindDelMH for
details. (This seems to be very rare.)

## Examples

```
Canonicalize1Del("xyAAAqr", del.seq = "A", pos = 3) # "DEL:T:1:2"
Canonicalize1Del("xyAAAqr", del.seq = "A", pos = 4) # "DEL:T:1:2"
Canonicalize1Del("xyAqr", del.seq = "A", pos = 3)   # "DEL:T:1:0"
```

---

CatalogRowOrder *Standard order of row names in a catalog*

---

### Description

This data is designed for those who need to create their own catalogs from formats not supported by this package. The rownames denote the mutation types. For example, for SBS96 catalogs, the rowname AGAT represents a mutation from AGA > ATA.

### Usage

```
catalog.row.order
```

### Format

A list of character vectors indicating the standard orders of row names in catalogs.

An object of class `list` of length 8.

### ID classification

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. `CACACACA`, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

### Note

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+. In ID83 catalogs, deletion repeat sizes range from 0 to 5.

### Examples

```
catalog.row.order$SBS96
# "ACAA" "ACCA" "ACGA" "ACTA" "CCAA" "CCCA" "CCGA" "CCTA" ...
# There are altogether 96 row names to denote the mutation types
# in SBS96 catalog.
```

---

CollapseCatalog        *"Collapse" a catalog*

---

**Description**

1. Take a mutational spectrum or signature catalog that is based on a fined-grained set of features (for example, single-nucleotide substitutions in the context of the preceding and following 2 bases).

2. Collapse it to a catalog based on a coarser-grained set of features (for example, single-nucleotide substitutions in the context of the immediately preceding and following bases).

Collapse192CatalogTo96 Collapse an SBS 192 catalog to an SBS 96 catalog.

Collapse1536CatalogTo96 Collapse an SBS 1536 catalog to an SBS 96 catalog.

Collapse144CatalogTo78 Collapse a DBS 144 catalog to a DBS 78 catalog.

**Usage**

```
Collapse192CatalogTo96(catalog)

Collapse1536CatalogTo96(catalog)

Collapse144CatalogTo78(catalog)
```

**Arguments**

catalog          A catalog as defined in ICAMS.

**Value**

A catalog as defined in ICAMS.

**Examples**

```
# Create an SBS192 catalog and collapse it to an SBS96 catalog
object <- matrix(1, nrow = 192, ncol = 1,
                 dimnames = list(catalog.row.order$SBS192))
catSBS192 <- as.catalog(object, region = "transcript")
catSBS96 <- Collapse192CatalogTo96(catSBS192)
```

---

FindDelMH              *Return the length of microhomology at a deletion*

---

**Description**

Return the length of microhomology at a deletion

**Usage**

```
FindDelMH(context, deleted.seq, pos, trace = 0, warn.cryptic = TRUE)
```

**Arguments**

| | |
|---|---|
| `context` | The deleted sequence plus ample surrounding sequence on each side (at least as long as `del.sequence`). |
| `deleted.seq` | The deleted sequence in `context`. |
| `pos` | The position of `del.sequence` in `context`. |
| `trace` | If > 0, then generate various messages showing how the computation is carried out. |
| `warn.cryptic` | if `TRUE` generating a warning if there is a cryptic repeat (see the example). |

**Details**

This function is primarily for internal use, but we export it to document the underlying logic.

Example:

`GGCTAGTT` aligned to `GGCTAGAACTAGTT` with a deletion represented as:

```
GGCTAGAACTAGTT
GG------CTAGTT GGCTAGTT GG[CTAGAA]CTAGTT
                            ----    ----
```

Presumed repair mechanism leading to this:

```
   ....
GGCTAGAACTAGTT
CCGATCTTGATCAA

=>


   ....
GGCTAG      TT
CC      GATCAA
        ....

=>

GGCTAGTT
CCGATCAA
```

Variant-caller software can represent the same deletion in several different, but completely equivalent, ways.

```
GGC------TAGTT GGCTAGTT GGC[TAGAAC]TAGTT
                           *  ---   *  ---

GGCT------AGTT GGCTAGTT GGCT[AGAACT]AGTT
                            **  --   **  --

GGCTA------GTT GGCTAGTT GGCTA[GAACTA]GTT
```

```
                                  *** -   *** -

GGCTAG-------TT GGCTAGTT GGCTAG[AACTAG]TT
                                  ****    ****
```

This function finds:

1. The maximum match of undeleted sequence to the left of the deletion that is identical to the right end of the deleted sequence, and

2. The maximum match of undeleted sequence to the right of the deletion that is identical to the left end of the deleted sequence.

The microhomology sequence is the concatenation of items (1) and (2).

**Warning**

A deletion in a *repeat* can also be represented in several different ways. A deletion in a repeat is abstractly equivalent to a deletion with microhomology that spans the entire deleted sequence. For example;

```
GACTAGCTAGTT
GACTA----GTT GACTAGTT GACTA[GCTA]GTT
                               *** -*** -
```

is really a repeat

```
GACTAG----TT GACTAGTT GACTAG[CTAG]TT
                               **** ----

GACT----AGTT GACTAGTT GACT[AGCT]AGTT
                             ** --** --
```

**This function only flags these "cryptic repeats" with a -1 return; it does not figure out the repeat extent.**

**Value**

The length of the maximum microhomology of del.sequence in context.

**ID classification**

See https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for Canonicalize1Del which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see FindMaxRepeatDel), and if the deletion is not in a simple repeat, looks for microhomology (see FindDelMH).

See the code for unexported function CanonicalizeID and the functions it calls for handling of insertions.

## Examples

```
# GAGAGG[CTAGAA]CTAGTT
#         ----    ----
FindDelMH("GGAGAGGCTAGAACTAGTTAAAAA", "CTAGAA", 8, trace = 0)  # 4

# A cryptic repeat
#
# TAAATTATTTATTAATTTATTG
# TAAATTA----TTAATTTATTG = TAAATTATTAATTTATTG
#
# equivalent to
#
# TAAATTATTTATTAATTTATTG
# TAAAT----TATTAATTTATTG = TAAATTATTAATTTATTG
#
# and
#
# TAAATTATTTATTAATTTATTG
# TAAA----TTATTAATTTATTG = TAAATTATTAATTTATTG

FindDelMH("TAAATTATTTATTAATTTATTG", "TTTA", 8, warn.cryptic = FALSE) # -1
```

---

FindMaxRepeatDel     *Return the number of repeat units in which a deletion is embedded*

---

## Description

Return the number of repeat units in which a deletion is embedded

## Usage

```
FindMaxRepeatDel(context, rep.unit.seq, pos)
```

## Arguments

| | |
|---|---|
| context | A string that embeds rep.unit.seq at position pos |
| rep.unit.seq | A substring of context at pos to pos + nchar(rep.unit.seq) −1, which is the repeat unit sequence. |
| pos | The position of rep.unit.seq in context. |

## Details

This function is primarily for internal use, but we export it to document the underlying logic.

For example FindMaxRepeatDel("xyaczt","ac",3) returns 0.

If substr(context,pos,pos + nchar(rep.unit.seq) −1) != rep.unit.seq then stop.

If this functions returns 0, then it is necessary to look for microhomology using the function FindDelMH.

### Warning

This function depends on the variant caller having "aligned" the deletion within the context of the repeat.

For example, a deletion of CAG in the repeat

```
GTCAGCAGCATGT
```

can have 3 "aligned" representations as follows:

```
CT---CAGCAGGT
CTCAG---CAGGT
CTCAGCAG---GT
```

In these cases this function will return 2. (Please not that the return value does not include the `rep.uni.seq` in the count.)

However, the same deletion can also have an "unaligned" representation, such as

```
CTCAGC---AGGT
```

(a deletion of `AGC`).

In this case this function will return 1 (a deletion of `AGC` in a 2-element repeat of `AGC`).

## Value

The number of repeat units in which `rep.unit.seq` is embedded, not including the input `rep.unit.seq` in the count.

## ID classification

See https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. `CACACACA`, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

## Examples

```
FindMaxRepeatDel("xyACACzt", "AC", 3) # 1
FindMaxRepeatDel("xyACACzt", "CA", 4) # 0
```

---

GeneExpressionData *Example gene expression data from two cell lines*

---

## Description

This data is designed to be used as an example in function `PlotTransBiasGeneExp` and `PlotTransBiasGeneExpToPdf`.

## Usage

```
gene.expression.data.HepG2

gene.expression.data.MCF10A
```

## Format

A [data.table](#) which contains the expression values of genes.

An object of class data.table (inherits from data.frame) with 57736 rows and 4 columns.

An object of class data.table (inherits from data.frame) with 57736 rows and 4 columns.

## Examples

```
gene.expression.data.HepG2
# Ensembl.gene.ID  gene.symbol  counts            TPM
# ENSG00000000003      TSPAN6    6007   33.922648455
# ENSG00000000005        TNMD       0    0.000000000
# ENSG00000000419        DPM1    4441   61.669371091
# ENSG00000000457        SCYL3    1368    3.334619195
# ENSG00000000460      C1orf112     916    2.416263423
#                ...          ...     ...            ...
```

---

```
GeneratePlotPFMmatrix
```
                    *Generate PFMmatrix (Position Frequency Matrix) from a given list of*
                    *sequences*

---

## Description

Generate PFMmatrix (Position Frequency Matrix) from a given list of sequences

## Usage

```
GeneratePlotPFMmatrix(
  sequences,
  indel.class,
  flank.length = 5,
  plot.dir = NULL,
  plot.title = NULL
)
```

## Arguments

| | |
|---|---|
| sequences | A list of strings returned from [SymmetricalContextsFor1BPIndel](#). |
| indel.class | A single character string that denotes a 1 base pair insertion or deletion, as taken from ICAMS::catalog.row.order$ID. Insertions or deletions into or from 5+ base-pair homopolymers are not supported. |
| flank.length | The length of flanking bases around the position or homopolymer targeted by the indel. |
| plot.dir | If provided, make a dot-line plot for PFMmatrix. |
| plot.title | The title of the dot-line plot |

## Value

A matrix recording the frequency of each base (A, C, G, T) on each position of the sequence.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
split.vcfs <- ReadAndSplitVCFs(file, variant.caller = "mutect")
ID.catalog <- VCFsToIDCatalogs(list.of.vcfs = split.vcfs$ID,
                               ref.genome = "hg19",
                               region = "genome",
                               return.annotated.vcfs = TRUE)
annotated.vcf <- ID.catalog$annotated.vcfs$Mutect.GRCh37.s1
extended.seq.contexts <-
  SymmetricalContextsFor1BPIndel(annotated.vcf = annotated.vcf,
                                 indel.class = "DEL:T:1:0")
GeneratePlotPFMmatrix(sequences = extended.seq.contexts,
                      indel.class = "DEL:T:1:0",
                      plot.dir = file.path(tempdir(), "test.pdf"),
                      plot.title = "Deletion of 1T from 1T")
```

---

| GetVAF | *Extract the VAFs (variant allele frequencies) and read depth information from a VCF file* |
|---|---|

---

## Description

Extract the VAFs (variant allele frequencies) and read depth information from a VCF file

## Usage

```
GetStrelkaVAF(vcf, name.of.VCF = NULL)

GetMutectVAF(vcf, name.of.VCF = NULL, tumor.col.name = NA)

GetFreebayesVAF(vcf, name.of.VCF = NULL)

GetPCAWGConsensusVAF(vcf, mc.cores = 1)
```

## Arguments

| | |
|---|---|
| `vcf` | An in-memory VCF data frame. |
| `name.of.VCF` | Name of the VCF file. |
| `tumor.col.name` | |
| | Optional. Only applicable to **Mutect** VCF. Name or index of the column in **Mutect** VCF which contains the tumor sample information. It **must** have quotation marks if specifying the column name. If `tumor.col.name` is equal to `NA`(default), this function will use the 10th column to calculate VAFs. |
| `mc.cores` | The number of cores to use. Not available on Windows unless `mc.cores = 1`. |

**Value**

The original `vcf` with two additional columns added which contain the VAF(variant allele frequency) and read depth information.

**Note**

GetPCAWGConsensusVAF is analogous to GetMutectVAF, calculating VAF and read depth from PCAWG7 consensus vcfs

**Examples**

```
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
MakeDataFrameFromVCF <- getFromNamespace("MakeDataFrameFromVCF", "ICAMS")
df <- MakeDataFrameFromVCF(file)
df1 <- GetStrelkaVAF(df)
```

---

HaplotypePlot *Generate Haplotype plot from a given list of sequences*

---

**Description**

Generate Haplotype plot from a given list of sequences

**Usage**

```
HaplotypePlot(
  sequences,
  indel.class,
  flank.length = 5,
  title = "Haplotype Plot"
)
```

**Arguments**

| | |
|---|---|
| sequences | A list of strings returned from SymmetricalContextsFor1BPIndel. |
| indel.class | A single character string that denotes a 1 base pair insertion or deletion, as taken from ICAMS::catalog.row.order$ID. Insertions or deletions into or from 5+ base-pair homopolymers are not supported. |
| flank.length | The length of flanking bases around the position or homopolymer targeted by the indel. |
| title | The title of the haplotype plot |

**Value**

A ggplot2 object

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
split.vcfs <- ReadAndSplitVCFs(file, variant.caller = "mutect")
ID.catalog <- VCFsToIDCatalogs(list.of.vcfs = split.vcfs$ID,
                               ref.genome = "hg19",
                               region = "genome",
                               return.annotated.vcfs = TRUE)
annotated.vcf <- ID.catalog$annotated.vcfs$Mutect.GRCh37.s1
extended.seq.contexts <-
  SymmetricalContextsFor1BPIndel(annotated.vcf = annotated.vcf,
                                 indel.class = "INS:T:1:4")
ggplot.object <- HaplotypePlot(sequences = extended.seq.contexts,
                               indel.class = "INS:T:1:4",
                               title = "Deletion of 1T from 4Ts")
plot(ggplot.object)
```

---

| ICAMS | *ICAMS: In-depth Characterization and Analysis of Mutational Signatures* |
|---|---|

---

### Description

Analysis and visualization of experimentally elucidated mutational signatures – the kind of analysis and visualization in Boot et al., "In-depth characterization of the cisplatin mutational signature in human cell lines and in esophageal and liver tumors",
*Genome Research* 2018 https://doi.org/10.1101/gr.230219.117 and "Characterization of colibactin-associated mutational signature in an Asian oral squamous cell carcinoma and in other mucosal tumor types", *Genome Research* 2020, https://doi.org/10.1101/gr.255620.119. "ICAMS" stands for In-depth Characterization and Analysis of Mutational Signatures. "ICAMS" has functions to read in variant call files (VCFs) and to collate the corresponding catalogs of mutational spectra and to analyze and plot catalogs of mutational spectra and signatures.

### Details

"ICAMS" can read in VCFs generated by Strelka, Mutect or other variant callers, and collate the mutations into "catalogs" of mutational spectra. "ICAMS" can create and plot catalogs of mutational spectra or signatures for single base substitutions (SBS), doublet base substitutions (DBS), and small insertions and deletions (ID). It can also read and write these catalogs.

### Catalogs

A key data type in "ICAMS" is a "catalog" of mutation counts, of mutation densities (see below), or of mutational signatures.

Catalogs are S3 objects of class `matrix` and one of several additional classes that specify the types of the mutations represented in the catalog. The additional class is one of

- `SBS96Catalog` (strand-agnostic single base substitutions in trinucleotide context)
- `SBS192Catalog` (transcription-stranded single-base substitutions in trinucleotide context)
- `SBS1536Catalog`

- `DBS78Catalog`
- `DBS144Catalog`
- `DBS136Catalog`
- `IndelCatalog`

`as.catalog` is the main constructor.

Conceptually, a catalog also has one of the following types, indicated by the attribute `catalog.type`:

1. Matrix of mutation counts (one column per sample), representing (counts-based) mutational spectra (`catalog.type = "counts"`).

2. Matrix of mutation **densities**, i.e. mutations per occurrences of source sequences (one column per sample), representing (density-based) mutational spectra (`catalog.type = "density"`).

3. Matrix of mutational signatures, which are similar to spectra. However where spectra consist of counts or densities of mutations in each mutation class (e.g. ACA > AAA, ACA > AGA, ACA > ATA, ACC > AAC, ...), signatures consist of the proportions of mutations in each class (with all the proportions summing to 1). A mutational signature can be based on either:

   - mutation counts (a "counts-based mutational signature", `catalog.type = "counts.signature"`), or
   - mutation densities (a "density-based mutational signature", `catalog.type = "density.signature"`).

Catalogs also have the attribute `abundance`, which contains the counts of different source sequences for mutations. For example, for SBSs in trinucleotide context, the abundances would be the counts of each trinucleotide in the human genome, exome, or in the transcribed region of the genome. See `TransformCatalog` for more information. Abundances logically depend on the species in question and on the part of the genome being analyzed.

In "ICAMS" abundances can sometimes be inferred from the `catalog` class attribute and the function arguments `region`, `ref.genome`, and `catalog.type`. Otherwise abundances can be provided as an `abundance` argument. See `all.abundance` for examples.

Possible values for `region` are the strings `genome`, `transcript`, `exome`, and `unknown`; `transcript` includes entire transcribed regions, i.e. the introns as well as the exons.

If you need to create a catalog from a source other than this package (i.e. other than with `ReadCatalog` or `VCFsToCatalogs`, `VCFsToZipFile`, etc.), then use `as.catalog`.

**Creating catalogs from variant call files (VCF files)**

- `VCFsToCatalogs` creates 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and ID (small insertions and deletions) catalog from the VCFs.

**Plotting catalogs**

- `PlotCatalog` function plots mutational spectra for **one** sample or plot **one** mutational signature.
- `PlotCatalogToPdf` function plots catalogs of mutational spectra or of mutational signatures to a PDF file.

**Wrapper function to create catalogs from VCFs and plot the catalogs to PDF files**

- `VCFsToCatalogsAndPlotToPdf` creates all types of SBS, DBS and ID catalogs from VCFs and plots the catalogs.

**Wrapper function to create a zip file which contains catalogs and plot PDFs from VCF files**

- `VCFsToZipFile` creates a zip file which contains SBS, DBS and ID catalogs and plot PDFs from VCF files.

**The** `ref.genome` **(reference genome) argument**

Many functions take the argument `ref.genome`.

To create a mutational spectrum catalog from a VCF file, "ICAMS" needs the reference genome sequence that matches the VCF file. The `ref.genome` argument provides this.

`ref.genome` must be one of

1. A variable from the Bioconductor `BSgenome` package that contains a particular reference genome, for example BSgenome.Hsapiens.1000genomes.hs37d5.
2. The strings `"hg38"` or `"GRCh38"`, which specify BSgenome.Hsapiens.UCSC.hg38.
3. The strings `"hg19"` or `"GRCh37"`, which specify BSgenome.Hsapiens.1000genomes.hs37d5.
4. The strings `"mm10"` or `"GRCm38"`, which specify BSgenome.Mmusculus.UCSC.mm10.

All needed reference genomes must be installed separately by the user. Further instructions are at https://bioconductor.org/packages/release/bioc/html/BSgenome.html.

Use of "ICAMS" with reference genomes other than the 2 human genomes and 1 mouse genome specified above is restricted to `catalog.type` of `counts` or `counts.signature` unless the user also creates the necessary abundance vectors. See `all.abundance`.

Use `available.genomes()` to get the list of available genomes.

**Writing catalogs to files**

- `WriteCatalog` function writes a catalog to a file.

**Reading catalogs**

- `ReadCatalog` function reads a file that contains a catalog in standardized format.

**Transforming catalogs**

`TransformCatalog` function transforms catalogs of mutational spectra or signatures to account for differing abundances of the source sequence of the mutations in the genome.

For example, mutations from ACG are much rarer in the human genome than mutations from ACC simply because CG dinucleotides are rare in the genome. Consequently, there are two possible representations of mutational spectra or signatures. One representation is based on mutation counts as observed in a given genome or exome, and this approach is widely used, as, for example, at https://cancer.sanger.ac.uk/signatures/, which presents signatures based on observed mutation counts in the human genome. We call these "counts-based spectra" or "counts-based signatures".

Alternatively, mutational spectra or signatures can be represented as mutations per source sequence, for example the number of ACT > AGT mutations occurring at all ACT 3-mers in a genome. We call these "density-based spectra" or "density-based signatures".

This function can also transform spectra based on observed genome-wide counts to "density"-based catalogs. In density-based catalogs mutations are expressed as mutations per source sequences. For example, a density-based catalog represents the proportion of ACCs mutated to ATCs, the proportion of ACGs mutated to ATGs, etc. This is different from counts-based mutational spectra catalogs, which contain the number of ACC > ATC mutations, the number of ACG > ATG mutations, etc.

This function can also transform observed-count based spectra or signatures from genome to exome based counts, or between different species (since the abundances of source sequences vary between genome and exome and between species).

### Collapsing catalogs

CollapseCatalog function

1. Takes a mutational spectrum or signature catalog that is based on a fined-grained set of features (for example, single-nucleotide substitutions in the context of the preceding and following 2 bases).

2. Collapses it to a catalog based on a coarser-grained set of features (for example, single-nucleotide substitutions in the context of the immediately preceding and following bases).

### Data

1. CatalogRowOrder Standard order of rownames in a catalog. The rownames encode the type of each mutation. For example, for SBS96 catalogs, the rowname AGAT represents a mutation from AGA > ATA.

2. TranscriptRanges Transcript ranges and strand information for a particular reference genome.

3. all.abundance The counts of different source sequences for mutations.

4. GeneExpressionData Example gene expression data from two cell lines.

---

MutectVCFFilesToCatalog

**[Deprecated, use VCFsToCatalogs(variant.caller = "mutect") instead]** *Create SBS, DBS and Indel catalogs from Mutect VCF files*

---

### Description

**[Deprecated, use VCFsToCatalogs(variant.caller = "mutect") instead]** Create 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and Indel catalog from the Mutect VCFs specified by files

### Usage

```
MutectVCFFilesToCatalog(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the Mutect VCF files. |
| `ref.genome` | A `ref.genome` argument as described in [ICAMS](). |
| `trans.ranges` | Optional. If `ref.genome` specifies one of the [BSgenome]() object |

      1. `BSgenome.Hsapiens.1000genomes.hs37d5`

      2. `BSgenome.Hsapiens.UCSC.hg38`

      3. `BSgenome.Mmusculus.UCSC.mm10`

      then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to [TranscriptRanges]() for more details. If `is.null(trans.ranges)` do not add transcript range information.

| | |
|---|---|
| `region` | A character string designating a genomic region; see [as.catalog]() and [ICAMS](). |
| `names.of.VCFs` | |

      Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

`tumor.col.names`

      Optional. Vector of column names or column indices in VCFs which contain the tumor sample information. The order of elements in `tumor.col.names` should match the order of VCFs specified in `files`. If `tumor.col.names` is equal to `NA`(default), this function will use the 10th column in all the VCFs to calculate VAFs. See [GetMutectVAF]() for more details.

`flag.mismatches`

      Deprecated. If there are ID variants whose `REF` do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See [AnnotateIDVCF]() for more details.

`return.annotated.vcfs`

      Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

`suppress.discarded.variants.warnings`

      Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls [VCFsToSBSCatalogs](), [VCFsToDBSCatalogs]() and [VCFsToIDCatalogs]()

## Value

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).
- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).
- `catID`: Matrix of ID (small insertions and deletions) catalog.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.
- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of elements:
  - `SBS`: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
  - `DBS`: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.
  - `ID`: ID VCF annotated by `AnnotateIDVCF` with one new column `ID.class` showing the mutation class for each ID variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

### ID classification

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. `CACACACA`, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

### Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

### Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

### Examples

```
## Not run:
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <- MutectVCFFilesToCatalog(file, ref.genome = "hg19",
                                      trans.ranges = trans.ranges.GRCh37,
                                      region = "genome")}

## End(Not run)
```

---

MutectVCFFilesToCatalogAndPlotToPdf

> **[Deprecated, use VCFsToCatalogsAndPlotToPdf(variant.caller = "mutect") instead]** *Create SBS, DBS and Indel catalogs from Mutect VCF files and plot them to PDF*

---

### Description

**[Deprecated, use VCFsToCatalogsAndPlotToPdf(variant.caller = "mutect") instead]** Create 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and Indel catalog from the Mutect VCFs specified by `files` and plot them to PDF

### Usage

```
MutectVCFFilesToCatalogAndPlotToPdf(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  output.file = "",
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

### Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the Mutect VCF files. |
| `ref.genome` | A `ref.genome` argument as described in [ICAMS](#). |
| `trans.ranges` | Optional. If `ref.genome` specifies one of the [BSgenome](#) object |

     1. BSgenome.Hsapiens.1000genomes.hs37d5
     2. BSgenome.Hsapiens.UCSC.hg38
     3. BSgenome.Mmusculus.UCSC.mm10

     then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to [TranscriptRanges](#) for more details. If `is.null(trans.ranges)` do not add transcript range information.

| | |
|---|---|
| `region` | A character string designating a genomic region; see [as.catalog](#) and [ICAMS](#). |
| `names.of.VCFs` | |

     Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

`tumor.col.names`

     Optional. Vector of column names or column indices in VCFs which contain the tumor sample information. The order of elements in `tumor.col.names`

should match the order of VCFs specified in `files`. If `tumor.col.names` is equal to `NA`(default), this function will use the 10th column in all the VCFs to calculate VAFs. See `GetMutectVAF` for more details.

output.file      Optional. The base name of the PDF files to be produced; multiple files will be generated, each ending in $x$.`pdf`, where $x$ indicates the type of catalog plotted in the file.

flag.mismatches

Deprecated. If there are ID variants whose `REF` do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See `AnnotateIDVCF` for more details.

return.annotated.vcfs

Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls `MutectVCFFilesToCatalog` and `PlotCatalogToPdf`

## Value

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).
- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).
- `catID`: Matrix of ID (small insertions and deletions) catalog.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.
- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of elements:
  - `SBS`: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
  - `DBS`: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.
  - `ID`: ID VCF annotated by `AnnotateIDVCF` with one new column `ID.class` showing the mutation class for each ID variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

## Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

## Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

## ID classification

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_`
`indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. `CACACACA`, see
`FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology
(see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

## Examples

```
## Not run:
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    MutectVCFFilesToCatalogAndPlotToPdf(file, ref.genome = "hg19",
                                        trans.ranges = trans.ranges.GRCh37,
                                        region = "genome",
                                        output.file =
                                        file.path(tempdir(), "Mutect"))}

## End(Not run)
```

---

MutectVCFFilesToZipFile

> **[Deprecated, use VCFsToZipFile(variant.caller = "mutect") instead]** *Create a zip file which contains catalogs and plot PDFs from Mutect VCF files*

---

## Description

**[Deprecated, use VCFsToZipFile(variant.caller = "mutect") instead]** Create 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and Indel catalog from the Mutect VCFs specified by `dir`, save the catalogs as CSV files, plot them to PDF and generate a zip archive of all the output files.

## Usage

```
MutectVCFFilesToZipFile(
  dir,
  zipfile,
  ref.genome,
```

```
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  base.filename = "",
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

dir
: Pathname of the directory which contains **only** the Mutect VCF files. Each Mutect VCF **must** have a file extension ".vcf" (case insensitive) and share the **same** `ref.genome` and `region`.

zipfile
: Pathname of the zip file to be created.

ref.genome
: A `ref.genome` argument as described in [ICAMS].

trans.ranges
: Optional. If `ref.genome` specifies one of the [BSgenome] object

    1. BSgenome.Hsapiens.1000genomes.hs37d5
    2. BSgenome.Hsapiens.UCSC.hg38
    3. BSgenome.Mmusculus.UCSC.mm10

    then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to [TranscriptRanges] for more details. If `is.null(trans.ranges)` do not add transcript range information.

region
: A character string designating a genomic region; see [as.catalog] and [ICAMS].

names.of.VCFs
: Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCFs listed in `dir`. If NULL(default), this function will remove all of the path up to and including the last path separator (if any) in `dir` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

tumor.col.names
: Optional. Vector of column names or column indices in VCFs which contain the tumor sample information. The order of elements in `tumor.col.names` should match the order of VCFs listed in `dir`. If `tumor.col.names` is equal to NA(default), this function will use the 10th column in all the VCFs to calculate VAFs. See [GetMutectVAF] for more details.

base.filename
: Optional. The base name of the CSV and PDF files to be produced; multiple files will be generated, each ending in $x$.csv or $x$.pdf, where $x$ indicates the type of catalog.

flag.mismatches
: Deprecated. If there are ID variants whose REF do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See [AnnotateIDVCF] for more details.

return.annotated.vcfs
: Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

> Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

**Details**

This function calls `MutectVCFFilesToCatalog`, `PlotCatalogToPdf`, `WriteCatalog` and `zip::zipr`.

**Value**

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).
- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).
- `catID`: Matrix of ID (small insertions and deletions) catalog.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.
- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of elements:
  - `SBS`: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
  - `DBS`: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.
  - `ID`: ID VCF annotated by `AnnotateIDVCF` with one new column `ID.class` showing the mutation class for each ID variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

**ID classification**

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

**Note**

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

## Comments

To add or change attributes of the catalog, you can use function attr.
For example, attr(catalog,"abundance") <-custom.abundance.

## Examples

```
## Not run:
dir <- c(system.file("extdata/Mutect-vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    MutectVCFFilesToZipFile(dir,
                            zipfile = file.path(tempdir(), "test.zip"),
                            ref.genome = "hg19",
                            trans.ranges = trans.ranges.GRCh37,
                            region = "genome",
                            base.filename = "Mutect")
  unlink(file.path(tempdir(), "test.zip"))}

## End(Not run)
```

---

PlotCatalog                 *Plot* **one** *spectrum or signature*

---

## Description

Plot the spectrum of **one** sample or plot **one** signature. The type of graph is based on attribute("catalog.type")
of the input catalog. You can first use TransformCatalog to get different types of catalog and
then do the plotting.

## Usage

```
PlotCatalog(
  catalog,
  plot.SBS12 = NULL,
  cex = NULL,
  grid = NULL,
  upper = NULL,
  xlabels = NULL,
  ylabels = NULL,
  ylim = NULL
)
```

## Arguments

catalog          A catalog as defined in ICAMS with attributes added. See as.catalog for
                 more details. catalog can also be a numeric matrix, numeric data.frame,
                 or a vector denoting the mutation **counts**, but **must** be in the correct row or-
                 der used in ICAMS. See CatalogRowOrder for more details. If catalog
                 is a vector, it will be converted to a 1-column matrix with rownames taken
                 from the element names of the vector and with column name "Unknown".

| | |
|---|---|
| plot.SBS12 | Only meaningful for class SBS192Catalog; if TRUE, generate an abbreviated plot of only SBS without context, i.e. C>A, C>G, C>T, T>A, T>C, T>G each on transcribed and untranscribed strands, rather than SBS in trinucleotide context, e.g. ACA > AAA, ACA > AGA, ..., TCT > TAT, ... There are 12 bars in the graph. |
| cex | Has the usual meaning. Taken from par("cex") by default. Only implemented for SBS96Catalog, SBS192Catalog and DBS144Catalog. |
| grid | A logical value indicating whether to draw grid lines. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. |
| upper | A logical value indicating whether to draw horizontal lines and the names of major mutation class on top of graph. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. |
| xlabels | A logical value indicating whether to draw x axis labels. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. If FALSE then plot x axis tick marks for SBS96Catalog; set par(tck = 0) to suppress. |
| ylabels | A logical value indicating whether to draw y axis labels. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. |
| ylim | Has the usual meaning. Only implemented for SBS96Catalog and IndelCatalog. |

## Value

An **invisible** list whose first element is a logic value indicating whether the plot is successful. For SBS96Catalog, SBS192Catalog, DBS78Catalog, DBS144Catalog and IndelCatalog, the list will have a second element, which is a numeric vector giving the coordinates of all the bar midpoints drawn, useful for adding to the graph. For **SBS192Catalog** with "counts" catalog.type and non-NULL abundance and plot.SBS12 = TRUE, the list will have an additional element which is a list containing the strand bias statistics.

## Comments

For **SBS192Catalog** with "counts" catalog.type and non-NULL abundance and plot.SBS12 = TRUE, the strand bias statistics are Benjamini-Hochberg q-values based on two-sided binomial tests of the mutation counts on the transcribed and untranscribed strands relative to the actual abundances of C and T on the transcribed strand. On the SBS12 plot, asterisks indicate q-values as follows *, $Q < 0.05$; **, $Q < 0.01$; ***, $Q < 0.001$.

## Note

The sizes of repeats involved in deletions range from 0 to 5+ in the mutational-spectra and signature catalog rownames, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

## Examples

```
file <- system.file("extdata",
                    "strelka.regress.cat.sbs.96.csv",
                    package = "ICAMS")
catSBS96 <- ReadCatalog(file)
colnames(catSBS96) <- "sample"
PlotCatalog(catSBS96)
```

| PlotCatalogToPdf | *Plot catalog to a PDF file* |
|---|---|

### Description

Plot catalog to a PDF file. The type of graph is based on attribute("catalog.type") of the input catalog. You can first use TransformCatalog to get different types of catalog and then do the plotting.

### Usage

```
PlotCatalogToPdf(
  catalog,
  file,
  plot.SBS12 = NULL,
  cex = NULL,
  grid = NULL,
  upper = NULL,
  xlabels = NULL,
  ylabels = NULL,
  ylim = NULL
)
```

### Arguments

| | |
|---|---|
| catalog | A catalog as defined in ICAMS with attributes added. See as.catalog for more details. catalog can also be a numeric matrix, numeric data.frame, or a vector denoting the mutation **counts**, but **must** be in the correct row order used in ICAMS. See CatalogRowOrder for more details. If catalog is a vector, it will be converted to a 1-column matrix with rownames taken from the element names of the vector and with column name "Unknown". |
| file | The name of the PDF file to be produced. |
| plot.SBS12 | Only meaningful for class SBS192Catalog; if TRUE, generate an abbreviated plot of only SBS without context, i.e. C>A, C>G, C>T, T>A, T>C, T>G each on transcribed and untranscribed strands, rather than SBS in trinucleotide context, e.g. ACA > AAA, ACA > AGA, ..., TCT > TAT, ... There are 12 bars in the graph. |
| cex | Has the usual meaning. A default value has been used by the program internally. Only implemented for SBS96Catalog, SBS192Catalog and DBS144Catalog. |
| grid | A logical value indicating whether to draw grid lines. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. |
| upper | A logical value indicating whether to draw horizontal lines and the names of major mutation class on top of graph. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. |
| xlabels | A logical value indicating whether to draw x axis labels. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. If FALSE then plot x axis tick marks for SBS96Catalog; set par(tck = 0) to suppress. |
| ylabels | A logical value indicating whether to draw y axis labels. Only implemented for SBS96Catalog, DBS78Catalog, IndelCatalog. |
| ylim | Has the usual meaning. Only implemented for SBS96Catalog and IndelCatalog. |

**Value**

An **invisible** list whose first element is a logic value indicating whether the plot is successful. For **SBS192Catalog** with "counts" catalog.type and non-null abundance and `plot.SBS12 = TRUE`, the list will have a second element which is a list containing the strand bias statistics.

**Comments**

For **SBS192Catalog** with "counts" catalog.type and non-NULL abundance and `plot.SBS12 = TRUE`, the strand bias statistics are Benjamini-Hochberg q-values based on two-sided binomial tests of the mutation counts on the transcribed and untranscribed strands relative to the actual abundances of C and T on the transcribed strand. On the SBS12 plot, asterisks indicate q-values as follows \*, $Q < 0.05$; \*\*, $Q < 0.01$; \*\*\*, $Q < 0.001$.

**Note**

The sizes of repeats involved in deletions range from 0 to 5+ in the mutational-spectra and signature catalog rownames, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

**Examples**

```
file <- system.file("extdata",
                    "strelka.regress.cat.sbs.96.csv",
                    package = "ICAMS")
catSBS96 <- ReadCatalog(file)
colnames(catSBS96) <- "sample"
PlotCatalogToPdf(catSBS96, file = file.path(tempdir(), "test.pdf"))
```

---

PlotTransBiasGeneExp

*Plot transcription strand bias with respect to gene expression values*

---

**Description**

Plot transcription strand bias with respect to gene expression values

**Usage**

```
PlotTransBiasGeneExp(
  annotated.SBS.vcf,
  expression.data,
  Ensembl.gene.ID.col,
  expression.value.col,
  num.of.bins,
  plot.type,
  damaged.base = NULL,
  ymax = NULL
)
```

## Arguments

annotated.SBS.vcf

> An SBS VCF annotated by `AnnotateSBSVCF`. It **must** have transcript range information added.

expression.data

> A `data.table` which contains the expression values of genes. See `GeneExpressionData` for more details.

Ensembl.gene.ID.col

> Name of column which has the Ensembl gene ID information in `expression.data`.

expression.value.col

> Name of column which has the gene expression values in `expression.data`.

num.of.bins    The number of bins that will be plotted on the graph.

plot.type      A character string indicating one mutation type to be plotted. It should be one of "C>A", "C>G", "C>T", "T>A", "T>C", "T>G".

damaged.base   One of `NULL`, `"purine"` or `"pyrimidine"`. This function allocates approximately equal numbers of mutations from `damaged.base` into each of `num.of.bins` bin by expression level. E.g. if `damaged.base` is `"purine"`, then mutations from A and G will be allocated in approximately equal numbers to each expression-level bin. The rationale for the name `damaged.base` is that the direction of strand bias is a result of whether the damage occurs on a purine or pyrimidine. If `NULL`, the function attempts to infer the `damaged.base` based on mutation counts.

ymax           Limit for the y axis. If not specified, it defaults to NULL and the y axis limit equals 1.5 times of the maximum mutation counts in a specific mutation type.

## Value

A list whose first element is a logic value indicating whether the plot is successful. The second element is a named numeric vector containing the p-values printed on the plot.

## Note

The p-values are calculated by logistic regression using function `glm`. The dependent variable is labeled "1" and "0" if the mutation from `annotated.SBS.vcf` falls onto the untranscribed and transcribed strand respectively. The independent variable is the binary logarithm of the gene expression value from `expression.data` plus one, i.e. $log_2(x + 1)$ where $x$ stands for gene expression value.

## Examples

```
file <- c(system.file("extdata/Strelka-SBS-vcf/",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitVCFs(file, variant.caller = "strelka")
SBS.vcf <- list.of.vcfs$SBS[[1]]
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.SBS.vcf <- AnnotateSBSVCF(SBS.vcf, ref.genome = "hg19",
                                      trans.ranges = trans.ranges.GRCh37)
  PlotTransBiasGeneExp(annotated.SBS.vcf = annotated.SBS.vcf,
                       expression.data = gene.expression.data.HepG2,
                       Ensembl.gene.ID.col = "Ensembl.gene.ID",
                       expression.value.col = "TPM",
```

```
                              num.of.bins = 4, plot.type = "C>A")
  }
```

---

```
PlotTransBiasGeneExpToPdf
                    Plot transcription strand bias with respect to gene expression values
                    to a PDF file
```

---

### Description

Plot transcription strand bias with respect to gene expression values to a PDF file

### Usage

```
PlotTransBiasGeneExpToPdf(
  annotated.SBS.vcf,
  file,
  expression.data,
  Ensembl.gene.ID.col,
  expression.value.col,
  num.of.bins,
  plot.type = c("C>A", "C>G", "C>T", "T>A", "T>C", "T>G"),
  damaged.base = NULL
)
```

### Arguments

annotated.SBS.vcf
                 An SBS VCF annotated by `AnnotateSBSVCF`. It **must** have transcript range
                 information added.

file             The name of output file.

expression.data
                 A `data.table` which contains the expression values of genes.
                 See `GeneExpressionData` for more details.

Ensembl.gene.ID.col
                 Name of column which has the Ensembl gene ID information in `expression.data`.

expression.value.col
                 Name of column which has the gene expression values in `expression.data`.

num.of.bins      The number of bins that will be plotted on the graph.

plot.type        A vector of character indicating types to be plotted. It can be one or more types
                 from "C>A", "C>G", "C>T", "T>A", "T>C", "T>G". The default is to print all
                 the six mutation types.

damaged.base     One of `NULL`, `"purine"` or `"pyrimidine"`. This function allocates ap-
                 proximately equal numbers of mutations from `damaged.base` into each of
                 `num.of.bins` bin by expression level. E.g. if `damaged.base` is `"purine"`,
                 then mutations from A and G will be allocated in approximately equal numbers
                 to each expression-level bin. The rationale for the name `damaged.base` is
                 that the direction of strand bias is a result of whether the damage occurs on a
                 purine or pyrimidine. If `NULL`, the function attempts to infer the `damaged.base`
                 based on mutation counts.

## Value

A list whose first element is a logic value indicating whether the plot is successful. The second element is a named numeric vector containing the p-values printed on the plot.

## Note

The p-values are calculated by logistic regression using function `glm`. The dependent variable is labeled "1" and "0" if the mutation from `annotated.SBS.vcf` falls onto the untranscribed and transcribed strand respectively. The independent variable is the binary logarithm of the gene expression value from `expression.data` plus one, i.e. $log_2(x + 1)$ where $x$ stands for gene expression value.

## Examples

```
file <- c(system.file("extdata/Strelka-SBS-vcf/",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitVCFs(file, variant.caller = "strelka")
SBS.vcf <- list.of.vcfs$SBS[[1]]
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.SBS.vcf <- AnnotateSBSVCF(SBS.vcf, ref.genome = "hg19",
                                      trans.ranges = trans.ranges.GRCh37)
  PlotTransBiasGeneExpToPdf(annotated.SBS.vcf = annotated.SBS.vcf,
                            expression.data = gene.expression.data.HepG2,
                            Ensembl.gene.ID.col = "Ensembl.gene.ID",
                            expression.value.col = "TPM",
                            num.of.bins = 4,
                            plot.type = c("C>A","C>G","C>T","T>A","T>C"),
                            file = file.path(tempdir(), "test.pdf"))
}
```

---

```
ReadAndSplitMutectVCFs
```
**[Deprecated, use ReadAndSplitVCFs(variant.caller = "mutect") instead]** *Read and split Mutect VCF files*

---

## Description

**[Deprecated, use ReadAndSplitVCFs(variant.caller = "mutect") instead]** Read and split Mutect VCF files

## Usage

```
ReadAndSplitMutectVCFs(
  files,
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  suppress.discarded.variants.warnings = TRUE
)
```

### Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the Mutect VCF files. |
| `names.of.VCFs` | |
| | Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files. |
| `tumor.col.names` | |
| | Optional. Vector of column names or column indices in VCFs which contain the tumor sample information. The order of elements in `tumor.col.names` should match the order of VCFs specified in `files`. If `tumor.col.names` is equal to `NA`(default), this function will use the 10th column in all the VCFs to calculate VAFs. See `GetMutectVAF` for more details. |
| `suppress.discarded.variants.warnings` | |
| | Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE. |

### Value

A list containing the following objects:

- `SBS`: List of VCFs with only single base substitutions.
- `DBS`: List of VCFs with only doublet base substitutions as called by Mutect.
- `ID`: List of VCFs with only small insertions and deletions.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

### See Also

`MutectVCFFilesToCatalog`

### Examples

```
## Not run:
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitMutectVCFs(file)

## End(Not run)
```

---

`ReadAndSplitStrelkaSBSVCFs`

> **[Deprecated, use ReadAndSplitVCFs(variant.caller = "strelka") instead]** *Read and split Strelka SBS VCF files*

---

### Description

**[Deprecated, use ReadAndSplitVCFs(variant.caller = "strelka") instead]** The function will find and merge adjacent SBS pairs into DBS if their VAFs are very similar. The default threshold value for VAF is 0.02.

## Usage

```
ReadAndSplitStrelkaSBSVCFs(
  files,
  names.of.VCFs = NULL,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

files            Character vector of file paths to the Strelka SBS VCF files.

names.of.VCFs

Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Value

A list of elements as follows:

- `SBS.vcfs`: List of data.frames of pure SBS mutations – no DBS or 3+BS mutations.

- `DBS.vcfs`: List of data.frames of pure DBS mutations – no SBS or 3+BS mutations.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

## See Also

[StrelkaSBSVCFFilesToCatalog](StrelkaSBSVCFFilesToCatalog)

## Examples

```
## Not run:
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitStrelkaSBSVCFs(file)

## End(Not run)
```

---

ReadAndSplitVCFs            *Read and split VCF files*

---

## Description

Read and split VCF files

## Usage

```
ReadAndSplitVCFs(
  files,
  variant.caller = "unknown",
  num.of.cores = 1,
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  filter.status = "PASS",
  get.vaf.function = NULL,
  ...,
  max.vaf.diff = 0.02,
  suppress.discarded.variants.warnings = TRUE,
  always.merge.SBS = FALSE
)
```

## Arguments

`files`           Character vector of file paths to the VCF files.

`variant.caller`

Name of the variant caller that produces the VCF, can be either `"strelka"`,
`"mutect"`, `"freebayes"` or `"unknown"`. This information is needed to
calculate the VAFs (variant allele frequencies). If variant caller is `"unknown"`(default)
and `get.vaf.function` is NULL, then VAF and read depth will be NAs. If
variant caller is `"mutect"`, do **not** merge SBSs into DBS.

`num.of.cores`    The number of cores to use. Not available on Windows unless `num.of.cores`
= 1.

`names.of.VCFs`

Character vector of names of the VCF files. The order of names in `names.of.VCFs`
should match the order of VCF file paths in `files`. If NULL(default), this func-
tion will remove all of the path up to and including the last path separator (if any)
and file paths without extensions (and the leading dot) will be used as the names
of the VCF files.

`tumor.col.names`

Optional. Only applicable to **Mutect** VCFs. Vector of column names or column
indices in **Mutect** VCFs which contain the tumor sample information. The order
of elements in `tumor.col.names` should match the order of **Mutect** VCFs
specified in `files`. If `tumor.col.names` is equal to `NA`(default), this func-
tion will use the 10th column in all the **Mutect** VCFs to calculate VAFs. See
`GetMutectVAF` for more details.

`filter.status`

The character string in column `FILTER` of the VCF that indicates that a variant
has passed all the variant caller's filters. Variants (lines in the VCF) for which
the value in column `FILTER` does not equal `filter.status` are silently
excluded from the output. If `NULL`, all variants are retained. In almost all cases,
the default value of `"PASS"` is what the user would want.

`get.vaf.function`

Optional. Only applicable when `variant.caller` is **"unknown"**. Function
to calculate VAF(variant allele frequency) and read depth information from orig-
inal VCF. See `GetMutectVAF` as an example. If NULL(default) and `variant.caller`
is "unknown", then VAF and read depth will be NAs.

`...`             Optional arguments to `get.vaf.function`.

max.vaf.diff **Not** applicable if `variant.caller = "mutect"`. The maximum difference of VAF, default value is 0.02. If the absolute difference of VAFs for adjacent SBSs is bigger than `max.vaf.diff`, then these adjacent SBSs are likely to be "merely" asynchronous single base mutations, opposed to a simultaneous doublet mutation or variants involving more than two consecutive bases. Use negative value (e.g. -1) to suppress merging adjacent SBSs to DBS.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

always.merge.SBS

If `TRUE` merge adjacent SBSs as DBSs regardless of VAFs and regardless of the value of `max.vaf.diff` and regardless of the value of `get.vaf.function`. It is an error to set this to `TRUE` when `variant.caller = "mutect"`.

## Value

A list containing the following objects:

- `SBS`: List of VCFs with only single base substitutions.
- `DBS`: List of VCFs with only doublet base substitutions.
- `ID`: List of VCFs with only small insertions and deletions.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

## See Also

[VCFsToCatalogs](#)

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadAndSplitVCFs(file, variant.caller = "mutect")
```

---

ReadCatalog                 *Read catalog*

---

## Description

Read a catalog in standardized format from path.

## Usage

```
ReadCatalog(
  file,
  ref.genome = NULL,
  region = "unknown",
  catalog.type = "counts",
  strict = NULL,
  stop.on.error = TRUE
)
```

## Arguments

| | |
|---|---|
| `file` | Path to a catalog on disk in a standardized format. The recognized formats are: |

- ICAMS formatted SBS96, SBS192, SBS1536, DBS78, DBS136, DBS144, ID (see `CatalogRowOrder`).
- SigProfiler-formatted SBS96, DBS78 and ID83 catalogs; see https://github.com/AlexandrovLab/SigProfilerExtractor.
- COSMIC-formatted SBS96, SBS192 (a.k.a. TSB192), DBS78, ID83 catalogs; see https://cancer.sanger.ac.uk/signatures/.

| | |
|---|---|
| `ref.genome` | A `ref.genome` argument as described in `ICAMS`. |
| `region` | region A character string designating a genomic region; see `as.catalog` and `ICAMS`. |
| `catalog.type` | One of "counts", "density", "counts.signature", "density.signature". |
| `strict` | Ignored and deprecated. |
| `stop.on.error` | |
| | If TRUE, call `stop` on error; otherwise return a 1-column matrix of NA's with the attribute "error" containing error information. The number of rows may not be the correct number for the expected catalog type. |

## Details

See also `WriteCatalog`

## Value

A catalog as an S3 object; see `as.catalog`.

## Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

## Note

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

## Examples

```
file <- system.file("extdata",
                     "strelka.regress.cat.sbs.96.csv",
                     package = "ICAMS")
catSBS96 <- ReadCatalog(file)
```

ReadStrelkaIDVCFs **[Deprecated, use ReadAndSplitVCFs(variant.caller = "strelka") instead]** *Read Strelka ID (small insertions and deletions) VCF files*

## Description

**[Deprecated, use ReadAndSplitVCFs(variant.caller = "strelka") instead]** Read Strelka ID (small insertions and deletions) VCF files

## Usage

```
ReadStrelkaIDVCFs(files, names.of.VCFs = NULL)
```

## Arguments

files           Character vector of file paths to the VCF files.

names.of.VCFs

Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

## Value

A list of data frames containing data lines of the VCF files.

## Note

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

## See Also

[StrelkaIDVCFFilesToCatalog](StrelkaIDVCFFilesToCatalog)

## Examples

```
## Not run:
file <- c(system.file("extdata/Strelka-ID-vcf",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadStrelkaIDVCFs(file)

## End(Not run)
```

---

## Description

Read VCF files

## Usage

```
ReadVCFs(
  files,
  variant.caller = "unknown",
  num.of.cores = 1,
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  filter.status = "PASS",
  get.vaf.function = NULL,
  ...
)
```

## Arguments

files            Character vector of file paths to the VCF files.

variant.caller

> Name of the variant caller that produces the VCF, can be either `"strelka"`,
> `"mutect"`, `"freebayes"` or `"unknown"`. This information is needed to
> calculate the VAFs (variant allele frequencies). If variant caller is `"unknown"`(default)
> and `get.vaf.function` is NULL, then VAF and read depth will be NAs. If
> variant caller is `"mutect"`, do **not** merge SBSs into DBS.

num.of.cores     The number of cores to use. Not available on Windows unless `num.of.cores`
                 `= 1`.

names.of.VCFs

> Character vector of names of the VCF files. The order of names in `names.of.VCFs`
> should match the order of VCF file paths in `files`. If NULL(default), this func-
> tion will remove all of the path up to and including the last path separator (if any)
> and file paths without extensions (and the leading dot) will be used as the names
> of the VCF files.

tumor.col.names

> Optional. Only applicable to **Mutect** VCFs. Vector of column names or column
> indices in **Mutect** VCFs which contain the tumor sample information. The order
> of elements in `tumor.col.names` should match the order of **Mutect** VCFs
> specified in `files`. If `tumor.col.names` is equal to NA(default), this func-
> tion will use the 10th column in all the **Mutect** VCFs to calculate VAFs. See
> `GetMutectVAF` for more details.

filter.status

> The character string in column `FILTER` of the VCF that indicates that a variant
> has passed all the variant caller's filters. Variants (lines in the VCF) for which
> the value in column `FILTER` does not equal `filter.status` are silently
> excluded from the output. If NULL, all variants are retained. In almost all cases,
> the default value of `"PASS"` is what the user would want.

```
get.vaf.function
```
> Optional. Only applicable when `variant.caller` is **"unknown"**. Function to calculate VAF(variant allele frequency) and read depth information from original VCF. See `GetMutectVAF` as an example. If `NULL`(default) and `variant.caller` is "unknown", then VAF and read depth will be NAs.

```
...
```
> Optional arguments to `get.vaf.function`.

## Value

A list of data frames storing data lines of the VCF files with two additional columns added which contain the VAF(variant allele frequency) and read depth information.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadVCFs(file, variant.caller = "mutect")
```

---

revc                    *Reverse complement every string in* string.vec

---

## Description

Based on `reverseComplement`. Handles IUPAC ambiguity codes but not "u" (uracil).
(see <https://en.wikipedia.org/wiki/Nucleic_acid_notation>).

## Usage

```
revc(string.vec)
```

## Arguments

```
string.vec    A character vector.
```

## Value

A character vector with the reverse complement of every string in `string.vec`.

## Examples

```
revc("aTgc") # GCAT

# A vector and strings with ambiguity codes
revc(c("ATGC", "aTGc", "wnTCb")) # GCAT GCAT VGANW

## Not run:
revc("ACGU") # An error
## End(Not run)
```

---

SimpleReadVCF *Read a VCF file into a data frame with minimal processing.*

---

### Description

Read a VCF file into a data frame with minimal processing.

### Usage

```
SimpleReadVCF(file)
```

### Arguments

file          The name/path of the VCF file, or a complete URL.

### Details

Header lines beginning "##" are removed, and column "#CHROM" is renamed to "CHROM". Other column names are unchanged. Columns "#CHROM", "POS", "REF", and "ALT" must be in the input.

### Value

A data frame storing mutation records of a VCF file.

### Examples

```
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
df <- SimpleReadVCF(file)
```

---

SplitListOfVCFs *Split each VCF into SBS, DBS, and ID VCFs (plus VCF-like data frame with left-over rows)*

---

### Description

Split each VCF into SBS, DBS, and ID VCFs (plus VCF-like data frame with left-over rows)

### Usage

```
SplitListOfVCFs(
  list.of.vcfs,
  variant.caller,
  max.vaf.diff = 0.02,
  num.of.cores = 1,
  suppress.discarded.variants.warnings = TRUE,
  always.merge.SBS = FALSE
)
```

## Arguments

`list.of.vcfs` List of VCFs as in-memory data frames. The VCFs should have `VAF` and `read.depth` information added. See `ReadVCFs` for more details.

`variant.caller`

Name of the variant caller that produces the VCF, can be either `"strelka"`, `"mutect"`, `"freebayes"` or `"unknown"`. If variant caller is `"mutect"`, do **not** merge SBSs into DBS.

`max.vaf.diff` The maximum difference of VAF, default value is 0.02. If the absolute difference of VAFs for adjacent SBSs is bigger than `max.vaf.diff`, then these adjacent SBSs are likely to be "merely" asynchronous single base mutations, opposed to a simultaneous doublet mutation or variants involving more than two consecutive bases. Use negative value (e.g. -1) to suppress merging adjacent SBSs to DBS.

`num.of.cores` The number of cores to use. Not available on Windows unless `num.of.cores = 1`.

`suppress.discarded.variants.warnings`

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

`always.merge.SBS`

If `TRUE` merge adjacent SBSs as DBSs regardless of VAFs and regardless of the value of `max.vaf.diff` and regardless of the value of `get.vaf.function`. It is an error to set this to `TRUE` when `variant.caller = "mutect"`.

## Value

A list containing the following objects:

- `SBS`: List of VCFs with only single base substitutions.
- `DBS`: List of VCFs with only doublet base substitutions as called by Mutect.
- `ID`: List of VCFs with only small insertions and deletions.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.vcfs <- ReadVCFs(file, variant.caller = "mutect")
split.vcfs <- SplitListOfVCFs(list.of.vcfs, variant.caller = "mutect")
```

---

`StrelkaIDVCFFilesToCatalog`

**[Deprecated, use VCFsToCatalogs(variant.caller = "strelka") instead]** *Create ID (small insertions and deletions) catalog from Strelka ID VCF files*

---

## Description

**[Deprecated, use VCFsToCatalogs(variant.caller = "strelka") instead]** Create ID (small insertions and deletions) catalog from the Strelka ID VCFs specified by `files`

## Usage

```
StrelkaIDVCFFilesToCatalog(
  files,
  ref.genome,
  region = "unknown",
  names.of.VCFs = NULL,
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

`files`              Character vector of file paths to the Strelka ID VCF files.

`ref.genome`    A `ref.genome` argument as described in `ICAMS`.

`region`          A character string designating a genomic region; see `as.catalog` and `ICAMS`.

`names.of.VCFs`

Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If NULL(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

`flag.mismatches`

Deprecated. If there are ID variants whose `REF` do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See `AnnotateIDVCF` for more details.

`return.annotated.vcfs`

Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

`suppress.discarded.variants.warnings`

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls `VCFsToIDCatalogs`

## Value

A **list** of elements:

- `catalog`: The ID (small insertions and deletions) catalog with attributes added. See `as.catalog` for more details.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs = TRUE`. A list of data frames which contain the original VCF's ID mutation rows with three additional columns `seq.context.width`, `seq.context` and `ID.class` added. The category assignment of each ID mutation in VCF can be obtained from `ID.class` column.

**ID classification**

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

**Note**

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

**Examples**

```
## Not run:
file <- c(system.file("extdata/Strelka-ID-vcf",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catID <- StrelkaIDVCFFilesToCatalog(file, ref.genome = "hg19",
                                      region = "genome")}

## End(Not run)
```

---

StrelkaIDVCFFilesToCatalogAndPlotToPdf

*[Deprecated, use VCFsToCatalogsAndPlotToPdf(variant.caller = "strelka") instead] Create ID (small insertions and deletions) catalog from Strelka ID VCF files and plot them to PDF*

---

**Description**

**[Deprecated, use VCFsToCatalogsAndPlotToPdf(variant.caller = "strelka") instead]** Create ID (small insertions and deletions) catalog from the Strelka ID VCFs specified by `files` and plot them to PDF

**Usage**

```
StrelkaIDVCFFilesToCatalogAndPlotToPdf(
  files,
  ref.genome,
  region = "unknown",
  names.of.VCFs = NULL,
  output.file = "",
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the Strelka ID VCF files. |
| `ref.genome` | A `ref.genome` argument as described in `ICAMS`. |
| `region` | A character string designating a genomic region; see `as.catalog` and `ICAMS`. |
| `names.of.VCFs` | |
| | Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files. |
| `output.file` | Optional. The base name of the PDF file to be produced; the file is ending in `catID.pdf`. |
| `flag.mismatches` | |
| | Deprecated. If there are ID variants whose `REF` do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See `AnnotateIDVCF` for more details. |
| `return.annotated.vcfs` | |
| | Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE. |
| `suppress.discarded.variants.warnings` | |
| | Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE. |

## Details

This function calls `StrelkaIDVCFFilesToCatalog` and `PlotCatalogToPdf`

## Value

A **list** of elements:

- `catalog`: The ID (small insertions and deletions) catalog with attributes added. See `as.catalog` for more details.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.
- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of data frames which contain the original VCF's ID mutation rows with three additional columns `seq.context.width`, `seq.context` and `ID.class` added. The category assignment of each ID mutation in VCF can be obtained from `ID.class` column.

## ID classification

See https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. `CACACACA`, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

**Note**

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

**Examples**

```
## Not run:
file <- c(system.file("extdata/Strelka-ID-vcf",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catID <-
    StrelkaIDVCFFilesToCatalogAndPlotToPdf(file, ref.genome = "hg19",
                                           region = "genome",
                                           output.file =
                                           file.path(tempdir(), "StrelkaID"))}

## End(Not run)
```

---

StrelkaIDVCFFilesToZipFile

> **[Deprecated, use VCFsToZipFile(variant.caller = "strelka") instead]** *Create a zip file which contains ID (small insertions and deletions) catalog and plot PDF from Strelka ID VCF files*

---

**Description**

**[Deprecated, use VCFsToZipFile(variant.caller = "strelka") instead]** Create ID (small insertions and deletions) catalog from the Strelka ID VCFs specified by `dir`, save the catalog as CSV file, plot it to PDF and generate a zip archive of all the output files.

**Usage**

```
StrelkaIDVCFFilesToZipFile(
  dir,
  zipfile,
  ref.genome,
  region = "unknown",
  names.of.VCFs = NULL,
  base.filename = "",
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

**Arguments**

| | |
|---|---|
| `dir` | Pathname of the directory which contains **only** the Strelka ID VCF files. Each Strelka ID VCF **must** have a file extension ".vcf" (case insensitive) and share the **same** `ref.genome` and `region`. |
| `zipfile` | Pathname of the zip file to be created. |

ref.genome        A `ref.genome` argument as described in `ICAMS`.

region            A character string designating a genomic region; see `as.catalog` and `ICAMS`.

names.of.VCFs

        Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCFs listed in `dir`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `dir` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

base.filename

        Optional. The base name of the CSV and PDF file to be produced; the file is ending in `catID.csv` and `catID.pdf` respectively.

flag.mismatches

        Deprecated. If there are ID variants whose `REF` do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See `AnnotateIDVCF` for more details.

return.annotated.vcfs

        Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

        Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls `StrelkaIDVCFFilesToCatalog`, `PlotCatalogToPdf`, `WriteCatalog` and `zip::zipr`.

## Value

A **list** of elements:

- `catalog`: The ID (small insertions and deletions) catalog with attributes added. See `as.catalog` for more details.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of data frames which contain the original VCF's ID mutation rows with three additional columns `seq.context.width`, `seq.context` and `ID.class` added. The category assignment of each ID mutation in VCF can be obtained from `ID.class` column.

## ID classification

See https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_ indel_classification_2021_09_03.xlsx for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

**Note**

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

**Examples**

```
## Not run:
dir <- c(system.file("extdata/Strelka-ID-vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    StrelkaIDVCFFilesToZipFile(dir,
                               zipfile = file.path(tempdir(), "test.zip"),
                               ref.genome = "hg19",
                               region = "genome",
                               base.filename = "Strelka-ID")
  unlink(file.path(tempdir(), "test.zip"))}

## End(Not run)
```

---

StrelkaSBSVCFFilesToCatalog

**[Deprecated, use VCFsToCatalogs(variant.caller = "strelka") instead]** *Create SBS and DBS catalogs from Strelka SBS VCF files*

---

**Description**

**[Deprecated, use VCFsToCatalogs(variant.caller = "strelka") instead]** Create 3 SBS catalogs (96, 192, 1536) and 3 DBS catalogs (78, 136, 144) from the Strelka SBS VCFs specified by `files`. The function will find and merge adjacent SBS pairs into DBS if their VAFs are very similar. The default threshold value for VAF is 0.02.

**Usage**

```
StrelkaSBSVCFFilesToCatalog(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

**Arguments**

| | |
|---|---|
| files | Character vector of file paths to the Strelka SBS VCF files. |
| ref.genome | A `ref.genome` argument as described in ICAMS. |
| trans.ranges | Optional. If `ref.genome` specifies one of the BSgenome object |

1. BSgenome.Hsapiens.1000genomes.hs37d5
2. BSgenome.Hsapiens.UCSC.hg38

3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer trans.ranges automatically. Otherwise, user will need to provide the necessary trans.ranges. Please refer to TranscriptRanges for more details. If is.null(trans.ranges) do not add transcript range information.

region          A character string designating a genomic region; see as.catalog and ICAMS.

names.of.VCFs

Optional. Character vector of names of the VCF files. The order of names in names.of.VCFs should match the order of VCF file paths in files. If NULL(default), this function will remove all of the path up to and including the last path separator (if any) in files and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

return.annotated.vcfs

Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls VCFsToSBSCatalogs and VCFsToDBSCatalogs.

## Value

A list containing the following objects:

- catSBS96, catSBS192, catSBS1536: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).

- catDBS78, catDBS136, catDBS144: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).

- discarded.variants: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column discarded.reason for more details.

- annotated.vcfs: **Non-NULL only if** return.annotated.vcfs = TRUE. A list of elements:

    - SBS: SBS VCF annotated by AnnotateSBSVCF with three new columns SBS96.class, SBS192.class and SBS1536.class showing the mutation class for each SBS variant.
    - DBS: DBS VCF annotated by AnnotateDBSVCF with three new columns DBS78.class, DBS136.class and DBS144.class showing the mutation class for each DBS variant.

If trans.ranges is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See as.catalog for more details.

## Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions.

## Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

## Examples

```
## Not run:
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <- StrelkaSBSVCFilesToCatalog(file, ref.genome = "hg19",
                                         trans.ranges = trans.ranges.GRCh37,
                                         region = "genome")}

## End(Not run)
```

---

StrelkaSBSVCFFilesToCatalogAndPlotToPdf

> **[Deprecated, use VCFsToCatalogsAndPlotToPdf(variant.caller = "strelka") instead]** *Create SBS and DBS catalogs from Strelka SBS VCF files and plot them to PDF*

---

## Description

**[Deprecated, use VCFsToCatalogsAndPlotToPdf(variant.caller = "strelka") instead]** Create 3 SBS catalogs (96, 192, 1536) and 3 DBS catalogs (78, 136, 144) from the Strelka SBS VCFs specified by `files` and plot them to PDF. The function will find and merge adjacent SBS pairs into DBS if their VAFs are very similar. The default threshold value for VAF is 0.02.

## Usage

```
StrelkaSBSVCFFilesToCatalogAndPlotToPdf(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  output.file = "",
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the Strelka SBS VCF files. |
| `ref.genome` | A `ref.genome` argument as described in ICAMS. |
| `trans.ranges` | Optional. If `ref.genome` specifies one of the BSgenome object |

1. BSgenome.Hsapiens.1000genomes.hs37d5
2. BSgenome.Hsapiens.UCSC.hg38

3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to `TranscriptRanges` for more details. If `is.null(trans.ranges)` do not add transcript range information.

region     A character string designating a genomic region; see `as.catalog` and `ICAMS`.

names.of.VCFs

       Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

output.file  Optional. The base name of the PDF files to be produced; multiple files will be generated, each ending in $x$.pdf, where $x$ indicates the type of catalog plotted in the file.

return.annotated.vcfs

       Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

       Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls `StrelkaSBSVCFFilesToCatalog` and `PlotCatalogToPdf`

## Value

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).
- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.
- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of elements:
  - SBS: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
  - DBS: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

## Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions.

**Comments**

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

**Examples**

```
## Not run:
file <- c(system.file("extdata/Strelka-SBS-vcf",
                      "Strelka.SBS.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    StrelkaSBSVCFFilesToCatalogAndPlotToPdf(file, ref.genome = "hg19",
                                            trans.ranges = trans.ranges.GRCh37,
                                            region = "genome",
                                            output.file =
                                            file.path(tempdir(), "StrelkaSBS"))}

## End(Not run)
```

---

StrelkaSBSVCFFilesToZipFile

> **[Deprecated, use VCFsToZipFile(variant.caller = "strelka") in-
> stead]** *Create a zip file which contains catalogs and plot PDFs from
> Strelka SBS VCF files*

---

**Description**

**[Deprecated, use VCFsToZipFile(variant.caller = "strelka") instead]** Create 3 SBS catalogs
(96, 192, 1536), 3 DBS catalogs (78, 136, 144) from the Strelka SBS VCFs specified by `dir`, save
the catalogs as CSV files, plot them to PDF and generate a zip archive of all the output files. The
function will find and merge adjacent SBS pairs into DBS if their VAFs are very similar. The default
threshold value for VAF is 0.02.

**Usage**

```
StrelkaSBSVCFFilesToZipFile(
  dir,
  zipfile,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  base.filename = "",
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

dir             Pathname of the directory which contains **only** the Strelka SBS VCF files. Each
                Strelka SBS VCF **must** have a file extension ".vcf" (case insensitive) and share
                the **same** `ref.genome` and `region`.

zipfile         Pathname of the zip file to be created.

ref.genome      A `ref.genome` argument as described in [ICAMS](#).

trans.ranges    Optional. If `ref.genome` specifies one of the [BSgenome](#) object

                1. `BSgenome.Hsapiens.1000genomes.hs37d5`
                2. `BSgenome.Hsapiens.UCSC.hg38`
                3. `BSgenome.Mmusculus.UCSC.mm10`

                then the function will infer `trans.ranges` automatically. Otherwise, user
                will need to provide the necessary `trans.ranges`. Please refer to [TranscriptRanges](#)
                for more details. If `is.null(trans.ranges)` do not add transcript range
                information.

region          A character string designating a genomic region; see [as.catalog](#) and [ICAMS](#).

names.of.VCFs
                Optional. Character vector of names of the VCF files. The order of names in
                `names.of.VCFs` should match the order of VCFs listed in `dir`. If NULL(default),
                this function will remove all of the path up to and including the last path sepa-
                rator (if any) in `dir` and file paths without extensions (and the leading dot) will
                be used as the names of the VCF files.

base.filename
                Optional. The base name of the CSV and PDF files to be produced; multiple
                files will be generated, each ending in $x$.csv or $x$.pdf, where $x$ indicates the
                type of catalog.

return.annotated.vcfs
                Logical. Whether to return the annotated VCFs with additional columns show-
                ing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings
                Logical. Whether to suppress warning messages showing information about the
                discarded variants. Default is TRUE.

## Details

This function calls [StrelkaSBSVCFFilesToCatalog](#), [PlotCatalogToPdf](#), [WriteCatalog](#)
and `zip::zipr`.

## Value

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192,
  and 1536).

- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136,
  and 144).

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from
  the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of
  elements:

- SBS: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
- DBS: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

### Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions.

### Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

### Examples

```
## Not run:
dir <- c(system.file("extdata/Strelka-SBS-vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    StrelkaSBSVCFFilesToZipFile(dir,
                                zipfile = file.path(tempdir(), "test.zip"),
                                ref.genome = "hg19",
                                trans.ranges = trans.ranges.GRCh37,
                                region = "genome",
                                base.filename = "Strelka-SBS")
  unlink(file.path(tempdir(), "test.zip"))}

## End(Not run)
```

---

```
SymmetricalContextsFor1BPIndel
```
*Get all the sequence contexts of the indels in a given 1 base-pair indel class from a VCF*

---

### Description

Get all the sequence contexts of the indels in a given 1 base-pair indel class from a VCF

### Usage

```
SymmetricalContextsFor1BPIndel(annotated.vcf, indel.class, flank.length = 5)
```

## Arguments

`annotated.vcf`

> An in-memory `data.frame` or similar table containing "VCF" (variant call format) data as created by `VCFsToIDCatalogs` with argument `return.annotated.vcfs = TRUE`.

`indel.class` A single character string that denotes a 1 base pair insertion or deletion, as taken from `ICAMS::catalog.row.order$ID`. Insertions or deletions into or from 5+ base-pair homopolymers are not supported.

`flank.length` The length of flanking bases around the position or homopolymer targeted by the indel.

## Value

A list of all sequence contexts for the specified `indel.class`.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
split.vcfs <- ReadAndSplitVCFs(file, variant.caller = "mutect")
ID.catalog <- VCFsToIDCatalogs(list.of.vcfs = split.vcfs$ID,
                               ref.genome = "hg19",
                               region = "genome",
                               return.annotated.vcfs = TRUE)
annotated.vcf <- ID.catalog$annotated.vcfs$Mutect.GRCh37.s1
extended.seq.contexts <-
  SymmetricalContextsFor1BPIndel(annotated.vcf = annotated.vcf,
                                 indel.class = "DEL:T:1:0")
```

---

`TranscriptRanges`  *Transcript ranges data*

---

## Description

Transcript ranges and strand information for a particular reference genome.

## Usage

```
trans.ranges.GRCh37

trans.ranges.GRCh38

trans.ranges.GRCm38
```

## Format

A `data.table` which contains transcript range and strand information for a particular reference genome. `colnames` are `chrom`, `start`, `end`, `strand`, `Ensembl.gene.ID`, `gene.symbol`. It uses one-based coordinates.

An object of class `data.table` (inherits from `data.frame`) with 19083 rows and 6 columns.

An object of class `data.table` (inherits from `data.frame`) with 19096 rows and 6 columns.

An object of class `data.table` (inherits from `data.frame`) with 20325 rows and 6 columns.

## Details

This information is needed to generate catalogs that depend on transcriptional strand information, for example catalogs of class `SBS192Catalog`.

`trans.ranges.GRCh37`: **Human** GRCh37.

`trans.ranges.GRCh38`: **Human** GRCh38.

`trans.ranges.GRCm38`: **Mouse** GRCm38.

For these two tables, only genes that are associated with a CCDS ID are kept for transcriptional strand bias analysis.

This information is needed for `StrelkaSBSVCFFilesToCatalog`, `StrelkaSBSVCFFilesToCatalogAndPlotToPdf`, `MutectVCFFilesToCatalog`, `MutectVCFFilesToCatalogAndPlotToPdf`, `VCFsToSBSCatalogs` and `VCFsToDBSCatalogs`.

## Source

`ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/`
`GRCh37_mapping/gencode.v30lift37.annotation.gff3.gz`

`ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/`
`gencode.v30.annotation.gff3.gz`

`ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M21/`
`gencode.vM21.annotation.gff3.gz`

## Examples

```
trans.ranges.GRCh37
# chrom    start       end strand Ensembl.gene.ID  gene.symbol
#     1    65419     71585      + ENSG00000186092         OR4F5
#     1   367640    368634      + ENSG00000235249        OR4F29
#     1   621059    622053      - ENSG00000284662        OR4F16
#     1   859308    879961      + ENSG00000187634        SAMD11
#     1   879583    894689      - ENSG00000188976         NOC2L
#   ...      ...       ...    ...             ...           ...
```

---

TransformCatalog          *Transform between counts and density spectrum catalogs and counts*
                          *and density signature catalogs*

---

### Description

Transform between counts and density spectrum catalogs and counts and density signature catalogs

### Usage

```
TransformCatalog(
  catalog,
  target.ref.genome = NULL,
  target.region = NULL,
  target.catalog.type = NULL,
  target.abundance = NULL
)
```

### Arguments

catalog            An SBS or DBS catalog as described in ICAMS; must **not** be an ID (small in-
                   sertions and deletions) catalog.

target.ref.genome
                   A ref.genome argument as described in ICAMS. If NULL, then defaults to
                   the ref.genome attribute of catalog.

target.region
                   A region argument; see as.catalog and ICAMS. If NULL, then defaults to
                   the region attribute of catalog.

target.catalog.type
                   A character string acting as a catalog type identifier, one of "counts", "density",
                   "counts.signature", "density.signature"; see as.catalog. If NULL, then de-
                   faults to the catalog.type attribute of catalog.

target.abundance
                   A vector of counts, one for each source K-mer for mutations (e.g. for strand-
                   agnostic single nucleotide substitutions in trinucleotide – i.e. 3-mer – con-
                   text, one count each for ACA, ACC, ACG, ... TTT). See all.abundance.
                   If NULL, the function tries to infer target.abundace from the class of
                   catalog and the value of the target.ref.genome, target.region,
                   and target.catalog.type.

### Details

Only the following transformations are legal:

1. counts -> counts (deprecated, generates a warning; we strongly suggest that you work
   with densities if comparing spectra or signatures generated from data with different underlying
   abundances.)

2. counts -> density

3. counts -> (counts.signature, density.signature)

4. `density -> counts` (the semantics are to infer the genome-wide or exome-wide counts based on the densities)

5. `density -> density` (a null operation, generates a warning)

6. `density -> (counts.signature,density.signature)`

7. `counts.signature -> counts.signature` (used to transform between the source abundance and `target.abundance`)

8. `counts.signature -> density.signature`

9. `counts.signature -> (counts,density)` (generates an error)

10. `density.signature -> density.signature` (a null operation, generates a warning)

11. `density.signature -> counts.signature`

12. `density.signature -> (counts,density)` (generates an error)

**Value**

A catalog as defined in ICAMS.

**Rationale**

The TransformCatalog function transforms catalogs of mutational spectra or signatures to account for differing abundances of the source sequence of the mutations in the genome.

For example, mutations from ACG are much rarer in the human genome than mutations from ACC simply because CG dinucleotides are rare in the genome. Consequently, there are two possible representations of mutational spectra or signatures. One representation is based on mutation counts as observed in a given genome or exome, and this approach is widely used, as, for example, at https://cancer.sanger.ac.uk/cosmic/signatures, which presents signatures based on observed mutation counts in the human genome. We call these "counts-based spectra" or "counts-based signatures".

Alternatively, mutational spectra or signatures can be represented as mutations per source sequence, for example the number of ACT > AGT mutations occurring at all ACT 3-mers in a genome. We call these "density-based spectra" or "density-based signatures".

This function can also transform spectra based on observed genome-wide counts to "density"-based catalogs. In density-based catalogs mutations are expressed as mutations per source sequences. For example, a density-based catalog represents the proportion of ACCs mutated to ATCs, the proportion of ACGs mutated to ATGs, etc. This is different from counts-based mutational spectra catalogs, which contain the number of ACC > ATC mutations, the number of ACG > ATG mutations, etc.

This function can also transform observed-count based spectra or signatures from genome to exome based counts, or between different species (since the abundances of source sequences vary between genome and exome and between species).

**Examples**

```
file <- system.file("extdata",
                    "strelka.regress.cat.sbs.96.csv",
                    package = "ICAMS")
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catSBS96.counts <- ReadCatalog(file, ref.genome = "hg19",
                                 region = "genome",
                                 catalog.type = "counts")
  catSBS96.density <- TransformCatalog(catSBS96.counts,
```

```
                                                      target.ref.genome = "hg19",
                                                      target.region = "genome",
                                                      target.catalog.type = "density")}
```

| VCFsToCatalogs | *Create SBS, DBS and Indel catalogs from VCFs* |
|---|---|

### Description

Create 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and Indel catalog from the Mutect VCFs specified by `files`

### Usage

```
VCFsToCatalogs(
  files,
  ref.genome,
  variant.caller = "unknown",
  num.of.cores = 1,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  filter.status = "PASS",
  get.vaf.function = NULL,
  ...,
  max.vaf.diff = 0.02,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

### Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the VCF files. |
| `ref.genome` | A `ref.genome` argument as described in `ICAMS`. |
| `variant.caller` | |
| | Name of the variant caller that produces the VCF, can be either `"strelka"`, `"mutect"`, `"freebayes"` or `"unknown"`. This information is needed to calculate the VAFs (variant allele frequencies). If variant caller is `"unknown"`(default) and `get.vaf.function` is NULL, then VAF and read depth will be NAs. If variant caller is `"mutect"`, do **not** merge SBSs into DBS. |
| `num.of.cores` | The number of cores to use. Not available on Windows unless `num.of.cores` `= 1`. |
| `trans.ranges` | Optional. If `ref.genome` specifies one of the `BSgenome` object |

1. BSgenome.Hsapiens.1000genomes.hs37d5
2. BSgenome.Hsapiens.UCSC.hg38
3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to [`TranscriptRanges`] for more details. If `is.null(trans.ranges)` do not add transcript range information.

region     A character string designating a genomic region; see [`as.catalog`] and [`ICAMS`].

names.of.VCFs

Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

tumor.col.names

Optional. Only applicable to **Mutect** VCFs. Vector of column names or column indices in **Mutect** VCFs which contain the tumor sample information. The order of elements in `tumor.col.names` should match the order of **Mutect** VCFs specified in `files`. If `tumor.col.names` is equal to `NA`(default), this function will use the 10th column in all the **Mutect** VCFs to calculate VAFs. See [`GetMutectVAF`] for more details.

filter.status

The character string in column `FILTER` of the VCF that indicates that a variant has passed all the variant caller's filters. Variants (lines in the VCF) for which the value in column `FILTER` does not equal `filter.status` are silently excluded from the output. If `NULL`, all variants are retained. In almost all cases, the default value of `"PASS"` is what the user would want.

get.vaf.function

Optional. Only applicable when `variant.caller` is **"unknown"**. Function to calculate VAF(variant allele frequency) and read depth information from original VCF. See [`GetMutectVAF`] as an example. If `NULL`(default) and `variant.caller` is "unknown", then VAF and read depth will be NAs.

...     Optional arguments to `get.vaf.function`.

max.vaf.diff   **Not** applicable if `variant.caller = "mutect"`. The maximum difference of VAF, default value is 0.02. If the absolute difference of VAFs for adjacent SBSs is bigger than `max.vaf.diff`, then these adjacent SBSs are likely to be "merely" asynchronous single base mutations, opposed to a simultaneous doublet mutation or variants involving more than two consecutive bases. Use negative value (e.g. -1) to suppress merging adjacent SBSs to DBS.

return.annotated.vcfs

Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls [`VCFsToSBSCatalogs`], [`VCFsToDBSCatalogs`] and [`VCFsToIDCatalogs`]

## Value

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).
- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).
- `catID`: Matrix of ID (small insertions and deletions) catalog.
- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.
- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs = TRUE`. A list of elements:
  - `SBS`: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
  - `DBS`: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.
  - `ID`: ID VCF annotated by `AnnotateIDVCF` with one new column `ID.class` showing the mutation class for each ID variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

### ID classification

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

### Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

### Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

### Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <- VCFsToCatalogs(file, ref.genome = "hg19",
                             variant.caller = "mutect", region = "genome")}
```

---

VCFsToCatalogsAndPlotToPdf

*Create SBS, DBS and Indel catalogs from VCFs and plot them to PDF*

---

#### Description

Create 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and Indel catalog from the VCFs specified by `files` and plot them to PDF

#### Usage

```
VCFsToCatalogsAndPlotToPdf(
  files,
  output.dir,
  ref.genome,
  variant.caller = "unknown",
  num.of.cores = 1,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  filter.status = "PASS",
  get.vaf.function = NULL,
  ...,
  max.vaf.diff = 0.02,
  base.filename = "",
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

#### Arguments

| | |
|---|---|
| `files` | Character vector of file paths to the VCF files. |
| `output.dir` | The directory where the PDF files will be saved. |
| `ref.genome` | A `ref.genome` argument as described in [ICAMS](#). |
| `variant.caller` | Name of the variant caller that produces the VCF, can be either `"strelka"`, `"mutect"`, `"freebayes"` or `"unknown"`. This information is needed to calculate the VAFs (variant allele frequencies). If variant caller is `"unknown"`(default) and `get.vaf.function` is NULL, then VAF and read depth will be NAs. If variant caller is `"mutect"`, do **not** merge SBSs into DBS. |
| `num.of.cores` | The number of cores to use. Not available on Windows unless `num.of.cores` = 1. |
| `trans.ranges` | Optional. If `ref.genome` specifies one of the [BSgenome](#) object |

> 1. BSgenome.Hsapiens.1000genomes.hs37d5
> 2. BSgenome.Hsapiens.UCSC.hg38
> 3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to `TranscriptRanges` for more details. If `is.null(trans.ranges)` do not add transcript range information.

region
: A character string designating a genomic region; see `as.catalog` and `ICAMS`.

names.of.VCFs

: Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCF file paths in `files`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `files` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

tumor.col.names

: Optional. Only applicable to **Mutect** VCFs. Vector of column names or column indices in **Mutect** VCFs which contain the tumor sample information. The order of elements in `tumor.col.names` should match the order of **Mutect** VCFs specified in `files`. If `tumor.col.names` is equal to NA(default), this function will use the 10th column in all the **Mutect** VCFs to calculate VAFs. See `GetMutectVAF` for more details.

filter.status

: The character string in column `FILTER` of the VCF that indicates that a variant has passed all the variant caller's filters. Variants (lines in the VCF) for which the value in column `FILTER` does not equal `filter.status` are silently excluded from the output. If `NULL`, all variants are retained. In almost all cases, the default value of `"PASS"` is what the user would want.

get.vaf.function

: Optional. Only applicable when `variant.caller` is **"unknown"**. Function to calculate VAF(variant allele frequency) and read depth information from original VCF. See `GetMutectVAF` as an example. If `NULL`(default) and `variant.caller` is "unknown", then VAF and read depth will be NAs.

...
: Optional arguments to `get.vaf.function`.

max.vaf.diff
: **Not** applicable if `variant.caller = "mutect"`. The maximum difference of VAF, default value is 0.02. If the absolute difference of VAFs for adjacent SBSs is bigger than `max.vaf.diff`, then these adjacent SBSs are likely to be "merely" asynchronous single base mutations, opposed to a simultaneous doublet mutation or variants involving more than two consecutive bases. Use negative value (e.g. -1) to suppress merging adjacent SBSs to DBS.

base.filename

: Optional. The base name of the PDF files to be produced; multiple files will be generated, each ending in $x$.pdf, where $x$ indicates the type of catalog plotted in the file.

return.annotated.vcfs

: Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

: Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Details

This function calls `VCFsToCatalogs` and `PlotCatalogToPdf`

**Value**

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).

- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).

- `catID`: Matrix of ID (small insertions and deletions) catalog.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. A list of elements:

  - `SBS`: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.
  - `DBS`: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.
  - `ID`: ID VCF annotated by `AnnotateIDVCF` with one new column `ID.class` showing the mutation class for each ID variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

**ID classification**

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

**Note**

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

**Comments**

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    VCFsToCatalogsAndPlotToPdf(file, ref.genome = "hg19",
                               output.dir = tempdir(),
                               variant.caller = "mutect",
                               region = "genome",
                               base.filename = "Mutect")}
```

---

VCFsToDBSCatalogs    *Create DBS catalogs from VCFs*

---

## Description

Create a list of 3 catalogs (one each for DBS78, DBS144 and DBS136) out of the contents in list.of.DBS.vcfs. The VCFs must not contain any type of mutation other then DBSs.

## Usage

```
VCFsToDBSCatalogs(
  list.of.DBS.vcfs,
  ref.genome,
  num.of.cores = 1,
  trans.ranges = NULL,
  region = "unknown",
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

list.of.DBS.vcfs

List of in-memory data frames of pure DBS mutations – no SBS or 3+BS mutations. The list names will be the sample ids in the output catalog.

ref.genome    A ref.genome argument as described in [ICAMS].

num.of.cores  The number of cores to use. Not available on Windows unless num.of.cores = 1.

trans.ranges  Optional. If ref.genome specifies one of the [BSgenome] object

1. BSgenome.Hsapiens.1000genomes.hs37d5
2. BSgenome.Hsapiens.UCSC.hg38
3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer trans.ranges automatically. Otherwise, user will need to provide the necessary trans.ranges. Please refer to [TranscriptRanges] for more details. If is.null(trans.ranges) do not add transcript range information.

region        A character string designating a genomic region; see [as.catalog] and [ICAMS].

```
return.annotated.vcfs
```
> Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

```
suppress.discarded.variants.warnings
```
> Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Value

A list containing the following objects:

- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs` = TRUE. DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

## Comments

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

## Note

DBS 144 catalog only contains mutations in transcribed regions.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.DBS.vcfs <- ReadAndSplitVCFs(file, variant.caller = "mutect")$DBS
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs.DBS <- VCFsToDBSCatalogs(list.of.DBS.vcfs, ref.genome = "hg19",
                                    trans.ranges = trans.ranges.GRCh37,
                                    region = "genome")}
```

---

VCFsToIDCatalogs    *Create ID (small insertions and deletions) catalog from ID VCFs*

---

## Description

Create ID (small insertions and deletions) catalog from ID VCFs

## Usage

```
VCFsToIDCatalogs(
  list.of.vcfs,
  ref.genome,
  num.of.cores = 1,
  region = "unknown",
  flag.mismatches = 0,
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

## Arguments

list.of.vcfs    List of in-memory ID VCFs. The list names will be the sample ids in the output
                catalog.

ref.genome      A `ref.genome` argument as described in ICAMS.

num.of.cores    The number of cores to use. Not available on Windows unless `num.of.cores`
                `= 1`.

region          A character string acting as a region identifier, one of "genome", "exome".

flag.mismatches

        Deprecated. If there are ID variants whose `REF` do not match the extracted sequence from `ref.genome`, the function will automatically discard these variants and an element `discarded.variants` will appear in the return value. See AnnotateIDVCF for more details.

return.annotated.vcfs

        Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

        Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Value

A **list** of elements:

- `catalog`: The ID (small insertions and deletions) catalog with attributes added. See as.catalog for details.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs = TRUE`. A list of data frames which contain the original VCF's ID mutation rows with three additional columns `seq.context.width`, `seq.context` and `ID.class` added. The category assignment of each ID mutation in VCF can be obtained from `ID.class` column.

## Note

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

### ID classification

See [https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_](https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_) [indel_classification_2021_09_03.xlsx](indel_classification_2021_09_03.xlsx) for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. CACACACA, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

### Examples

```
file <- c(system.file("extdata/Strelka-ID-vcf/",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.ID.vcfs <- ReadAndSplitVCFs(file, variant.caller = "strelka")$ID
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5",
 quietly = TRUE)) {
  catID <- VCFsToIDCatalogs(list.of.ID.vcfs, ref.genome = "hg19",
                            region = "genome")}
```

---

VCFsToSBSCatalogs     *Create SBS catalogs from SBS VCFs*

---

### Description

Create a list of 3 catalogs (one each for 96, 192, 1536) out of the contents in list.of.SBS.vcfs. The SBS VCFs must not contain DBSs, indels, or other types of mutations.

### Usage

```
VCFsToSBSCatalogs(
  list.of.SBS.vcfs,
  ref.genome,
  num.of.cores = 1,
  trans.ranges = NULL,
  region = "unknown",
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

### Arguments

`list.of.SBS.vcfs`

> List of in-memory data frames of pure SBS mutations – no DBS or 3+BS mutations. The list names will be the sample ids in the output catalog.

`ref.genome`     A `ref.genome` argument as described in [ICAMS](ICAMS).

`num.of.cores`   The number of cores to use. Not available on Windows unless `num.of.cores` = 1.

trans.ranges   Optional. If ref.genome specifies one of the BSgenome object

1. BSgenome.Hsapiens.1000genomes.hs37d5
2. BSgenome.Hsapiens.UCSC.hg38
3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer trans.ranges automatically. Otherwise, user will need to provide the necessary trans.ranges. Please refer to TranscriptRanges for more details. If is.null(trans.ranges) do not add transcript range information.

region            A character string designating a genomic region; see as.catalog and ICAMS.

return.annotated.vcfs

Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

## Value

A list containing the following objects:

- catSBS96, catSBS192, catSBS1536: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).

- discarded.variants: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column discarded.reason for more details.

- annotated.vcfs: **Non-NULL only if** return.annotated.vcfs = TRUE. SBS VCF annotated by AnnotateSBSVCF with three new columns SBS96.class, SBS192.class and SBS1536.class showing the mutation class for each SBS variant.

If trans.ranges is not provided by user and cannot be inferred by ICAMS, SBS 192 catalog will not be generated. Each catalog has attributes added. See as.catalog for more details.

## Comments

To add or change attributes of the catalog, you can use function attr.
For example, attr(catalog,"abundance") <-custom.abundance.

## Note

SBS 192 catalogs only contain mutations in transcribed regions.

## Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                      "Mutect.GRCh37.s1.vcf",
                      package = "ICAMS"))
list.of.SBS.vcfs <- ReadAndSplitVCFs(file, variant.caller = "mutect")$SBS
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs.SBS <- VCFsToSBSCatalogs(list.of.SBS.vcfs, ref.genome = "hg19",
                                    trans.ranges = trans.ranges.GRCh37,
                                    region = "genome")}
```

---

VCFsToZipFile                *Create a zip file which contains catalogs and plot PDFs from VCFs*

---

### Description

Create 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and Indel catalog from the VCFs specified by `dir`, save the catalogs as CSV files, plot them to PDF and generate a zip archive of all the output files.

### Usage

```
VCFsToZipFile(
  dir,
  files,
  zipfile,
  ref.genome,
  variant.caller = "unknown",
  num.of.cores = 1,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  filter.status = "PASS",
  get.vaf.function = NULL,
  ...,
  max.vaf.diff = 0.02,
  base.filename = "",
  return.annotated.vcfs = FALSE,
  suppress.discarded.variants.warnings = TRUE
)
```

### Arguments

| | |
|---|---|
| `dir` | Pathname of the directory which contains VCFs that come from the **same** variant caller. Each VCF **must** have a file extension ".vcf" (case insensitive) and share the **same** `ref.genome` and `region`. |
| `files` | Character vector of file paths to the VCF files. Only **one** of argument `dir` or `files` need to be specified. |
| `zipfile` | Pathname of the zip file to be created. |
| `ref.genome` | A `ref.genome` argument as described in [ICAMS](#). |
| `variant.caller` | |
| | Name of the variant caller that produces the VCF, can be either `"strelka"`, `"mutect"`, `"freebayes"` or `"unknown"`. This information is needed to calculate the VAFs (variant allele frequencies). If variant caller is `"unknown"`(default) and `get.vaf.function` is NULL, then VAF and read depth will be NAs. If variant caller is `"mutect"`, do **not** merge SBSs into DBS. |
| `num.of.cores` | The number of cores to use. Not available on Windows unless `num.of.cores = 1`. |
| `trans.ranges` | Optional. If `ref.genome` specifies one of the [BSgenome](#) object |

1. BSgenome.Hsapiens.1000genomes.hs37d5

2. BSgenome.Hsapiens.UCSC.hg38

3. BSgenome.Mmusculus.UCSC.mm10

then the function will infer `trans.ranges` automatically. Otherwise, user will need to provide the necessary `trans.ranges`. Please refer to `TranscriptRanges` for more details. If `is.null(trans.ranges)` do not add transcript range information.

region                A character string designating a genomic region; see `as.catalog` and `ICAMS`.

names.of.VCFs

Optional. Character vector of names of the VCF files. The order of names in `names.of.VCFs` should match the order of VCFs listed in `dir`. If `NULL`(default), this function will remove all of the path up to and including the last path separator (if any) in `dir` and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

tumor.col.names

Optional. Only applicable to **Mutect** VCFs. Vector of column names or column indices in **Mutect** VCFs which contain the tumor sample information. The order of elements in `tumor.col.names` should match the order of **Mutect** VCFs specified in `files`. If `tumor.col.names` is equal to `NA`(default), this function will use the 10th column in all the **Mutect** VCFs to calculate VAFs. See `GetMutectVAF` for more details.

filter.status

The character string in column `FILTER` of the VCF that indicates that a variant has passed all the variant caller's filters. Variants (lines in the VCF) for which the value in column `FILTER` does not equal `filter.status` are silently excluded from the output. If `NULL`, all variants are retained. In almost all cases, the default value of `"PASS"` is what the user would want.

get.vaf.function

Optional. Only applicable when `variant.caller` is **"unknown"**. Function to calculate VAF(variant allele frequency) and read depth information from original VCF. See `GetMutectVAF` as an example. If `NULL`(default) and `variant.caller` is "unknown", then VAF and read depth will be NAs.

...                   Optional arguments to `get.vaf.function`.

max.vaf.diff          **Not** applicable if `variant.caller = "mutect"`. The maximum difference of VAF, default value is 0.02. If the absolute difference of VAFs for adjacent SBSs is bigger than `max.vaf.diff`, then these adjacent SBSs are likely to be "merely" asynchronous single base mutations, opposed to a simultaneous doublet mutation or variants involving more than two consecutive bases. Use negative value (e.g. -1) to suppress merging adjacent SBSs to DBS.

base.filename

Optional. The base name of the CSV and PDF files to be produced; multiple files will be generated, each ending in $x$.csv or $x$.pdf, where $x$ indicates the type of catalog.

return.annotated.vcfs

Logical. Whether to return the annotated VCFs with additional columns showing mutation class for each variant. Default is FALSE.

suppress.discarded.variants.warnings

Logical. Whether to suppress warning messages showing information about the discarded variants. Default is TRUE.

**Details**

This function calls `VCFsToCatalogs`, `PlotCatalogToPdf`, `WriteCatalog` and `zip::zipr`.

**Value**

A list containing the following objects:

- `catSBS96`, `catSBS192`, `catSBS1536`: Matrix of 3 SBS catalogs (one each for 96, 192, and 1536).

- `catDBS78`, `catDBS136`, `catDBS144`: Matrix of 3 DBS catalogs (one each for 78, 136, and 144).

- `catID`: Matrix of ID (small insertions and deletions) catalog.

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

- `annotated.vcfs`: **Non-NULL only if** `return.annotated.vcfs = TRUE`. A list of elements:

  - `SBS`: SBS VCF annotated by `AnnotateSBSVCF` with three new columns `SBS96.class`, `SBS192.class` and `SBS1536.class` showing the mutation class for each SBS variant.

  - `DBS`: DBS VCF annotated by `AnnotateDBSVCF` with three new columns `DBS78.class`, `DBS136.class` and `DBS144.class` showing the mutation class for each DBS variant.

  - `ID`: ID VCF annotated by `AnnotateIDVCF` with one new column `ID.class` showing the mutation class for each ID variant.

If `trans.ranges` is not provided by user and cannot be inferred by ICAMS, SBS 192 and DBS 144 catalog will not be generated. Each catalog has attributes added. See `as.catalog` for more details.

**ID classification**

See `https://github.com/steverozen/ICAMS/raw/master/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx` for additional information on ID (small insertions and deletions) mutation classification.

See the documentation for `Canonicalize1Del` which first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units, (e.g. `CACACACA`, see `FindMaxRepeatDel`), and if the deletion is not in a simple repeat, looks for microhomology (see `FindDelMH`).

See the code for unexported function `CanonicalizeID` and the functions it calls for handling of insertions.

**Note**

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

**Comments**

To add or change attributes of the catalog, you can use function `attr`.
For example, `attr(catalog,"abundance") <-custom.abundance`.

### Examples

```
dir <- c(system.file("extdata/Mutect-vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    VCFsToZipFile(dir,
                  zipfile = file.path(tempdir(), "test.zip"),
                  ref.genome = "hg19",
                  variant.caller = "mutect",
                  region = "genome",
                  base.filename = "Mutect")
  unlink(file.path(tempdir(), "test.zip"))}
```

---

WriteCatalog *Write a catalog*

---

### Description

Write a catalog to a file.

### Usage

```
WriteCatalog(catalog, file, strict = TRUE)
```

### Arguments

| | |
|---|---|
| catalog | A catalog as defined in ICAMS; see also as.catalog. |
| file | The path to the file to be created. |
| strict | If TRUE, do additional checks on the input, and stop if the checks fail. |

### Details

See also ReadCatalog.

### Note

In ID (small insertions and deletions) catalogs, deletion repeat sizes range from 0 to 5+, but for plotting and end-user documentation deletion repeat sizes range from 1 to 6+.

### Examples

```
file <- system.file("extdata",
                     "strelka.regress.cat.sbs.96.csv",
                     package = "ICAMS")
catSBS96 <- ReadCatalog(file)
WriteCatalog(catSBS96, file = file.path(tempdir(), "catSBS96.csv"))
```

# Index