

Package ‘ICAMS’

December 16, 2019

Type Package

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

Version 2.1.0.9000

Author Steve Rozen, Nanhai Jiang, Arnoud Boot, Mo Liu

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>. '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" catalogs of mutational spectra or signatures.

License GPL-3

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,
GenomeInfoDb,
GenomicRanges,
graphics,
grDevices,
IRanges,
RColorBrewer,
stats,
stringi,
utils

Depends R (>= 3.5),

RoxygenNote 7.0.2

Suggests testthat,

BSgenome.Hsapiens.1000genomes.hs37d5,

BSgenome.Hsapiens.UCSC.hg38,

BSgenome.Mmusculus.UCSC.mm10

R topics documented:

all.abundance	3
AnnotateDBSVCF	3
AnnotateIDVCF	4
AnnotateSBSVCF	5
as.catalog	6
Canonicalize1Del	7
CatalogRowOrder	8
CollapseCatalog	8
FindDelMH	9
FindMaxRepeatDel	12
GeneExpressionData	13
GetVAF	13
ICAMS	14
MutectVCFFilesToCatalog	17
MutectVCFFilesToCatalogAndPlotToPdf	19
MutectVCFFilesToZipFile	20
PlotCatalog	22
PlotCatalogToPdf	23
PlotTransBiasGeneExp	24
PlotTransBiasGeneExpToPdf	26
ReadAndSplitMutectVCFs	27
ReadAndSplitStrelkaSBSVCFs	28
ReadCatalog	29
ReadStrelkaIDVCFs	30
revc	31
StrelkaIDVCFFilesToCatalog	31
StrelkaIDVCFFilesToCatalogAndPlotToPdf	32
StrelkaSBSVCFFilesToCatalog	34
StrelkaSBSVCFFilesToCatalogAndPlotToPdf	35
TranscriptRanges	36
TransformCatalog	37
VCFsToDBSCatalogs	39
VCFsToIDCatalogs	40
VCFsToSBSCatalogs	41
WriteCatalog	42

Index

43

all.abundance	<i>K-mer abundances.</i>
---------------	--------------------------

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	<i>Add sequence context and transcript information to an in-memory DBS VCF.</i>
----------------	---

Description

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

Usage

```
AnnotateDBSVCF(DBS.vcf, ref.genome, trans.ranges = 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	A data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details. If is.null(trans.ranges) do not add transcript range information.

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.GRCh37.vcf",
                     package = "ICAMS"))
list.of.vcfs <- ReadAndSplitStrelkaSBSVCFs(file)
DBS.vcf <- list.of.vcfs$DBS.vcfs[[1]]
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.DBS.vcf <- AnnotateDBSVCF(DBS.vcf, ref.genome = "hg19",
                                     trans.ranges = trans.ranges.GRCh37)}
```

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

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)
```

Arguments

<code>ID.vcf</code>	An in-memory ID (insertion/deletion) VCF as a <code>data.frame</code> . This function expects that there is a "context base" to the left, for example <code>REF = ACG</code> , <code>ALT = A</code> (deletion of CG) or <code>REF = A</code> , <code>ALT = ACC</code> (insertion of CC).
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>flag.mismatches</code>	If > 0 , then if there are mismatches to references, generate messages showing the mismatched rows and continue. Otherwise stop if there are mismatched rows.

Value

A data frame with 2 new columns added to the input data frame:

1. `seq.context` The sequence embedding the variant.
2. `seq.context.width` The width of `seq.context` to the left

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 .
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	<i>Given a deletion and its sequence context, categorize it.</i>
------------------	--

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 mutation 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 warning 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	<i>Standard order of row names in a catalog.</i>
-----------------	--

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
```

```
catalog.row.order.sp
```

Format

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

Note

In ID (small insertion and deletion) 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.

catalog.row.order.sp$ID83
# "DEL:C:1:0" "DEL:C:1:1" "DEL:C:1:2" "DEL:C:1:3" ...
# There are altogether 83 row names to denote the mutation types
# in ID83 catalog.
```

CollapseCatalog	<i>"Collapse" a catalog.</i>
-----------------	------------------------------

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.

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	<i>Return the number of repeat units in which a deletion is embedded.</i>
------------------	---

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.

Examples

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

GeneExpressionData	<i>Example gene expression data from two cell lines.</i>
--------------------	--

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.

Examples

```
gene.expression.data.HepG2
# Ensembl.gene.ID gene.symbol counts      TPM
# ENSG000000000003      TSPAN6    6007  33.922648455
# ENSG000000000005      TNMD       0    0.000000000
# ENSG0000000000419      DPM1    4441  61.669371091
# ENSG0000000000457      SCYL3    1368   3.334619195
# ENSG0000000000460    C1orf112    916   2.416263423
#           ...           ...     ...           ...
```

GetVAF	<i>Extract the VAFs (variant allele frequencies) from a VCF file.</i>
--------	---

Description

Extract the VAFs (variant allele frequencies) from a VCF file.

Usage

```
GetStrelkaVAF(vcf)

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

Arguments

<code>vcf</code>	Said VCF as a <code>data.frame</code> .
<code>name.of.VCF</code>	Name of the VCF file.
<code>tumor.col.name</code>	Name of the column in VCF which contains the tumor sample information. It must have quotation marks. If <code>tumor.col.name</code> is equal to <code>NA</code> (default), this function will use the 10th column to calculate VAFs.

Value

A vector of VAFs, one for each row of `vcf`.

Examples

```
file <- c(system.file("extdata",
                     "Strelka.SBS.GRCh37.vcf",
                     package = "ICAMS"))
MakeDataFrameFromStrelkaSBSVCF <-
  getFromNamespace("MakeDataFrameFromStrelkaSBSVCF", "ICAMS")
df <- MakeDataFrameFromStrelkaSBSVCF(file)
vaf <- GetStrelkaVAF(df)
```

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>. "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" catalogs of mutational spectra or signatures.

Details

"ICAMS" can read in VCFs generated by Strelka or Mutect, 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), double 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, 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 possible 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 [StrelkaSBSVCFFilesToCatalog](#), [MutectVCFFilesToCatalog](#), etc.), then use `as.catalog`.

Creating catalogs from variant call files (VCF files)

1. [StrelkaSBSVCFFilesToCatalog](#) creates 3 SBS catalogs (96, 192, 1536) and 3 DBS catalogs (78, 136, 144) from the Strelka SBS VCFs.
2. [StrelkaIDVCFFilesToCatalog](#) creates an ID (small insertion and deletion) catalog from the Strelka ID VCFs.
3. [MutectVCFFilesToCatalog](#) creates 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and ID (small insertion and deletion) catalog from the Mutect VCFs.

Plotting catalogs

The [PlotCatalog](#) functions plot mutational spectra for **one** sample or plot **one** mutational signature.

The [PlotCatalogToPdf](#) functions plot catalogs of mutational spectra or of mutational signatures to a PDF file.

Wrapper functions to create catalogs from VCFs and plot the catalogs to PDF files

1. [StrelkaSBSVCFFilesToCatalogAndPlotToPdf](#) creates all type of SBS and DBS catalogs from Strelka SBS VCFs and plots the catalogs.
2. [StrelkaIDVCFFilesToCatalogAndPlotToPdf](#) creates an ID (small insertion and deletion) catalog from Strelka ID VCFs and plot it.
3. [MutectVCFFilesToCatalogAndPlotToPdf](#) creates all types of SBS, DBS, and ID catalogs from Mutect VCFs and plots the catalogs.

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

The [WriteCatalog](#) functions write a catalog to a file.

Reading catalogs

The [ReadCatalog](#) functions read a file that contains a catalog in standardized format.

Transforming catalogs

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).

Collapsing catalogs

The [CollapseCatalog](#) functions

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).

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. [GeneExpressionData](#) Example gene expression data from two cell lines.

MutectVCFFilesToCatalog

Create SBS, DBS and Indel catalogs from Mutect VCF files

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

```
MutectVCFFilesToCatalog(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA
)
```

MutectVCFFilesToCatalogAndPlotToPdf

Create SBS, DBS and Indel catalogs from Mutect VCF files and plot them to PDF

Description

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 = ""
)
```

Arguments

files	Character vector of file paths to the Mutect VCF files.
ref.genome	A ref.genome argument as described in ICAMS .
trans.ranges	A data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details.
region	A character string designating a genomic region; see as.catalog and ICAMS .
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.
tumor.col.names	Character vector of column names in VCFs which contain the tumor sample information. The order of names 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	The base name of the PDF files to be produced; multiple files will be generated, each ending in <i>x.pdf</i> , where <i>x</i> indicates the type of catalog plotted in the file.

Details

This function calls [MutectVCFFilesToCatalog](#) and [PlotCatalogToPdf](#)

Value

A list of 3 SBS catalogs (one each for 96, 192, and 1536), 3 DBS catalogs (one each for 78, 136, and 144), Indel catalog and their graphs plotted to PDF with specified file name. If `trans.ranges = NULL`, SBS 192 and DBS 144 catalog will not be generated and plotted. 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 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertion and deletion) 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 <- c(system.file("extdata/Mutect-vcf",
                     "Mutect.GRCh37.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"))}
```

MutectVCFFilesToZipFile

Create a zip file which contains catalogs and plot PDFs from Mutect VCF files

Description

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

Usage

```
MutectVCFFilesToZipFile(
  file,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  tumor.col.names = NA,
  output.file = "",
  zipfile.name = "output"
)
```

Arguments

<code>file</code>	Full path of the directory which contains the Mutect VCF files. Each Mutect VCF must have a file extension ".vcf" (case insensitive) and share the same <code>ref.genome</code> and region.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>trans.ranges</code>	a data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details.
<code>region</code>	A character string designating a genomic region; see as.catalog and ICAMS .
<code>names.of.VCFs</code>	Character vector of names of the VCF files. The order of names in <code>names.of.VCFs</code> should match the order of VCFs listed in <code>file</code> . If <code>NULL</code> (default), this function will remove all of the path up to and including the last path separator (if any) in <code>file</code> and file paths without extensions (and the leading dot) will be used as the names of the VCF files.
<code>tumor.col.names</code>	Character vector of column names in VCFs which contain the tumor sample information. The order of names in <code>tumor.col.names</code> should match the order of VCFs listed in <code>file</code> . If <code>tumor.col.names</code> is equal to <code>NA</code> (default), this function will use the 10th column in all the VCFs to calculate VAFs. See GetMutectVAF for more details.
<code>output.file</code>	The base name of the CSV and PDF files to be produced; multiple files will be generated, each ending in <code>x.csv</code> or <code>x.pdf</code> , where <code>x</code> indicates the type of catalog.
<code>zipfile.name</code>	The name of the zip file to be created. It should not contain the file extension ".zip" and if not specified, the default is "output".

Details

This function calls [MutectVCFFilesToCatalog](#), [PlotCatalogToPdf](#), [WriteCatalog](#) and [zip](#).

Value

A list of 3 SBS catalogs (one each for 96, 192, and 1536), 3 DBS catalogs (one each for 78, 136, and 144) and Indel catalog. If `trans.ranges = NULL`, SBS 192 and DBS 144 catalog will not be generated and plotted. 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 and DBS 144 catalogs include only mutations in transcribed regions. In ID (small insertion and deletion) 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 <- c(system.file("extdata/Mutect-vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <-
    MutectVCFFilesToZipFile(file, ref.genome = "hg19",
```

```

        trans.ranges = trans.ranges.GRCh37,
        region = "genome",
        output.file = file.path(tempdir(), "Mutect"),
        zipfile.name = "test")

unlink("test.zip")

```

PlotCatalog

*Plot **one** spectrum or signature.*

Description

Plot the spectrum of **one** sample or plot **one** signature. The type of graph is based on one 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,
  xlabel = NULL
)

```

Arguments

catalog	A catalog as defined in ICAMS with attributes added. See as.catalog for more details.
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, ...
cex	A numerical value giving the amount by which mutation class labels, mutation counts(if it exists), y axis and its labels, x axis labels and its annotations(if it exists), sample name and legend(if it exists) should be magnified relative to the default. Only implemented for SBS96Catalog, SBS192Catalog and DBS144Catalog.
grid	A logical value indicating whether to draw grid lines. Only implemented for SBS96Catalog.
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.
xlabels	A logical value indicating whether to draw x axis labels. Only implemented for SBS96Catalog.

Value

A list whose first element is a logic value indicating whether the plot is successful. The second element is a numeric vector giving the coordinates of all the bar midpoints drawn, useful for adding to the graph(**currently only implemented for SBS96Catalog**).

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	<i>Plot catalog to a PDF file.</i>
------------------	------------------------------------

Description

Plot catalog to a PDF file. The type of graph is based on one 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
)
```

Arguments

catalog	A catalog as defined in ICAMS with attributes added. See as.catalog for more details.
file	The name of the PDF file to be produced.
plot.SBS12	Only meaningful for class <code>SBS192Catalog</code> ; 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	A numerical value giving the amount by which mutation class labels, mutation counts(if it exists), y axis and its labels, x axis labels and its annotations(if it exists), sample name and legend(if it exists) should be magnified relative to the default. Only implemented for <code>SBS96Catalog</code> , <code>SBS192Catalog</code> and <code>DBS144Catalog</code> .

grid	A logical value indicating whether to draw grid lines. Only implemented for SBS96Catalog.
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.
xlabels	A logical value indicating whether to draw x axis labels. Only implemented for SBS96Catalog.

Value

invisible(TRUE)

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

<code>annotated.SBS.vcf</code>	An SBS VCF annotated by AnnotateSBSVCF . It must have transcript range information added.
<code>expression.data</code>	A data.table which contains the expression values of genes. See GeneExpressionData for more details.
<code>Ensembl.gene.ID.col</code>	Name of column which has the Ensembl gene ID information in <code>expression.data</code> .
<code>expression.value.col</code>	Name of column which has the gene expression values in <code>expression.data</code> .
<code>num.of.bins</code>	The number of bins that will be plotted on the graph.
<code>plot.type</code>	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".
<code>damaged.base</code>	One of NULL, "purine" or "pyrimidine". This function allocates approximately equal numbers of mutations from <code>damaged.base</code> into each of <code>num.of.bins</code> bin by expression level. E.g. if <code>damaged.base</code> 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 <code>damaged.base</code> 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 <code>damaged.base</code> based on mutation counts.
<code>ymax</code>	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.GRCh37.vcf",
                     package = "ICAMS"))
list.of.vcfs <- ReadAndSplitStrelkaSBSVCFs(file)
SBS.vcf <- list.of.vcfs$SBS.vcfs[[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 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.

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.GRCh37.vcf",
                     package = "ICAMS"))
list.of.vcfs <- ReadAndSplitStrelkaSBSVCFs(file)
SBS.vcf <- list.of.vcfs$SBS.vcfs[[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

Read and split Mutect VCF files.

Description

Read and split Mutect VCF files.

Usage

```
ReadAndSplitMutectVCFs(files, names.of.VCFs = NULL, tumor.col.names = NA)
```

Arguments

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

tumor.col.names

Character vector of column names in VCFs which contain the tumor sample information. The order of names 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.

Value

A list with 3 in-memory VCFs and two left-over VCF-like data frames with rows that were not incorporated into the first 3 VCFs, as follows:

1. SBS VCF with only single base substitutions.
2. DBS VCF with only doublet base substitutions as called by Mutect.
3. ID VCF with only small insertions and deletions.
4. other.subs VCF like data.frame with rows for coordinate substitutions involving 3 or more nucleotides, e.g. ACT > TGA or AACT > GGTA.
5. multiple.alternative.alleles VCF like data.frame with rows for variants with multiple alternative alleles, for example ACT mutated to both AGT and ACT at the same position.

See Also

[MutectVCFFilesToCatalog](#)

Examples

```
file <- c(system.file("extdata/Mutect-vcf",
                     "Mutect.GRCh37.vcf",
                     package = "ICAMS"))
list.of.vcfs <- ReadAndSplitMutectVCFs(file)
```

ReadAndSplitStrelkaSBSVCFs

Read and split Strelka SBS VCF files.

Description

Read and split Strelka SBS VCF files.

Usage

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

Arguments

files	Character vector of file paths to the Strelka SBS 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) and file paths without extensions (and the leading dot) will be used as the names of the VCF files.

Value

A list of 3 in-memory objects as follows:

1. `SBS.vcfs` List of data.frames of pure SBS mutations – no DBS or 3+BS mutations.
2. `DBS.vcfs` List of data.frames of pure DBS mutations – no SBS or 3+BS mutations.
3. `ThreePlus` List of data.tables with the key `CHROM`, `LOW.POS`, `HIGH.POS`. containing rows that that in the input that did not represent SBSs or DBSs.

See Also

[StrelkaSBSVCFFilesToCatalog](#)

Examples

```
file <- c(system.file("extdata",
                     "Strelka.SBS.GRCh37.vcf",
                     package = "ICAMS"))
list.of.vcfs <- ReadAndSplitStrelkaSBSVCFs(file)
```

ReadCatalog

Read catalog.

Description

Read a catalog in standardized format from path.

Usage

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

Arguments

<code>file</code>	Path to a catalog on disk in the standardized format.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>region</code>	region A character string designating a genomic region; see as.catalog and ICAMS .
<code>catalog.type</code>	One of "counts", "density", "counts.signature", "density.signature".
<code>strict</code>	If TRUE, do additional checks on the input, and stop if the checks fail.

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 insertion and deletion) 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	<i>Read Strelka ID (small insertion and deletion) VCF files.</i>
-------------------	--

Description

Read Strelka ID (small insertion and deletion) VCF files.

Usage

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

Arguments

<code>files</code>	Character vector of file paths to the VCF files.
<code>names.of.VCFs</code>	Character vector of names of the VCF files. The order of names in <code>names.of.VCFs</code> should match the order of VCF file paths in <code>files</code> . If <code>NULL</code> (default), this function 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.

Value

A list of vcfs from files.

Note

In ID (small insertion and deletion) 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](#)

Examples

```
file <- c(system.file("extdata",
                     "Strelka.ID.GRCh37.vcf",
                     package = "ICAMS"))
list.of.vcfs <- ReadStrelkaIDVCFs(file)
```

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)
```

StrelkaIDVCFFilesToCatalog

Create ID (small insertion and deletion) catalog from Strelka ID VCF files

Description

Create ID (small insertion and deletion) catalog from the Strelka ID VCFs specified by files

Usage

```
StrelkaIDVCFFilesToCatalog(
  files,
  ref.genome,
  region = "unknown",
  names.of.VCFs = NULL
)
```

Arguments

<code>files</code>	Character vector of file paths to the Strelka ID VCF files.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>region</code>	A character string designating a genomic region; see as.catalog and ICAMS .
<code>names.of.VCFs</code>	Character vector of names of the VCF files. The order of names in <code>names.of.VCFs</code> should match the order of VCF file paths in <code>files</code> . If <code>NULL</code> (default), this function 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.

Details

This function calls [VCFsToIDCatalogs](#)

Value

A list of two elements. 1st element is an S3 object containing an ID (small insertion and deletion) catalog with class "IndelCatalog". See [as.catalog](#) for more details. 2nd element is a list of further annotated VCFs.

Note

In ID (small insertion and deletion) 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 <- c(system.file("extdata",
                     "Strelka.ID.GRCh37.vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catID <- StrelkaIDVCFFilesToCatalog(file, ref.genome = "hg19",
                                     region = "genome")}
```

StrelkaIDVCFFilesToCatalogAndPlotToPdf

Create ID (small insertion and deletion) catalog from Strelka ID VCF files and plot them to PDF

Description

Create ID (small insertion and deletion) 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 = ""
)
```

Arguments

<code>files</code>	Character vector of file paths to the Strelka ID VCF files.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>region</code>	A character string designating a genomic region; see as.catalog and ICAMS .
<code>names.of.VCFs</code>	Character vector of names of the VCF files. The order of names in <code>names.of.VCFs</code> should match the order of VCF file paths in <code>files</code> . If <code>NULL</code> (default), this function 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.
<code>output.file</code>	The base name of the PDF file to be produced; the file is ending in <code>catID.pdf</code> .

Details

This function calls [StrelkaIDVCFFilesToCatalog](#) and [PlotCatalogToPdf](#)

Value

A list whose first element is an ID (small insertion and deletion) catalog with its graph plotted to PDF with specified file name. The ID catalog has attributes added. See [as.catalog](#) for more details. The second element of the returned list is a list of further annotated VCFs.

Note

In ID (small insertion and deletion) 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

[illegible]

StrelkaSBSVCFFilesToCatalog

Create SBS and DBS catalogs from Strelka SBS VCF files.

Description

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

Usage

```
StrelkaSBSVCFFilesToCatalog(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL
)
```

Arguments

<code>files</code>	Character vector of file paths to the Strelka SBS VCF files.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>trans.ranges</code>	A data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details.
<code>region</code>	A character string designating a genomic region; see as.catalog and ICAMS .
<code>names.of.VCFs</code>	Character vector of names of the VCF files. The order of names in <code>names.of.VCFs</code> should match the order of VCF file paths in <code>files</code> . If <code>NULL</code> (default), this function 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.

Details

This function calls [VCFsToSBSCatalogs](#) and [VCFsToDBSCatalogs](#).

Value

A list of 3 SBS catalogs (one each for 96, 192, and 1536) and 3 DBS catalogs (one each for 78, 136, and 144). If `trans.ranges = NULL`, SBS 192 and 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

SBS 192 and DBS 144 catalog only contains mutations in transcribed regions.

Examples

```
file <- c(system.file("extdata",
                     "Strelka.SBS.GRCh37.vcf",
                     package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <- StrelkaSBSVCFFilesToCatalog(file, ref.genome = "hg19",
                                          trans.ranges = trans.ranges.GRCh37,
                                          region = "genome")}
```

StrelkaSBSVCFFilesToCatalogAndPlotToPdf

Create SBS and DBS catalogs from Strelka SBS VCF files and plot them to PDF

Description

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

Usage

```
StrelkaSBSVCFFilesToCatalogAndPlotToPdf(
  files,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown",
  names.of.VCFs = NULL,
  output.file = ""
)
```

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	A data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details.
region	A character string designating a genomic region; see as.catalog and ICAMS .
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) and file paths without extensions (and the leading dot) will be used as the names of the VCF files.
output.file	The base name of the PDF files to be produced; multiple files will be generated, each ending in <i>x.pdf</i> , where <i>x</i> indicates the type of catalog plotted in the file.

Details

This function calls [StrelkaSBSVCFFilesToCatalog](#) and [PlotCatalogToPdf](#)

Value

A list of 3 SBS catalogs (one each for 96, 192, and 1536), 3 DBS catalogs (one each for 78, 136, and 144) and their graphs plotted to PDF with specified file name. If `trans.ranges = NULL`, SBS 192 and DBS 144 catalog will not be generated and plotted. 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 and DBS 144 catalogs include only mutations in transcribed regions.

Examples

```
file <- c(system.file("extdata",
                     "Strelka.SBS.GRCh37.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"))}
```

TranscriptRanges	<i>Transcript ranges data</i>
------------------	-------------------------------

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.

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	<i>Transform between counts and density spectrum catalogs and counts and density signature catalogs.</i>
------------------	--

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

<code>catalog</code>	An SBS or DBS catalog as described in ICAMS ; must not be an ID (small insertion and deletion) catalog.
<code>target.ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS . If NULL, then defaults to the <code>ref.genome</code> attribute of <code>catalog</code> .
<code>target.region</code>	A region argument; see as.catalog and ICAMS . If NULL, then defaults to the <code>region</code> attribute of <code>catalog</code> .
<code>target.catalog.type</code>	A character string acting as a catalog type identifier, one of "counts", "density", "counts.signature", "density.signature"; see as.catalog . If NULL, then defaults to the <code>catalog.type</code> attribute of <code>catalog</code> .
<code>target.abundance</code>	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 – context, one count each for ACA, ACC, ACG, ... TTT). See all.abundance . If NULL, the function tries to infer <code>target.abundance</code> from the class of <code>catalog</code> and the value of the <code>target.ref.genome</code> , <code>target.region</code> , and <code>target.catalog.type</code> . If the <code>target.abundance</code> can be inferred and is different from a supplied non-NULL value of <code>target.abundance</code> , raise an error.

Details

Only the following transformations are legal:

1. `counts -> counts` (used to transform between the source abundance and `target.abundance`)
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](#).

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")}
```

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 than DBSs.

Usage

```
VCFsToDBSCatalogs(
  list.of.DBS.vcfs,
  ref.genome,
  trans.ranges = NULL,
  region = "unknown"
)
```

Arguments

<code>list.of.DBS.vcfs</code>	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.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>trans.ranges</code>	A data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details.
<code>region</code>	A character string designating a genomic region; see as.catalog and ICAMS .

Value

A list of 3 DBS catalogs, one each for 78, 144, 136: `catDBS78 catDBS144 catDBS136`. If `trans.ranges = NULL`, 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.vcf",
                     package = "ICAMS"))
list.of.DBS.vcfs <- ReadAndSplitMutectVCFs(file)$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 insertion and deletion) catalog from ID VCFs

Description

Create ID (small insertion and deletion) catalog from ID VCFs

Usage

```
VCFsToIDCatalogs(list.of.vcfs, ref.genome, region = "unknown")
```

Arguments

list.of.vcfs	List of in-memory VCFs. The list names will be the sample ids in the output catalog.
ref.genome	A ref.genome argument as described in ICAMS .
region	A character string acting as a region identifier, one of "genome", "exome".

Value

A list of two elements. 1st element is an S3 object containing an ID (small insertion and deletion) catalog with class "IndelCatalog". See [as.catalog](#) for more details. 2nd element is a list of further annotated VCFs.

Note

In ID (small insertion and deletion) 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 <- c(system.file("extdata",
                     "Strelka.ID.GRCh37.vcf",
                     package = "ICAMS"))
list.of.ID.vcfs <- ReadStrelkaIDVCFs(file)
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,
  trans.ranges = NULL,
  region = "unknown"
)
```

Arguments

<code>list.of.SBS.vcfs</code>	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.
<code>ref.genome</code>	A <code>ref.genome</code> argument as described in ICAMS .
<code>trans.ranges</code>	A data.table which contains transcript range and strand information. Please refer to TranscriptRanges for more details.
<code>region</code>	A character string designating a genomic region; see as.catalog and ICAMS .

Value

A list of 3 SBS catalogs, one each for 96, 192, 1536: `catSBS96 catSBS192 catSBS1536`. If `trans.ranges = NULL`, 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.vcf",
  package = "ICAMS"))
list.of.SBS.vcfs <- ReadAndSplitMutectVCFs(file)$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")}
```

WriteCatalog	<i>Write a catalog</i>
--------------	------------------------

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 insertion and deletion) 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

*Topic **datasets**

- all.abundance, [3](#)
- CatalogRowOrder, [8](#)
- GeneExpressionData, [13](#)
- TranscriptRanges, [36](#)
- all.abundance, [3](#), [6](#), [15](#), [16](#), [38](#)
- AnnotateDBSVCF, [3](#)
- AnnotateIDVCF, [4](#)
- AnnotateSBSVCF, [5](#), [25](#), [26](#)
- as.catalog, [6](#), [15](#), [18–23](#), [29](#), [30](#), [32–36](#), [38–42](#)
- attr, [18](#), [20](#), [21](#), [30](#), [34](#), [36](#), [39](#), [41](#)
- available.genomes, [16](#)
- BSgenome, [16](#)
- BSgenome.Hsapiens.1000genomes.hs37d5, [16](#)
- BSgenome.Hsapiens.UCSC.hg38, [16](#)
- BSgenome.Mmusculus.UCSC.mm10, [16](#)
- Canonicalize1Del, [7](#)
- CanonicalizeID, [7](#)
- catalog.row.order (CatalogRowOrder), [8](#)
- CatalogRowOrder, [6](#), [8](#), [17](#)
- Collapse144CatalogTo78 (CollapseCatalog), [8](#)
- Collapse1536CatalogTo96 (CollapseCatalog), [8](#)
- Collapse192CatalogTo96 (CollapseCatalog), [8](#)
- CollapseCatalog, [8](#), [17](#)
- data.table, [3](#), [5](#), [13](#), [18](#), [19](#), [21](#), [25](#), [26](#), [34–36](#), [39](#), [41](#)
- FindDelMH, [7](#), [9](#), [12](#)
- FindMaxRepeatDel, [7](#), [12](#)
- gene.expression.data.HepG2 (GeneExpressionData), [13](#)
- gene.expression.data.MCF10A (GeneExpressionData), [13](#)
- GeneExpressionData, [13](#), [17](#), [25](#), [26](#)
- GetMutectVAF, [18](#), [19](#), [21](#), [28](#)
- GetMutectVAF (GetVAF), [13](#)
- GetStrelkaVAF (GetVAF), [13](#)
- GetVAF, [13](#)
- glm, [25](#), [27](#)
- ICAMS, [3–6](#), [9](#), [14](#), [18](#), [19](#), [21–23](#), [29](#), [32–35](#), [38–42](#)
- MutectVCFFilesToCatalog, [15](#), [17](#), [19](#), [21](#), [28](#), [37](#)
- MutectVCFFilesToCatalogAndPlotToPdf, [16](#), [19](#), [37](#)
- MutectVCFFilesToZipFile, [20](#)
- PlotCatalog, [15](#), [22](#)
- PlotCatalogToPdf, [15](#), [19](#), [21](#), [23](#), [33](#), [35](#)
- PlotTransBiasGeneExp, [13](#), [24](#)
- PlotTransBiasGeneExpToPdf, [13](#), [26](#)
- ReadAndSplitMutectVCFs, [27](#)
- ReadAndSplitStrelkaSBSVCFs, [28](#)
- ReadCatalog, [15](#), [16](#), [29](#), [42](#)
- ReadStrelkaIDVCFs, [30](#)
- revc, [31](#)
- reverseComplement, [31](#)
- StrelkaIDVCFFilesToCatalog, [15](#), [30](#), [31](#), [33](#)
- StrelkaIDVCFFilesToCatalogAndPlotToPdf, [16](#), [32](#)
- StrelkaSBSVCFFilesToCatalog, [15](#), [29](#), [34](#), [35](#), [37](#)
- StrelkaSBSVCFFilesToCatalogAndPlotToPdf, [16](#), [35](#), [37](#)
- trans.ranges.GRCh37 (TranscriptRanges), [36](#)
- trans.ranges.GRCh38 (TranscriptRanges), [36](#)
- trans.ranges.GRCm38 (TranscriptRanges), [36](#)
- TranscriptRanges, [3](#), [5](#), [17–19](#), [21](#), [34](#), [35](#), [36](#), [39](#), [41](#)
- TransformCatalog, [15](#), [16](#), [22](#), [23](#), [37](#)

VCFsToDBSCatalogs, [18](#), [34](#), [37](#), [39](#)

VCFsToIDCatalogs, [18](#), [32](#), [40](#)

VCFsToSBSCatalogs, [18](#), [34](#), [37](#), [41](#)

WriteCatalog, [16](#), [21](#), [29](#), [42](#)

zip, [21](#)