Package 'ICAMS'

September 27, 2019

Type Package

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

Version 2.0.9.9007

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 6.1.1

Suggests testthat,
BSgenome.Hsapiens.1000genomes.hs37d5,
BSgenome.Hsapiens.UCSC.hg38,
BSgenome.Mmusculus.UCSC.mm10

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)
```

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` 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` *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)
```

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`	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

Examples

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

`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)
```

Arguments

`SBS.vcf`	An in-memory SBS 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 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` 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)
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)}
```

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 than 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` *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 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 *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.

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 (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 (insertion and deletion) 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 *"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)
```

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.

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 this case 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
```

`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.

Examples

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

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

Description

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

Usage

```
GetStrelkaVAF(vcf)

GetMutectVAF(vcf)
```

Arguments

vcf	Said VCF as a data.frame.

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 ID (indel) catalog from the Strelka ID VCFs.
3. `MutectVCFFilesToCatalog` creates 3 SBS catalogs (96, 192, 1536), 3 DBS catalogs (78, 136, 144) and ID (indel) 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 (indel) 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. The rownames denote the mutation types. 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.

`ICAMS.to.SigPro.ID` *Linker from PCAWG(ICAMS)-formatted to SigProExtractor-formatted indel names.*

Description

This data is designed for converting ICAMS-formatted indel names to SigProExtractor-formatted indel names.

Usage

```
ICAMS.to.SigPro.ID
```

Format

A 83*1 matrix. Its contents (first column) contain SigProExtractor formatted indel names in SigProExtractor order. Its rownames refer to the corresponding PCAWG(ICAMS)-formatted indel names.

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")
```

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`.

Details

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

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 ID catalog. 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.

Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (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.GRCh37.vcf",
                      package = "ICAMS"))
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  catalogs <- MutectVCFFilesToCatalog(file, ref.genome = "hg19",
                                      trans.ranges = trans.ranges.GRCh37,
                                      region = "genome")}
```

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", 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`.
`output.file`	The name of the PDF file to be produced.

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.

Note

SBS 192 and DBS 144 catalogs include only mutations in transcribed regions. In ID (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.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.pdf"))}
```

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, xlabels = 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

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"
PlotCatalog(catSBS96)
```

PlotCatalogToPdf *Plot catalog to a PDF file.*

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 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	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

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"))
```

`ReadAndSplitMutectVCFs` *Read and split Mutect VCF files.*

Description

Read and split Mutect VCF files.

Usage

```
ReadAndSplitMutectVCFs(files)
```

Arguments

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

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.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)
```

Arguments

files	Character vector of file paths to the Strelka SBS 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

file	Path to a catalog on disk in the standardized format.
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	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.

Note

In ID (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 *Read Strelka ID (insertion and deletion) VCF files.*

Description

Read Strelka ID (insertion and deletion) VCF files.

Usage

```
ReadStrelkaIDVCFs(files)
```

Arguments

files	Character vector of file paths to the VCF files.

Value

A list of vcfs from files.

Note

In ID (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)
```

SigPro.to.ICAMS.ID — *Linker from SigProExtractor-formatted to PCAWG(ICAMS)-formatted indel names.*

Description

This data is designed for converting SigProExtractor-formatted indel names to ICAMS-formatted indel names.

Usage

```
SigPro.to.ICAMS.ID
```

Format

A 83*1 matrix. Its contents (first column) contain PCAWG(ICAMS)- formatted indel names in PCAWG(ICAMS) order. Its rownames refer to the corresponding SigProExtractor indel names.

StrelkaIDVCFFilesToCatalog — *Create ID (indel) catalog from Strelka ID VCF files*

Description

Create ID (indel) catalog from the Strelka ID VCFs specified by `files`

Usage

```
StrelkaIDVCFFilesToCatalog(files, ref.genome, region = "unknown")
```

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`.

Details

This function calls `VCFsToIDCatalogs`

Value

An ID (indel) catalog with attributes added. See `as.catalog` for more details.

Note

In ID (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 (indel) catalog from Strelka ID VCF files and plot them to PDF

Description

Create ID (indel) catalog from the Strelka ID VCFs specified by `files` and plot them to PDF

Usage

```
StrelkaIDVCFFilesToCatalogAndPlotToPdf(files, ref.genome,
  region = "unknown", output.file)
```

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`.
`output.file`	The name of the PDF file to be produced.

Details

This function calls `StrelkaIDVCFFilesToCatalog` and `PlotCatalogToPdf`

Value

An ID (indel) catalog and its graph plotted to PDF with specified file name. The ID (indel) catalog has attributes added. See `as.catalog` for more details.

Note

In ID (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 <-
    StrelkaIDVCFFilesToCatalogAndPlotToPdf(file, ref.genome = "hg19",
                                           region = "genome",
```

```
                                        output.file =
                                        file.path(tempdir(), "StrelkaID.pdf"))}
```

`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")
```

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`.

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.

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", 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`.
`output.file`	The name of the PDF file to be produced.

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.

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.pdf"))}
```

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, 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 gene.name
#     1    65419    71585      +     OR4F5
#     1   367640   368634      +    OR4F29
#     1   621059   622053      -    OR4F16
#     1   859308   879961      +    SAMD11
```

```
#     1   879583   894689      -     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 (indel) 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 defaults to the `catalog.type` attribute of `catalog`.
`target.abundance`	A vector of counts different source K-mer sequences for mutations. See `all.abundance`. If `NULL`, then the function attempt to infer the `target.abundace` from the class of `catalog` and the values of the `target.ref.genome`, `target.region`, and `target.catalog.type`. It is an error if the inferred abundance is different from an non-`NULL` `target.abundance`.

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 then DBSs.

Usage

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

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.
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.

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.

Note

DBS 144 catalog only contains mutations in transcribed regions.

Examples

```
file <- c(system.file("extdata",
                      "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 (insertion and deletion) catalog from ID VCFs*

Description

Create ID (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

An S3 object containing an ID (indel) catalog with class "catalog". See `as.catalog` for more details.

Note

In ID (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.GRCh37.vcf",
                      package = "ICAMS"))
list.of.ID.vcfs <- ReadAndSplitMutectVCFs(file)$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, trans.ranges = NULL,
  region = "unknown")
```

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.
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.

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.

Note

SBS 192 catalogs only contain mutations in transcribed regions.

Examples

```
file <- c(system.file("extdata",
                      "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 *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 (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