Package 'ICAMSxtra'

January 8, 2021

Title ICAMSxtra

Version 0.0.4

Author Steve Rozen, Nanhai Jiang, Arnoud Boot, Mo Liu, Chang Jia Geng

Maintainer Steve Rozen <steverozen@gmail.com>

Description ICAMSxtra.

biocViews

Imports boot,
data.table,
ICAMS,
simpleboot

Depends R (>= 3.5)

License GPL-3

Encoding UTF-8

LazyData true

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

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

R topics documented:

`AnnotateIDVCFsWithTransRanges` *Add sequence context and transcript information to an in-memory ID VCF*

Description

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

Usage

```
AnnotateIDVCFsWithTransRanges(
  ID.vcfs,
  ref.genome,
  trans.ranges = NULL,
  vcf.names = NULL
)
```

Arguments

`ID.vcfs`	A list of in-memory ID VCF as a `data.frame`.
`ref.genome`	A `ref.genome` argument as described in `ICAMS`.
`trans.ranges`	Optional. If `ref.genome` specifies one of the `BSgenome` object 1. `BSgenome.Hsapiens.1000genomes.hs37d5`

2. BSgenome.Hsapiens.UCSC.hg38
3. BSgenome.Mmusculus.UCSC.mm10

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

vcf.names — list of names of the vcfs

Value

A list of in-memory ID VCFs each a data.table. These have been annotated with the sequence context (column name seq.context) 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-ID-vcf",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMSxtra"))
list.of.vcfs <- ICAMS::ReadStrelkaIDVCFs(file)
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.ID.vcfs <- AnnotateIDVCFsWithTransRanges(list.of.vcfs, ref.genome = "hg19",
                                                     trans.ranges = ICAMS::trans.ranges.GRCh37)
  }
```

---

as.catalog.for.ID115 *Create a catalog from a* matrix, data.frame, *or* vector

---

Description

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

Usage

```
as.catalog.for.ID115(
  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.

Canonicalize1Del115 *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

```
Canonicalize1Del115(context, del.seq, pos, trace = 0, strand)
```

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.
strand	NULL by default. But when called by PlotTransBiasInternal, strand is either + or -. The return value will include :trans or :nontrans indicating whether the deletion occurred on the transcribed or non-transcribed strand.

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.)

@keywords internal

CatalogRowOrder *Standard order of row names in a catalog*

Description

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

Usage

```
catalog.row.order
```

Format

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

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$ID115
# There are altogether 115 row names to denote the mutation types
# in ID115 catalog.
# The difference from the .$ID class in \code{\link{ICAMS::catalog.row.order}} is that
# single base nonhomopolymer indels have trinucleotide context added to them in the format
# INS/DEL:C/T:1:0_PF where P is the preceding base and F is the following base.
```

Collapse115CatalogTo83 *"Collapse" a catalog*

Description

Collapse115CatalogTo83 Collapse a ID 115 catalog to a ID 83 catalog.

Usage

```
Collapse115CatalogTo83(catalog)
```

Arguments

catalog	A catalog as defined in ICAMS.

Value

A catalog as defined in ICAMS.

CollapseID115CatalogsToID83s *"Collapse" a matrix of ID 115 catalogs to ID 83 catalog*

Description

"Collapse" a matrix of ID 115 catalogs to ID 83 catalog

Usage

```
CollapseID115CatalogsToID83s(catalogs)
```

Arguments

catalogs	A catalog as defined in ICAMS.

Value

A catalog as defined in ICAMS.

GRCh37.proportions *Genic/intergenic size and proportions for H.sapiens BSgenome GRCh37*

Description

This data is designed to be used in function PlotTranscriptionAssociatedDamageToPdf

Usage

```
GRCh37.proportions
```

Format

A list of 5 numbers with the names:

1. total.bp 2. coding.bp 3. noncoding.bp 4. prop.coding and 5. prop.noncoding

GRCh38.proportions *Genic/intergenic size and proportions for H.sapiens BSgenome GRCh38*

Description

This data is designed to be used in function PlotTranscriptionAssociatedDamageToPdf

Usage

```
GRCh38.proportions
```

Format

A list of 5 numbers with the names:

1. total.bp 2. coding.bp 3. noncoding.bp 4. prop.coding and 5. prop.noncoding

Match1Sig *Find signatures in* other.sigs *with the highest cosine similarity to* query.sig.

Description

Find signatures in other.sigs with the highest cosine similarity to query.sig.

Usage

```
Match1Sig(query.sig, other.sigs)
```

Arguments

query.sig	A single signature.
other.sigs	Matrix with each column being one signature.

Value

The maximum similarity between query.sig and any signature in other.sigs; the name of the single element in the vector is the name of a signature with the maximum similarity.

MatchSigs1Direction — *Find the closest match in* other.sigs *for each signature in* query.sigs

Description

Find the closest match in other.sigs for each signature in query.sigs

Usage

```
MatchSigs1Direction(query.sigs, other.sigs)
```

Arguments

query.sigs	A signature matrix; signatures for which to find the closest match in other.sigs. The colnames are used as the identifiers of the signatures.
other.sigs	A signature matrix; find the closest matches to a signature in this matrix. The colnames are used as the identifiers of the signatures.

Value

A list with one element for each signature in query.sigs. The names of the list elements are the colnames of query.sigs. Each list element is a vector of length 1, and the name of the vector element is the name of the closest matching signature in other.sigs, and the value is the cosine similarity between the given signature in query.sigs and the matching signature in other.sigs.

MatchSigs2Directions — *Bidirectional closest similarities between two sets of signatures.*

Description

Bidirectional closest similarities between two sets of signatures.

Usage

```
MatchSigs2Directions(sigs1, sigs2)
```

Arguments

sigs1	Matrix of signatures; colnames are used as signature identifiers, and the colnames in sigs1 should be distinguishable from those in sigs2.
sigs2	Matrix of signatures; colnames are used as signature identifiers.

Value

A list with the elements:

averCosSim: the average of the cosine similarities between each signature in sigs1 and its closest match in sigs2 and the closest match between each signature in sigs2 and its closest match in sigs1.

match1: a data frame with rownames being signature identifiers from sigs1, the signature identifier of the closest match in sigs1 in the 1st column, and the cosine similarity between them in the 2nd column.

match2: a data frame with the rownames being signature identifiers from sigs2, the signature identifier of the closest match in sigs1 in the 1st column, and the cosine similarity between them in the 2nd column.

match1 and match2 might not have the same number of rows.

Examples

```
seta <- matrix(c(1, 3,   4, 1, 2, 4), ncol = 2)
setb <- matrix(c(1, 3.1, 4, 5, 1, 1, 1, 2.8, 4), ncol = 3)
colnames(seta) <- c("A.1", "A.2")
colnames(setb) <- c("B.1", "B.2", "B.3")
MatchSigs2Directions(seta, setb)
```

MatchSigsAndRelabel *A somewhat asymmetrical analysis of a set of "ground truth" and "extracted" signatures.*

Description

A somewhat asymmetrical analysis of a set of "ground truth" and "extracted" signatures.

Usage

```
MatchSigsAndRelabel(ex.sigs, gt.sigs, exposure = NULL)
```

Arguments

ex.sigs	Newly extracted signatures to be compared to gt.sigs
gt.sigs	"Ground truth" signatures.
exposure	If NULL, then match ex.sigs against all signatures in gt.sigs. Otherwise this should be ground-truth exposures used generate the synthetic spectra from which ex.sigs were extracted. In this case we do not match to ground-truth signatures to that were not in the ground truth exposure.

Value

A list with the elements averCosSim, match1, match2 as for MatchSigs2Directions, with match1 being matches for the the extracted signatures (ex.sigs) and match2 being the matches for the ground truth signatures (gt.sigs). The return list also echos the input arguments ex.sigs and gt.sigs.

Examples

```
gt.sigs <- matrix(c(1, 3,   4, 1, 2, 4), ncol = 2)
ex.sigs <- matrix(c(1, 3.1, 4, 5, 1, 1, 1, 2.8, 4), ncol = 3)
colnames(gt.sigs) <- c("gt.1", "gt.2")
colnames(ex.sigs) <- c("ex.1", "ex.2", "ex.3")
tout <- MatchSigsAndRelabel(gt.sigs = gt.sigs, ex.sigs = ex.sigs)
tout
```

MeanOfSpectraAsSig *Return the mean of multiple spectra as a signature*

Description

Return the mean of multiple spectra as a signature

Usage

```
MeanOfSpectraAsSig(
  spectra,
  mean.weighted = TRUE,
  title = "sig.from.spectra.mean"
)
```

Arguments

spectra	An ICAMS spectrum catalog. Convert each spectrum to a signature and then compute the mean.
mean.weighted	Logical. Whether to weigh the samples according to the number of mutations in them to calculate the weighted mean as the consensus signature. Default is TRUE. If FALSE, then arithmetic mean will be calculated.
title	The name of the output signature.

Value

The mean of the spectra as a signature and the constituent spectra as signatures.

NumFromId *Get the numerical parts of identifiers.*

Description

Get the numerical parts of identifiers.

Usage

```
NumFromId(s)
```

Arguments

s	A character vector.

Details

Not very sophisticated.

Value

A vector, each element of which is the integer corresponding to the first string of digits of an element of s.

Examples

```
x<- c("SBS22", "SBS2", "SBS7b", "SBS7a")
NumFromId(x)
x[order(NumFromId(x))]
```

PlotExposure *Plot exposures in multiple plots each with a manageable number of samples*

Description

Plot exposures in multiple plots each with a manageable number of samples

Usage

```
PlotExposure(
  exposure,
  samples.per.line = 30,
  plot.proportion = FALSE,
  xlim = NULL,
  ylim = NULL,
  legend.x = NULL,
  legend.y = NULL,
  cex.legend = 0.9,
  ...
)
```

Arguments

exposure	Exposures as a numerical `matrix` (or `data.frame`) with signatures in rows and samples in columns. Rownames are taken as the signature names and column names are taken as the sample IDs. If you want `exposure` sorted from largest to smallest, use `SortExposure`. Do not use column names that start with multiple underscores. The exposures will often be mutation counts, but could also be e.g. mutations per megabase.
samples.per.line	Number of samples to show in each plot.

`plot.proportion`
: Plot exposure proportions rather than counts.

`xlim, ylim`
: Limits for the x and y axis. If `NULL`(default), the function tries to do something reasonable.

`legend.x, legend.y`
: The x and y co-ordinates to be used to position the legend.

`cex.legend`
: A numerical value giving the amount by which legend plotting text and symbols should be magnified relative to the default.

`...`
: Other arguments passed to `barplot`. If `ylab` is not included, it defaults to a value depending on `plot.proportion`. If `col` is not supplied the function tries to do something reasonable.

Value

An **invisible** 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.

Examples

```
file <- system.file("extdata",
                    "synthetic.exposure.csv",
                    package = "ICAMSxtra")
exposure <- ReadExposure(file)
PlotExposure(exposure[, 1:30])
```

PlotExposureToPdf *Plot exposures in multiple plots each with a manageable number of samples to PDF*

Description

Plot exposures in multiple plots each with a manageable number of samples to PDF

Usage

```
PlotExposureToPdf(
  exposure,
  file,
  mfrow = c(2, 1),
  mar = c(6, 4, 3, 2),
  oma = c(3, 2, 0, 2),
  samples.per.line = 30,
  plot.proportion = FALSE,
  xlim = NULL,
  ylim = NULL,
  legend.x = NULL,
  legend.y = NULL,
  cex.legend = 0.9,
  ...
)
```

Arguments

exposure	Exposures as a numerical matrix (or data.frame) with signatures in rows and samples in columns. Rownames are taken as the signature names and column names are taken as the sample IDs. If you want exposure sorted from largest to smallest, use SortExposure. Do not use column names that start with multiple underscores. The exposures will often be mutation counts, but could also be e.g. mutations per megabase.
file	The name of the PDF file to be produced.
mfrow	A vector of the form c(nr,nc). Subsequent figures will be drawn in an nr-by-nc array on the device by rows.
mar	A numerical vector of the form c(bottom,left,top,right) which gives the number of lines of margin to be specified on the four sides of the plot.
oma	A vector of the form c(bottom,left,top,right) giving the size of the outer margins in lines of text.
samples.per.line	Number of samples to show in each plot.
plot.proportion	Plot exposure proportions rather than counts.
xlim, ylim	Limits for the x and y axis. If NULL(default), the function tries to do something reasonable.
legend.x, legend.y	The x and y co-ordinates to be used to position the legend.
cex.legend	A numerical value giving the amount by which legend plotting text and symbols should be magnified relative to the default.
...	Other arguments passed to barplot. If ylab is not included, it defaults to a value depending on plot.proportion. If col is not supplied the function tries to do something reasonable.

Value

An **invisible** 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.

Examples

```
file <- system.file("extdata",
                    "synthetic.exposure.csv",
                    package = "ICAMSxtra")
exposure <- ReadExposure(file)
PlotExposureToPdf(exposure, file = file.path(tempdir(), "exposure.pdf"))
```

PlotID115AsID83ToPdf *Plot an ID 115 signatures (default) or catalog as standard ID83 and save as pdf file*

Description

Plot an ID 115 signatures (default) or catalog as standard ID83 and save as pdf file

Usage

```
PlotID115AsID83ToPdf(catalog, file, ylim = NULL)
```

Arguments

catalog	A catalog as defined in ICAMS.
file	The name of the PDF file to be produced.
ylim	Has the usual meaning. Only implemented for SBS96Catalog and IndelCatalog.

Value

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

PlotID115Catalog *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

```
PlotID115Catalog(catalog, ylim = NULL)
```

Arguments

catalog	A catalog as defined in ICAMS with attributes added. See as.catalog for more details.
ylim	Has the usual meaning. Only implemented for SBS96Catalog and IndelCatalog.

PlotID115CatalogToPdf *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

```
PlotID115CatalogToPdf(catalog, file, ylim = 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.
ylim	Has the usual meaning. Only implemented for SBS96Catalog and IndelCatalog.

Value

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

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+.

PlotSpectraAsSigsWithUncertainty

Convert spectra to signatures and then plot mean with "error" bars

---

Description

Convert spectra to signatures and then plot mean with "error" bars

Usage

```
PlotSpectraAsSigsWithUncertainty(
  spectra,
  mean.weighted = TRUE,
  conf.int = 0.95,
  num.of.bootstrap.replicates = 10^4,
  title = "Mean.as.signature"
)
```

Arguments

spectra	An ICAMS spectrum catalog. Convert each spectrum to a signature and then compute the mean.
mean.weighted	Logical. Whether to weigh the samples according to the number of mutations in them to calculate the weighted mean as the consensus signature. Default is TRUE. If FALSE, then arithmetic mean will be calculated.
conf.int	A number specifying the required confidence interval. The error bars will be plotted as bootstrap confidence interval for the mean. If NULL, then use the maximum and minimum value of the spectra to plot error bars.
num.of.bootstrap.replicates	The number of bootstrap replicates to use. Default is 10000.
title	The name of the output signature.

Value

The mean of the spectra as a signature, the constituent spectra as signatures, and the y positions of the arrowheads.

`PlotTransBiasID115` *Plot transcription strand bias*

Description

Plot transcription strand bias

Usage

```
PlotTransBiasID115(annotated.ID.vcf, pool, damaged.base = NULL, ymax = NULL)
```

Arguments

`annotated.ID.vcf`	An ID VCF annotated by `AnnotateIDVCFsWithTransRanges`. It **must** have transcript range information added.
`pool`	if true, 36 categories will be pooled to 4 categories by removing trinucleotide context. This can be done if the counts of individual categories are too low, to increase power.
`damaged.base`	One of `NULL`, `"purine"` or `"pyrimidine"`. This function allocates approximately equal numbers of mutations from `damaged.base` into each of `num.of.bins` bin by expression level. E.g. if `damaged.base` is `"purine"`, then mutations from A and G will be allocated in approximately equal numbers to each expression-level bin. The rationale for the name `damaged.base` is that the direction of strand bias is a result of whether the damage occurs on a purine or pyrimidine. If `NULL`, the function attempts to infer the `damaged.base` based on mutation counts.
`ymax`	Limit for the y axis. If not specified, it defaults to NULL and the y axis limit equals 1.5 times of the maximum mutation counts in a specific mutation type.

Value

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

Note

The strand bias statistics are Benjamini-Hochberg q-values based on two-sided binomial tests of the mutation counts on the transcribed and untranscribed strands relative to the actual abundances of C and T on the transcribed strand. On the plot, asterisks indicate q-values as follows *, $Q < 0.05$; **, $Q < 0.01$; ***, $Q < 0.001$.

Examples

```
file <- c(system.file("extdata/Strelka-ID-vcf/",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMSxtra"))
ID.vcf <- ICAMS::ReadStrelkaIDVCFs(file)
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.ID.vcf <- AnnotateIDVCFsWithTransRanges(ID.vcf, ref.genome = "hg19",
                                                    trans.ranges = ICAMS::trans.ranges.GRCh37,
                                                       vcf.names = "Strelka.ID.GRCh37.s1.vcf")
  #' alternatively run below code to skip call to AnnotateIDVCFsWithTransRanges
  #' load(c(system.file("extdata/annotated.ID.vcf.rda",package = "ICAMSxtra")))
  PlotTransBiasID115(annotated.ID.vcf = annotated.ID.vcf[[1]],
                      pool = TRUE)
}
```

PlotTransBiasID115ToPdf

Plot transcription strand bias to a PDF file

Description

Plot transcription strand bias to a PDF file

Usage

```
PlotTransBiasID115ToPdf(annotated.ID.vcfs, file, pool, damaged.base = NULL)
```

Arguments

annotated.ID.vcfs	ID vcfs which have been annotated with AnnotateIDVCFsWithTransRanges.
file	The name of output file.
pool	if true, 36 categories will be pooled to 4 categories by removing trinucleotide context. This can be done if the counts of individual categories are too low, to increase power.
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 strand bias statistics are Benjamini-Hochberg q-values based on two-sided binomial tests of the mutation counts on the transcribed and untranscribed strands relative to the actual abundances of C and T on the transcribed strand. On the plot, asterisks indicate q-values as follows *, $Q < 0.05$; **, $Q < 0.01$; ***, $Q < 0.001$.

Examples

```
library(ICAMS)
file <- c(system.file("extdata/Strelka-ID-vcf/",
                      "Strelka.ID.GRCh37.s1.vcf",
                      package = "ICAMSxtra"))
ID.vcf <- ICAMS::ReadStrelkaIDVCFs(file)
if (requireNamespace("BSgenome.Hsapiens.1000genomes.hs37d5", quietly = TRUE)) {
  annotated.ID.vcf <- AnnotateIDVCFsWithTransRanges(ID.vcf, ref.genome = "hg19",
                                    trans.ranges = ICAMS::trans.ranges.GRCh37,
                                    vcf.names = "Strelka.ID.GRCh37.s1")
  PlotTransBiasID115ToPdf(annotated.ID.vcfs = annotated.ID.vcf,
                          file = file.path(tempdir(), "test.pdf"),
                          pool = TRUE)
}
```

PlotTranscriptionAssociatedDamageToPdf

Plot indel counts on transcribed and nontranscribed strands to pdf

---

Description

Plot indel counts on transcribed and nontranscribed strands to pdf

Usage

```
PlotTranscriptionAssociatedDamageToPdf(
  list.of.vcfs,
  ref.genome,
  names.of.vcfs,
  proportions,
  file
)
```

Arguments

`list.of.vcfs`	List of in-memory ID VCFs. The list names will be the sample ids in the output catalog.
`ref.genome`	A `ref.genome` argument as described in ICAMS.
`names.of.vcfs`	list of names of vcfs
`proportions`	The gene proportions for the genome e.g. `GRCh37.proportions` or `GRCh38.proportions`
`file`	The name of the PDF file to be produced.

Value

a list of tables of p-values for each vcf

Note

The strand bias statistics are Benjamini-Hochberg q-values based on two-sided binomial tests of the mutation counts on the transcribed and untranscribed strands relative to the actual abundances of C and T on the transcribed strand. On the plot, asterisks indicate q-values as follows *, $Q < 0.05$; **, $Q < 0.01$; ***, $Q < 0.001$.

Examples

```
dirs <- c(system.file("extdata/Strelka-ID-vcf", "Strelka.ID.GRCh37.s1.vcf", package="ICAMSxtra"),
       system.file("extdata/Strelka-ID-vcf", "Strelka.ID.GRCh37.s2.vcf", package="ICAMSxtra"))

list.of.vcfs <- ICAMS::ReadStrelkaIDVCFs(dirs, names.of.VCFs = c("s1","s2"))
PlotTranscriptionAssociatedDamageToPdf(list.of.vcfs = list.of.vcfs,
                                       ref.genome = "hg19",
                                       names.of.vcfs = c("s1","s2"),
                                       proportions = GRCh37.proportions,
                                       file = file.path(tempdir(), "test.pdf"))
```

ReadExposure *Read an exposure matrix from a file*

Description

Read an exposure matrix from a file

Usage

```
ReadExposure(file, check.names = FALSE)
```

Arguments

file	CSV file containing an exposure matrix.
check.names	Passed to read.csv. **IMPORTANT**: If TRUE this will replace the double colon in identifiers of the form <tumor_type>::<sample_id> with two periods (i.e. <tumor_type>..<sample_id>. If check.names is true, generate a warning if double colons were present.

Value

Matrix of exposures.

Examples

```
file <- system.file("extdata",
                    "synthetic.exposure.csv",
                    package = "ICAMSxtra")
exposure <- ReadExposure(file)
```

`ReadID115Catalog` *Read catalog*

Description

Read a catalog in standardized format from path.

Usage

```
ReadID115Catalog(
  file,
  ref.genome = NULL,
  region = "unknown",
  catalog.type = "counts"
)
```

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".

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+.

Reverse — *Transcription bias of indels classified into 115 categories (purine)*

Description

This data is designed to be used as an example in function PlotTransBiasID115 and PlotTransBiasID115ToPdf.

Usage

```
reverse
```

Format

A vector which contains the 115 categories of indel events, but in purine format

An object of class character of length 36.

Reverse_pooled — *Transcription bias of indels classified into 13 categories (purine)*

Description

This data is designed to be used as an example in function PlotTransBiasID115 and PlotTransBiasID115 when pool = TRUE.

Usage

```
reverse_pooled
```

Format

A vector which contains the 13 categories of indel events, standardised to purine format.

An object of class character of length 4.

SortExposure — *Sort columns of an exposure matrix from largest to smallest (or vice versa)*

Description

Sort columns of an exposure matrix from largest to smallest (or vice versa)

Usage

```
SortExposure(exposure, decreasing = TRUE)
```

Arguments

exposure	Exposures as a numerical matrix (or data.frame) with signatures in rows and samples in columns. Rownames are taken as the signature names and column names are taken as the sample IDs.
decreasing	If TRUE, sort from largest to smallest.

Value

The original exposure with columns sorted.

Examples

```
file <- system.file("extdata",
                    "synthetic.exposure.csv",
                    package = "ICAMSxtra")
exposure <- ReadExposure(file)
exposure.sorted <- SortExposure(exposure)
```

Target *Transcription bias of indels classified into 115 categories (pyrimidine)*

Description

This data is designed to be used as an example in function PlotTransBiasID115 and PlotTransBiasID115ToPdf.

Usage

```
target
```

Format

A vector which contains the 115 categories of indel events, standardised to pyrimidine format.

An object of class character of length 36.

Target_pooled *Transcription bias of indels classified into 13 categories (pyrimidine)*

Description

This data is designed to be used as an example in function PlotTransBiasID115 and PlotTransBiasID115

Usage

```
target_pooled
```

Format

A vector which contains the 13 categories of indel events, standardised to pyrimidine format.

An object of class character of length 4.

VCFsToID115Catalogs *Create ID (small insertion and deletion) catalog from ID VCFs*

Description

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

Usage

```
VCFsToID115Catalogs(
  list.of.vcfs,
  ref.genome,
  region = "unknown",
  flag.mismatches = 0
)
```

Arguments

list.of.vcfs	List of in-memory ID VCFs. The list names will be the sample ids in the output catalog.
ref.genome	A ref.genome argument as described in ICAMS.
region	A character string acting as a region identifier, one of "genome", "exome".
flag.mismatches	Optional. If > 0, then if there are mismatches to references in the ID (insertion/deletion) VCF, generate messages showing the mismatched rows and continue. Otherwise stop if there are mismatched rows. See AnnotateIDVCF for more details.

Value

A list of two elements. 1st element catalog is the ID (small insertion and deletion) catalog with attributes added. See as.catalog for more details. 2nd element annotated.vcfs is a list of data frames which contain the original VCF with three additional columns seq.context.width, seq.context and ID.class added. The category assignment of each ID mutation in VCF can be obtained from ID.class column.

Note

In ID (small 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+.

VCFsToID115CatalogsAndPlotToPdf *Read a list of vcfs and plot ID115 catalogs as pdf*

Description

Read a list of vcfs and plot ID115 catalogs as pdf

Usage

```
VCFsToID115CatalogsAndPlotToPdf(
  list.of.vcfs,
  ref.genome,
  region = "unknown",
  flag.mismatches = 0,
  file,
  ylim = NULL
)
```

Arguments

list.of.vcfs	List of in-memory ID VCFs. The list names will be the sample ids in the output catalog.
ref.genome	A ref.genome argument as described in ICAMS.
region	A character string acting as a region identifier, one of "genome", "exome".
flag.mismatches	Optional. If > 0, then if there are mismatches to references in the ID (insertion/deletion) VCF, generate messages showing the mismatched rows and continue. Otherwise stop if there are mismatched rows. See AnnotateIDVCF for more details.
file	The name of the PDF file to be produced.
ylim	Has the usual meaning. Only implemented for SBS96Catalog and IndelCatalog.

WriteExposure *Write an exposure matrix to a file*

Description

Write an exposure matrix to a file

Usage

```
WriteExposure(exposure, file)
```

Arguments

exposure	Exposures as a numerical matrix (or data.frame) with signatures in rows and samples in columns. Rownames are taken as the signature names and column names are taken as the sample IDs.
file	File to which to write the exposure matrix (as a CSV file).

Examples

```
file <- system.file("extdata",
                    "synthetic.exposure.csv",
                    package = "ICAMSxtra")
exposure <- ReadExposure(file)
WriteExposure(exposure, file = file.path(tempdir(), "synthetic.exposure.csv"))
```

`WriteID115Catalog` *Write a catalog to a file.*

Description

Write a catalog to a file.

Usage

```
WriteID115Catalog(catalog, file, strict = TRUE)
```

Arguments

`catalog`	A catalog as defined in `ICAMS` with attributes added. See `as.catalog` for more details.
`file`	The path of the file to be written.
`strict`	If TRUE, then stop if additional checks on the input fail.

Index