Package 'seqCAT'

April 29, 2024

Title High Throughput Sequencing Cell Authentication Toolkit

Version 1.25.0

Description The seqCAT package uses variant calling data (in the form of VCF files) from high throughput sequencing technologies to authenticate and validate the source, function and characteristics of biological samples used in scientific endeavours.

Depends R (>= 3.6), GenomicRanges (>= 1.26.4), VariantAnnotation(>= 1.20.3)

Imports dplyr (>= 0.5.0), GenomeInfoDb (>= 1.13.4), ggplot2 (>= 2.2.1), grid (>= 3.5.0), IRanges (>= 2.8.2), methods, rtracklayer, rlang, scales (>= 0.4.1), S4Vectors (>= 0.12.2), stats, SummarizedExperiment (>= 1.4.0), tidyr (>= 0.6.1), utils

Suggests knitr, BiocStyle, rmarkdown, testthat, BiocManager

biocViews Coverage, GenomicVariation, Sequencing, VariantAnnotation

License MIT + file LICENCE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

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

git_branch devel

git_last_commit 9cc4004

git_last_commit_date 2023-10-24

Repository Bioconductor 3.19

Date/Publication 2024-04-28

Author Erik Fasterius [aut, cre]

Maintainer Erik Fasterius <erik.fasterius@outlook.com>

Contents

`calculate_similarity` *SNV profile similarity calculations*

Description

Calculate the similarity statistics for SNV profile comparisons.

Usage

```
calculate_similarity(data, similarity = NULL, a = 1, b = 5)
```

Arguments

`data`	The input SNV data dataframe.
`similarity`	Optional dataframe to add results to.
`a`	Similarity score parameter a (integer).
`b`	Similarity score parameter b (integer).

Details

This function calculates various summary statistics and sample similarities for a given profile comparison dataframe. It returns a small dataframe with the overall similarity score (whose parameters ‘a‘ and ‘b‘ can be adjusted in the function call), total SNV data, the concordance of the data and the sample names in question. This dataframe can also be given to the function, in which case it will simply add another row for the current samples, facilitating downstream aggregate analyses.

Value

A dataframe with summary statistics.

Examples

```
# Load test data
data(test_comparison)

# Calculate similarities
similarity <- calculate_similarity(test_comparison)

# Add another row of summary statistics
calculate_similarity(test_comparison, similarity = similarity)
```

---

compare_many *Comparisons of many SNV profiles*

Description

Overlap and compare genotypes in many SNV profiles.

Usage

```
compare_many(many, one = NULL, a = 1, b = 5)
```

Arguments

many	SNV profiles to be compared (list of dataframes).
one	SNV profile to be compared to all others (dataframe).
a	Similarity score parameter a (integer).
b	Similarity score parameter b (integer).

Details

This is a function that compares all the combinations of the SNV profiles input to it, either in a one-to-many or many-to-many manner. It returns both a dataframe containing summary statistics for all unique combinations and a list of dataframes with all the performed comparisons, for easy re-use and downstream analyses of said comparisons.

Value

A list of summary statistics and comparisons.

Examples

```
# Load test data
data(test_profile_1)
data(test_profile_2)

# Perform many-to-many comparisons
profiles <- list(test_profile_1, test_profile_2)
comparisons <- compare_many(profiles)

# View aggregate similarities
## Not run: comparisons[[1]])

# View data of first comparison
## Not run: head(comparisons[[2]][[1]])
```

---

compare_profiles *Binary SNV profile comparisons*

Description

Overlap and compare genotypes in two SNV profiles.

Usage

```
compare_profiles(profile_1, profile_2, mode = "intersection")
```

Arguments

profile_1	The first SNV profile (GRanges object).
profile_2	The second SNV profile (GRanges object).
mode	Merge profiles using "union" or "intersection" (character).

Details

This is a function for finding overlapping variants in two different SNV profiles (stored as GenomicRanges objects), followed by comparing the genotypes of the overlapping variants. The "compare_overlaps" function calls the "add_metadata" function twice in succession in order to merge the metadata for the two profiles (supplied as GRanges objects), returns the results as a dataframe, compares the genotypes of the overlapping variants using the "compare_genotypes" function and, finally, returns the final dataframe with all variant overlaps and their similarity.

Value

A dataframe.

Examples

```
# Load test data
data(test_profile_1)
data(test_profile_2)

# Compare the two profiles
comparison <- compare_profiles(test_profile_1, test_profile_2)
```

`create_profile` *SNV profile creation*

Description

Create an SNV profile from data in a VCF file.

Usage

```
create_profile(vcf_file, sample, min_depth = 10, filter_vc = TRUE,
  filter_mt = TRUE, filter_ns = TRUE, filter_gd = TRUE,
  filter_pd = FALSE)
```

Arguments

`vcf_file`	The VCF file from which the profile will be created (path).
`sample`	The sample in the VCF for which a profile will be created (character).
`min_depth`	Filter variants below this sequencing depth (integer).
`filter_vc`	Filter variants failing variant caller criteria (boolean).
`filter_mt`	Filter mitochondrial variants (boolean).
`filter_ns`	Filter non-standard chromosomes (boolean).
`filter_gd`	Filter duplicate variants at the gene-level (boolean).
`filter_pd`	Filter duplicate variants at the position-level (boolean).

Details

This function creates a SNV profile from a given VCF file by extracting the variants that pass the filtering criterias. Profile creation is performed to facilitate and accelerate the cell authentication procedures, which is especially relevant when more than one pairwise comparison will be performed on the same sample.

Value

A data frame.

Examples

```
# Path to the test VCF file
vcf_file = system.file("extdata", "test.vcf.gz", package = "seqCAT")

# Create SNV profiles
profile_1 <- create_profile(vcf_file, "sample1")
profile_1 <- create_profile(vcf_file, "sample1", min_depth = 15)
```

`create_profiles` *SNV profile creation*

Description

Create SNV profiles from all VCF files in a directory

Usage

```
create_profiles(vcf_dir, min_depth = 10, filter_vc = TRUE,
  filter_mt = TRUE, filter_ns = TRUE, filter_gd = TRUE,
  filter_pd = FALSE, pattern = NULL, recursive = FALSE)
```

Arguments

`vcf_dir`	The VCF directory from which the profiles will be created (path).
`min_depth`	Remove variants below this sequencing depth (integer).
`filter_vc`	Filter variants failing variant caller criteria (boolean).
`filter_mt`	Filter mitochondrial variants (boolean).
`filter_ns`	Filter non-standard chromosomes (boolean).
`filter_gd`	Filter duplicate variants at the gene-level (boolean).
`filter_pd`	Filter duplicate variants at the position-level (boolean).
`pattern`	Only create profiles for a subset of files corresponding to this pattern (character).
`recursive`	Find VCF files recursively in sub-directories as well (boolean).

Details

This functions is a convenience-wrapper for the ‘create_profile‘ function, which will create SNV profiles for each and every VCF file in the provided directory. The file naming scheme used is ‘<sample>.vcf‘ and will dictate the each profile’s sample name.

Value

A list of data frames.

Examples

```
# Path to the test VCF directory
vcf_dir = system.file("extdata", package = "seqCAT")

# Create SNV profiles
profiles <- create_profiles(vcf_dir, pattern = "test", recursive = TRUE)
```

`filter_duplicates` *Variant de-duplication*

Description

Filter duplicated variants.

Usage

```
filter_duplicates(data, filter_gd = TRUE, filter_pd = FALSE)
```

Arguments

`data`	The dataframe containing the variant data to be filtered.
`filter_gd`	Filter duplicate variants at the gene-level (boolean).
`filter_pd`	Filter duplicate variants at the position-level (boolean).

Details

This is a function for filtering duplicated variants either on the gene-level or the position-level.

Value

A data frame containing the filtered variants.

Examples

```
# Load test comparisons
data(test_profile_1)

# Filter variants
filtered_gene <- filter_duplicates(test_profile_1)
filtered_position <- filter_duplicates(test_profile_1, filter_pd = TRUE)
```

`filter_variants` *Variant filtering*

Description

Filter variants on several criteria.

Usage

```
filter_variants(data, min_depth = 10, filter_vc = FALSE,
  filter_mt = FALSE, filter_ns = FALSE)
```

Arguments

`data`	The dataframe containing the variant data to be filtered.
`min_depth`	Threshold for variant depth (integer).
`filter_vc`	Filter variants not passing filtering criteria (boolean).
`filter_mt`	Filter mitochondrial variants (boolean).
`filter_ns`	Filter non-standard chromosomes (boolean).

Details

This is a function for filtering SNV profiles on several criteria: sequencing depth, variant caller-specific filtering, mitochondrial variants and variants in non-standard chromosomes. Only filters by sequencing depth by default.

Value

A data frame containing the filtered variants.

Examples

```
# Load test comparisons
data(test_profile_1)

# Filter variants
filtered <- filter_variants(test_profile_1, min_depth = 15)
```

list_cosmic *List COSMIC sample names*

Description

List all available samples in the COSMIC database

Usage

```
list_cosmic(file_path)
```

Arguments

file_path	The file containing COSMIC data (path).

Details

This function lists the available sample names in the provided COSMIC file (e.g. CosmicCLP_MutantExport.tsv.gz), and takes about half the time it takes to read the full file with the read_cosmic function, making it useful for just seeing if your particular sample is listed in COSMIC or not.

Value

A vector of sample names

Examples

```
file <- system.file("extdata",
                    "subset_CosmicCLP_MutantExport.tsv.gz",
                    package = "seqCAT")
cosmic_samples <- list_cosmic(file)
```

list_variants *List known variants*

Description

List known variants present in SNV profiles

Usage

```
list_variants(profiles, known_variants)
```

Arguments

profiles	The SNV profiles to analyse (list)
known_variants	The known variants to look for (dataframe)

Details

This is a function for listing known variants present in SNV profiles. Input is a list of profiles and a dataframe of known variants, containing at least the genomic locations ("chr" and "pos"). Any additional columns will be retained.

Value

A dataframe containing the known variant genotypes in each profile.

Examples

```
# Load test data
data(test_profile_1)
data(test_profile_2)

# Create some variants to analyse
known_variants <- data.frame(chr = 1, pos = 16229, gene = "DDX11L1")

# List the known variants in each profile
profiles <- list(test_profile_1, test_profile_2)
known_variants <- list_variants(profiles, known_variants)
```

---

plot_heatmap *Plot similarity heatmap*

Description

Plot a heatmap of similarities from many-to-many SNV profile comparisons.

Usage

```
plot_heatmap(similarities, cluster = TRUE, annotate = TRUE,
  annotate_size = 9, legend = TRUE, legend_size = c(36, 8),
  limits = c(0, 50, 90, 100), text_size = 14, colour = "#1954A6")
```

Arguments

similarities	The long-format dataframe containing the data.
cluster	Cluster the samples based on similarity (boolean).
annotate	Annotate each cell with the score (boolean).
annotate_size	Text size of the annotations (numeric).
legend	Show a legend for the colour gradient (boolean).
legend_size	Height and width of the legend (vector of two integers).
limits	The limits for the colour gradient (vector of four integers).
text_size	Text size for axes, labels and legend (numeric).
colour	The main colour to use for the gradient (character).

Details

This function creates publication-ready plots of heatmaps for many-to-many sample comparisons, taking a long-format dataframe containing the summary statistics of each comparison as input.

Value

A ggplot2 graphical object.

Examples

```
# Load test similarities
data(test_similarities)

# Plot a similarity heatmap
heatmap <- plot_heatmap(test_similarities)
```

---

plot_impacts *Plot SNV impact distribution*

Description

Plot SNV impact distributions for a binary SNV profile comparison.

Usage

```
plot_impacts(comparison, legend = TRUE, annotate = TRUE,
  annotate_size = 9, text_size = 14, palette = c("#0D2D59",
  "#1954A6"))
```

Arguments

comparison	The SNV profile comparison to be plotted.
legend	Show the legend (boolean).
annotate	Annotate each category (boolean).
annotate_size	Text size for annotations (numeric).
text_size	Text size for axes, ticks and legend (numeric).
palette	Colour palette for filling of bars (character vector).

Details

This function creates publication-ready plots of the impact distribution from a binary dataset comparison across the matched/mismatched SNVs.

Value

A ggplot2 graphical object.

Examples

```
# Load test comparison data
data(test_comparison)

# Plot the impact distribution
impacts <- plot_impacts(test_comparison)
```

`plot_variant_list` *Plot known variants list*

Description

Plot a genotype grid from a list of known variants

Usage

```
plot_variant_list(variant_list, legend = TRUE, legend_size = 22,
  text_size = 14, palette = c("#4e8ce4", "#a6c6f2", "#999999",
  "#cccccc"))
```

Arguments

`variant_list`	The data containing the variants (dataframe)
`legend`	Show a legend for the genotype colours (boolean)
`legend_size`	Size of the legend (numeric).
`text_size`	Text size for axes and legend (numeric).
`palette`	Nucleotide colour palette (4-element character vector)

Details

This function creates publication-ready plots from lists of known variants, taking a dataframe containing all the genotypes (on "A1/A2" format) for each sample (columns) and variant (row names).

Value

A ggplot2 graphical object.

Examples

```
# Load test variant list
data(test_variant_list)

# Plot each variant's genotype per sample
genotype_grid <- plot_variant_list(test_variant_list)
```

`read_cosmic` *Read COSMIC data*

Description

Read COSMIC sample-specific mutational data.

Usage

```
read_cosmic(file_path, sample_name = NULL, primary_site = NULL)
```

Arguments

`file_path`	The COSMIC data file path (path).
`sample_name`	Subset the data on sample name (character).
`primary_site`	Subset the data on primary tumour site (character).

Details

This function reads the COSMIC data files (e.g. "CosmicCLP_MutantExport.tsv.gz") and returns a GRanges object with all the listed mutations for the specified sample, which can then be use in downstream profile comparisons. Only non-duplicated (gene-level) SNVs are included in COSMIC profiles.

Value

A dataframe with COSMIC SNVs.

Examples

```
# Path to COSMIC test data
file <- system.file("extdata",
                    "subset_CosmicCLP_MutantExport.tsv.gz",
                    package = "seqCAT")

# Read COSMIC test data for the HCT116 cell line
cosmic_hct116 <- read_cosmic(file, "HCT116")
```

read_profile *Read SNV profile*

Description

Read an SNV profile stored on disk.

Usage

```
read_profile(file, sample_name = NULL)
```

Arguments

file	The SNV profile to be read (path).
sample_name	Sample name to be added; overrides profile sample if it already exists (character).

Details

This is a function for reading SNV profiles created from VCF files that were stored on disk.

Value

A data frame.

Examples

```
# Path to test data
profile = system.file("extdata",
                      "test_1.profile.txt.gz",
                      package = "seqCAT")

# Read test profile
profile <- read_profile(profile)
```

read_profiles *Read SNV profiles*

Description

Read SNV profiles in a directory.

Usage

```
read_profiles(profile_dir, pattern = ".profile.txt",
  sample_names = FALSE)
```

Arguments

profile_dir	The directory containing the profiles to be read (path).
pattern	Pattern for file name or extension to be read (character).
sample_names	Add sample name based on file name; overrides profile sample if it already exists (boolean).

Details

This is a wrapper function for reading multiple SNV profiles present in a directory (and its subdirectories in recursive mode).

Value

A list of data frames.

Examples

```
# Path to test data
profile_dir = system.file("extdata", package = "seqCAT")

# Read test profiles
profile_list <- read_profiles(profile_dir, pattern = "profile.txt")
```

---

seqCAT *seqCAT: High Throughput Sequencing Cell Authentication Toolkit*

Description

The *seqCAT* package provides a number of functions for performing evaluation, characterisation and authentication of biological samples through analysis of high throughput sequencing data.

---

test_comparison *Overlapping and compared SNVs*

Description

Overlapping and compared variants from "sample1" and "sample2" originating from the example.vcf file included in the inst/extdata directory, for use in unit tests.

Usage

```
data(test_comparison)
```

Format

A dataframe with 51 rows and 39 columns:

chr chromosome

pos SNV position

DP.sample_1 total variant depth, sample 1

AD1.sample_1 allelic depth, allele 1, sample 1

AD2.sample_1 allelic depth, allele 2, sample 1

A1.sample_1 allele 1, sample 1

A2.sample_1 allele 2, sample 1

warnings.sample_1 warnings from variant calling, sample 1

DP.sample_2 total variant depth, sample 2

AD1.sample_2 allelic depth, allele 1, sample 2

AD2.sample_2 allelic depth, allele 2, sample 2

A1.sample_2 allele 1, sample 2

A2.sample_2 allele 2, sample 2

warnings.sample_2 warnings from variant calling, sample 2

sample_1 name, sample 1

sample_2 name, sample 2

match status of genotype comparison

rsID mutation ID

gene associated gene

ENSGID ensembl gene ID

ENSTID ensembl transcript ID

REF reference allele

ALT alternative allele

impact putative variant impact

effect variant effect

feature transcript feature

biotype transcript biotype

`test_profile_1` *SNV profile 1*

Description

SNV profile in GRanges format from "sample1", originating from the test_profile_1.txt in the inst/extdata directory, for use in unit tests.

Usage

```
data(test_profile_1)
```

Format

A GRanges object with 383 elements and 17 metadata columns:

rsID mutation ID, if available

gene associated gene

ENSGID ensembl gene ID

ENSTID ensembl transcript ID

REF reference allele

ALT alternative allele

impact putative variant impact

effect variant effect

feature transcript feature

biotype transcript biotype

DP total variant depth

AD1 allelic depth, allele 1

AD2 allelic depth, allele 2

A1 allele 1

A2 allele 2

warnings warnings from variant calling

sample sample name

`test_profile_2` *SNV profile 2*

Description

SNV profile in GRanges format from "sample2", originating from the test_profile_2.txt in the inst/extdata directory, for use in unit tests.

Usage

```
data(test_profile_2)
```

Format

A GRanges object with 382 elements and 17 metadata columns:

rsID mutation ID, if available

gene associated gene

ENSGID ensembl gene ID

ENSTID ensembl transcript ID

REF reference allele

ALT alternative allele

impact putative variant impact

effect variant effect

feature transcript feature

biotype transcript biotype

DP total variant depth

AD1 allelic depth, allele 1

AD2 allelic depth, allele 2

A1 allele 1

A2 allele 2

warnings warnings from variant calling

sample sample name

test_profile_3 *SNV profile 3*

Description

SNV profile in GRanges format from "sample3", originating from the test_profile_3.txt in the inst/extdata directory, for use in unit tests.

Usage

```
data(test_profile_3)
```

Format

A GRanges object with 99 elements and 9 metadata columns:

rsID mutation ID, if available

REF reference allele

ALT alternative allele

DP total variant depth

AD1 allelic depth, allele 1

AD2 allelic depth, allele 2

A1 allele 1

A2 allele 2

sample sample name

test_similarities *Collated similarities object*

Description

Collated similarities of multiple sample comparisons from "sample1" and "sample" from the example.vcf file, for use in unit tests.

Usage

```
data(test_similarities)
```

Format

A dataframe with 3 rows and 6 columns:

sample_1 name of sample 1

sample_2 name of sample 2

overlaps the number of overlaps for the comparison

matches the number of matches for the comparison

concordance the concordance of the profiles

similarity_score the similarity score of the profiles

---

test_variant_list *Modified variant list object*

Description

A variant list object from the ‘list_variants‘ function, where the row names have been defined as "chr: pos (gene)" and the corresponding columns removed, for use in plotting.

Usage

```
data(test_variant_list)
```

Format

A dataframe with 2 rows and 2 columns:

sample1 the genotypes of sample1

sample2 the genotypes of sample2

---

write_profile *Write SNV profile*

Description

Write an SNV profile to a file for later re-use.

Usage

```
write_profile(profile, file)
```

Arguments

profile	The SNV profile to be written (data frame).
file	The file to write to (path).

Details

This is a function for writing SNV profiles (created from VCF files) to disk for later re-use. Several formats are allowed, including BED, GTF, GFF and normal text files, which are automatically recognised based on the supplied filename.

Value

None; writes to disk only.

Examples

```
# Load test profile
data(test_profile_1)

# Write test profile to file
write_profile(test_profile_1, "test_profile_1.txt")
```

---

write_profiles *Write SNV profiles*

Description

Write several SNV profiles to file for later re-use.

Usage

```
write_profiles(profile_list, format = "TXT", directory = "./")
```

Arguments

profile_list	The SNV profiles to be written (list).
format	The desired file format (character).
directory	The directory to write to (path).

Details

This is a wrapper function for writing multiple SNV profiles present in a directory (and its sub-directories in recursive mode).

Value

None; writes to disk only.

Examples

```
# Load test profiles
data(test_profile_1)
data(test_profile_2)
profiles <- list(test_profile_1, test_profile_2)

# Write test profile to file
write_profiles(profiles, format = "TXT", directory = "./")
```

Index