Package ‘deconstructSigs’

August 29, 2016

Type Package

Title Identifies Signatures Present in a Tumor Sample

Version 1.8.0

Date 2016-07-26

Author Rachel Rosenthal

Maintainer Rachel Rosenthal `<rachel.rosenthal.14@ucl.ac.uk>`

Description Takes sample information in the form of the fraction of mutations in each of 96 trinucleotide contexts and identifies the weighted combination of published signatures that, when summed, most closely reconstructs the mutational profile.

License GPL (>= 2)

Imports reshape2, BSgenome, BSgenome.Hsapiens.UCSC.hg19, GenomeInfoDb, grDevices, graphics, utils

Suggests VariantAnnotation

LazyData yes

URL `https://github.com/raerose01/deconstructSigs`

BugReports `https://github.com/raerose01/deconstructSigs/issues`

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-29 18:27:06

R topics documented:

deconstructSigs 2
example.output 2
makePie 3
mut.to.sigs.input 4
plotSignatures 5
plotTumor 5

randomly.generated.tumors 6
sample.mut.ref 6
signatures.cosmic 7
signatures.nature2013 7
tri.counts.exome 7
tri.counts.genome 8
vcf.to.sigs.input 8
whichSignatures 9

Index **11**

`deconstructSigs` *deconstructSigs*

Description

Takes sample information in the form of the fraction of mutations in each of 96 trinucleotide contexts and identifies the weighted combination of published signatures that, when summed, most closely reconstructs the mutational profile.

Main functions

- `whichSignatures`
- `mut.to.sigs.input`
- `plotSignatures`
- `makePie`

Author(s)

Rachel Rosenthal rachel.rosenthal.14@ucl.ac.uk

`example.output` *Example output of whichSignatures()*

Description

A list that was generated from running whichSignatures(). It contains the following:

Format

A list of four elements

Details

- weights - weight of each signature found in the sample
- tumor - tricontext fractions that were used as input
- product - product of the weights by the reference signatures used
- diff - difference between the tumor tricontexts fractions used as input and those generated as the product
- unknown - the fraction of the tumor profile that could not be assigned to a signature

---

`makePie` *Plots the weights from whichSignatures()*

Description

Uses the output from whichSignatures() and creates a pie chart of the weights outputted

Usage

```
makePie(sigs.output, sub = "", add.color = NULL)
```

Arguments

`sigs.output`	The list output from whichSignatures()
`sub`	A character vector that specifies cancer subtype for plot title, if wanted
`add.color`	Optional character vector to assign additional colors for novel signatures

Value

Plots a pie chart of the weights calculated in the given tumor sample

Examples

```
makePie(example.output)
```

`mut.to.sigs.input` *Converts mutation list to correct input format*

Description

Given a mutation list, outputs a data frame with counts of how frequently a mutation is found within each trinucleotide context per sample ID. Output can be used as input into getTriContextFraction.

Usage

```
mut.to.sigs.input(mut.ref, sample.id = "Sample", chr = "chr", pos = "pos",
  ref = "ref", alt = "alt", bsg = NULL)
```

Arguments

`mut.ref`	Location of the mutation file that is to be converted or name of data frame in environment
`sample.id`	Column name in the mutation file corresponding to the Sample ID
`chr`	Column name in the mutation file corresponding to the chromosome
`pos`	Column name in the mutation file corresponding to the mutation position
`ref`	Column name in the mutation file corresponding to the reference base
`alt`	Column name in the mutation file corresponding to the alternate base
`bsg`	Only set if another genome build is required. Must be a BSgenome object.

Details

The context sequence is taken from the BSgenome.Hsapiens.UCSC.hgX::Hsapiens object. Therefore the coordinates must correspond to the human hgX assembly. Default is set to the UCSC hg19 assembly, which corresponds to the GRCh37 assembly. If another assembly is required, it must already be present in the R workspace and fed as a parameter. This method will translate chromosome names from other versions of the assembly like NCBI or Ensembl. For instance, the following transformation will be done: "1" -> "chr1"; "MT" -> "chrM"; "GL000245.1" -> "chrUn_gl000245"; etc.

Value

A data frame that contains sample IDs for the rows and trinucleotide contexts for the columns. Each entry is the count of how many times a mutation with that trinucleotide context is seen in the sample.

Examples

```
## Not run:
sigs.input = mut.to.sigs.input(mut.ref = sample.mut.ref, sample.id = "Sample",
chr = "chr", pos = "pos", ref = "ref", alt = "alt", bsg =
BSgenome.Hsapiens.UCSC.hg19)

## End(Not run)
```

`plotSignatures` *Plots the result from whichSignatures()*

Description

Uses the output from whichSignatures() and creates a plot of the given tumor mutational spectrum and the calculated one

Usage

```
plotSignatures(sigs.output, sub = "")
```

Arguments

`sigs.output`	The list output from whichSignatures()
`sub`	A character vector that specifies cancer subtype for plot title, if wanted

Value

Plots the trinucleotide frequency for the given tumor on the top panel, the calculated one on the middle panel, and the difference between the two on the bottom panel.

Examples

```
plotSignatures(example.output, sub = "example")
```

`plotTumor` *Plots a tumor profile*

Description

Uses the output from whichSignatures() and creates a plot of the given tumor mutational spectrum and the calculated one

Usage

```
plotTumor(tumor, sub = "")
```

Arguments

`tumor`	A tumor matrix
`sub`	A character vector that specifies cancer subtype for plot title, if wanted

Value

Plots the trinucleotide frequency for the given tumor

Examples

```
plotTumor(example.output[['tumor']], sub = "example")
```

`randomly.generated.tumors` *Values for 100 randomly generated tumor samples*

Description

A dataset containing the value for the 96-trinucleotide contexts of 100 tumors generated by a random linear combination of the Nature 2013 signatures.

Format

A data frame of 100 rows and 96 columns

`sample.mut.ref` *Example input to mut.to.sigs.input()*

Description

A data frame containing example mutations that can be used as input to mut.to.sigs.input(). Data comes from two TCGA LUAD patients (http://cancergenome.nih.gov/) Contains the following columns:

Format

A data frame of 657 rows and 5 column that contains example mutations which could be used in mut.to.sigs.input()

Details

- Sample - sample name
- chr - chromosome number
- pos - chromosome position
- ref - reference base
- alt - alternate base

Source

http://cancergenome.nih.gov/

`signatures.cosmic` *Published Signatures from Sanger COSMIC*

Description

A dataset containing the additional signatures identified read into R as a data frame. Can be used the 'signatures.ref' parameter in whichSignatures().

Format

A data frame of 30 rows and 96 columns

Source

http://cancer.sanger.ac.uk/cosmic/signatures

`signatures.nature2013` *Published Signatures from Alexandrov et al 2013*

Description

A dataset containing the published signatures from Alexandrov et al read into R as a data frame. Can be used the 'signatures.ref' parameter in whichSignatures(). Data obtained: Nature 2013 PMID:23945592

Format

A data frame of 27 rows and 96 columns

Source

http://www.ncbi.nlm.nih.gov/pubmed/23945592

`tri.counts.exome` *The counts of every trinuclotide frequency in an exome*

Description

A datset containing the number of times each trinucleotide (ex: ACA) is found in the exome region captured by sequencing. Can be used as the 'trimer.counts.loc' parameter in whichSignatures().

Format

A data frame of 32 rows and 1 column that contains the counts

`tri.counts.genome` *The counts of every trinuclotide frequency in a genome*

Description

A datset containing the number of times each trinucleotide (ex: ACA) is found in the hg19 genome. Can be used as the 'trimer.counts.loc' parameter in whichSignatures().

Format

A data frame of 32 rows and 1 column that contains the counts

`vcf.to.sigs.input` *Converts a VCF file to correct input format*

Description

Given a VCF file, outputs a data frame with counts of how frequently a mutation is found within each trinucleotide context per sample ID. Output can be used as input into getTriContextFraction.

Usage

```
vcf.to.sigs.input(vcf, bsg = NULL)
```

Arguments

`vcf`	Location of the VCF file that is to be converted
`bsg`	Only set if another genome build is required. Must be a BSgenome object.

Details

The context sequence is taken from the BSgenome.Hsapiens.UCSC.hg19::Hsapiens object, therefore the coordinates must correspond to the human hg19 assembly, the UCSC version of the GRCh37 Homo sapiens assembly. This method will to its best to translate chromosome names from other versions of the assembly like NCBI or Ensembl. For instance, the following transformation will be done: "1" -> "chr1"; "MT" -> "chrM"; "GL000245.1" -> "chrUn_gl000245"; etc.

This method relies on the VariantAnnotation package to read the VCF file.

Value

A data frame that contains sample IDs for the rows and trinucleotide contexts for the columns. Each entry is the count of how many times a mutation with that trinucleotide context is seen in the sample.

Examples

```
## Not run:
sigs.input = vcf.to.sigs.input(vcf = "variants.vcf")

## End(Not run)
```

`whichSignatures` *Which signatures are present*

Description

Determines how much of each signature is present in the sample given

Usage

```
whichSignatures(tumor.ref = NA, sample.id,
  signatures.ref = signatures.nature2013, associated = c(),
  signatures.limit = NA, signature.cutoff = 0.06, contexts.needed = FALSE,
  tri.counts.method = "default")
```

Arguments

`tumor.ref`	Either a data frame or location of input text file, where rows are samples, columns are trinucleotide contexts
`sample.id`	Name of sample – should be rowname of tumor.ref. Optional if the tumor.ref contains one single sample
`signatures.ref`	Either a data frame or location of signature text file, where rows are signatures, columns are trinucleotide contexts
`associated`	Vector of associated signatures. If given, will narrow the signatures tested to only the ones listed.
`signatures.limit`	Number of signatures to limit the search to
`signature.cutoff`	Discard any signature contributions with a weight less than this amount
`contexts.needed`	FALSE if tumor.file is a context file, TRUE if it is only mutation counts
`tri.counts.method`	Set to either: • 'default' – no further normalization • 'exome' – normalized by number of times each trinucleotide context is observed in the exome • 'genome' – normalized by number of times each trinucleotide context is observed in the genome • 'exome2genome' – multiplied by a ratio of that trinucleotide's occurence in the genome to the trinucleotide's occurence in the exome

- 'genome2exome' – multiplied by a ratio of that trinucleotide's occurence in the exome to the trinucleotide's occurence in the genome
- data frame containing user defined scaling factor – count data for each trinucleotide context is multiplied by the corresponding value given in the data frame

Value

A list of the weights for each signatures, the product when those are multiplied on the signatures, the difference between the tumor sample and product, the tumor sample tricontext distribution given, and the unknown weight.

Normalization

If the input data frame only contains the counts of the mutations observed in each context, then the data frame must be normalized. In these cases, the value of 'contexts.needed' should be TRUE.
The parameter, 'tri.counts.method', determines any additional normalization performed. Any user provided data frames should match the format of 'tri.counts.exome' and 'tri.counts.genome'.
The method of normalization chosen should match how the input signatures were normalized. For exome data, the 'exome2genome' method is appropriate for the signatures included in this package. For whole genome data, use the 'default' method to obtain consistent results.

Examples

```
test = whichSignatures(tumor.ref = randomly.generated.tumors,
                       sample.id = "2",
                       contexts.needed = FALSE)
```

Index

∗Topic **datasets**
    example.output, 2
    randomly.generated.tumors, 6
    sample.mut.ref, 6
    signatures.cosmic, 7
    signatures.nature2013, 7
    tri.counts.exome, 7
    tri.counts.genome, 8

deconstructSigs, 2
deconstructSigs-package
        *(*deconstructSigs*)*, 2

example.output, 2

makePie, *2*, 3
mut.to.sigs.input, *2*, 4

plotSignatures, *2*, 5
plotTumor, 5

randomly.generated.tumors, 6

sample.mut.ref, 6
signatures.cosmic, 7
signatures.nature2013, 7

tri.counts.exome, 7
tri.counts.genome, 8

vcf.to.sigs.input, 8

whichSignatures, *2*, 9