

Package ‘ensemblVEP’

April 9, 2015

Version 1.6.2

Title R Interface to Ensembl Variant Effect Predictor

Author Valerie Obenchain <vobencha@fhcrc.org>,

Maintainer Valerie Obenchain <vobencha@fhcrc.org>

Depends methods, BiocGenerics, GenomicRanges, VariantAnnotation

Imports Biostrings

Suggests RUnit

Description Query the Ensembl Variant Effect Predictor via the perl API

SystemRequirements Ensembl VEP (API version 78) and the Perl package DBD::mysql must be installed. See the package README and Ensembl web site, <http://www.ensembl.org/info/docs/tools/vep/index.html> for installation instructions.

License Artistic-2.0

LazyLoad yes

biocViews Annotation

R topics documented:

| | |
|-----------------------------|----|
| ensemblVEP | 2 |
| parseCSQToGRanges | 4 |
| runtimeOptions | 5 |
| VEPParam-class | 10 |

| | |
|--------------|-----------|
| Index | 15 |
|--------------|-----------|

`ensemblVEP`*Query Ensembl Variant Effect Predictor*

Description

Retrieve variant annotation data from the Ensembl Variant Effect Predictor (VEP).

Usage

```
## S4 method for signature character
ensemblVEP(file, param=VEPParam(), ...)
```

Arguments

| | |
|--------------------|---|
| <code>file</code> | A character specifying the full path to the file, including the file name. Valid input file types are described on the Ensembl VEP web page. http://www.ensembl.org/info/docs/variation/vep/vep_script.html#running |
| <code>param</code> | An instance of <code>VEPParam</code> specifying runtime options. |
| <code>...</code> | Additional arguments passed to methods. |

Details

The Ensembl VEP tool is described in detail on the home page (link in ‘see also’ section). The `ensemblVEP` function wraps the perl API and requires a local install of the Ensembl VEP available in the user’s path. The `VEPParam` class provides a way to specify runtime options. Results are returned from Ensembl VEP as `GRanges` (default) or `VCF` objects. Alternatively, results can be written directly to a file.

Value

Default behavior returns a `GRanges` object. Options can be set to return a `VCF` object or write a file to disk.

Author(s)

Valerie Obenchain <vobencha@fhcrc.org>

References

Ensembl VEP Home: <http://www.ensembl.org/info/docs/tools/vep/index.html>

Human Genome Variation Society (hgvs): <http://www.hgvs.org/mutnomen/>

See Also

[VEPParam-class](#)

Examples

```

## -----
## Results returned as GRanges or VCF objects
## -----
## The default behavior returns a GRanges with the consequence
## data as metadata columns.
file <- system.file("extdata", "ex2.vcf", package="VariantAnnotation")
gr <- ensemblVEP(file)
gr[1:3]

## When the vcf option is TRUE, a VCF object is returned.
myparam <- VEPParam(dataformat=c(vcf=TRUE))
vcf <- ensemblVEP(file, param=myparam)
vcf

## The consequence data are returned as the CSQ column in info.
info(vcf)$CSQ

## To parse this column use parseCSQToGRanges().
csq <- parseCSQToGRanges(vcf)
head(csq, 4)

## The columns returned are controlled by the fields option.
## By default all fields are returned. See ?VEPParam for details.

## When comparing ensemblVEP() results to the data in the
## input vcf we see that variant 20:1230237 was not returned.
vcf_input <- readVcf(file, "hg19")
rowData(vcf_input)
rowData(vcf)

## This variant has no alternate allele and is called a
## monomorphic reference. The Ensembl VEP automatically
## drops these variants.
rowData(vcf)[,c("REF", "ALT")]

## -----
## Results written to disk
## -----
## Write a file to disk by providing a path and file name as output_file.
## Different output file formats are specified using the dataformat
## runtime options.

## Write a vcf file to myfile.vcf:
myparam <- VEPParam(dataformat=c(vcf=TRUE),
                    input=c(output_file="/path/myfile.vcf"))
## Write a gvf file to myfile.gvf:
myparam <- VEPParam(dataformat=c(gvf=TRUE),
                    input=c(output_file="/path/myfile.gvf"))

## -----
## Runtime options

```

```

## -----
## All runtime options are controlled by specifying a VEPParam.
## See ?VEPParam for complete details.
param <- VEPParam()

## Logical options are turned on/off with TRUE/FALSE. By
## default, quiet is FALSE.
basic(param)$quiet

## Setting quiet to TRUE will suppress all status and warnings.
basic(param)$quiet <- TRUE

## Character options are turned on/off by specifying a character
## value or an empty character (i.e., character()). By default no
## sift results are returned.
output(param)$sift

## Setting sift to b will return both predictions and scores.
output(param)$sift <- b

## Return sift to the original state of no results returned.
output(param)$sift <- character()

```

parseCSQToGRanges *Parse the CSQ column of a VCF object into a GRanges object*

Description

Parse the CSQ column in a VCF object returned from the Ensembl Variant Effect Predictor (VEP).

Usage

```

## S4 method for signature character
parseCSQToGRanges(x, VCFRowID=character(), ...)
## S4 method for signature VCF
parseCSQToGRanges(x, VCFRowID=character(), ...)

```

Arguments

| | |
|----------|---|
| x | The character name of a vcf file on disk or a VCF object |
| VCFRowID | A character vector of rownames from the original VCF. When provided, the result includes a metadata column named 'VCFRowID' which maps the result back to the row (variant) in the original VCF. When VCFRowID is not provided no 'VCFRowID' column is included. |
| ... | Arguments passed to other methods. Currently not used. |

Details

When `ensemblVEP` returns a VCF object, the consequence data are returned unparsed in the 'CSQ' INFO column. `parseCSQToGRanges` parses these data into a `GRanges` object that is expanded to match the dimension of the 'CSQ' data. Because each variant can have multiple matches, the ranges in the `GRanges` are repeated.

If rownames from the original VCF are provided as `VCFRowID` a metadata column is included in the result that maps back to the row (variant) in the original VCF.

Value

Returns a `GRanges` object with consequence data as the metadata columns.

Author(s)

Valerie Obenchain <vobencha@fhcrc.org>

References

Ensembl VEP Home: <http://uswest.ensembl.org/info/docs/tools/vep/index.html>

See Also

[ensemblVEP VEPParam-class](#)

Examples

```
file <- system.file("extdata", "ex2.vcf", package="VariantAnnotation")
vep <- ensemblVEP(file, param=VEPParam(dataformat=c(vcf=TRUE)))

## The returned CSQ data are unparsed.
info(vep)$CSQ

## Parse into a GRanges and include the VCFRowID column.
vcf <- readVcf(file, "hg19")
csq <- parseCSQToGRanges(vep, VCFRowID=rownames(vcf))
csq[1:4]
```

runtimeOptions

VEPParam runtime options

Description

Runtime options for the most current API version of the Ensembl Variant Effect Predictor.

Details

VEPParam objects store the runtime options for querying the Ensembl Variant Effect Predictor (VEP). This page describes only the most current runtime options and is a condensed version of what is listed on the Ensembl web site:

http://uswest.ensembl.org/info/docs/tools/vep/script/vep_options.html

Runtime options for archived versions can be found on the corresponding archive page.

<http://useast.ensembl.org/info/website/archives/index.html>

Runtime options:

Data in the VEPParam are organized into the following categories, 'basic', 'input', 'cache', 'output', 'identifier', 'colocatedVariants', 'dataformat', 'filterqc', 'database' and 'advanced'. Each category is a list of runtime options. logical options are turned on/off with TRUE/FALSE. character and numeric are 'on' when a character string is provided and 'off' when they contain an empty value (i.e., character() or numeric()).

'identifier', 'colocatedVariants', 'dataformat' are supported for VEPParam73 and later.

basic list of the following options:

- verbose: logical, default FALSE; output status messages
- quiet: logical, default FALSE; suppress status/warnings
- no_progress: logical, default FALSE; don't show progress bars
- config: character, default character(); name of config file
- everything: logical, default FALSE; shortcut to switch on 12 options (sift, polyphen, ccds, hgvs, hgnc, numbers, domains, regulatory, cell_type, canonical, protein and gmaf).
- fork: numeric, default numeric(); enable forking

input list of the the following options:

- species: character, default 'homo_sapiens'; species for the data
- assembly: character, default character(); select assembly version if more than one available
- format: character, default character(); one of the following input file formats, 'ensembl', 'vcf', 'pileup', 'hgvs', 'id' or 'vep'. By default the script auto-detects the input file format.
- output_file: character, default writes to temp file; path and file name of output file
- force_overwrite: logical, default FALSE; overwrite the output file if it currently exists
- stats_file: character, default character(); summary stats file name
- no_stats: logical, default FALSE; do not generate a stats file
- stats_text: logical, default FALSE; generate a plain text stats file instead of html
- html: logical, default FALSE; generate html version of the output file

cache list of the following options:

- cache: logical, default FALSE; enable use of cache
- dir: character, default '\$HOME/.vep/'; cache/plugin to be used
- dir_cache: character, default '\$HOME/.vep/'; cache to be used
- dir_plugins: character, default '\$HOME/.vep/'; plugin to be used

- `offline`: logical, default FALSE; enable offline mode, no database connections will be made
- `fasta`: character, default `character()`; FASTA filename or directory to files to use for reference sequences

`output` list of the following options:

- `sift`: character, default `character()`; output prediction, score or both, valid strings are 'p', 's' or 'b'
- `polyphen`: character, default `character()`; output prediction, score or both, valid strings are 'p', 's' or 'b'
- `humdiv`: logical, default FALSE; retrieve the humDiv PolyPhen prediction instead of humVar
- `regulatory`: logical, default FALSE; identify overlaps with regulatory regions
- `cell_type`: character, default `character()`; only report regulatory regions found in the given cell type(s)
- `custom`: character, default `character()`; name of custom annotation file to add to output. Currently only a single annotation is supported.
- `plugin`: character, default `character()`; name of plugin module. Currently only a single module is supported.
- `individual`: character, default `character()`; consider only alternate alleles present in the genotypes of 'all' or a character vector of specified individuals
- `phased`: logical, default FALSE; force VCF genotypes to be interpreted as phased
- `allele_number`: logical, default FALSE; identify allele number from VCF input (1=first ALT, 2=second ALT, etc.)
- `total_length`: character, default `character()`; cDNA, CDS and protein positions as position/length
- `numbers`: logical, default FALSE; output affected exon and intron numbering, format is Number/Total
- `domains`: logical, default FALSE; output names of overlapping protein domains
- `no_escape`: logical, default FALSE; don't URI escape HGVS string
- `terms`: character, default 'so'; type of consequence terms to output, valid strings are 'ensembl' or 'so'

`identifiers` list of the following options:

- `hgvs`: logical, default FALSE; add hgvs ID's
- `protein`: logical, default FALSE; add Ensembl protein ID's
- `symbol`: logical, default FALSE; add gene symbol (e.g. HGNC) (where available) to the output
- `ccds`: logical, default FALSE; add CCDS transcript ID's
- `canonical`: logical, default FALSE; indicate if transcript is cononical transcript for the gene
- `uniprot`: logical, default FALSE; adds identifiers for translated protein products from three UniProt-related databases
- `tsl`: logical, default FALSE; adds the transcript support level for this transcript
- `biotype`: logical, default FALSE; add biotype of transcript
- `xref_seq`: logical, default FALSE; output aligned refseq mRNA ID

colocatedVariants list of the following options:

- check_existing: logical, default FALSE; check for co-located variants
- check_alleles: logical, default FALSE; when checking for co-located variants only report them if none of the alleles supplied are novel
- check_svsv: logical, default FALSE; check for structural variants that overlap the input variants
- gmaf: logical, default FALSE; add global minor allele frequency (MAF) from 1000 Genomes Phase 1 data
- maf_1kg: logical, default FALSE; add MAF from continental populations of 1000 Genomes Phase 1 data; must be use with `-cache`
- maf_esp: logical, default FALSE; add MAF from NHLBI-ESP populations; must be used with `-cache`
- old_maf: logical, default FALSE; for maf_1kg and maf_esp report only the frequency (no allele) and convert so it is always a minor frequency, i.e. < 0.5
- pubmed: logical, default FALSE; report Pubmed IDs for publications that cite existing variant; must be used with `-cache`
- failed: logical, default FALSE; when checking for co-located variants include or exclude variants that have been flagged as failed

dataformat list of the following options:

- vcf: logical, default FALSE; write output in vcf format
- json: logical, default FALSE; write output in json format
- gvf: logical, default FALSE; write output in gcf format
- original: logical, default FALSE; writes output as filtered set of input. Must be used with `-filter`.
- fields: character, default fields are 'Uploaded_variation', 'Location', 'Allele', 'Gene', 'Feature', 'Feature_type', 'Consequence', 'cDNA_position', 'CDS_position', 'Protein_position', 'Amino_acids', 'Codons' and 'Extra'. See http://www.ensembl.org/info/docs/variation/vep/vep_formats.html#sv for details.
- convert: character, default character(); converts input file to one of 'ensembl', 'vcf', or 'pileup'

filterqc list of the following options:

- check_ref: logical, default FALSE; force check of supplied reference allele against the sequence stored in Ensembl Core database
- coding_only: logical, default FALSE; return consequences in coding regions only
- chr: character, default character(); select a subset of chromosomes to be analyzed
- no_intergenic: logical, default FALSE; do not include intergenic consequences
- pick: logical, default FALSE; pick once line of consequence data per variant
- pick_allele: logical, default FALSE; pick once line of consequence data per variant allele
- flag_pick: logical, default FALSE; as per `-pick`, but adds the PICK flag to the chosen block of consequence data and retains others.
- flag_pick_allele: logical, default FALSE; as per `-pick_allele`, but adds the PICK flag to the chosen block of consequence data and retains others.
- per_gene: logical, default FALSE; output only the most severe consequence per gene

- `pick_order`: integer, default `integer()`; customise the order of criteria applied when choosing a block of annotation data with e.g. `-pick`.
- `most_severe`: logical, default `FALSE`; output only most severe consequence per variation
- `summary`: logical, default `FALSE`; output a comma-separated list of all observed consequences per variation, transcript-specific columns will be left blank
- `filter_common`: logical, default `FALSE`; shortcut flag to turn on filters, See web page for details.
- `check_frequency`: logical, default `FALSE`; turn on frequency filtering, must also specify all of the `-freq_*` flags. See web page for details.
- `freq_pop`: character, default `character()`; population to use in frequency filter
- `freq_freq`: numeric, default `numeric()`; MAF to use in frequency filter
- `freq_gt_lt`: character, default `character()`; specify whether the frequency of the co-located variant must be greater than or less than the value specified. Values are 'gt' or 'lt'. in the `freq_freq` option.
- `freq_filter`: character, default `character()`; specify whether to exclude or include variants that pass the frequency filter. Values are 'exclude' or 'include'.
- `filter`: character, default `character()`; filter the output on consequence type. See web page for details.
- `allow_non_variant`: logical, default `FALSE`; when using VCF format as input and output, by default VEP will skip all non-variant lines of input (i.e., where the ALT is NULL). When this option is enabled, lines will be printed in the VCF output with no consequence data added.

database list of the following options:

- `database`: logical, default `TRUE`; enable the VEP to use local or remote databases
- `host`: character, default `'useast.ensembl.db.org'`; database host
- `user`: character default `character()`; database user
- `password`: character, default `character()`; database password
- `port`: numeric, default `character()`; database port
- `genomes`: logical, default `FALSE`; override default connection settings with those for the Ensembl Genomes public MySQL server
- `gencode_basic`: logical, default `FALSE`; limit analysis to transcripts in GENCODE basic set
- `refseq`: logical, default `FALSE`; use otherfeatures database to retrieve transcripts
- `merged`: logical, default `FALSE`; use the merged Ensembl and RefSeq cache
- `all_refseq`: logical, default `FALSE`; include e.g. CCDS and Ensembl EST transcripts
- `lrg`: logical, default `FALSE`; map input variants to LRG coordinates
- `db_version`: numeric, default `character()`; force connection to specific version
- `registry`: character, default `character()`; provide file to override default connection settings

advanced list of the following options:

- `no_whole_genome`: logical, default `FALSE`; run in non-whole genome mode, variants analyzed one at a time, no caching

- `buffer_size`: numeric, default 5000; internal buffer size corresponding to number of variations read into memory simultaneously
- `write_cache`: logical, default FALSE; enable writing to the cache
- `build`: character, default `character()`; build cache for the selected species from the database (See `-chr` flag)
- `compress`: character, default `character()`; specify utility to decompress cached files (zcat is default)
- `skip_db_check`: logical, default FALSE; force the script to use a cache built from a different host than specified with `-host`
- `cache_region_size`: numeric, default `numeric()`; size in base-pairs of the region covered by one file in the cache, see full description of this flag on the web site for details

Author(s)

Valerie Obenchain <vobencha@fhcrc.org>

See Also

- The `ensemblVEP` function man page.
- The `VEPPParam` class man page.

Examples

```
## See ?VEPPParam for examples of constructing instances of a
## VEPPParam object with different runtime options.
```

VEPPParam-class

VEPPParam objects

Description

VEPPParam is a VIRTUAL class with concrete subclasses `VEPPParam67`, `VEPPParam73`, `VEPPParam75`, `VEPPParam77` and `VEPPParam78`. These objects store runtime options for the Ensembl Variant Effect Predictor.

Details

- The `VEPPParam` family of objects stores runtime options for querying the Ensembl Variant Effect Predictor (VEP). Brief descriptions of the options are provided below. For complete details, see the Ensembl VEP web page.
http://uswest.ensembl.org/info/docs/tools/vep/script/vep_options.html
- `VEPPParam` subclasses: As of `ensemblVEP` version `>= 1.3.3` a new subclass of `VEPPParam` will be created when there is a substantial change in the Ensembl variant database API. The subclass will support all versions up until the next 'substantial' version change. The version accessor on a `VEPPParam` object lists all versions that particular `VEPPParam` class supports. By default the `VEPPParam` constructor creates a subclass that supports the current version. To create a `VEPPParam` for a specific version, supply the version as an argument.

- Archived versions: Past versions of the VEP script and corresponding data are available on the Ensembl archive page.

<http://uswest.ensembl.org/info/website/archives/index.html>

Archived scripts can query cached data or perform a live query against the European mirror. (The US mirror, useastdb, hosts the current release only.) The appropriate version of cached data can be downloaded from the same page as the archived script.

- Multiple versions: By default, the `variant_effect_predictor.pl` script found in the `PATH` is the script used. If multiple versions of the script are installed locally a few steps must be taken to ensure the proper version is used.
 - `scriptPath` Provide the full path to the `variant_effect_predictor.pl` script (including the `.pl` extension) as the `scriptPath` argument to the `VEPParam` constructor.
 - `cache directories` By default, the cache runtime options `dir`, `dir_cache` and `dir_plugins` are set to `$HOME/.vep`. These locations need to be specified if the data are not in the default `$HOME/.vep` location.

Runtime options

For a description of runtime options for the most current version of the API see the `?runtimeOptions` man page or the Ensembl web site:

http://www.ensembl.org/info/docs/tools/vep/script/vep_options.html

Runtime options for archived versions can be found on the corresponding archive page.

Constructor

`VEPParam(version=max(unlist(currentVEP()), basic=basicOpts(), input=inputOpts(), cache=cacheOpts()), ...)`
Creates a `VEPParam` object.

`version` Numeric specifying the Ensembl API version(s) supported.

`basic` list of basic options

`input` list of input options

`cache` list of cache options

`output` list of output options

`filterqc` list of filterqc options

`database` list of database options

`advanced` list of advanced options

`scriptPath` character path to `variant_effect_predictor.pl` script; applicable when multiple versions of the script are installed locally

Supported for `VEPParam73` and later:

`identifier` list of identifier options

`colocatedVariants` list of `colocatedVariants` options

`dataformat` list of `dataformat` options

Accessors

In the following code, `x` is a `VEPPParam` object and `value` is a named list or character vector.

```
basic(x), basic(x) <- value
input(x), input(x) <- value
cache(x), cache(x) <- value
output(x), output(x) <- value
filterqc(x), filterqc(x) <- value
database(x), database(x) <- value
advanced(x), advanced(x) <- value
version(x), version(x) <- value
scriptPath(x), scriptPath(x) <- value
```

Supported for `VEPPParam73` and later:

```
identifier(x), identifier(x) <- value
colocatedVariants(x), colocatedVariants(x) <- value
dataformat(x), dataformat(x) <- value
```

Helper functions

`currentVEP()`: Invoked with no arguments. Returns the most current `VEPPParam` class and supported Ensembl API versions.

`supportedVEP()`: Invoked with no arguments. Returns a list of `VEPPParam` subclasses and the the Ensembl API versions they support.

The following functions create a list of runtime options and are used in the `VEPPParam` constructor.

```
basicOpts(version, ...)
inputOpts(version, ...)
cacheOpts(version, ...)
outputOpts(version, ...)
filterqcOpts(version, ...)
databaseOpts(version, ...)
advancedOpts(version, ...)
```

Supported for `VEPPParam73` and later:

```
identifierOpts(version, ...)
colocatedVariantsOpts(version, ...)
dataformatOpts(version, ...)
```

Author(s)

Valerie Obenchain <vobencha@fhcrc.org>

See Also

- The runtimeOptions man page.
- The ensemblVEP function man page.

Examples

```
## -----
## Current API version
## -----
## The default constructor supports the most current version
## of the Ensembl variant API.
param <- VEPParam()
class(param)

## The version slot lists all API versions supported by the class.
version(param)

## The supportedVEP helper returns a list of VEPParam classes and the
## corresponding API versions they support.
supportedVEP()

## -----
## Archived API versions
## -----
## Create a VEPParam for an archived version by supplying the
## version to the constructor.
param67 <- VEPParam(67)
param67

## Archive versions can query the cache or the European mirror. The
## default host for live queries is set to ensembl.db.ensembl.org.
database(param67)$host

## By default the VEP script used is the one found in the PATH.
## To specify a script in a non-standard location use the scriptPath
## setter. Include the full path and the name of the script with the
## .pl extension.
## Not run:
scriptPath(param) <- "fullPathToScript/variant_effect_predictor.pl"

## End(Not run)

## -----
## Manipulation
## -----

## View the values in basic and input.
basic(param)
input(param)

## Change the value of the everything to TRUE.
basic(param)$everything
```

```
basic(param)$everything <- TRUE
basic(param)$everything

## Replace multiple values using a named list.
basic(param) <- list(verbose=TRUE, config="myconfig.txt")
basic(param)

## Write the output to myfile.vcf instead of returning a VCF object.
## Return the sift and polyphen predictions only (not scores).
param <- VEPPParam(input=c(output_file="path/myfile.vcf"),
                   output=c(sift="p", polyphen="p"))

## sift and polyphen are runtime options that require
## a character value, (i.e., s, p, or b).
output(param)$sift

## To turn off sift or polyphen set the value to an
## empty character (i.e., character()).
output(param)$sift <- character()
```

Index

*Topic **classes**

VEPPParam-class, 10

*Topic **methods**

ensemblVEP, 2

parseCSQToGRanges, 4

VEPPParam-class, 10

*Topic **utilities**

runtimeOptions, 5

advanced (VEPPParam-class), 10

advanced, VEPPParam-method
(VEPPParam-class), 10

advanced<- (VEPPParam-class), 10

advanced<- , VEPPParam-method
(VEPPParam-class), 10

advancedOpts (VEPPParam-class), 10

basic (VEPPParam-class), 10

basic, VEPPParam-method (VEPPParam-class),
10

basic<- (VEPPParam-class), 10

basic<- , VEPPParam-method
(VEPPParam-class), 10

basicOpts (VEPPParam-class), 10

cache (VEPPParam-class), 10

cache, VEPPParam-method (VEPPParam-class),
10

cache<- (VEPPParam-class), 10

cache<- , VEPPParam-method
(VEPPParam-class), 10

cacheOpts (VEPPParam-class), 10

class:VEPPParam (VEPPParam-class), 10

class:VEPPParam67 (VEPPParam-class), 10

class:VEPPParam73 (VEPPParam-class), 10

class:VEPPParam75 (VEPPParam-class), 10

class:VEPPParam77 (VEPPParam-class), 10

class:VEPPParam78 (VEPPParam-class), 10

colocatedVariants (VEPPParam-class), 10

colocatedVariants, VEPPParam73-method
(VEPPParam-class), 10

colocatedVariants, VEPPParam75-method
(VEPPParam-class), 10

colocatedVariants, VEPPParam77-method
(VEPPParam-class), 10

colocatedVariants, VEPPParam78-method
(VEPPParam-class), 10

colocatedVariants<- (VEPPParam-class), 10

colocatedVariants<- , VEPPParam73-method
(VEPPParam-class), 10

colocatedVariants<- , VEPPParam75-method
(VEPPParam-class), 10

colocatedVariants<- , VEPPParam77-method
(VEPPParam-class), 10

colocatedVariants<- , VEPPParam78-method
(VEPPParam-class), 10

colocatedVariantsOpts (VEPPParam-class),
10

currentVEP (VEPPParam-class), 10

database (VEPPParam-class), 10

database, VEPPParam-method
(VEPPParam-class), 10

database<- (VEPPParam-class), 10

database<- , VEPPParam-method
(VEPPParam-class), 10

databaseOpts (VEPPParam-class), 10

dataformat (VEPPParam-class), 10

dataformat, VEPPParam73-method
(VEPPParam-class), 10

dataformat, VEPPParam75-method
(VEPPParam-class), 10

dataformat, VEPPParam77-method
(VEPPParam-class), 10

dataformat, VEPPParam78-method
(VEPPParam-class), 10

dataformat<- (VEPPParam-class), 10

dataformat<- , VEPPParam73-method
(VEPPParam-class), 10

- dataformat<- ,VEPParam75-method
(VEPParam-class), 10
- dataformat<- ,VEPParam77-method
(VEPParam-class), 10
- dataformat<- ,VEPParam78-method
(VEPParam-class), 10
- dataformatOpts (VEPParam-class), 10
- ensemblVEP, 2, 5
- ensemblVEP, character-method
(ensemblVEP), 2

- filterqc (VEPParam-class), 10
- filterqc, VEPParam-method
(VEPParam-class), 10
- filterqc<- (VEPParam-class), 10
- filterqc<- ,VEPParam-method
(VEPParam-class), 10
- filterqcOpts (VEPParam-class), 10

- identifier (VEPParam-class), 10
- identifier, VEPParam73-method
(VEPParam-class), 10
- identifier, VEPParam75-method
(VEPParam-class), 10
- identifier, VEPParam77-method
(VEPParam-class), 10
- identifier, VEPParam78-method
(VEPParam-class), 10
- identifier<- (VEPParam-class), 10
- identifier<- ,VEPParam73-method
(VEPParam-class), 10
- identifier<- ,VEPParam75-method
(VEPParam-class), 10
- identifier<- ,VEPParam77-method
(VEPParam-class), 10
- identifier<- ,VEPParam78-method
(VEPParam-class), 10
- identifierOpts (VEPParam-class), 10
- input (VEPParam-class), 10
- input, VEPParam-method (VEPParam-class),
10
- input<- (VEPParam-class), 10
- input<- ,VEPParam-method
(VEPParam-class), 10
- inputOpts (VEPParam-class), 10

- output (VEPParam-class), 10
- output, VEPParam-method
(VEPParam-class), 10

- output<- (VEPParam-class), 10
- output<- ,VEPParam-method
(VEPParam-class), 10
- outputOpts (VEPParam-class), 10

- parseCSQToGRanges, 4
- parseCSQToGRanges, character-method
(parseCSQToGRanges), 4
- parseCSQToGRanges, VCF-method
(parseCSQToGRanges), 4

- runtimeOptions, 5

- scriptPath (VEPParam-class), 10
- scriptPath<- (VEPParam-class), 10
- show, VEPParam-method (VEPParam-class),
10
- supportedVEP (VEPParam-class), 10

- VCF, 2
- VEPParam (VEPParam-class), 10
- VEPParam-class, 2, 5, 10
- VEPParam67-class (VEPParam-class), 10
- VEPParam73-class (VEPParam-class), 10
- VEPParam75-class (VEPParam-class), 10
- VEPParam77-class (VEPParam-class), 10
- VEPParam78-class (VEPParam-class), 10
- version (VEPParam-class), 10
- version<- (VEPParam-class), 10