

Overview of ensemblVEP

Valerie Obenchain

Last modified: December 2012; Compiled: October 17, 2016

Contents

1 Introduction

Ensembl provides the facility to predict functional consequences of known and unknown variants using the Variant Effect Predictor (VEP). The `ensemblVEP` package wraps Ensembl VEP and returns the results as `Robjects` or a file on disk. To use this package the Ensembl VEP perl script must be installed in your path. See the package README for details. Downloads: <http://uswest.ensembl.org/info/docs/tools/vep/index.html>
Complete documentation for runtime options: http://uswest.ensembl.org/info/docs/tools/vep/script/vep_options.html

To test that Ensembl VEP is properly installed, enter the name of the script from the command line:

```
variant_effect_predictor.pl
```

2 Results as R objects

```
> library(ensemblVEP)
```

The `ensemblVEP` function can return variant consequences from Ensembl VEP as `Robjects` (`GRanges` or `VCF`) or write them to a file. The default behavior returns a `GRanges`. Runtime options are stored in a `VEPParam` object and allow a great deal of control over the content and format of the results. See the man pages for more details.

```
> ?ensemblVEP
```

```
> ?VEPParam
```

The default runtime options can be inspected by creating a `VEPParam`.

```
> param <- VEPParam()
```

```
> param
```

```
class: VEPParam82
identifier():
colocatedVariants():
dataformat():
basic():
input(1): species
cache(3): dir, dir_cache, dir_plugins
output(1): terms
filterqc():
database(2): host, database
advanced(1): buffer_size
version: 82
scriptPath:
```

```
> basic(param)
```

```

$verbose
[1] FALSE

$quiet
[1] FALSE

$no_progress
[1] FALSE

$config
character(0)

$everything
[1] FALSE

$fork
numeric(0)

```

Using a vcf file from VariantAnnotation as input, we query Ensembl VEP with the default runtime parameters.

```

> fl <- system.file("extdata", "gl_chr1.vcf", package="VariantAnnotation")
> gr <- ensemblVEP(fl)

```

Consequence data are parsed into the metadata columns of the GRanges. To control the type and amount of data returned see the options in output(VEPParam()).

```

> head(gr, 3)
GRanges object with 3 ranges and 23 metadata columns:

```

seqnames	ranges	strand	Allele																				
<Rle>	<IRanges>	<Rle>	<factor>																				
rs6054257	20 [14370, 14370]	*	A																				
20:17330_T/A	20 [17330, 17330]	*	A																				
rs6040355	20 [1110696, 1110696]	*	G																				
	Consequence	IMPACT	SYMBOL																				
	<factor>	<factor>	<factor>																				
rs6054257	intergenic_variant	MODIFIER	<NA>																				
20:17330_T/A	intergenic_variant	MODIFIER	<NA>																				
rs6040355	upstream_gene_variant	MODIFIER	PSMF1																				
	Feature_type	Feature	BIOTYPE																				
	<factor>	<factor>	<factor>																				
rs6054257	<NA>	<NA>	<NA>																				
20:17330_T/A	<NA>	<NA>	<NA>																				
rs6040355	Transcript	ENST00000479715	processed_transcript																				
	INTRON	HGVSc	HGVSp	cDNA_position																			
	<factor>	<factor>	<factor>	<factor>																			
rs6054257	<NA>	<NA>	<NA>	<NA>																			
20:17330_T/A	<NA>	<NA>	<NA>	<NA>																			
rs6040355	<NA>	<NA>	<NA>	<NA>																			
	Protein_position	Amino_acids	Codons	Existing_variation																			
	<factor>	<factor>	<factor>	<factor>																			
rs6054257	<NA>	<NA>	<NA>	<NA>																			
20:17330_T/A	<NA>	<NA>	<NA>	<NA>																			
rs6040355	<NA>	<NA>	<NA>	<NA>																			
	DISTANCE	STRAND	FLAGS	SYMBOL_SOURCE																			
	<factor>	<factor>	<factor>	<factor>																			
rs6054257	<NA>	<NA>	<NA>	<NA>																			
20:17330_T/A	<NA>	<NA>	<NA>	<NA>																			
rs6040355	2610	1	<NA>	HGNC																			
				HGNC																			

```

-----
seqinfo: 1 sequence from genome

```

Next we use a vcf of structural variants as input

```
> fl <- system.file("extdata", "structural.vcf", package="VariantAnnotation")
```

and request that a VCF object be returned by setting the *vcf* option in the *dataformat* slot to TRUE.

```
> param <- VEPParam(dataformat=c(vcf=TRUE))
```

An call to *ensemblVEP* results in an error.

```
> vcf <- ensemblVEP(fl, param)
2012-12-03 16:40:55 - Starting...
ERROR: Could not detect input file format
```

In most situations Ensembl VEP can auto-detect the input format. In this case, however, it cannot so we explicitly set the *format* option to 'vcf'.

```
> input(param)$format <- "vcf"
```

Try again.

```
> vep <- ensemblVEP(fl, param)
```

Success! When a VCF is returned, consequence data are included as an unparsed INFO column labeled *CSQ*.

```
> info(vep)$CSQ

CharacterList of length 5
[[1]] deletion|intron_variant&non_coding_transcript_variant&feature_truncatio...
[[2]] -|intergenic_variant|
[[3]] insertion|upstream_gene_variant|MODIFIER|SETD5|ENSG00000168137|Transcri...
[[4]] duplication|upstream_gene_variant|MODIFIER|RAF1|ENSG00000132155|Transcr...
[[5]] -|intergenic_variant|
```

The *parseCSQToGRanges* function parses these data into a *GRanges*. When the rownames of the original VCF are provided as *VCFRowID* a metadata column of the same name is included in the output.

```
> vcf <- readVcf(fl, "hg19")
> csq <- parseCSQToGRanges(vep, VCFRowID=rownames(vcf))
> head(csq, 3)
```

GRanges object with 3 ranges and 24 metadata columns:

	seqnames	ranges	strand	VCFRowID
	<Rle>	<IRanges>	<Rle>	<integer>
	2:321682_T/	2 [321682, 321682]	*	3
	2:321682_T/	2 [321682, 321682]	*	3
	2:14477084_C/<DEL:ME:ALU>	2 [14477084, 14477084]	*	4
	Allele			
	<character>			
	2:321682_T/	deletion		
	2:321682_T/	deletion		
	2:14477084_C/<DEL:ME:ALU>	-		
				Consequence
				<character>
	2:321682_T/	intron_variant&non_coding_transcript_variant&feature_truncation		
	2:321682_T/	intron_variant&non_coding_transcript_variant&feature_truncation		
	2:14477084_C/<DEL:ME:ALU>	intergenic_variant		
	IMPACT	SYMBOL	Gene	
	<character>	<character>	<character>	
	2:321682_T/	MODIFIER AC079779.6	ENSG00000233684	
	2:321682_T/	MODIFIER AC079779.6	ENSG00000233684	
	2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	
	Feature_type	Feature	BIOTYPE	

	<character>	<character>	<character>
2:321682_T/	Transcript	ENST00000430529	lincRNA
2:321682_T/	Transcript	ENST00000436808	lincRNA
2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	<NA>
	EXON	INTRON	HGVSc
	<character>	<character>	<character>
2:321682_T/	<NA>	1/1	<NA>
2:321682_T/	<NA>	1/3	<NA>
2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	<NA>
	cDNA_position	CDS_position	Protein_position
	<character>	<character>	<character>
2:321682_T/	<NA>	<NA>	<NA>
2:321682_T/	<NA>	<NA>	<NA>
2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	<NA>
	Amino_acids	Codons	Existing_variation
	<character>	<character>	<character>
2:321682_T/	<NA>	<NA>	<NA>
2:321682_T/	<NA>	<NA>	<NA>
2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	<NA>
	DISTANCE	STRAND	FLAGS
	<character>	<character>	<character>
2:321682_T/	<NA>	1	<NA>
2:321682_T/	<NA>	1	<NA>
2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	<NA>
	SYMBOL_SOURCE	HGNC_ID	
	<character>	<character>	
2:321682_T/	Clone_based_vega_gene	<NA>	
2:321682_T/	Clone_based_vega_gene	<NA>	
2:14477084_C/<DEL:ME:ALU>	<NA>	<NA>	

seqinfo: 3 sequences from genome; no seqlengths

The `VCFRowID` columns maps the expanded *CSQ* data back to the rows in the *VCF* object. This index can be used to subset the original VCF.

```
> vcf[csq$"VCFRowID"]
```

```
class: CollapsedVCF
```

```
dim: 22 1
```

```
rowRanges(vcf):
```

```
GRanges with 5 metadata columns: paramRangeID, REF, ALT, QUAL, FILTER
```

```
info(vcf):
```

```
DataFrame with 10 columns: BKPTID, CIEND, CIPOS, END, HOMLEN, HOMSEQ, IMPR...
```

```
info(header(vcf)):
```

	Number	Type	Description
BKPTID	.	String	ID of the assembled alternate allele in the asse...
CIEND	2	Integer	Confidence interval around END for imprecise var...
CIPOS	2	Integer	Confidence interval around POS for imprecise var...
END	1	Integer	End position of the variant described in this re...
HOMLEN	.	Integer	Length of base pair identical micro-homology at ...
HOMSEQ	.	String	Sequence of base pair identical micro-homology a...
IMPRECISE	0	Flag	Imprecise structural variation
MEINFO	4	String	Mobile element info of the form NAME,START,END,P...
SVLEN	.	Integer	Difference in length between REF and ALT alleles
SVTYPE	1	String	Type of structural variant

```
geno(vcf):
```

```
SimpleList of length 4: GT, GQ, CN, CNQ
```

```
geno(header(vcf)):
```

Number	Type	Description
--------	------	-------------

GT	1	String	Genotype
GQ	1	Float	Genotype quality
CN	1	Integer	Copy number genotype for imprecise events
CNQ	1	Float	Copy number genotype quality for imprecise events

3 Write results to a file

In the previous section we saw Ensembl VEP results returned as R objects in the workspace. Alternatively, these results can be written directly to a file. The flag that controls how the data are returned is the *output_file* flag in the *input* options.

When *output_file* is an empty character (default), the results are returned as either a *GRanges* or *VCF* object.

```
> input(param)$output_file
character(0)
```

To write results directly to a file, specify a file name for the *output_file* flag.

```
> input(param)$output_file <- "/mypath/myfile"
```

The file can be written as a *vcf* or *gvf* by setting the options in the *dataformat* slot to TRUE. If neither of *vcf* or *gvf* are TRUE the file is written out as tab delimited.

```
> ## 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"))
> ## Write a tab delimited file to myfile.txt:
> myparam <- VEPParam(input=c(output_file="/path/myfile.txt"))
```

4 Configuring runtime options

The Ensembl VEP web page has complete descriptions of all runtime options. http://uswest.ensembl.org/info/docs/tools/vep/script/vep_options.html Below are examples of how to configure the runtime options in the *VEPParam* for specific situations. Investigate the differences in results using a sample file from *VariantAnnotation*.

```
> fl <- system.file("extdata", "ex2.vcf", package="VariantAnnotation")
```

- Add regulatory region consequences:

```
> param <- VEPParam(output=c(regulatory=TRUE))
> gr <- ensemblVEP(fl, param)
```

- Specify input file format as VCF, add HGNC gene identifiers, output SO consequence terms:

```
> param <- VEPParam(input=c(format="vcf"),
+                   output=c(terms="so"),
+                   identifiers=c(symbol=TRUE))
> gr <- ensemblVEP(fl, param)
```

- Check for co-located variants, output only coding sequence consequences, output HGVS names:

```
> param <- VEPParam(filterqc=c(coding_only=TRUE),
+                   colocatedVariants=c(check_existing=TRUE),
+                   identifiers=c(symbol=TRUE))
> gr <- ensemblVEP(fl, param)
```

- Add SIFT score and prediction, PolyPhen prediction only, output results as VCF:

```

fl <- system.file("extdata", "chr22.vcf.gz", package="VariantAnnotation")
param <- VEPParam(output=c(sift="b", polyphen="p"),
                  dataformat=c(vcf=TRUE))
vcf <- ensemblVEP(fl, param)
csq <- parseCSQToGRanges(vcf)

> head(levels(mcols(csq)$SIFT))
[1] "deleterious(0.01)" "deleterious(0.02)" "deleterious(0.03)"
[4] "deleterious(0.04)" "deleterious(0.05)" "deleterious(0)"

> levels(mcols(csq)$PolyPhen)
[1] "benign"           "possibly_damaging" "probably_damaging"
[4] "unknown"

```

5 sessionInfo()

```
> sessionInfo()
```

```

R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

```

```
locale:
```

```
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
attached base packages:
```

```

[1] stats4    parallel  stats      graphics  grDevices  utils      datasets
[8] methods   base

```

```
other attached packages:
```

```

[1] ensemblVEP_1.14.0      VariantAnnotation_1.20.0
[3] Rsamtools_1.26.0       Biostrings_2.42.0
[5] XVector_0.14.0         SummarizedExperiment_1.4.0
[7] Biobase_2.34.0         GenomicRanges_1.26.0
[9] GenomeInfoDb_1.10.0    IRanges_2.8.0
[11] S4Vectors_0.12.0       BiocGenerics_0.20.0

```

```
loaded via a namespace (and not attached):
```

```

[1] AnnotationDbi_1.36.0    GenomicAlignments_1.10.0  zlibbioc_1.20.0
[4] BiocParallel_1.8.0      BSgenome_1.42.0          lattice_0.20-34
[7] tools_3.3.1            grid_3.3.1              DBI_0.5-1
[10] Matrix_1.2-7.1          rtracklayer_1.34.0       bitops_1.0-6
[13] RCurl_1.95-4.8          biomaRt_2.30.0           RSQLite_1.0.0
[16] GenomicFeatures_1.26.0  XML_3.98-1.4

```