

Overview of ensemblVEP

Valerie Obenchain

Last modified: December 2012; Compiled: December 23, 2014

Contents

1	Introduction	1
2	Results as R objects	1
3	Write results to a file	4
4	Configuring runtime options	5
5	sessionInfo()	6

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 R objects 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 R objects (`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: VEPParam78
```

```
identifier(0):
```

```
colocatedVariants(0):
```

```
dataformat(0):
```

```
basic(0):
```

```
input(1): species
```

```
cache(3): dir, dir_cache, dir_plugins
```

```
output(1): terms
```

```

filterqc(0):
database(2): host, database
advanced(1): buffer_size
version: 78
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 13 metadata columns:

	seqnames	ranges	strand	Allele	Gene
	<Rle>	<IRanges>	<Rle>	<factor>	<factor>
rs58108140	1	[10583, 10583]	*	A	ENSG00000223972
rs58108140	1	[10583, 10583]	*	A	ENSG00000227232
rs58108140	1	[10583, 10583]	*	A	ENSG00000223972
	Feature	Feature_type		Consequence	cDNA_position
	<factor>	<factor>		<factor>	<factor>
rs58108140	ENST00000456328	Transcript	upstream_gene_variant		<NA>
rs58108140	ENST00000488147	Transcript	downstream_gene_variant		<NA>
rs58108140	ENST00000450305	Transcript	upstream_gene_variant		<NA>
	CDS_position	Protein_position	Amino_acids	Codons	
	<factor>	<factor>	<factor>	<factor>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
	Existing_variation	DISTANCE	STRAND		
	<factor>	<factor>	<factor>		
rs58108140	<NA>	1286	1		
rs58108140	<NA>	3821	-1		
rs58108140	<NA>	1427	1		

```

-----
seqinfo: 1 sequence from genome; no seqlengths

```

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|ENSG00000233684|ENST00000430529|Transcript|intron_variant&non...
[[2]] -||||intergenic_variant|||||||
[[3]] insertion|ENSG00000168137|ENST00000468208|Transcript|intron_variant&non...
[[4]] duplication|ENSG00000132155|ENST00000423275|Transcript|upstream_gene_va...
[[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 14 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		Gene		Feature
	<factor>		<factor>		<factor>
	2:321682_T/	deletion	ENSG00000233684	ENST00000430529	
	2:321682_T/	deletion	ENSG00000233684	ENST00000436808	
	2:14477084_C/<DEL:ME:ALU>	-	<NA>	<NA>	
	Feature_type				
	<factor>				
	2:321682_T/	Transcript			
	2:321682_T/	Transcript			
	2:14477084_C/<DEL:ME:ALU>	<NA>			
					Consequence
					<factor>
	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			
	cDNA_position	CDS_position	Protein_position		
	<factor>	<factor>	<factor>		
	2:321682_T/	<NA>	<NA>	<NA>	

```

      2:321682_T/<DEL>      <NA>      <NA>      <NA>
2:14477084_C/<DEL:ME:ALU>  <NA>      <NA>      <NA>
      Amino_acids  Codons Existing_variation  DISTANCE
      <factor> <factor>      <factor> <factor>
      2:321682_T/<DEL>      <NA>      <NA>      <NA>      <NA>
      2:321682_T/<DEL>      <NA>      <NA>      <NA>      <NA>
2:14477084_C/<DEL:ME:ALU>  <NA>      <NA>      <NA>      <NA>
      STRAND
      <factor>
      2:321682_T/<DEL>      1
      2:321682_T/<DEL>      1
2:14477084_C/<DEL:ME:ALU>  <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
rowData(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 3.1.2 (2014-10-31)
```

```
Platform: x86_64-apple-darwin13.4.0 (64-bit)
```

```
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_1.6.2      VariantAnnotation_1.12.7 Rsamtools_1.18.2  
[4] Biostrings_2.34.1     XVector_0.6.0           GenomicRanges_1.18.3  
[7] GenomeInfoDb_1.2.4    IRanges_2.0.1           S4Vectors_0.4.0  
[10] BiocGenerics_0.12.1
```

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

```
[1] AnnotationDbi_1.28.1  BBmisc_1.8              BSgenome_1.34.0  
[4] BatchJobs_1.5         Biobase_2.26.0          BiocParallel_1.0.0  
[7] DBI_0.3.1             GenomicAlignments_1.2.1 GenomicFeatures_1.18.3  
[10] RCurl_1.95-4.5        RSQLite_1.0.0           XML_3.98-1.1  
[13] base64enc_0.1-2       biomaRt_2.22.0          bitops_1.0-6  
[16] brew_1.0-6            checkmate_1.5.1         codetools_0.2-9  
[19] digest_0.6.7          fail_1.2                 foreach_1.4.2  
[22] iterators_1.0.7       rtracklayer_1.26.2      sendmailR_1.2-1  
[25] stringr_0.6.2         tools_3.1.2             zlibbioc_1.12.0
```