

# Overview of ensemblVEP

Lori Shepherd and Valerie Obenchain

July 12, 2022

## Contents

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

## 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.

NOTE: As of Ensembl version 88 the VEP script has been renamed from `variant_effect_predictor.pl` to `vep`. The ensemblVEP package code and documentation have been updated to reflect this change.

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](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:

```
vep
```

## 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 VEPFlags object and allow a great deal of control over the content and format of the results. See the man pages for more details.

```
> ?ensemblVEP
```

```
> ?VEPFlags
```

The default runtime options can be inspected by creating a VEPFlags.

```
> param <- VEPFlags()
```

```
> param
```

```
class: VEPFlags
flags(1): database
version: 105
scriptPath:
```

```
> flags(param)
```

```
$database
```

```
[1] TRUE
```

```
$vcf
```

```
[1] FALSE
```

Of note is the 'host'. The default 'host' for queries is character() and therefore is set to the vep default of 'ensemldb.ensembl.org'. Users in the US may find connection and transfer speeds quicker using our East coast mirror, 'useastdb.ensembl.org'.

```
> param <- VEPFlags(flags=list(host="useastdb.ensembl.org"))
```

Using a vcf file from VariantAnnotation as input, we query Ensembl VEP with the default runtime parameters. Consequence data are parsed into the metadata columns of the GRanges. To control the type and amount of data returned see the output options at [http://uswest.ensembl.org/info/docs/tools/vep/script/vep\\_options.html](http://uswest.ensembl.org/info/docs/tools/vep/script/vep_options.html).

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

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

```
> head(gr, 3)
```

GRanges object with 3 ranges and 23 metadata columns:

	seqnames	ranges	strand	Allele	Consequence
	<Rle>	<IRanges>	<Rle>	<character>	<character>
rs58108140	1	10583	*	A	upstream_gene_variant
rs58108140	1	10583	*	A	upstream_gene_variant
rs58108140	1	10583	*	A	downstream_gene_vari..
	IMPACT	SYMBOL	Gene	Feature_type	
	<character>	<character>	<character>	<character>	
rs58108140	MODIFIER	DDX11L1	ENSG00000223972	Transcript	
rs58108140	MODIFIER	DDX11L1	ENSG00000223972	Transcript	
rs58108140	MODIFIER	WASH7P	ENSG00000227232	Transcript	
	Feature	BIOTYPE	EXON	INTRON	
	<character>	<character>	<character>	<character>	
rs58108140	ENST00000450305	transcribed_unproces..	<NA>	<NA>	
rs58108140	ENST00000456328	processed_transcript	<NA>	<NA>	
rs58108140	ENST00000488147	unprocessed_pseudogene	<NA>	<NA>	
	HGVSc	HGVSp	cDNA_position	CDS_position	
	<character>	<character>	<character>	<character>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
	Protein_position	Amino_acids	Codons	Existing_variation	
	<character>	<character>	<character>	<character>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
rs58108140	<NA>	<NA>	<NA>	<NA>	
	DISTANCE	STRAND	FLAGS	SYMBOL_SOURCE	HGNC_ID
	<character>	<character>	<character>	<character>	<character>
rs58108140	1427	1	<NA>	HGNC	HGNC:37102
rs58108140	1286	1	<NA>	HGNC	HGNC:37102
rs58108140	3821	-1	<NA>	HGNC	HGNC:38034

```
-----
```

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

Next we request that a VCF object be returned by setting the vcf option in the flags slot to TRUE.

```
> param <- VEPFlags(flags=list(vcf=TRUE, host="useastdb.ensembl.org"))
```

```
> 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 3

```
[[1]] A|upstream_gene_variant|MODIFIER|DDX11L1|ENSG00000223972|Transcript|ENS...
[[2]] T|non_coding_transcript_exon_variant|MODIFIER|DDX11L1|ENSG00000223972|T...
[[3]] T|downstream_gene_variant|MODIFIER|FAM138A|ENSG00000237613|Transcript|E...
```

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(vcf, VCFRowID=rownames(vcf))
> head(csq, 3)
```

`GRanges` object with 3 ranges and 24 metadata columns:

```
      seqnames      ranges strand | VCFRowID      Allele
      <Rle> <IRanges> <Rle> | <integer> <character>
rs58108140      1      10583      * |      1      A
rs58108140      1      10583      * |      1      A
rs58108140      1      10583      * |      1      A
      Consequence      IMPACT      SYMBOL      Gene
      <character> <character> <character> <character>
rs58108140 upstream_gene_variant      MODIFIER      DDX11L1      ENSG00000223972
rs58108140 upstream_gene_variant      MODIFIER      DDX11L1      ENSG00000223972
rs58108140 downstream_gene_vari..      MODIFIER      WASH7P      ENSG00000227232
      Feature_type      Feature      BIOTYPE      EXON
      <character> <character> <character> <character>
rs58108140 Transcript      ENST00000450305 transcribed_unproces.. <NA>
rs58108140 Transcript      ENST00000456328 processed_transcript <NA>
rs58108140 Transcript      ENST00000488147 unprocessed_pseudogene <NA>
      INTRON      HGVS_c      HGVS_p      cDNA_position      CDS_position
      <character> <character> <character> <character> <character>
rs58108140 <NA> <NA> <NA> <NA> <NA>
rs58108140 <NA> <NA> <NA> <NA> <NA>
rs58108140 <NA> <NA> <NA> <NA> <NA>
      Protein_position      Amino_acids      Codons      Existing_variation
      <character> <character> <character> <character>
rs58108140 <NA> <NA> <NA> <NA>
rs58108140 <NA> <NA> <NA> <NA>
rs58108140 <NA> <NA> <NA> <NA>
      DISTANCE      STRAND      FLAGS      SYMBOL_SOURCE      HGNC_ID
      <character> <character> <character> <character> <character>
rs58108140      1427      1 <NA>      HGNC      HGNC:37102
rs58108140      1286      1 <NA>      HGNC      HGNC:37102
rs58108140      3821     -1 <NA>      HGNC      HGNC:38034
```

-----

seqinfo: 1 sequence from genome; no seqlengths

The `VCFRowID` column 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: 13 85

rowRanges(vcf):

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

info(vcf):

DataFrame with 22 columns: `LDAF`, `AVGPOST`, `RSQ`, `ERATE`, `THETA`, `CIEND`, `CIPOS`, ...

info(header(vcf)):

	Number	Type	Description
LDAF	1	Float	MLE Allele Frequency Accounting for LD

AVGPOST	1	Float	Average posterior probability from MaCH/Thunder
RSQ	1	Float	Genotype imputation quality from MaCH/Thunder
ERATE	1	Float	Per-marker Mutation rate from MaCH/Thunder
THETA	1	Float	Per-marker Transition rate from MaCH/Thunder
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...
SVLEN	1	Integer	Difference in length between REF and ALT alleles
SVTYPE	1	String	Type of structural variant
AC	.	Integer	Alternate Allele Count
AN	1	Integer	Total Allele Count
AA	1	String	Ancestral Allele, ftp://ftp.1000genomes.ebi.ac.u...
AF	1	Float	Global Allele Frequency based on AC/AN
AMR_AF	1	Float	Allele Frequency for samples from AMR based on A...
ASN_AF	1	Float	Allele Frequency for samples from ASN based on A...
AFR_AF	1	Float	Allele Frequency for samples from AFR based on A...
EUR_AF	1	Float	Allele Frequency for samples from EUR based on A...
VT	1	String	indicates what type of variant the line represents
SNPSOURCE	.	String	indicates if a snp was called when analysing the...

geno(vcf):  
List of length 3: GT, DS, GL

geno(header(vcf)):

Number	Type	Description
GT	1	String Genotype
DS	1	Float Genotype dosage from MaCH/Thunder
GL	.	Float Genotype Likelihoods

### 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.

When *output\_file* is NULL (default), the results are returned as either a *GRanges* or *VCF* object.

```
> flags(param)$output_file
```

```
NULL
```

To write results directly to a file, specify a file name for the *output\_file* flag.

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

The file can be written as a *vcf* or *gvf* by setting the options of the 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 <- VEPFlags(flags=list(vcf=TRUE,
+                               output_file="/path/myfile.vcf"))
> ## Write a gvf file to myfile.gvf:
> myparam <- VEPFlags(flags=list(gvf=TRUE,
+                               output_file="/path/myfile.gvf"))
> ## Write a tab delimited file to myfile.txt:
> myparam <- VEPFlags(flags=list(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](http://uswest.ensembl.org/info/docs/tools/vep/script/vep_options.html) Below are examples of how to configure the runtime options in the *VEPFlags* 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 <- VEPFlags(flags=list(regulatory=TRUE, host="useastdb.ensembl.org"))
> gr <- ensemblVEP(fl, param)
```
- Specify input file format as VCF, add HGNC gene identifiers, output SO consequence terms:

```
> param <- VEPFlags(flag=list(format="vcf",
+                             terms="SO",
+                             symbol=TRUE, host="useastdb.ensembl.org"))
> gr <- ensemblVEP(fl, param)
```
- Check for co-located variants, output only coding sequence consequences, output HGVS names:

```
> param <- VEPFlags(flags=list(coding_only=TRUE,
+                               check_existing=TRUE,
+                               symbol=TRUE, host="useastdb.ensembl.org"))
> 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 <- VEPFlags(flags=list(sift="b", polyphen="p",
                             vcf=TRUE, host="useastdb.ensembl.org"))
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 4.2.1 (2022-06-23)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.4 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.15-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.15-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
 [3] LC_TIME=en_GB            LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
 [9] LC_ADDRESS=C             LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4      stats      graphics  grDevices  utils      datasets  methods
[8] base
```

other attached packages:

[1]	ensemblVEP_1.38.0	VariantAnnotation_1.42.1
[3]	Rsamtools_2.12.0	Biostrings_2.64.0
[5]	XVector_0.36.0	SummarizedExperiment_1.26.1
[7]	Biobase_2.56.0	MatrixGenerics_1.8.1
[9]	matrixStats_0.62.0	GenomicRanges_1.48.0
[11]	GenomeInfoDb_1.32.2	IRanges_2.30.0
[13]	S4Vectors_0.34.0	BiocGenerics_0.42.0

loaded via a namespace (and not attached):

[1]	Rcpp_1.0.9	lattice_0.20-45	prettyunits_1.1.1
[4]	png_0.1-7	assertthat_0.2.1	digest_0.6.29
[7]	utf8_1.2.2	BiocFileCache_2.4.0	R6_2.5.1
[10]	RSQLite_2.2.14	httr_1.4.3	pillar_1.7.0
[13]	zlibbioc_1.42.0	rlang_1.0.4	GenomicFeatures_1.48.3
[16]	progress_1.2.2	curl_4.3.2	blob_1.2.3
[19]	Matrix_1.4-1	BiocParallel_1.30.3	stringr_1.4.0
[22]	RCurl_1.98-1.7	bit_4.0.4	biomaRt_2.52.0
[25]	DelayedArray_0.22.0	rtracklayer_1.56.1	compiler_4.2.1
[28]	pkgconfig_2.0.3	tidyselect_1.1.2	KEGGREST_1.36.3
[31]	tibble_3.1.7	GenomeInfoDbData_1.2.8	codetools_0.2-18
[34]	XML_3.99-0.10	fansi_1.0.3	crayon_1.5.1
[37]	dplyr_1.0.9	dbplyr_2.2.1	GenomicAlignments_1.32.0
[40]	bitops_1.0-7	rappdirs_0.3.3	grid_4.2.1
[43]	lifecycle_1.0.1	DBI_1.1.3	magrittr_2.0.3
[46]	cli_3.3.0	stringi_1.7.8	cachem_1.0.6
[49]	xml2_1.3.3	ellipsis_0.3.2	filelock_1.0.2
[52]	vctrs_0.4.1	generics_0.1.3	rjson_0.2.21
[55]	restfulr_0.0.15	tools_4.2.1	bit64_4.0.5
[58]	BSgenome_1.64.0	glue_1.6.2	purrr_0.3.4
[61]	hms_1.1.1	yaml_2.3.5	parallel_4.2.1
[64]	fastmap_1.1.0	AnnotationDbi_1.58.0	memoise_2.0.1
[67]	BiocIO_1.6.0		