

Overview of `ensemblVEP` Pre Ensembl 90

Valerie Obenchain and Lori Shepherd

March 31, 2022

Contents

1	Introduction	1
2	Results as R objects	1
3	Write results to a file	6
4	Configuring runtime options	7
5	<code>sessionInfo()</code>	8

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.

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

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 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(version=88)
```

```
> param
```

```
class: VEPParam88
identifier(0):
colocatedVariants(0):
dataformat(0):
basic(0):
input(1): species
```

```

cache(3): dir, dir_cache, dir_plugins
output(1): terms
filterqc(0):
database(1): database
advanced(1): buffer_size
version: 88
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	Gene			
		<factor>	<factor>	<factor>	<factor>			
	rs6054257	intergenic_variant	MODIFIER	<NA>	<NA>			
	20:17330_T/A	intergenic_variant	MODIFIER	<NA>	<NA>			
	rs6040355	upstream_gene_variant	MODIFIER	PSMF1	ENSG00000125818			
		Feature_type	Feature	BIOTYPE	EXON			
		<factor>	<factor>	<factor>	<factor>			
	rs6054257	<NA>	<NA>	<NA>	<NA>			
	20:17330_T/A	<NA>	<NA>	<NA>	<NA>			
	rs6040355	Transcript	ENST00000479715	processed_transcript	<NA>			
		INTRON	HGVSc	HGVSp	cDNA_position	CDS_position		
		<factor>	<factor>	<factor>	<factor>	<factor>		
	rs6054257	<NA>	<NA>	<NA>	<NA>	<NA>		
	20:17330_T/A	<NA>	<NA>	<NA>	<NA>	<NA>		
	rs6040355	<NA>	<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  HGNC_ID
      <factor> <factor> <factor>      <factor> <factor>
rs6054257     <NA>      <NA>      <NA>      <NA>      <NA>
20:17330_T/A  <NA>      <NA>      <NA>      <NA>      <NA>
rs6040355     2610      1        <NA>      HGNC HGNC:9571
-----

```

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), version=88)
```

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 6
[[1]] -|intergenic_variant|MODIFIER|
[[2]] deletion|intron_variant&non_coding_transcript_variant&feature_trunctio...
[[3]] deletion|intergenic_variant|MODIFIER|
[[4]] insertion|intron_variant&feature_elongation|MODIFIER|SETD5|ENSG00000168...
[[5]] duplication|upstream_gene_variant|MODIFIER|RAF1|ENSG00000132155|Transcr...
[[6]] duplication|intron_variant&non_coding_transcript_variant&feature_elonga...

```

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
                                     <Rle>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C 1
                                     2:321682_T/<DEL> 2
                                     2:321682_T/<DEL> 2
                                     ranges
                                     <IRanges>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C 2827693-2827762
                                     2:321682_T/<DEL> 321682
                                     2:321682_T/<DEL> 321682

```

		strand
		<Rle>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		*
	2:321682_T/	*
	2:321682_T/	*
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		
	2:321682_T/	
	2:321682_T/	
		VCFRowID
		<integer>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		2
	2:321682_T/	3
	2:321682_T/	3
		Allele
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		-
	2:321682_T/	deletion
	2:321682_T/	deletion
		Consequence
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		intergenic_variant
	2:321682_T/	intron_variant&non_c..
	2:321682_T/	intron_variant&non_c..
		IMPACT
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		MODIFIER
	2:321682_T/	MODIFIER
	2:321682_T/	MODIFIER
		SYMBOL
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	LINC01865
	2:321682_T/	LINC01865
		Gene
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	ENSG00000233684
	2:321682_T/	ENSG00000233684
		Feature_type
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	Transcript
	2:321682_T/	Transcript
		Feature
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	ENST00000430529
	2:321682_T/	ENST00000436808
		BIOTYPE
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	lncRNA
	2:321682_T/	lncRNA
		EXON
		<character>
1:2827693_CCGTGGATGCGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>

	2:321682_T/	<NA>
	2:321682_T/	<NA>
		INTRON
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	1/1
	2:321682_T/	1/3
		HGVSc
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		HGVSp
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		cDNA_position
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		CDS_position
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		Protein_position
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		Amino_acids
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		Codons
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		Existing_variation
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		DISTANCE
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	<NA>
	2:321682_T/	<NA>
		STRAND
		<character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C		<NA>
	2:321682_T/	1
	2:321682_T/	1
		FLAGS

```

1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C      <character>
                                                                <NA>
                                                                2:321682_T/<DEL>      <NA>
                                                                2:321682_T/<DEL>      <NA>
                                                                SYMBOL_SOURCE
                                                                <character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C      <NA>
                                                                2:321682_T/<DEL>      HGNC
                                                                2:321682_T/<DEL>      HGNC
                                                                HGNC_ID
                                                                <character>
1:2827693_CCGTGGATGCGGGGACCCGCATCCCCTCTCCCTTCACAGCTGAGTGACCCACATCCCCTCTCCCCTCGCA/C      <NA>
                                                                2:321682_T/<DEL>      HGNC:52684
                                                                2:321682_T/<DEL>      HGNC:52684

```

```

-----
seqinfo: 4 sequences 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: 28 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):
  List 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"), version=88)
> ## Write a gvf file to myfile.gvf:
> myparam <- VEPParam(dataformat=c(gvf=TRUE),
+                       input=c(output_file="/path/myfile.gvf"), version=88)
> ## Write a tab delimited file to myfile.txt:
> myparam <- VEPParam(input=c(output_file="/path/myfile.txt"), version=88)
```

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 *VEP-Param* 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), version=88)
> 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), version=88)
> 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), version=88)
> 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), version=88)
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 4.1.3 (2022-03-10)
```

```
Platform: x86_64-pc-linux-gnu (64-bit)
```

```
Running under: Ubuntu 20.04.4 LTS
```

```
Matrix products: default
```

```
BLAS: /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
```

```
LAPACK: /home/biocbuild/bbs-3.14-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_1.36.1          VariantAnnotation_1.40.0
[3] Rsamtools_2.10.0           Biostrings_2.62.0
[5] XVector_0.34.0             SummarizedExperiment_1.24.0
[7] Biobase_2.54.0             MatrixGenerics_1.6.0
[9] matrixStats_0.61.0        GenomicRanges_1.46.1
[11] GenomeInfoDb_1.30.1       IRanges_2.28.0
[13] S4Vectors_0.32.4          BiocGenerics_0.40.0
```

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

```
[1] Rcpp_1.0.8.3                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_2.2.1     R6_2.5.1
[10] RSQLite_2.2.11             httr_1.4.2               pillar_1.7.0
[13] zlibbioc_1.40.0            rlang_1.0.2              GenomicFeatures_1.46.5
[16] progress_1.2.2             curl_4.3.2               blob_1.2.2
[19] Matrix_1.4-1               BiocParallel_1.28.3     stringr_1.4.0
[22] RCurl_1.98-1.6            bit_4.0.4                biomaRt_2.50.3
[25] DelayedArray_0.20.0        rtracklayer_1.54.0      compiler_4.1.3
[28] pkgconfig_2.0.3           tidysselect_1.1.2       KEGGREST_1.34.0
[31] tibble_3.1.6               GenomeInfoDbData_1.2.7  XML_3.99-0.9
[34] fansi_1.0.3                crayon_1.5.1            dplyr_1.0.8
[37] dbplyr_2.1.1               GenomicAlignments_1.30.0 bitops_1.0-7
[40] rappdirs_0.3.3            grid_4.1.3              lifecycle_1.0.1
[43] DBI_1.1.2                  magrittr_2.0.3          cli_3.2.0
[46] stringi_1.7.6             cachem_1.0.6            xml2_1.3.3
[49] ellipsis_0.3.2           filelock_1.0.2         vctrs_0.4.0
[52] generics_0.1.2           rjson_0.2.21           restfulr_0.0.13
[55] tools_4.1.3               bit64_4.0.5            BSgenome_1.62.0
[58] glue_1.6.2                purrr_0.3.4            hms_1.1.1
[61] yaml_2.3.5                 parallel_4.1.3         fastmap_1.1.0
[64] AnnotationDbi_1.56.2      memoise_2.0.1          BiocIO_1.4.0
```