Tempus_Assessment_4_4_22

Genelle F Harrison

4/4/2022

Ensembl REST Annotation Pipeline (ERAP) was built to provide a basic quality control assessment as well as annotation information for individual variant call files (VCF). Input data is a VCF file containing variant calls from an indivudual. The vcf_parser function creates an object called parsed_vcf with columns containing the chromosome (CHROM), position (POS), reference allele (REF), alternate allele (ALT), total coverage for region containing the variant (Total_Coverage), and the number of reads supporting the variant (Reads_w_Variant). The percentage of reads carrying the alternate (Percent_Reads_W_Variant) and reference (Percent_Reads_W_Reference) alleles is calculated. A data frame containing variants with multiple segregating alleles (Multi_Allelic_Sites) is created and these sites are filtered out.

Variant sites are annotated using the Ensembl REST API Endpoints database (https://rest.ensembl.org/#VEP). The format of this data is JSON files (nested lists). Each position in the VCF is annotated with the gene in which the variant is found, the severity of the variant (synonymous, intronic, missense, in-frame insertion, frameshift etc), the biotype information (protein coding, pseudogene, nonsense mediated decay etc), the impact of the variant (low, moderate, modifier, high), and the minor allele frequency. "Missing" is printed for fields in which data is unavailable in the Ensembl database. Once the pipeline has finished, the final data frame is written in .csv format to the current working directory as a file names "ERAP_Annotated_VCF.csv". ERAP should be used with R version 4.1.3 and can be used to annotate data mapped with either GRCh37 or GRCh38. The current pipeline uses GRCh37 as the provided test VCF was mapped to GRCh37 (/data/ref_genome/human_g1k_v37.fasta).

This is a beta version of the pipeline. This document walks through each step of this pipeline. The R-notebook must be run with RStudio. The R-notebook and results file can be found at https://github.com/GFHarrison/ERAP

To run the ERAP pipeline in its entirety simply clone the R-notebook from GIT, load the notebook "ERAP_Notebook.Rmd" into R, and follow these steps:

1 – Navigate the directory containing your VCF file 2 – Designate the name of the VCF file (line 11) 3 – Set the reference (line 12) 4 – Hit Run

Depending on the number of variants in the VCF the run time may be a few hours.

Input information

Specify the name of the VCF file to be annotated and the reference to which this VCF was mapped.

```r
vcf_path = paste0(getwd(),"/test_vcf_data.vcf")
Reference = "grch37"
```

Install dependencies

These packages are needed to run ERAP

```r
if (!("reshape2" %in% rownames(installed.packages()))) {
  install.packages("reshape2")
}
```

```r
if (!("data.table" %in% rownames(installed.packages()))) {
  install.packages("data.table")
}
if (!("dplyr" %in% rownames(installed.packages()))) {
  install.packages("dplyr")
}
if (!("tidyr" %in% rownames(installed.packages()))) {
  install.packages("tidyr")
}
if (!("httr" %in% rownames(installed.packages()))) {
  install.packages("httr")
}
if (!("jsonlite" %in% rownames(installed.packages()))) {
  install.packages("jsonlite")
}
if (!("xml2" %in% rownames(installed.packages()))) {
  install.packages("xml2")
}
if (!("ggplot2" %in% rownames(installed.packages()))) {
  install.packages("ggplot2")
}
```

Load dependencies

These packages are needed to run ERAP

```r
library(xml2)
library(ggplot2)
library(reshape2)
library(data.table)
library(dplyr)
library(tidyr)
library(httr)
library(jsonlite)
```

Running the vcf_parser function:

This piece of code will parse the VCF into a useable data frame and will add columns containing the total coverage, total number of reads supporting the variant, the percent of reads supporting the variant and the percent of reads supporting the reference.

```r
Features = c("CHROM", "POS", "REF", "ALT", "INFO")

vcf_parser = function(vcf_path, col_of_interest){
     vcf = fread(vcf_path)
     colnames(vcf)[1]= "CHROM"
     ncols <- max(stringr::str_count(vcf$INFO, ";")) + 1
     vcf_INFO = vcf %>%
     dplyr::select(col_of_interest) %>%
     separate(INFO, paste0("col", 1:ncols), sep= ";") %>%
     separate(col15, c("name", "Total_Coverage"), sep="=") %>%
     separate(col18, c("name", "Reads_w_Variant"), sep="=") %>%
  select(-c("col1","col2", "col3","col4","col5","col6", "col7","col8", "col9",
            "col10", "col11", "col12", "col13", "col14", "col16", "col17", "col19",
            "col20", "name"))
}
parsed_vcf = vcf_parser(vcf_path, Features)

parsed_vcf$Percent_Reads_W_Variant =
  signif(100*(as.numeric(parsed_vcf$Reads_w_Variant) /
                as.numeric(parsed_vcf$Total_Coverage)), digits=4)
parsed_vcf$Percent_Reads_W_Reference =
  signif(100 - as.numeric(parsed_vcf$Percent_Reads_W_Variant),
                                              digits=4)
Multi_Allelic_Sites = parsed_vcf[is.na(parsed_vcf$Percent_Reads_W_Variant),]
parsed_vcf = parsed_vcf[!is.na(parsed_vcf$Percent_Reads_W_Variant),]
```

VCF Annotation using the Ensembl REST API Endpoints database:

A blank dataframe is created to be populated by the loop

```
CHROM = "chromosome"
POS = "position"
id = "missing"
Gene_Symbol = "missing"
Impact = "missing"
BioType = "missing"
Variant_Severity = "Missing"
minor_allele_freq = 0

vcf_ensembl_endpoint_cat =
  as.data.frame(cbind(CHROM, POS, id, Gene_Symbol, Impact, BioType,
                      Variant_Severity, minor_allele_freq))
```

For every variant in the parsed_vcf file, this loop will print the web address containing the JSON file; create a table from the JSON file (vcf_ensembl_endpoint); create a table containing the desired annotation information (vcf_ensembl_endpoint_cat_Interim) and concatinate the annotations into a final data frame (vcf_ensembl_endpoint_cat)

```
for (i in 1:nrow(parsed_vcf)) {
# Build the start frame to add to
CHROM = "chromosome"
POS = "position"
id = 0
Gene_Symbol = "missing"
Impact = "missing"
BioType = "missing"
Variant_Severity = "Missing"
minor_allele_freq = 0

vcf_ensembl_endpoint_cat_Interim = as.data.frame(cbind(CHROM, POS, id, Gene_Symbol, Impact,
  BioType, Variant_Severity, minor_allele_freq))

address = print(paste("https://", Reference, paste(".rest.ensembl.org/vep/human/region/"),
  parsed_vcf$CHROM[i], paste(":"), parsed_vcf$POS[i], paste(":"),parsed_vcf$POS[i],
  paste("/"), parsed_vcf$ALT[i], paste("?")), mirror = "uswest")

address = gsub(" ", "", address)
```

```r
vcf_ensembl_endpoint <- GET(paste(address,sep = ""), content_type("application/json"))
vcf_ensembl_endpoint = data.frame(t(sapply(content(vcf_ensembl_endpoint),c)))
vcf_ensembl_endpoint_cat_Interim$CHROM = parsed_vcf$CHROM[i]
vcf_ensembl_endpoint_cat_Interim$POS = parsed_vcf$POS[i]

### Loading wanted data:
vcf_ensembl_endpoint_trans_consequences =
  data.frame(vcf_ensembl_endpoint$transcript_consequences)
vcf_ensembl_endpoint_pheno_consequences =
  data.frame(vcf_ensembl_endpoint$most_severe_consequence)
vcf_ensembl_endpoint_ID =
  data.frame(vcf_ensembl_endpoint$id)
vcf_ensembl_endpoint_colocated =
  data.frame(vcf_ensembl_endpoint$colocated_variants)

# ID of variant in VCF:
if (nrow(vcf_ensembl_endpoint_ID) == 0) {
  vcf_ensembl_endpoint_cat_Interim$id = "missing"
} else {
  vcf_ensembl_endpoint_cat_Interim$id = vcf_ensembl_endpoint_ID$id
}

# Gene Name:
if (nrow(vcf_ensembl_endpoint_trans_consequences) == 0) {
  vcf_ensembl_endpoint_cat_Interim$Gene_Symbol = "missing"
} else if (!"transcript_consequences.gene_symbol"
           %in% colnames(vcf_ensembl_endpoint_trans_consequences)) {
  vcf_ensembl_endpoint_cat_Interim$Gene_Symbol = "missing"
} else {
  vcf_ensembl_endpoint_cat_Interim$Gene_Symbol =
  vcf_ensembl_endpoint_trans_consequences$transcript_consequences.gene_symbol
}

# Impact of variant:
if (nrow(vcf_ensembl_endpoint_trans_consequences) == 0) {
  vcf_ensembl_endpoint_cat_Interim$Impact = "missing"
} else if (!"transcript_consequences.impact"
  %in% colnames(vcf_ensembl_endpoint_trans_consequences)) {
  vcf_ensembl_endpoint_cat_Interim$Impact = "missing"
} else {
  vcf_ensembl_endpoint_cat_Interim$Impact =
  vcf_ensembl_endpoint_trans_consequences$transcript_consequences.impact
}

# BioType of variant:
if (nrow(vcf_ensembl_endpoint_trans_consequences) == 0) {
  vcf_ensembl_endpoint_cat_Interim$BioType = "missing"
} else if (!"transcript_consequences.biotype"
  %in% colnames(vcf_ensembl_endpoint_trans_consequences)) {
  vcf_ensembl_endpoint_cat_Interim$BioType = "missing"
} else {
  vcf_ensembl_endpoint_cat_Interim$BioType =
  vcf_ensembl_endpoint_trans_consequences$transcript_consequences.biotype
}

# Severity of Variant:
if (nrow(vcf_ensembl_endpoint_pheno_consequences) == 0) {
  vcf_ensembl_endpoint_cat_Interim$Variant_Severity = "missing"
} else if (!"most_severe_consequence" %in% colnames(vcf_ensembl_endpoint_pheno_consequences)) {
  vcf_ensembl_endpoint_cat_Interim$Variant_Severity = "missing"
} else {
  vcf_ensembl_endpoint_cat_Interim$Variant_Severity = vcf_ensembl_endpoint_pheno_consequences$most_seve
}

# Minor Allele frequency
if (nrow(vcf_ensembl_endpoint_colocated) == 0) {
  vcf_ensembl_endpoint_cat_Interim$minor_allele_freq = 0
} else if (!"colocated_variants.minor_allele_freq"
  %in% colnames(vcf_ensembl_endpoint_colocated)) {
  vcf_ensembl_endpoint_cat_Interim$minor_allele_freq = 0
} else {
  vcf_ensembl_endpoint_cat_Interim$minor_allele_freq =
  vcf_ensembl_endpoint_colocated$colocated_variants.minor_allele_freq

}
  vcf_ensembl_endpoint_cat = rbind(vcf_ensembl_endpoint_cat,vcf_ensembl_endpoint_cat_Interim)
}
```

Final annotation file

The annotation table is merged with the parsed_vcf by chromosome and position and the results are written to the current working directory as a .csv file.

```
Final_OutPut = merge(vcf_ensembl_endpoint_cat, parsed_vcf,
  by = c("CHROM", "POS"), all = FALSE)
write.csv(Final_OutPut, paste0(getwd(),"/ERAP_Annotated_VCF.csv"), quote =F, row.names = F)
```

Bioinformatics is hard but you are doing Great!