ACMG EH

Ali Saadat

04/12/2023

Load required packages

```
library(tidyverse)
```

Import data

```
data_anno <- read_delim("data/toy_example/annovar_annotations.txt",
                        col_names = TRUE,
                        na = ".",
                        delim="\t")

data_info <- read_delim("data/toy_example/sample_info.txt",
                        delim="\t",
                        col_select = c(CHROM:NCALLED)) %>%
  dplyr::rename(case_ac=AC, case_af=AF, case_an=AN)

data_vep_csq <- read_delim("data/toy_example/vep_annotations.txt",
                           na=".",
                           col_names = c("CHROM", "POS", "REF", "ALT",  "Allele", "Consequence",
                                         "IMPACT", "SYMBOL", "Gene", "Feature_type", "Feature",
                                         "BIOTYPE", "EXON", "INTRON", "HGVSc", "HGVSp", "cDNA_position"
                                         "CDS_position", "Protein_position", "Amino_acids", "Codons",
                                         "Existing_variation", "DISTANCE", "STRAND", "FLAGS", "SYMBOL_S
                                         "HGNC_ID", "CANONICAL", "MANE_SELECT", "MANE_PLUS_CLINICAL", "
                                         "GENE_PHENO", "DOMAINS", "HGVS_OFFSET", "LoF", "LoF_filter", "
                                         "LoF_info", "SpliceAI_pred_DP_AG", "SpliceAI_pred_DP_AL", "Spl
                                         "SpliceAI_pred_DP_DL", "SpliceAI_pred_DS_AG", "SpliceAI_pred_D
                                         "SpliceAI_pred_DS_DL", "SpliceAI_pred_SYMBOL", "existing_InFra
                                         "existing_OutOfFrame_oORFs", "existing_uORFs",
                                         "five_prime_UTR_variant_annotation", "five_prime_UTR_variant_c

data_GT = read_delim("data/toy_example/sample_GT.txt")

clinvar <- read_delim("data/processed/clinvar_PLP_BLB_CLNSIG_CLNSTAT.tsv.gz")

uniprot_variations <- read_delim("data/processed/humsvar_corrected_withID.txt.gz") %>%
  filter(Variant_category_uniprot=="LP/P") %>%
  mutate(POS_uniprot=str_extract(AAchange_uniprot, "[0-9]+"),
         AA1_uniprot=str_sub(AAchange_uniprot, 3, 5),
         AA2_uniprot=str_sub(AAchange_uniprot, -3, -1))
```

```
pm1_functional_domains_withP <- read_delim("data/processed/uniprot_functional_regions_no_benign_1_patho

pm1_functional_domains <- read_delim("data/processed/uniprot_functional_regions_no_benign.tsv.gz")

PER <- readxl::read_xlsx("data/processed/PER_coordinates.xlsx", sheet = 4)

repeat_masker <- read_delim("data/processed/hg38_rmsk_cleaned.txt.gz", col_names = c("Chr", "Start", "En
  mutate(Start=Start+1) #data from UCSC is 0-based. Add 1 to Start to become 1-based.

bp1_genes <- read_delim("data/processed/BP1_genes.txt.gz", delim="\n") %>%
  mutate(bp1_gene=1)

pp2_genes <- read_delim("data/processed/PP2_genes.txt.gz", delim="\n") %>%
  mutate(pp2_gene=1)
```

```
data <- data_anno %>%
  bind_cols(data_vep_csq) %>%
  bind_cols(data_info %>% dplyr::select(-CHROM, -POS, -REF, -ALT)) %>%
  bind_cols(data_GT %>% dplyr::select(-CHROM, -POS, -REF, -ALT))

rm(data_vep_csq, data_info, data_GT, data_anno)
```

```
data <- data %>%
  filter(IMPACT %in% c("HIGH", "MODERATE")) %>%
  filter(Chr != "chrM") %>%
  filter(VAR >= 1)
```

```
data <- data %>%
  replace_na(list(gnomad41_genome_AF=0,
                  gnomad41_exome_AF=0,
                  gnomad41_genome_AF_nfe=0,
                  gnomad41_genome_AF_afr=0,
                  gnomad41_genome_AF_amr=0,
                  gnomad41_genome_AF_sas=0,
                  gnomad41_genome_AF_eas=0))

data$popmax <- pmax(data$gnomad41_genome_AF,
                    data$gnomad41_exome_AF)

data$spliceai_max <- pmax(as.numeric(data$SpliceAI_pred_DS_AG),
                          as.numeric(data$SpliceAI_pred_DS_AL),
                          as.numeric(data$SpliceAI_pred_DS_DG),
                          as.numeric(data$SpliceAI_pred_DS_DL))
```

ACMG/AMP rules

- **PVS1 and PVS1_strong**

High confidence loss-of-function variants get $PVS1 = 1$ and low confidence loss-of-function variants get $PVS1\_strong = 1$. For both of them, the transcript should be CANONICAL or MANE:

```
data$pvs1 <- 0
data$pvs1_strong <- 0

data[((!is.na(data$MANE_SELECT)) | (!is.na(data$CANONICAL))) &
```

```
        (!is.na(data$LoF)) & data$LoF=="HC", "pvs1"] <- 1

data[((!is.na(data$MANE_SELECT)) | (!is.na(data$CANONICAL))) &
        (!is.na(data$LoF)) & data$LoF=="LC", "pvs1_strong"] <- 1
```

- **PS3**

If the variant is reported in ClinVar (with status of practice_guideline/reviewed_by_expert_pane) or Uniprot, is pathogenic/likely_pathogenic:

```
#clinvar
data <- data %>%
  left_join(clinvar %>% dplyr::select(CHROM:ALT, CLINSIG, CLINSTAT))

data$ps3 <- 0
data <- data %>%
  mutate(ps3 = case_when(str_detect(CLINSIG, "athogenic") &
                         (CLINSTAT %in% c("practice_guideline", "reviewed_by_expert_panel")) ~ 1,
                         TRUE ~ 0))

#uniprot
data <- data %>%
  separate(col=HGVSp, into=c("protein_id", "AAchange_data"), sep=":", remove = F) %>%
  mutate(POS_data=str_extract(AAchange_data, "[0-9]+"),
         AA1_data=str_sub(AAchange_data, 3, 5),
         AA2_data=str_sub(AAchange_data, -3, -1)) %>%
  left_join(uniprot_variations %>% dplyr::select(Gene, POS_uniprot, AA1_uniprot, AA2_uniprot, Variant_ca
            by=c("Gene"="Gene", "POS_data"="POS_uniprot", "AA1_data"="AA1_uniprot", "AA2_data"="AA2_unip

data[!is.na(data$Variant_category_uniprot) & data$Variant_category_uniprot=="LP/P" &
      ( (!str_detect(data$CLINSIG, "enign")) | (is.na(data$CLINSIG)) ), "ps3"] <- 1
```

- **PS1 and PM5**

If variant is missense, and reported as pathogenic in Clinvar/Uniprot, different nucleotide which results in the same amino acid should get PS1. Different nucleotide with different amino acide gets PM5:

```
#clinvar
clinvar_missense <- clinvar %>%
  filter(str_detect(CLINSIG, "athogenic")) %>%
  filter((str_detect(Consequence, "missense") |
          str_detect(Consequence, "start_lost") |
          str_detect(Consequence, "stop_lost") &
          (str_length(REF)==1 & str_length(ALT)==1))) %>%
  dplyr::rename(ALT_clinvar=ALT, Amino_acids_clinvar=Amino_acids) %>%
  dplyr::select(CHROM:ALT_clinvar, Amino_acids_clinvar)


data <- data %>%
  left_join(clinvar_missense)

#uniprot
data <- data %>%
  left_join(uniprot_variations %>%
              dplyr::select(Gene, POS_uniprot, AA1_uniprot, AA2_uniprot, Variant_category_uniprot) %>%
              dplyr::rename(Variant_category_uniprot_pm5=Variant_category_uniprot),
```

```
            by=c("Gene"="Gene", "POS_data"="POS_uniprot", "AA1_data"="AA1_uniprot"))

#ps1
data$ps1 <- 0
data[(!is.na(data$ALT_clinvar)) &
      (data$ALT != data$ALT_clinvar) &
      (data$Amino_acids == data$Amino_acids_clinvar) &
      (data$ps3==0) &
      (data$spliceai_max<0.1 | is.na(data$spliceai_max)) &
      (str_detect(data$Consequence, "missense") | str_detect(data$Consequence, "start_lost") | str_dete
      (str_length(data$REF)==1 & str_length(data$ALT)==1), "ps1"] <- 1

#pm5
data$pm5 <- 0
data[(!is.na(data$ALT_clinvar)) &
      (data$ALT != data$ALT_clinvar) &
      (data$Amino_acids != data$Amino_acids_clinvar) &
      (data$ps3==0) &
      (data$spliceai_max<0.1 | is.na(data$spliceai_max)) &
      (data$ps1==0) &
      (str_detect(data$Consequence, "missense") | str_detect(data$Consequence, "start_lost") | str_dete
      (str_length(data$REF)==1 & str_length(data$ALT)==1), "pm5"] <- 1

data[(!is.na(data$AA2_uniprot)) &
      (data$AA2_data!=data$AA2_uniprot) &
      (data$ps3==0) &
       (data$spliceai_max<0.1 | is.na(data$spliceai_max)) &
      (data$ps1==0) &
      (str_detect(data$Consequence, "missense") | str_detect(data$Consequence, "start_lost") | str_dete
      (str_length(data$REF)==1 & str_length(data$ALT)==1), "pm5"] <- 1
```

- **PM2 (suporting)**

Variant is absent from the controls and is rare in the population:

```
data$pm2_supporting <- 0
data[data$popmax<=0.01, "pm2_supporting"] <- 1
```

- **PM1**

Missense variant located in a functional domain without benign variant, or in a hotspot.

1. Functional domains (active site, binding site, calcium-binding region, DNA-binding region, metal ion-binding site, nucleotide phosphate-binding region, site, and zinc finger region) are retrieved from Uniprot and intersected with ClinVar. For a given domain, if there is no missense benign variant inside the domain, it is considered as functional domain.
2. Hotspots were retreive from "Identification of pathogenic variant enriched regions across genes and gene families" paper

```
data$pm1 <- 0

#domains
domains_input <- rbind(pm1_functional_domains, pm1_functional_domains_withP) %>%
  distinct()

data_ranges <- GenomicRanges::GRanges(seqnames = data$Chr,
```

```
                                    ranges = IRanges::IRanges(start = data$Start, end = data$End))

domains_ranges <- IRanges::reduce(GenomicRanges::GRanges(seqnames = domains_input$Chr,
                                                         ranges = IRanges::IRanges(start = domains_inpu
                                                                                   end = domains_input$

data_domain_overlap <- IRanges::subsetByOverlaps(data_ranges, domains_ranges)

if(length(data_domain_overlap)>0){
  data_domain_df <- data.frame(Chr=GenomicRanges::seqnames(data_domain_overlap),
                               Start=GenomicRanges::start(data_domain_overlap),
                               End=GenomicRanges::end(data_domain_overlap),
                               pm1_domain=1)

  data <- data %>%
    left_join(data_domain_df) %>%
    replace_na(list(pm1_domain=0))

  data[(data$pm1_domain==1) &
       (str_detect(data$Consequence, "missense") |
          str_detect(data$Consequence, "start_lost") |
          str_detect(data$Consequence, "stop_lost") &
       (str_length(data$REF)==1 & str_length(data$ALT)==1)), "pm1"] <- 1
}


#hotspots
PER$Codon <- as.character(PER$Codon)
data <- data %>%
  left_join(PER %>% dplyr::select(Transcript, Codon, Stats), by=c("Feature"="Transcript", "POS_data"="C

data[!is.na(data$Stats) &
     (str_detect(data$Consequence, "missense") |
     str_detect(data$Consequence, "start_lost") |
     str_detect(data$Consequence, "stop_lost")) &
     (str_length(data$REF)==1 & str_length(data$ALT)==1), "pm1"] <- 1
```

- **PM4**

If the variant is inframe indel and falls inside a non-repeat region or if the variant is start_lost/stop_lost. Also, variant should not be a loss-of-function. To detect repeat regions, we use repeat masker from UCSC:

```
data$pm4 <- 0

rmsk_ranges <- GenomicRanges::GRanges(seqnames = repeat_masker$Chr,
                                      ranges = IRanges::IRanges(start = repeat_masker$Start,
                                                                end = repeat_masker$End))

data_rmsk_overlap <- IRanges::subsetByOverlaps(data_ranges, rmsk_ranges)

if(length(data_rmsk_overlap)>0){

  data_rmsk_df <- data.frame(Chr=GenomicRanges::seqnames(data_rmsk_overlap),
                             Start=GenomicRanges::start(data_rmsk_overlap),
                             End=GenomicRanges::end(data_rmsk_overlap),
```

```
                                rmsk=1)

  data <- data %>%
    left_join(data_rmsk_df) %>%
    replace_na(list(rmsk=0))


  for(i in 1:nrow(data)){
    if(!is.na(str_locate(data$Consequence[i], "inframe"))[1] &
       (data$rmsk[i]==0 & (is.na(data$DOMAINS[i]) | !str_detect(data$DOMAINS[i], "ow_complexity"))) &
       is.na(data$LoF[i])){
      data$pm4[i] <- 1
    }
    if(!is.na(str_locate(data$Consequence[i], "stop_lost")[1]) & is.na(data$LoF[i])){
      data$pm4[i] <- 1
    }
    if(!is.na(str_locate(data$Consequence[i], "start_lost")[1]) & is.na(data$LoF[i])){
      data$pm4[i] <- 1
    }
  }
}
```

- **PP2**

If missense variant is inside a gene depleted from pathogenic missense variants. To consider a gene as a gene deplted from pathogenic missense, two criteria must be satisfied for it:

1. $\frac{pathogenic\_missense}{pathogenic\_total} >= 0.7$
2. $\frac{benign\_missense}{total\_missense} <= 0.3$
3. at least 5 pathogenic missense are reported

```
data$pp2 <- 0

data[(data$Gene %in% pp2_genes$Gene) &
     (data$pm1==0) &
     (str_detect(data$Consequence, "missense") |
        str_detect(data$Consequence, "start_lost") |
        str_detect(data$Consequence, "stop_lost") &
     (str_length(data$REF)==1 & str_length(data$ALT)==1)), "pp2"] <- 1
```

- **PP3 and BP4**

Computational prediction. PP3 is applied when $REVEL >= 0.5$ and BayesDel_noAF_pred predicts damaging

BP4 is applied if $REVEL < 0.4$ or BayesDel_noAF_pred predicts tolerated

```
data$pp3 <- 0
data[(!is.na(data$REVEL_score) & data$REVEL_score>=0.5) &
      (!is.na(data$BayesDel_noAF_pred) & data$BayesDel_noAF_pred=="D"), "pp3"] <- 1

data$bp4 <- 0
data[(!is.na(data$REVEL_score) & data$REVEL_score<0.4)|
       (!is.na(data$BayesDel_noAF_pred) & data$BayesDel_noAF_pred=="T"), "bp4"] <- 1
```

- **BA1**

If a variant is very common (>0.05) this rule is applied, and variant is considered as benign.

```
data$ba1 <- 0
data[data$gnomad41_genome_AF_nfe>0.05 |
     data$gnomad41_genome_AF_afr>0.05 |
     data$gnomad41_genome_AF_amr>0.05 |
     data$gnomad41_genome_AF_sas>0.05 |
     data$gnomad41_genome_AF_eas>0.05, "ba1"] <- 1
```

- **BS1**

If a variant is common (>0.02):

```
data$bs1 <- 0
data[data$gnomad41_genome_AF_nfe>0.02 |
     data$gnomad41_genome_AF_afr>0.02 |
     data$gnomad41_genome_AF_amr>0.02 |
     data$gnomad41_genome_AF_sas>0.02 |
     data$gnomad41_genome_AF_eas>0.02, "bs1"] <- 1
```

- **BS3**

Reported as benign in ClinVar and its status is practice_guideline/reviewed_by_expert_panel:

```
data$bs3 <- 0
data <- data %>%
  mutate(bs3 = case_when(str_detect(CLINSIG, "enign") &
                           (CLINSTAT %in% c("practice_guideline",
                                            "reviewed_by_expert_panel")) ~ 1,
                         TRUE ~ 0))
```

- **BP1**

Missense variant falls inside a gene which damaging mutations are mostly truncating. To detect such genes, we do the following steps:

1. Count pathogenic truncating ('frameshift_variant', 'splice_acceptor_variant', 'splice_donor_variant', 'stop_gained') variants for each gene
2. Count total pathogenic variants for each gene (at least 5 pathogenic must be found in a gene)
3. Keep genes that $\frac{truncating\_pathogenic}{total\_pathogenic} >= 0.7$

```
data$bp1 <- 0
data[(data$Gene %in% bp1_genes$Gene) &
       (str_detect(data$Consequence, "missense") |  str_detect(data$Consequence, "start_lost") | str_de
     , "bp1"] <- 1
```

- **BP3**

Variant is an inframe indel and falls inside a repeat region:

```
data$bp3 <- 0
if(length(data_rmsk_overlap)>0){
  for(i in 1:nrow(data)){
    if(!is.na(str_locate(data$Consequence[i], "inframe"))[1] &
       (data$rmsk[i]==1 | (str_detect(data$DOMAINS[i], "ow_complexity") & !is.na(data$DOMAINS[i])) ) ){
      data$bp3[i] <- 1
    }
  }
}
```

Pathogenicity probability

We use the recommended model in "modeling the ACMG/AMP variant classification guidelines as a Bayesian classification framework" to assign a pathogenicity probability to each variant:

```
data$probability <- 0
prior <- 0.1
for (i in 1:nrow(data)){
  PVS <- data$pvs1[i]
  PS <- data$ps1[i]+data$ps3[i]+data$pvs1_strong[i]
  PM <- data$pm1[i]+data$pm4[i]+data$pm5[i]
  PP <- data$pp2[i]+data$pp3[i]+data$pm2_supporting[i]
  BS <- data$bs1[i]+data$bs3[i]
  BP <- data$bp1[i]+data$bp3[i]+data$bp4[i]
  if (data$ba1[i]==1){
    data$probability[i] <- 0
  }
  else{
    odds_path <- 350^((PP/8)+(PM/4)+(PS/2)+(PVS/1)-(BP/8)-(BS/2))
    data$probability[i] <- (odds_path*prior)/(((odds_path-1)*prior) + 1)
  }
}

### to remove duplicated rows (for a given variant, choose the maximum probability)
temp <- data %>%
  group_by(CHROM, POS, REF, ALT) %>%
  summarize(probability=max(probability))

data <- data %>%
  dplyr::select(-probability) %>%
  left_join(temp) %>%
  distinct(CHROM, POS, REF, ALT, probability, .keep_all = T)
```

Extract Pathogenic Variants

The recommended threshold is 0.9, but there are some variants whose probability of pathogenicity is 0.899. We consider them as pathogenic as well.

```
### extract pathogenic
data_patho <- data %>%
  filter(probability>=0.89)
```