# Populate DB

Oksana Sorokin

10.10.2022

## 1 Make database

```
## <SQLiteResult>
##  SQL  CREATE TABLE BrainRegion (   ID       INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,   Name
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE Gene (   ID       INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,   MGI      varch
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE Localisation (   ID       INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,   Name
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE Method (   ID       INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,   Name     va
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE Paper (   PMID      numeric(19, 0) NOT NULL,   Year      integer(10) NOT NULL,   Name     va
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE PaperGene (   GeneID      integer(10) NOT NULL,   PaperPMID     numeric(19, 0) NOT NULL,
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE PPI (   ID    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,   A     integer(10) N
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE Species (   TaxID   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,   Name   varc
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##   SQL  CREATE UNIQUE INDEX GeneUI     ON Gene (HumanEntrez, MouseEntrez);
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
## <SQLiteResult>
##  SQL  CREATE TABLE GO (   GOID      varchar(255) NOT NULL,   Term      varchar(255) NOT NULL,   Domain     va
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE GOGene (   GeneID   integer(10) NOT NULL,   SpecieID integer(10) NOT NULL,   GOID    varc
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE Disease (   HDOID      varchar(255) NOT NULL,   Description varchar(255),   PRIMARY KEY (
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE DiseaseGene (   GeneID integer(10) NOT NULL,   HDOID  varchar(255) NOT NULL,   FOREIG
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE SpecieRegion (   BrainRegionID integer(10) NOT NULL,   TaxID      integer(10) NOT NULL,
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE Mutation ( ID         INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  GeneID
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE PaperMutation ( PMID     NUMERIC(19, 0) NOT NULL,  MutationID integer(10) NOT NULL,
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE TABLE PaperPPI ( PMID NUMERIC(19, 0) NOT NULL,  PPID    integer(10) NOT NULL,  FOREIGN
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE VIEW FullGenePaper AS SELECT p.GeneID,LocalisationID, MGI,HumanEntrez,MouseEntrez,HumanN
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE VIEW FullGenefullPaper AS SELECT p.GeneID,l.Name AS Localisation,  MGI,HumanEntrez,MouseEnt
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE VIEW FullGeneFullPaperFullRegion AS    SELECT p.GeneID,        l.Name AS Localisation,        MGI
##   ROWS Fetched: 0 [complete]
##        Changed: 0

## <SQLiteResult>
##  SQL  CREATE VIEW FullGeneFullDisease AS    SELECT HumanEntrez,        HumanName,        d.HDOID,        d.D
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

# 2 Populate database

## 2.1 Method

```
method.df<-data.frame(ID=1:2,
                      Name=c("Shotgun","Target"),
                      Description=c("Shotgun","Target"))
dbWriteTable(mydb, "method", method.df,append=TRUE)
```

```
## Warning: Closing open result set, pending rows
```

## 2.2 Species

```
species.df<-data.frame(TaxID=c(9606,10090,10116),
                       Name=c("human","mouse","rat"),
                       SciName=c("Homo sapiens",
                                 "Mus musculus",
                                 "Rattus norvegicus"))
dbWriteTable(mydb, "species", species.df,append=TRUE)
```

## 2.3 Brain Regions

```
brainReg.df <- read.table("BrainRegions_Oct22.txt", sep = "\t", header = TRUE, stringsAsFactors = FALSE )
brp <- read.delim("BrainRegPapers_Oct22.txt", sep = "\t", header = T, stringsAsFactors = FALSE)
idxBR <- match(brp$Name, brainReg.df$Name)

dbWriteTable(mydb, "BrainRegion", brainReg.df,append=TRUE)
```

## 2.4 SpecieRegion

```
sbr <- read.delim("BrainRegionSpecie_Oct22.txt", sep = "\t", header = TRUE, stringsAsFactors = FALSE)
dbWriteTable(mydb, "SpecieRegion", sbr,append=TRUE)
```

## 2.5 Localisation

```
loc.df<-data.frame(ID=1:4,
                   Name=c("Postsynaptic",
                          "Presynaptic",
                          "Synaptosome",
                          "Synaptic_Vesicle"),
                   Description=c("Postsynaptic",
                          "Presynaptic",
                          "Synaptosome",
                          "Synaptic_Vesicle"))
dbWriteTable(mydb, "localisation", loc.df,append=TRUE)
```

## 2.6 Papers

```
#papers<-read.delim("Paper_DB_summary.txt",sep='\t',header = TRUE, stringsAsFactors = FALSE)
papers<-read.delim("Paper_DB_summary_Oct22.txt",sep='\t',header = TRUE, stringsAsFactors = FALSE)
```

```
pmed <- read.csv("pubmed.named.csv", stringsAsFactors = FALSE)
any(papers$PubMed %in% pmed$PMID)
```

```
## [1] FALSE
```

```
pmed.df <- unique(pmed[,c("PMID", "Year","Name")])
names(pmed.df) <- c("PMID","Year","Name")
pmed.df$Description <- NA
# p.fq<-as.data.frame(table(pmed.df$Name))
# p.fq<-p.fq[p.fq$Freq>1,]
# for(nm in p.fq$Var1){
#   idx.pfq<-which(pmed.df$Name==nm)
#   pmed.df$Name[idx.pfq]<-paste0(pmed.df$Name[idx.pfq],letters[1:length(idx.pfq)])
# }
# pmed.df$Name[which(pmed.df$Name %in% papers$Name)]<-paste0(
#   pmed.df$Name[which(pmed.df$Name %in% papers$Name)], 'a')
p.df<-unique(papers[,c('PubMed','Year','Name','Short.description')])
names(p.df)<-c("PMID","Year","Name","Description")
p.df <- rbind(p.df, pmed.df)
p.df<-p.df[!is.na(p.df$Year),]
papers$taxId<-species.df$TaxID[match(papers$Species,species.df$Name)]
papers$methodId<-2
papers$methodId[papers$shotgun=="YES"]<-1
dbWriteTable(mydb, "paper", p.df,append=TRUE)
```

## 2.7 Genes

### 2.7.1 Genes table

```
full <- read.delim("Full_DB_Rat_Oct22.txt", sep = "\t", header = T, stringsAsFactors = FALSE)
full$ID<-1:dim(full)[1]
fg.df<- full[,c(dim(full)[2],1:7)]
names(fg.df)<-c('ID',
                'MGI',
                'MouseEntrez',
                'MouseName',
                'HumanEntrez',
                'HumanName',
                'RatEntrez',
                'RatName')
dbWriteTable(mydb, "gene", fg.df,append=TRUE)
fg.df$surkey<-paste(fg.df$MouseEntrez,
                    fg.df$HumanEntrez,
                    sep = ":")
```

### 2.7.2 Postsynaptic

```
#gene1<-read.delim("PSD_db_Sept19.txt",sep = '\t',header=TRUE, stringsAsFactors = FALSE)
gene1<-read.delim("PSD_db_Oct22.txt",sep ='\t',header=TRUE, stringsAsFactors = FALSE)
gene1 <- gene1[, -c(dim(gene1)[2])]
g1.df<-gene1[,1:5]
names(g1.df)<-c('mgi',
                'mouseentrez',
```

```
                'mousename',
                'humanentrez',
                'humanname')
surKey<-paste(g1.df$mouseentrez,g1.df$humanentrez,sep=":")
idx<-match(surKey,fg.df$surkey)
g1.df$id<-fg.df$ID[idx]
gene1$id<-fg.df$ID[idx]
mg1<-melt(gene1[,c(6:dim(gene1)[2])],id="id")
mg1<-mg1[mg1$value==1,]
mg1$locID=1
idx<-match(mg1$variable,p.df$Name)
mg1$pmid<-p.df$PMID[idx]
mg1$dataset<-'FULL'
mg1$taxId<-papers$taxId[idx]
mg1$methodId<-papers$methodId[idx]
l <- list()
for (i in 1:dim(brp)[1]){
  if (any(mg1$variable == brp$Paper[i])){
  mgt <- mg1[mg1$variable == brp$Paper[i],]
  mgt$BrainRegionID <- idxBR[i]
  l[[length(l)+1]] <- mgt
  }
}
mag1 <-do.call(rbind,l)
```

### 2.7.3 Presynaptic

```
gene2<-read.delim("Pres_DB_Oct22.txt",sep ='\t',header=TRUE, stringsAsFactors = FALSE)
gene2 <- gene2[, -c(dim(gene2)[2])]
g2.df<-gene2[,c("MGI.ID",
                "MOUSE.ENTREZ.ID",
                "MOUSE.GENE.NAME",
                "HUMAN.ENTREZ.ID",
                "HUMAN.GENE.NAME")]
names(g2.df)<-c('mgi',
                'mouseentrez',
                'mousename',
                'humanentrez',
                'humanname')
surKey<-paste(g2.df$mouseentrez,g2.df$humanentrez,sep=":")
idx<-match(surKey,fg.df$surkey)
g2.df$id<-fg.df$ID[idx]
gene2$id<-fg.df$ID[idx]
mg2<-melt(gene2[,c(11:dim(gene2)[2])],id="id")
mg2<-mg2[mg2$value==1,]
mg2$locID=2
idx<-match(mg2$variable,p.df$Name)
mg2$pmid<-p.df$PMID[idx]
mg2$dataset<-'FULL'
mg2$taxId<-papers$taxId[idx]
mg2$methodId<-papers$methodId[idx]
l <- list()
for (i in 1:dim(brp)[1]){
```

```
  if (any(mg2$variable == brp$Paper[i])){
  mgt <- mg2[mg2$variable == brp$Paper[i],]
  mgt$BrainRegionID <- idxBR[i]
  l[[length(l)+1]] <- mgt
  }
}
mag2 <-do.call(rbind,l)
```

#### 2.7.4 Synaptosome

```
gene3<-read.delim("Syn_Oct22.txt",sep ='\t',header=TRUE, stringsAsFactors = FALSE)
g3.df<-gene3[,1:4]
names(g3.df)<-c('mouseentrez',
                'mousename',
                'humanentrez',
                'humanname')
surKey<-paste(g3.df$mouseentrez,g3.df$humanentrez,sep=":")
idx<-match(surKey,fg.df$surkey)
g3.df$id<-fg.df$ID[idx]
gene3$id<-fg.df$ID[idx]
mg3<-melt(gene3[,c(5:dim(gene3)[2])],id="id")
mg3<-mg3[mg3$value==1,]
mg3$locID=3
idx<-match(mg3$variable,p.df$Name)
mg3$pmid<-p.df$PMID[idx]
mg3$dataset<-'FULL'
mg3$taxId<-papers$taxId[idx]
mg3$methodId<-papers$methodId[idx]
l <- list()
for (i in 1:dim(brp)[1]){
    if (any(mg3$variable == brp$Paper[i])){
        if('KOOPMANS_2018' == brp$Paper[i]){
            for(txid in species.df$TaxID){
            mgt <- mg3[mg3$variable == brp$Paper[i],]
            mgt$BrainRegionID <- idxBR[i]
            mgt$taxId <- txid
            l[[length(l)+1]] <- mgt
            }
        }else{
            mgt <- mg3[mg3$variable == brp$Paper[i],]
            mgt$BrainRegionID <- idxBR[i]
            l[[length(l)+1]] <- mgt
        }
    }
}
mag3 <-do.call(rbind,l)
```

#### 2.7.5 SV

```
gene4<-read.delim("SV_Oct22.txt",sep ='\t',header=TRUE, stringsAsFactors = FALSE)
gene4 <- gene4[, -c(dim(gene4)[2])]
g4.df<-gene4[,c("MGI.ID",
```

```
                "MOUSE.ENTREZ.ID",
                "MOUSE.GENE.NAME",
                "HUMAN.ENTREZ.ID",
                "HUMAN.GENE.NAME")]
names(g4.df)<-c('mgi',
                'mouseentrez',
                'mousename',
                'humanentrez',
                'humanname')
surKey<-paste(g4.df$mouseentrez,g4.df$humanentrez,sep=":")
idx<-match(surKey,fg.df$surkey)
g4.df$id<-fg.df$ID[idx]
gene4$id<-fg.df$ID[idx]
mg4<-melt(gene4[,c(8:dim(gene4)[2])],id="id")
mg4<-mg4[mg4$value==1,]
mg4$locID=4
mg4$dataset<-'FULL'
idx<-which(mg4$variable=='TAOUFIQ_2020_SV_RESIDENTS')
mg4$dataset[idx]<-'SV_RESIDENTS'
mg4$variable[idx]<-'TAOUFIQ_2020'
idx<-match(mg4$variable,p.df$Name)
mg4$pmid<-p.df$PMID[idx]
mg4$taxId<-papers$taxId[idx]
mg4$methodId<-papers$methodId[idx]
l <- list()
for (i in 1:dim(brp)[1]){
  if (any(mg4$variable == brp$Paper[i])){
  mgt <- mg4[mg4$variable == brp$Paper[i],]
  mgt$BrainRegionID <- idxBR[i]
  l[[length(l)+1]] <- mgt
  }
}
mag4 <-do.call(rbind,l)
```

#### 2.7.6 COmbine all localisation

```
totGene<-rbind(mag1[,c("id","locID","pmid","dataset",
                       "taxId","methodId", 'BrainRegionID')],
               mag2[,c("id","locID","pmid","dataset",
                       "taxId","methodId", 'BrainRegionID')],
               mag3[,c("id","locID","pmid","dataset",
                       "taxId","methodId", 'BrainRegionID')],
               mag4[,c("id","locID","pmid","dataset",
                       "taxId","methodId", 'BrainRegionID')])
names(totGene)<-c("GeneID","LocalisationID","PaperPMID","Dataset","SpeciesTaxID",
                  "MethodID", 'BrainRegionID')
totGene<-totGene[,c("GeneID",
"PaperPMID",
"Dataset",
"SpeciesTaxID",
"BrainRegionID",
"LocalisationID",
"MethodID")]
```

```
dbWriteTable(mydb, "papergene", totGene,append=TRUE)
```

# 3 PPI

```
#ppi.df<-read.delim("PPI_DB_Homo.txt",sep = "\t", header = TRUE, stringsAsFactors = FALSE)
ppi.df<-read.delim("PPI_DB_Oct22.txt",sep = "\t", header = TRUE, stringsAsFactors = FALSE)
idxA<-match(ppi.df$entA,fg.df$HumanEntrez)
idxB<-match(ppi.df$entB,fg.df$HumanEntrez)
ppi.df$A<-fg.df$ID[idxA]
ppi.df$B<-fg.df$ID[idxB]
ppi.df$ID<- 1:dim(ppi.df)[1]
ppi.t<-ppi.df[,c('ID','A','B','type','method')]
names(ppi.t)<-c('ID','A','B','type','method')
dbWriteTable(mydb, "ppi", ppi.t,append=TRUE)
```

```
# pmidx<-match(ppi.df$pmid,papers$PubMed)
# idx<-which(!is.na(pmidx))
# pap.ppi<-data.frame(PMID=papers$PubMed[pmidx[idx]],PPID=ppi.df$ID[idx])
pmidx<-match(ppi.df$pmid,pmed$PMID)
idx<-which(is.na(pmidx))
npmid<-as.numeric(ppi.df$pmid[idx])
```

```
## Warning: NAs introduced by coercion
```

```
nnaidx<-which(!is.na(npmid))
if(length(nnaidx)>0){
npmid.df<-data.frame(PMID=npmid[idx[nnaidx]],
                     Year=0,
                     Name=as.character(npmid[idx[nnaidx]]),
                     Description=NA)
dbWriteTable(mydb, "paper", npmid.df,append=TRUE)
p.df<-rbind(p.df,npmid.df)
}
pmidx<-match(ppi.df$pmid,p.df$PMID)
idx<-which(is.na(pmidx))
pap.ppi<-data.frame(PMID=ppi.df$pmid[-idx],PPID=ppi.df$ID[-idx])
dbWriteTable(mydb, "PaperPPI", pap.ppi,append=TRUE)
length(which(is.na(pmidx)))
```

```
## [1] 88
```

# 4 GO

```
orgDB<-org.Hs.eg.db
keytype <- "ENTREZID"
keys<-fg.df$HumanEntrez
keys<-as.character(keys[!is.na(keys)])
on <- AnnotationDbi::select(orgDB, keys,
                            columns = c("GO",'ONTOLOGY'),
                            keytype = keytype)
```

```
## 'select()' returned many:many mapping between keys and columns
```

```
on<-on[!is.na(on$GO),]
gogene.hs<-unique(on[,c('ENTREZID','GO','EVIDENCE')])
names(gogene.hs)<-c('GeneID','GOID','Evidence')
gogene.hs$SpecieID<-9606
go.hs<-AnnotationDbi::select(GO.db,
                             unique(on$GO),
                             column=c('TERM',"ONTOLOGY"),
                             keytype='GOID')
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
names(go.hs)<-c('GOID','Term','Domain')
```

```
orgDB<-org.Mm.eg.db
keytype <- "ENTREZID"
keys<-fg.df$MouseEntrez
keys<-as.character(keys[!is.na(keys)])
on <- AnnotationDbi::select(orgDB, keys,
                            columns = c("GO",'ONTOLOGY'),
                            keytype = keytype)
```

```
## 'select()' returned many:many mapping between keys and columns
```

```
on<-on[!is.na(on$GO),]
gogene.mm<-unique(on[,c('ENTREZID','GO','EVIDENCE')])
names(gogene.mm)<-c('GeneID','GOID','Evidence')
gogene.mm$SpecieID<-10090
go.mm<-AnnotationDbi::select(GO.db,
                             unique(on$GO),
                             column=c('TERM',"ONTOLOGY"),
                             keytype='GOID')
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
names(go.mm)<-c('GOID','Term','Domain')
```

```
orgDB<-org.Rn.eg.db
keytype <- "ENTREZID"
keys<-fg.df$RatEntrez
keys<-as.character(keys[!is.na(keys)])
on <- AnnotationDbi::select(orgDB, keys,
                            columns = c("GO",'ONTOLOGY'),
                            keytype = keytype)
```

```
## 'select()' returned many:many mapping between keys and columns
```

```
on<-on[!is.na(on$GO),]
gogene.rn<-unique(on[,c('ENTREZID','GO','EVIDENCE')])
names(gogene.rn)<-c('GeneID','GOID','Evidence')
gogene.rn$SpecieID<-10116
go.rn<-AnnotationDbi::select(GO.db,
                             unique(on$GO),
                             column=c('TERM',"ONTOLOGY"),
                             keytype='GOID')
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
names(go.rn)<-c('GOID','Term','Domain')
```

```
df.go <- unique(rbind(go.hs,go.mm,go.rn))
dbWriteTable(mydb, "GO", df.go, append=TRUE)
```

# 5 GoGene

```
gogene<-unique(rbind(gogene.hs,gogene.mm,gogene.rn))
gogene<-gogene[,c("GeneID", "SpecieID","GOID","Evidence")]
dbWriteTable(mydb, "GOGene", gogene, append=TRUE)
```

# 6 Disease

```
hdo <- read.csv("flatfile_human_HDO.csv", sep = "\t", header = FALSE, stringsAsFactors = FALSE)
hdo <- hdo[, c(1,2)]
hdoU <- unique(hdo)
names(hdoU) <- c("HDOID","Description")
dbWriteTable(mydb, "Disease", hdoU, append=TRUE)
```

# 7 DiseaseGene

```
hdo <- read.csv("flatfile_human_HDO.csv", sep = "\t", header = FALSE, stringsAsFactors = FALSE)
idx <- match(hdo$V3, fg.df$HumanEntrez)
hdo$GeneID <- fg.df$ID[idx]
hdog <- hdo[, c(4,1)]
names(hdog) <- c("GeneID", "HDOID")
dbWriteTable(mydb, "DiseaseGene", hdog, append=TRUE)
```

# 8 Mutations

## 8.1 Autism related mutation

Gene table contains two copies of GSTM1 HLA-A, so we decided that mapping of first row to mutations will be enough for our purposes.

{r prepare.asd.mut.table}
mut.asd <- read.csv("ASD_mut_combined_Oct22.txt",sep='\t', stringsAsFactors = FALSE)
idx <- match(mut.asd$Gene, fg.df$HumanName)
mut.asd$GeneID <- fg.df$ID[idx]
mut.asd$HDOID <- "DOID:0060041"
mut.asd.df <- unique(mut.asd[!is.na(mut.asd$GeneID), c("GeneID", "HDOID","Chr", "Position", "Variant","FunctionClass", "
mut.asd.df$ID <- 1:dim(mut.asd.df)[1]
mut.asd.df$EpilepsyGene <- "NO"
mut.asd.df <- mut.asd.df[, c("ID","GeneID", "HDOID","Chr", "Position", "Variant","FunctionClass", "cDnaVariant", "Protein
names(mut.asd.df) <- c("ID","GeneID", "HDOID","Chromosome", "Position", "Variant","FunctionClass", "cDNAvariant", "P

## 8.2 Epilepsy related mutation

{r prepare.epi.mut.table}

```
mut.epi <- read.csv("Epi_combined_Oct22.txt",sep='\t', stringsAsFactors = FALSE)
idx <- match(mut.epi$Human_Entrez, fg.df$HumanEntrez)
mut.epi$GeneID <- fg.df$ID[idx]
mut.epi$HDOID <- "DOID:1826"
mut.epi.df <- unique(mut.epi[!is.na(mut.epi$GeneID), c("GeneID", "HDOID","Chr", "Position", "Variant","FunctionClass", "c
mut.epi.df$ID <- 1:dim(mut.epi.df)[1]
mut.epi.df$SFARI <- "NO"
mut.epi.df <- mut.epi.df[, c("ID","GeneID", "HDOID","Chr", "Position", "Variant","FunctionClass", "cDnaVariant", "ProteinV
names(mut.epi.df) <- c("ID","GeneID", "HDOID","Chromosome", "Position", "Variant","FunctionClass", "cDNAvariant", "Pr
```

```
mut.df<-read.csv("Mut_combined.txt",sep=',', stringsAsFactors = FALSE)
idx <- match(mut.df$Human_Entrez, fg.df$HumanEntrez)
mut.df$GeneID <- fg.df$ID[idx]
mut.df <- mut.df[, c("ID","GeneID", "HDOID","Chr", "Position", "Variant","FunctionClass", "cDnaVariant", "ProteinVar
names(mut.df) <- c("ID","GeneID", "HDOID","Chromosome", "Position", "Variant","FunctionClass", "cDNAvariant", "P
dbWriteTable(mydb, "Mutation", mut.df, append=TRUE)
```

```
   mpub<-read.csv("Mut_PMID.txt",sep=',', stringsAsFactors = FALSE)
papmut.df <- mpub[,c(2,1)]
names(papmut.df) <- c("PMID", "MutationID")
dbWriteTable(mydb, "PaperMutation", papmut.df, append=TRUE)
```

# 9 Close database

```
dbDisconnect(mydb)
```

# 10 Appendix

## 10.1 Functions

```
ddlBR<-paste("CREATE TABLE BrainRegion (",
"  ID          INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
"  Name        varchar(255) NOT NULL UNIQUE, ",
"  Description varchar(4255), ",
"  InterlexID   varchar(255), ",
"  ParentID    integer(10) , ",
"  FOREIGN KEY(ParentID) REFERENCES BrainRegion(ID));")
ddlG<-paste("CREATE TABLE Gene (",
"  ID          INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
"  MGI         varchar(255), ",
"  HumanEntrez integer(10), ",
"  MouseEntrez integer(10), ",
"  HumanName   varchar(255), ",
"  MouseName   varchar(255), ",
"  RatEntrez   integer(10), ",
"  RatName     varchar(255));")
ddlL<-paste("CREATE TABLE Localisation (",
"  ID          INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
"  Name        varchar(255) UNIQUE, ",
"  Description varchar(4255));")
ddlM<-paste("CREATE TABLE Method (",
"  ID          INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
```

```
"  Name        varchar(255) NOT NULL UNIQUE, ",
"  Description varchar(4255));")
ddlP<-paste("CREATE TABLE Paper (",
"  PMID        numeric(19, 0) NOT NULL, ",
"  Year        integer(10) NOT NULL, ",
"  Name        varchar(255) NOT NULL UNIQUE, ",
"  Description varchar(1255), ",
"  PRIMARY KEY (PMID));")
ddlPG<-paste("CREATE TABLE PaperGene (",
"  GeneID         integer(10) NOT NULL, ",
"  PaperPMID      numeric(19, 0) NOT NULL, ",
"  Dataset    varchar(255) NOT NULL, ",
"  SpeciesTaxID   integer(10) NOT NULL, ",
"  BrainRegionID  integer(10) NOT NULL, ",
"  LocalisationID integer(10) NOT NULL, ",
"  MethodID       integer(10) NOT NULL, ",
"  PRIMARY KEY (GeneID, ",
"  PaperPMID, ",
"  Dataset, ",
"  SpeciesTaxID, ",
"  BrainRegionID, ",
"  LocalisationID), ",
"  FOREIGN KEY(GeneID) REFERENCES Gene(ID), ",
"  FOREIGN KEY(PaperPMID) REFERENCES Paper(PMID), ",
"  FOREIGN KEY(SpeciesTaxID) REFERENCES Species(TaxID), ",
"  FOREIGN KEY(BrainRegionID) REFERENCES BrainRegion(ID), ",
"  FOREIGN KEY(LocalisationID) REFERENCES Localisation(ID), ",
"  FOREIGN KEY(MethodID) REFERENCES Method(ID));")
ddlPPI<-paste("CREATE TABLE PPI (",
"  ID     INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
"  A      integer(10) NOT NULL, ",
"  B      integer(10) NOT NULL, ",
"  type   varchar(255) NOT NULL, ",
"  method varchar(255) NOT NULL, ",
#"  pmid   integer(10), ",
#"  taxID  integer(10) NOT NULL, ",
"  FOREIGN KEY(A) REFERENCES Gene(ID), ",
"  FOREIGN KEY(B) REFERENCES Gene(ID));")
ddlS<-paste("CREATE TABLE Species (",
"  TaxID   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
"  Name    varchar(255) NOT NULL UNIQUE, ",
"  SciName varchar(255));")
ddlUGE<-paste("CREATE UNIQUE INDEX GeneUI ",
"  ON Gene (HumanEntrez, MouseEntrez);")
ddlGO <- paste("CREATE TABLE GO (",
"  GOID        varchar(255) NOT NULL, ",
"  Term        varchar(255) NOT NULL, ",
"  Domain      varchar(255) NOT NULL, ",
"  PRIMARY KEY (GOID));")
ddlGOGene <- paste("CREATE TABLE GOGene (",
"  GeneID   integer(10) NOT NULL, ",
"  SpecieID integer(10) NOT NULL, ",
"  GOID     varchar(255) NOT NULL, ",
```

```
"  Evidence varchar(255) NOT NULL, ",
"  FOREIGN KEY(GeneID) REFERENCES Gene(ID), ",
"  FOREIGN KEY(SpecieID) REFERENCES Species(TaxID), ",
"  FOREIGN KEY(GOID) REFERENCES GO(GOID));")
ddlD <- paste("CREATE TABLE Disease (",
"  HDOID       varchar(255) NOT NULL, ",
"  Description varchar(255), ",
"  PRIMARY KEY (HDOID));")
ddlDG <- paste("CREATE TABLE DiseaseGene (",
"  GeneID integer(10) NOT NULL, ",
"  HDOID  varchar(255) NOT NULL, ",
"  FOREIGN KEY(GeneID) REFERENCES Gene(ID), ",
"  FOREIGN KEY(HDOID) REFERENCES Disease(HDOID));")
ddlMO <- paste("CREATE TABLE GeneToModel (",
"  GeneID integer(10) NOT NULL, ",
"  EntityID varchar(255) NOT NULL, ",
"  PMID numeric(19, 0) NOT NULL, ",
"  FOREIGN KEY(GeneID) REFERENCES Gene(ID), ",
"  FOREIGN KEY(PMID) REFERENCES Paper(PMID));")
ddlBRS <- paste("CREATE TABLE SpecieRegion (",
"  BrainRegionID integer(10) NOT NULL, ",
"  TaxID         integer(10) NOT NULL, ",
"  FOREIGN KEY(TaxID) REFERENCES Species(TaxID), ",
"  FOREIGN KEY(BrainRegionID) REFERENCES BrainRegion(ID));")
ddlMut <- paste("CREATE TABLE Mutation (",
  "ID             INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, ",
  "GeneID         integer(10) NOT NULL, ",
  "HDOID          varchar(255) NOT NULL, ",
  "Chromosome     varchar(2), ",
  "Position       integer(10), ",
  "Variant        text, ",
  "FunctionClass  text NOT NULL, ",
  "cDNAvariant    text NOT NULL, ",
  "ProteinVariant text, ",
  "ExonIntron     text, ",
  "DENOVO         boolean DEFAULT 'FALSE' NOT NULL, ",
  "SFARI          boolean DEFAULT 'FALSE' NOT NULL, ",
  "EpilepsyGene   boolean DEFAULT 'FALSE' NOT NULL, ",
  "ClinVar        boolean DEFAULT 'FALSE' NOT NULL, ",
  "FOREIGN KEY(HDOID) REFERENCES Disease(HDOID), ",
  "FOREIGN KEY(GeneID) REFERENCES Gene(ID));",
"CREATE INDEX Mutation_Chromosome ",
  "ON Mutation (Chromosome);",
"CREATE INDEX Mutation_Variant ",
  "ON Mutation (Variant);",
"CREATE INDEX Mutation_FunctionClass ",
  "ON Mutation (FunctionClass);",
"CREATE INDEX Mutation_DENOVO ",
  "ON Mutation (DENOVO);",
"CREATE INDEX Mutation_SFARI ",
  "ON Mutation (SFARI);",
"CREATE INDEX Mutation_HGMD ",
  "ON Mutation (HGMD);",
```

```
"CREATE INDEX Mutation_ClinVar ",
  "ON Mutation (ClinVar);")
#"CREATE INDEX Gene_HumanName ",
#"  ON Gene (HumanName);",
#"CREATE INDEX Gene_MouseName ",
#"  ON Gene (MouseName);")
ddlPapMut <- paste("CREATE TABLE PaperMutation (",
  "PMID       NUMERIC(19, 0) NOT NULL, ",
  "MutationID integer(10) NOT NULL, ",
  "FOREIGN KEY(PMID) REFERENCES Paper(PMID), ",
  "FOREIGN KEY(MutationID) REFERENCES Mutation(ID));")
ddlPapPPI <- paste("CREATE TABLE PaperPPI (",
  "PMID NUMERIC(19, 0) NOT NULL, ",
  "PPID     integer(10) NOT NULL, ",
  "FOREIGN KEY(PMID) REFERENCES Paper(PMID), ",
  "FOREIGN KEY(PPID) REFERENCES PPI(ID));")
ddlV1<-paste("CREATE VIEW FullGenePaper AS",
"SELECT p.GeneID,LocalisationID, MGI,HumanEntrez,MouseEntrez,HumanName,MouseName,PaperPMID,SpeciesTaxID,
"FROM Gene  g join PaperGene p on g.ID=p.GeneID;")
ddlV2<-paste("CREATE VIEW FullGenefullPaper AS",
"SELECT p.GeneID,l.Name AS Localisation, ",
"MGI,HumanEntrez,MouseEntrez,HumanName,",
"MouseName,PaperPMID,a.Name AS Paper,",
"a.Year AS Year,",
"SpeciesTaxID,MethodID",
"FROM Gene  g join PaperGene p on g.ID=p.GeneID ",
"join Localisation l on l.ID = p.LocalisationID ",
"join Paper a on a.PMID = p.PaperPMID;")
ddlV3<-paste("CREATE VIEW FullGeneFullPaperFullRegion AS",
"    SELECT p.GeneID,",
"           l.Name AS Localisation,",
"           MGI,",
"           HumanEntrez,",
"           MouseEntrez,",
"           HumanName,",
"           MouseName,",
"           PaperPMID,",
"           a.Name AS Paper,",
"           a.Year AS Year,",
"           SpeciesTaxID,",
"           MethodID,",
"           b.Name AS BrainRegion",
"      FROM Gene g",
"           JOIN",
"           PaperGene p ON g.ID = p.GeneID",
"           JOIN",
"           Localisation l ON l.ID = p.LocalisationID",
"           JOIN",
"           Paper a ON a.PMID = p.PaperPMID",
"           JOIN",
"           BrainRegion b ON b.ID = p.BrainRegionID;")
ddlV4<-paste("CREATE VIEW FullGeneFullDisease AS",
"    SELECT HumanEntrez,",
```

```
"          HumanName,",
"          d.HDOID,",
"          d.Description",
"      FROM Gene g",
"          JOIN",
"          DiseaseGene dg ON g.ID = dg.GeneID",
"          JOIN",
"          disease d ON dg.HDOID = d.HDOID;")
```

## 10.2 Setup R

```
## This chunk should contain global configuration commands.
## Use this to set knitr options and related things. Everything
## in this chunk will be included in an appendix to document the
## configuration used.
#output <- opts_knit$get("rmarkdown.pandoc.to")
opts_knit$set(stop_on_error = 2L)


## Cache options
opts_chunk$set(cache=FALSE)


## Set 'hide.fig.code' to FALSE to include code chunks that
## produce Figures in the output. Note that this affects all chunks
## that provide a figure caption.
opts_chunk$set(hold=TRUE, hide.fig.code=FALSE)

## Pander options
panderOptions("digits", 3)
panderOptions("table.split.table", 160)
```

## 10.3 Versions

### 10.3.1 Session Info

| | |
|---|---|
| version | R version 4.2.1 (2022-06-23) |
| os | macOS Big Sur ... 10.16 |
| system | x86_64, darwin17.0 |
| ui | X11 |
| language | (EN) |
| collate | en_US.UTF-8 |
| ctype | en_US.UTF-8 |
| tz | Asia/Tokyo |
| date | 2022-10-10 |
| pandoc | 2.19 @ /usr/local/bin/ (via rmarkdown) |

| | package | ondiskversion | loadedversion | attached | is_base | date | source |
|---|---|---|---|---|---|---|---|
| AnnotationDbi | AnnotationDbi | 1.59.1 | 1.59.1 | TRUE | FALSE | 2022-05-19 | Bioconductor |

| | package | ondiskversion | loadedversion | attached | is_base | date | source |
|---|---|---|---|---|---|---|---|
| assertthat | assertthat | 0.2.1 | 0.2.1 | FALSE | FALSE | 2019-03-21 | CRAN (R 4.2.0) |
| Biobase | Biobase | 2.57.1 | 2.57.1 | TRUE | FALSE | 2022-05-19 | Bioconductor |
| BiocGenerics | BiocGenerics | 0.43.4 | 0.43.4 | TRUE | FALSE | 2022-09-11 | Bioconductor |
| Biostrings | Biostrings | 2.65.6 | 2.65.6 | FALSE | FALSE | 2022-09-09 | Bioconductor |
| bit | bit | 4.0.4 | 4.0.4 | FALSE | FALSE | 2020-08-04 | CRAN (R 4.2.0) |
| bit64 | bit64 | 4.0.5 | 4.0.5 | FALSE | FALSE | 2020-08-30 | CRAN (R 4.2.0) |
| bitops | bitops | 1.0.7 | 1.0-7 | FALSE | FALSE | 2021-04-24 | CRAN (R 4.2.0) |
| blob | blob | 1.2.3 | 1.2.3 | FALSE | FALSE | 2022-04-10 | CRAN (R 4.2.0) |
| cachem | cachem | 1.0.6 | 1.0.6 | FALSE | FALSE | 2021-08-19 | CRAN (R 4.2.0) |
| callr | callr | 3.7.2 | 3.7.2 | FALSE | FALSE | 2022-08-22 | CRAN (R 4.2.0) |
| cli | cli | 3.4.1 | 3.4.1 | FALSE | FALSE | 2022-09-23 | CRAN (R 4.2.0) |
| colorspace | colorspace | 2.0.3 | 2.0-3 | FALSE | FALSE | 2022-02-21 | CRAN (R 4.2.0) |
| crayon | crayon | 1.5.2 | 1.5.2 | FALSE | FALSE | 2022-09-29 | CRAN (R 4.2.0) |
| data.table | data.table | 1.14.2 | 1.14.2 | FALSE | FALSE | 2021-09-27 | CRAN (R 4.2.0) |
| DBI | DBI | 1.1.3 | 1.1.3 | TRUE | FALSE | 2022-06-18 | CRAN (R 4.2.0) |
| dbplyr | dbplyr | 2.2.1 | 2.2.1 | TRUE | FALSE | 2022-06-27 | CRAN (R 4.2.0) |
| devtools | devtools | 2.4.4 | 2.4.4 | FALSE | FALSE | 2022-07-20 | CRAN (R 4.2.0) |
| digest | digest | 0.6.29 | 0.6.29 | FALSE | FALSE | 2021-12-01 | CRAN (R 4.2.0) |
| dplyr | dplyr | 1.0.10 | 1.0.10 | FALSE | FALSE | 2022-09-01 | CRAN (R 4.2.1) |
| dtplyr | dtplyr | 1.2.2 | 1.2.2 | TRUE | FALSE | 2022-08-20 | CRAN (R 4.2.0) |
| ellipsis | ellipsis | 0.3.2 | 0.3.2 | FALSE | FALSE | 2021-04-29 | CRAN (R 4.2.0) |
| evaluate | evaluate | 0.17 | 0.17 | FALSE | FALSE | 2022-10-07 | CRAN (R 4.2.1) |
| fansi | fansi | 1.0.3 | 1.0.3 | FALSE | FALSE | 2022-03-24 | CRAN (R 4.2.0) |
| fastmap | fastmap | 1.1.0 | 1.1.0 | FALSE | FALSE | 2021-01-25 | CRAN (R 4.2.0) |
| fs | fs | 1.5.2 | 1.5.2 | FALSE | FALSE | 2021-12-08 | CRAN (R 4.2.0) |
| generics | generics | 0.1.3 | 0.1.3 | FALSE | FALSE | 2022-07-05 | CRAN (R 4.2.0) |

| | package | ondiskversion | loadedversion | attached | is_base | date | source |
|---|---|---|---|---|---|---|---|
| GenomeInfoDb | GenomeInfoDb | 1.33.7 | 1.33.7 | FALSE | FALSE | 2022-09-07 | Bioconductor |
| GenomeInfoDbData | GenomeInfoDbData | 1.2.9 | 1.2.9 | FALSE | FALSE | 2022-10-04 | Bioconductor |
| ggplot2 | ggplot2 | 3.3.6 | 3.3.6 | TRUE | FALSE | 2022-05-03 | CRAN (R 4.2.0) |
| glue | glue | 1.6.2 | 1.6.2 | FALSE | FALSE | 2022-02-24 | CRAN (R 4.2.0) |
| GO.db | GO.db | 3.16.0 | 3.16.0 | TRUE | FALSE | 2022-10-04 | Bioconductor |
| gtable | gtable | 0.3.1 | 0.3.1 | FALSE | FALSE | 2022-09-01 | CRAN (R 4.2.1) |
| htmltools | htmltools | 0.5.3 | 0.5.3 | FALSE | FALSE | 2022-07-18 | CRAN (R 4.2.0) |
| htmlwidgets | htmlwidgets | 1.5.4 | 1.5.4 | FALSE | FALSE | 2021-09-08 | CRAN (R 4.2.0) |
| httpuv | httpuv | 1.6.6 | 1.6.6 | FALSE | FALSE | 2022-09-08 | CRAN (R 4.2.0) |
| httr | httr | 1.4.4 | 1.4.4 | FALSE | FALSE | 2022-08-17 | CRAN (R 4.2.1) |
| IRanges | IRanges | 2.31.2 | 2.31.2 | TRUE | FALSE | 2022-08-18 | Bioconductor |
| KEGGREST | KEGGREST | 1.37.3 | 1.37.3 | FALSE | FALSE | 2022-07-10 | Bioconductor |
| knitr | knitr | 1.40 | 1.40 | TRUE | FALSE | 2022-08-24 | CRAN (R 4.2.0) |
| later | later | 1.3.0 | 1.3.0 | FALSE | FALSE | 2021-08-18 | CRAN (R 4.2.0) |
| lifecycle | lifecycle | 1.0.3 | 1.0.3 | FALSE | FALSE | 2022-10-07 | CRAN (R 4.2.1) |
| magrittr | magrittr | 2.0.3 | 2.0.3 | FALSE | FALSE | 2022-03-30 | CRAN (R 4.2.0) |
| memoise | memoise | 2.0.1 | 2.0.1 | FALSE | FALSE | 2021-11-26 | CRAN (R 4.2.0) |
| mime | mime | 0.12 | 0.12 | FALSE | FALSE | 2021-09-28 | CRAN (R 4.2.0) |
| miniUI | miniUI | 0.1.1.1 | 0.1.1.1 | FALSE | FALSE | 2018-05-18 | CRAN (R 4.2.0) |
| munsell | munsell | 0.5.0 | 0.5.0 | FALSE | FALSE | 2018-06-12 | CRAN (R 4.2.0) |
| org.Hs.eg.db | org.Hs.eg.db | 3.16.0 | 3.16.0 | TRUE | FALSE | 2022-10-04 | Bioconductor |
| org.Mm.eg.db | org.Mm.eg.db | 3.16.0 | 3.16.0 | TRUE | FALSE | 2022-10-08 | Bioconductor |
| org.Rn.eg.db | org.Rn.eg.db | 3.16.0 | 3.16.0 | TRUE | FALSE | 2022-10-08 | Bioconductor |
| pander | pander | 0.6.5 | 0.6.5 | TRUE | FALSE | 2022-03-18 | CRAN (R 4.2.0) |
| pillar | pillar | 1.8.1 | 1.8.1 | FALSE | FALSE | 2022-08-19 | CRAN (R 4.2.0) |
| pkgbuild | pkgbuild | 1.3.1 | 1.3.1 | FALSE | FALSE | 2021-12-20 | CRAN (R 4.2.0) |

| | package | ondiskversion | loadedversion | attached | is_base | date | source |
|---|---|---|---|---|---|---|---|
| pkgconfig | pkgconfig | 2.0.3 | 2.0.3 | FALSE | FALSE | 2019-09-22 | CRAN (R 4.2.0) |
| pkgload | pkgload | 1.3.0 | 1.3.0 | FALSE | FALSE | 2022-06-27 | CRAN (R 4.2.0) |
| plyr | plyr | 1.8.7 | 1.8.7 | TRUE | FALSE | 2022-03-24 | CRAN (R 4.2.0) |
| png | png | 0.1.7 | 0.1-7 | FALSE | FALSE | 2013-12-03 | CRAN (R 4.2.0) |
| prettyunits | prettyunits | 1.1.1 | 1.1.1 | FALSE | FALSE | 2020-01-24 | CRAN (R 4.2.0) |
| processx | processx | 3.7.0 | 3.7.0 | FALSE | FALSE | 2022-07-07 | CRAN (R 4.2.0) |
| profvis | profvis | 0.3.7 | 0.3.7 | FALSE | FALSE | 2020-11-02 | CRAN (R 4.2.0) |
| promises | promises | 1.2.0.1 | 1.2.0.1 | FALSE | FALSE | 2021-02-11 | CRAN (R 4.2.0) |
| ps | ps | 1.7.1 | 1.7.1 | FALSE | FALSE | 2022-06-18 | CRAN (R 4.2.0) |
| purrr | purrr | 0.3.5 | 0.3.5 | FALSE | FALSE | 2022-10-06 | CRAN (R 4.2.1) |
| R6 | R6 | 2.5.1 | 2.5.1 | FALSE | FALSE | 2021-08-19 | CRAN (R 4.2.0) |
| Rcpp | Rcpp | 1.0.9 | 1.0.9 | FALSE | FALSE | 2022-07-08 | CRAN (R 4.2.0) |
| RCurl | RCurl | 1.98.1.9 | 1.98-1.9 | FALSE | FALSE | 2022-10-03 | CRAN (R 4.2.1) |
| remotes | remotes | 2.4.2 | 2.4.2 | FALSE | FALSE | 2021-11-30 | CRAN (R 4.2.0) |
| reshape2 | reshape2 | 1.4.4 | 1.4.4 | TRUE | FALSE | 2020-04-09 | CRAN (R 4.2.0) |
| rlang | rlang | 1.0.6 | 1.0.6 | FALSE | FALSE | 2022-09-24 | CRAN (R 4.2.0) |
| rmarkdown | rmarkdown | 2.17 | 2.17 | FALSE | FALSE | 2022-10-07 | CRAN (R 4.2.1) |
| RSQLite | RSQLite | 2.2.18 | 2.2.18 | TRUE | FALSE | 2022-10-04 | CRAN (R 4.2.0) |
| rstudioapi | rstudioapi | 0.14 | 0.14 | FALSE | FALSE | 2022-08-22 | CRAN (R 4.2.0) |
| S4Vectors | S4Vectors | 0.35.4 | 0.35.4 | TRUE | FALSE | 2022-09-18 | Bioconductor |
| scales | scales | 1.2.1 | 1.2.1 | FALSE | FALSE | 2022-08-20 | CRAN (R 4.2.0) |
| sessioninfo | sessioninfo | 1.2.2 | 1.2.2 | FALSE | FALSE | 2021-12-06 | CRAN (R 4.2.0) |
| shiny | shiny | 1.7.2 | 1.7.2 | FALSE | FALSE | 2022-07-19 | CRAN (R 4.2.0) |
| stringi | stringi | 1.7.8 | 1.7.8 | FALSE | FALSE | 2022-07-11 | CRAN (R 4.2.0) |
| stringr | stringr | 1.4.1 | 1.4.1 | FALSE | FALSE | 2022-08-20 | CRAN (R 4.2.1) |
| tibble | tibble | 3.1.8 | 3.1.8 | FALSE | FALSE | 2022-07-22 | CRAN (R 4.2.0) |

| | package | ondiskversion | loadedversion | attached | is_base | date | source |
|---|---|---|---|---|---|---|---|
| tidyselect | tidyselect | 1.1.2 | 1.1.2 | FALSE | FALSE | 2022-02-21 | CRAN (R 4.2.0) |
| urlchecker | urlchecker | 1.0.1 | 1.0.1 | FALSE | FALSE | 2021-11-30 | CRAN (R 4.2.0) |
| usethis | usethis | 2.1.6 | 2.1.6 | FALSE | FALSE | 2022-05-25 | CRAN (R 4.2.0) |
| utf8 | utf8 | 1.2.2 | 1.2.2 | FALSE | FALSE | 2021-07-24 | CRAN (R 4.2.0) |
| vctrs | vctrs | 0.4.2 | 0.4.2 | FALSE | FALSE | 2022-09-29 | CRAN (R 4.2.0) |
| withr | withr | 2.5.0 | 2.5.0 | FALSE | FALSE | 2022-03-03 | CRAN (R 4.2.0) |
| xfun | xfun | 0.33 | 0.33 | FALSE | FALSE | 2022-09-12 | CRAN (R 4.2.0) |
| xtable | xtable | 1.8.4 | 1.8-4 | FALSE | FALSE | 2019-04-21 | CRAN (R 4.2.0) |
| XVector | XVector | 0.37.1 | 0.37.1 | FALSE | FALSE | 2022-08-25 | Bioconductor |
| yaml | yaml | 2.3.5 | 2.3.5 | FALSE | FALSE | 2022-02-21 | CRAN (R 4.2.0) |
| zlibbioc | zlibbioc | 1.43.0 | 1.43.0 | FALSE | FALSE | 2022-05-05 | Bioconductor |