Project 2

John Kim

Introduction

Research Question

Are there nucleotide sequence similarities between cysteine/serine proteases that activate same allergic reaction pathways in humans such as house mite allergen, papain, ragweed pollen allergen, SplA, and if so, what structural similarities can be seen in the protein or the active site by looking at its 3D structure that may explain the function similiarities?

Background Research

A major class of allergens that induce airway stimulation are proteases that come from many different sources. Some of these protease allergens include house dust mite derived proteases Der p 1, ragweed protease allergen Amb a 1.2, papain, and bacteria derived SplA . All of these allergens have shown to activate the type 2 immunity in a similar fashion where they disturb the tight junctions in epithelial cells making the epithelial layer more permeable. It also activates the Protease Activated Receptor 2 by cleaving the receptor that further signals the activation of type 2 immunity (Matsumura, 2011). Papain has shown to trigger T cell differentiation with a bias towards Th2 cells that initiate type 2 immunity through the PAR2 pathway (Liang et al., 2012). Staphylococcus aureus derived SplA protease has also shown to increase Th2 cells in a similar fashion (Nordengrun et al, 2021). Ragweed allergy has been found to linked with the proteases allergen which is triggered by PAR2 (Hosoki et al., 2017). Moreover, these features were also observed in house dust mite allergen Der p 1 (Reithofer et al., 2017).

Hypothesis

If there are multiple instances of environmental allergens being proteases or having protease-like activity causing similar allergic responses in humans, then there must be either protein sequence or structural homology between these proteins for it to cause allergies in a similar manner in humans.

Analysis and Data

The analysis performed in this project invlude pairwise alignment as well as multiple sequence alignment to check for protein sequence similarities. The result of the pairwise alignment was plotted on a heamap to visualize the score differences between the comparisons, and multiple sequence alignment can be seen through the function MSAprettyprint which nicely organizes results of our MSA.

3D protein measurements were made through bio3d and pyMOL, and structural similiarities near the binding sites of the proteins were measured and visualized through it's 3D structure images. The binding sites were obtained through bio3d, and these residues and nearby residues were checked for structual similarities. These videos have been embedded into this notebook.

```
library("Biostrings")         # Allows us to read fastafiles through the function readAAStringSet()
```

The datasets were downloaded in UniProt and PDB

```
## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: XVector
```

```
## Loading required package: GenomeInfoDb
```

```
##
## Attaching package: 'Biostrings'
```

```
## The following object is masked from 'package:base':
##
##     strsplit
```

```
library(msa)                    # Performing multiple sequence alignments
library(tools)                  # R base package for various functions
library(text.alignment)         # Package to run walterman smith pairwise alignment


# get fasta files found on uniprot through ebi database

#assigning fasta file names to variables to call later
ragweed<- "PLY1_AMBAR.fasta"
splA<- "Q2FXC2.fasta"
hdm<- "P08176.fasta"
ppa<- "P00784.fasta"

proteins<- c(ragweed, splA, hdm, ppa)


# fasta name variables called here to make 1 variable containing all fastafiles
fassequences <- lapply(proteins, readAAStringSet)
# make loaded proteins sequences readable for msa format
fassequences <- do.call(c,fassequences)

#run msa
malignment<- msa(fassequences)
```

```
## use default substitution matrix
```

```
malignment
```

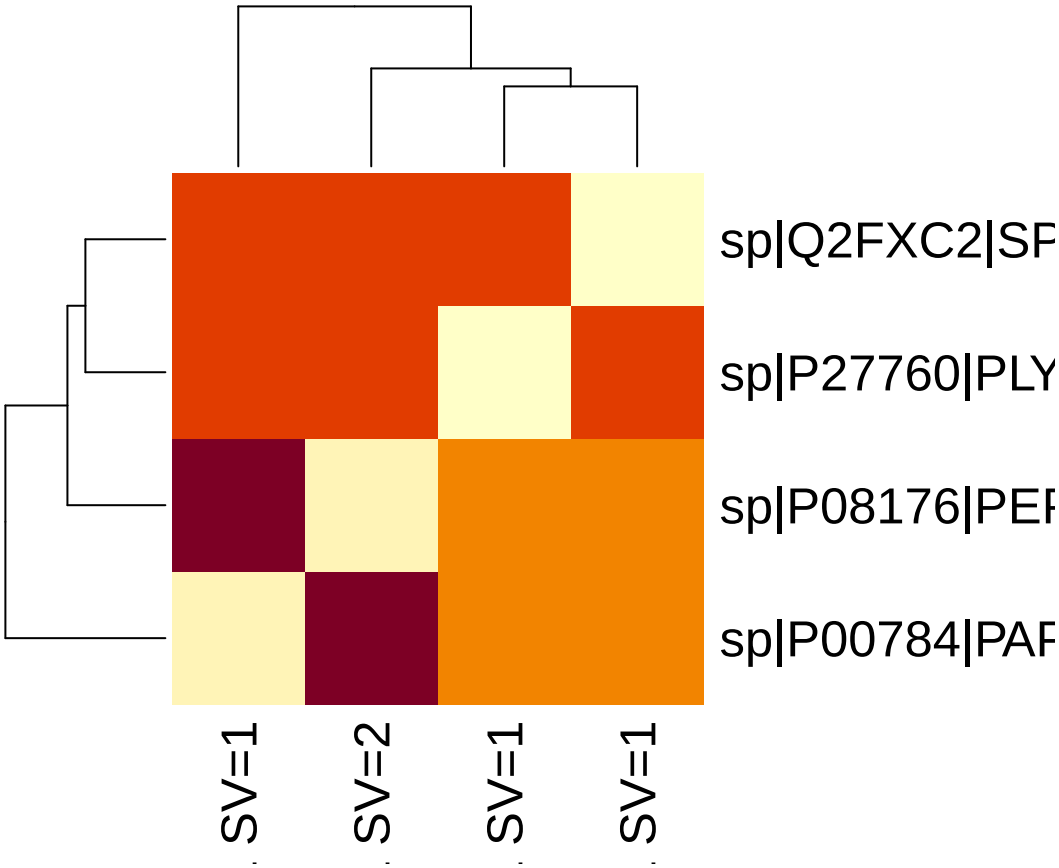
```
## CLUSTAL 2.1
##
## Call:
##    msa(fassequences)
##
## MsaAAMultipleAlignment with 4 rows and 398 columns
##     aln                                                  names
## [1] ---------MKIVLAIASLLALS---...FAANIDLMMIEEYPYVVIL------ sp|P08176|PEPT1_D...
## [2] ---------MAMIPSISKLLFVAICL...IKRGTGNSYGVCGLYTSSFYPVKN- sp|P00784|PAPA1_C...
## [3] MGIKHCCYILYFTLALVTLLQPVRSA...AEPGEAVLRLTSSAGVLSCHQGAPC sp|P27760|PLY1_AM...
## [4] ---------MNKNVMVKGLTALT---...------------------------ sp|Q2FXC2|SPLA_ST...
## Con ---------M???LAI??LLAL?---...???G??????????YV?S?------ Consensus
```

```
# MSAprettyprint is used to show the results of our msa in a nicely visualized figure.
# cannot view latex in non unix systems properly. but the code works
# Error: Functions that produce HTML output found in document targeting latex output.
# Please change the output type of this document to HTML. Alternatively, you can allow
# HTML output in non-HTML formats by adding this option to the YAML front-matter of
# your rmarkdown file:
# a<-msaPrettyPrint(malignment, output="html", showNames="none",
#             showLogo="none", askForOverwrite=FALSE, verbose=FALSE)


# trying msa without spla, which has the lowerst homology
nospla<-c(ragweed, hdm, ppa)


fassequences1 <- lapply(nospla, readAAStringSet)
fassequences1 <- do.call(c,fassequences1)

malignment1<- msa(fassequences1)


## use default substitution matrix

malignment1


## CLUSTAL 2.1
##
## Call:
##    msa(fassequences1)
##
## MsaAAMultipleAlignment with 3 rows and 398 columns
##     aln                                                 names
## [1] ---------MKIVLAIASLLALS---...FAANIDLMMIEEYPYVVIL------ sp|P08176|PEPT1_D...
## [2] ---------MAMIPSISKLLFVAICL...IKRGTGNSYGVCGLYTSSFYPVKN- sp|P00784|PAPA1_C...
## [3] MGIKHCCYILYFTLALVTLLQPVRSA...AEPGEAVLRLTSSAGVLSCHQGAPC sp|P27760|PLY1_AM...
## Con ---------M???LAI??LL??????...???G??????????YV?S??????- Consensus

# using pairwaise sequence alignment to see if there any two similar proteins



# code check for the upcoming function
# seqonly1<-fassequences
# seqonly1 <- seqonly1[!grepl('^$', seqonly1)]
# ids <- gsub('^.+\\|(\\w+)\\|.*$', '>\\1', seqonly1)
#
# smith_waterman(ids[[1]],ids[[3]],
#                         match = 1L,
#                         mismatch = -2L,
#                         gap = -2L,)

# brought back alignment code from earlier challenge problem to make heatmap of different sequences

# make function named compareall that compares all sequences in a fasta file
```

```
compareall<-function(seqlist){
#define empty matrix with the ncol and nrow as the list length
  WaM1<-matrix(0,nrow=length(seqlist),ncol=length(seqlist))

## useful code to only get sequence from fasta
## https://stackoverflow.com/questions/61082486/how-to-get-the-id-codes-from-a-fasta-file-using-r
seqonly<-seqlist



###### using grepl and gsub (BASE FUNCTIONS) to remove uncessary information in fasta files


seqonly <- seqonly[!grepl('^$', seqonly)]
ids <- gsub('^.+\\|(\\w+)\\|.*$', '>\\1', seqonly)
# nested for loop to check every two combinations in the matrix returns 0 if the comparisons are the sa
  for (i in 1:length(seqlist)) {
    for (j in 1:length(seqlist)){
      if (i==j){
        WaM1[i,j]<-0}
       else {
           #run smith_waterman from package text.alignment
          result<-smith_waterman(ids[[i]],ids[[j]],
                       match = 1L,
                       mismatch = -2L,
                       gap = -2L,)
          #codecheck
          #print(ids[i])
          #print(ids[j])
          WaM1[i,j]<- result$sw
          #print(WaM1)
          }

    }
  }
 return(WaM1)
}

WaM<-compareall(fassequences)
WaM
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    4    4    4
## [2,]    4    0    4    4
## [3,]    4    4    0    8
## [4,]    4    4    8    0
```

```
protnames<-as.vector(names(fassequences))

#draw out a heat map using names and matrix value
heatmap(WaM,labRow=protnames, labCol=protnames)
```




```
library(bio3d)                  # Allows 3d measurement of our proteins
```

The above method pairwise sequence alignment compares two sequences direction against each other. The algorithm used above has a scoring rubric, and scores based of match, mismatch, and gaps. The score shows us how similar the two sequences are. Multiple sequence alignment uses a different algorithm, such as clustalW, MUSCLE, and others to align and return consensus sequences between three or more sequences. The data types that are read in for this method are amino acid sequences.

```
##
## Attaching package: 'bio3d'

## The following object is masked from 'package:Biostrings':
##
##     mask

## The following object is masked from 'package:IRanges':
##
##     trim
```

```
library(r3dmol)

#call in all pdb files
ragwpdb<-read.pdb("5egw")
```

```
##   Note: Accessing on-line PDB file
```

```
ppapdb<- read.pdb("1PPP")
```

```
##   Note: Accessing on-line PDB file
```

```
ragwpdb<-read.pdb("5egw")
```

```
##   Note: Accessing on-line PDB file

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): C:
## \Users\JOHNKI~1\AppData\Local\Temp\RtmpY9BTSW/5egw.pdb exists. Skipping download
```

```
hdmpdb<-read.pdb("1xkg")
```

```
##   Note: Accessing on-line PDB file
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

```
#chain A of ragweed protein is weird (too different for nma), need to take it out and compare
ragwpdbB <-trim(ragwpdb, chain= 'B')
write.pdb(ragwpdbB)

pdbnames <- c("1PPP.pdb", "2w7u.pdb", "trim5egw.pdb", "1xkg.pdb")


# wanted to make comparisons with multople pdb files, but muscle is required, and cannot trouble shoot
# fixed with 'test@gmail.com
pdbs <- pdbaln(pdbnames, fit = TRUE, pqr = FALSE, ncore = 1, nseg.scale = 1, web.args=list(email='test@
```

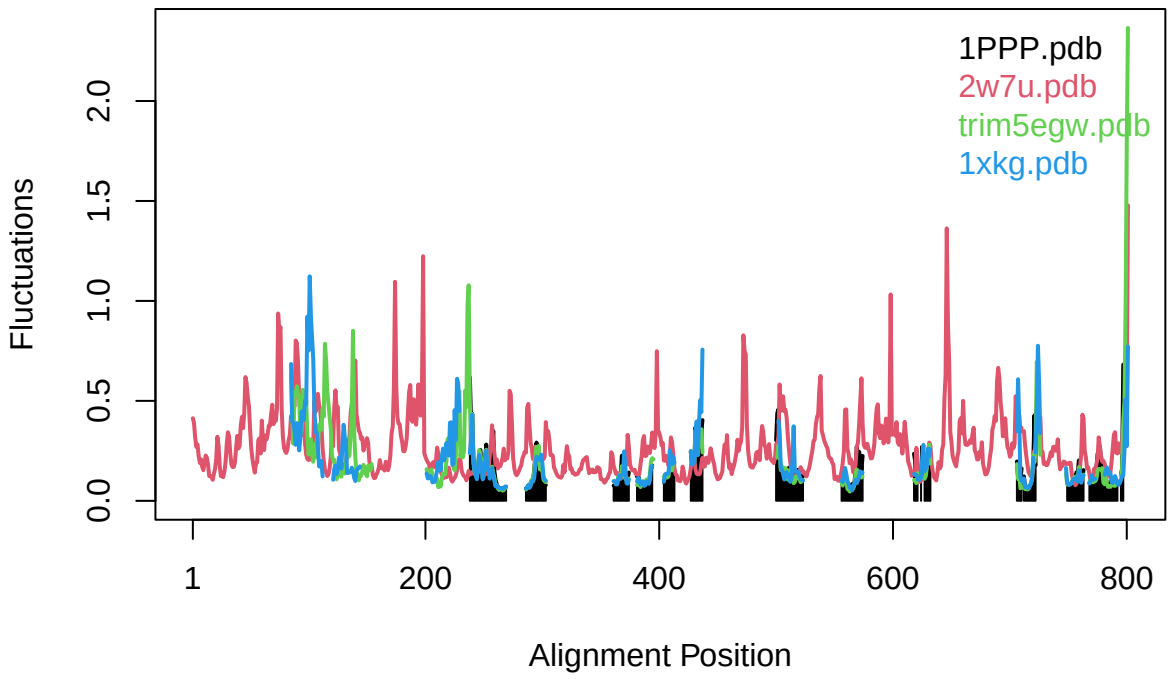
```
## Reading PDB files:
## 1PPP.pdb
## 2w7u.pdb
## trim5egw.pdb
## 1xkg.pdb
## ...   PDB has ALT records, taking A only, rm.alt=TRUE
## .
##
## Extracting sequences
##
##
## Will try to align sequences online...
##
## Job successfully submited (job ID: muscle-R20220603-230038-0076-90248230-p2m)
## Waiting for job to finish...Done.
##
```

```
## pdb/seq: 1   name: 1PPP.pdb
## pdb/seq: 2   name: 2w7u.pdb
## pdb/seq: 3   name: trim5egw.pdb
## pdb/seq: 4   name: 1xkg.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

```
#perform nma and plot
modes <- nma(pdbs, rm.gaps=FALSE)
```

```
## Warning in nma.pdbs(pdbs, rm.gaps = FALSE): 2w7u.pdb, trim5egw.pdb, 1xkg.pdb might have missing resi
##    Fluctuations at neighboring positions may be affected.

##
## Details of Scheduled Calculation:
##   ... 4 input structures
##   ... storing 597 eigenvectors for each structure
##   ... dimension of x$U.subspace: ( 2403x597x4 )
##   ... coordinate superposition prior to NM calculation
##   ... estimated memory usage of final 'eNMA' object: 43.8 Mb
##
##   |                                                                                 |
```

```
plot(modes)
```



```
# observe binding sites for each protein of interest that showed homology
ppabinding<- binding.site(ppapdb)
ppabinding
```

```
## $inds
##
##  Call:  NULL
##
##    Atom Indices#: 163  ($atom)
##    XYZ  Indices#: 489  ($xyz)
##
## + attr: atom, xyz
##
## $resnames
##  [1] "TYR 4 (A)"   "VAL 5 (A)"   "ASP 6 (A)"   "GLN 19 (A)"  "CYS 22 (A)"
##  [6] "GLY 23 (A)"  "SER 24 (A)"  "CYS 25 (A)"  "TRP 26 (A)"  "GLY 65 (A)"
## [11] "GLY 66 (A)"  "TYR 67 (A)"  "LYS 156 (A)" "VAL 157 (A)" "ASP 158 (A)"
## [16] "HIS 159 (A)" "ALA 160 (A)" "TYR 166 (A)" "TYR 170 (A)" "LEU 172 (A)"
##
## $resno
##  [1]   4   5   6  19  22  23  24  25  26  65  66  67 156 157 158 159 160 166 170
## [20] 172
##
## $chain
##  [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
## [20] "A"
##
## $call
## binding.site(a = ppapdb)
```

```
# ragwbinding<-binding.site(ragwpdb)
# not enough ligand in structure
hdmbinding <- binding.site(hdmpdb)
hdmbinding
```

```
## $inds
##
##  Call:  NULL
##
##    Atom Indices#: 289  ($atom)
##    XYZ  Indices#: 867  ($xyz)
##
## + attr: atom, xyz
##
## $resnames
##  [1] "MET 63 (A)"  "PHE 75 (A)"  "ARG 106 (A)" "MET 107 (A)" "GLY 112 (A)"
##  [6] "SER 113 (A)" "ALA 114 (A)" "TRP 115 (A)" "PHE 117 (A)" "GLU 124 (A)"
## [11] "ASP 136 (A)" "LEU 137 (A)" "ALA 138 (A)" "GLU 139 (A)" "GLY 153 (A)"
## [16] "ASP 154 (A)" "THR 155 (A)" "ILE 156 (A)" "GLU 171 (A)" "TYR 176 (A)"
## [21] "GLU 180 (A)" "GLN 181 (A)" "SER 182 (A)" "ARG 184 (A)" "ILE 221 (A)"
## [26] "GLN 240 (A)" "TYR 249 (A)" "HIS 250 (A)" "ALA 251 (A)" "GLY 262 (A)"
## [31] "VAL 263 (A)" "ASP 264 (A)" "ALA 285 (A)" "ALA 286 (A)" "ASN 287 (A)"
```

```
## [36] "ILE 288 (A)"
##
## $resno
##  [1]  63  75 106 107 112 113 114 115 117 124 136 137 138 139 153 154 155 156 171
## [20] 176 180 181 182 184 221 240 249 250 251 262 263 264 285 286 287 288
##
## $chain
##  [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
## [20] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
##
## $call
## binding.site(a = hdmpdb)
```

```
library("vembedr")   # allows embedding videos into notebook

embed_youtube("JPKnIxE1QHU")
```

```
embed_youtube("gPdUmwcW8qY")
```

```
embed_youtube("Cu_mL3Oefzw")
```

```
embed_youtube("OXW22tnwf_Y")
```

The above analysis method uses the pdb database and pdb files for proteins to make 3D measurement of protein structures in the database. Here, the function binding.site is called to check the binding sites. The binding sites were cross referenced with the MSA results to locate the conserved sequences that may play a role in the induction of similiar allergic responses.

The first video shows papain in cyan, and Der p 1 in yellow. We can see the sequence homologies that were shown in MSA, which all occur near the binding sites of the proteins respectively.

The second video shows Der p 1 in yellow, and ragweed pollen allergen Amb a 1.2 in grey. The same sequences that were shown as conserved in MSA were visualized

The third video shows papain in orange and Amb a 1.2 in grey. The same methods as above was used to visualize the similarities.

The fourth video shows all the conserved sequences between all three proteins on a papain that seem to play a role in the protease triggering an allergic effect.

Defining Variables

Global variables are variables that are created outside a function. This allows them to be called at all times

Local variables are variables that are created inside a function. This means that they can be only called inside the function, but not outside.

```
#example code to explain local and global variable

glob<- "Global Variable"


myf<- function(){
  glob<- "Local Variable"
  paste("This is a ", glob)
}

# use function to call local variable
print(myf())
```

This is used in my nested forloop to compare pairwise sequence alignment to all AA sequences I had

```
## [1] "This is a  Local Variable"
```

```
# call glob directly
print(glob)
```

```
## [1] "Global Variable"
```

Analysis

From the pairwise sequence alignment and the heatmap, we can see the papain and Der p 1 have the most sequence homology, and other proteins also show some sequence homology. Upon further investigation with multiple sequence alignment, it is clearly shown that the three proteins Der p 1, Amb a 1.2 and papain have commonly conserved sequences, but with SplA, it is not too clear if there are seqeunce homology with the other proteins. These proteins then were further analyzed to check for their binding site and the residues around the binding sites. These sites were then crossreferenced with the conserved sequences from our MSA, and it was found through pyMOL that the conserved sequences are around the binding sites, and these sites show structural similarities. Moreover, the secondary protein structures also show homology, which may indicate that these protein structures help the allergens bind specifically to the junction gap proteins between epithilial cells. The residues that help form the similiarities then were visualized and shown through PyMOL. In conclusion it can be said that the structure of these proteins are related in their functions in antagonizing the human immune system, and that the binding sites of the allergens, which show homology, seem to be driving this effect.