Identifying unexpected metabolic pathways arising from Alzheimer's Disease-related genes with KEGG API

Gabriel L. Ketron

December 24, 2022

Abstract

Alzheimer's disease is a neurodegenerative disease that causes most dementia cases around the world. Ongoing investigation into the mechanisms of Alzheimer's disease cover a spectrum of genomic, metabolic, and transcriptomic data. The aim of this paper is to demonstrate R software developed to congregate relevant multiomic data to identify unexpected metabolic pathways which may impact Alzheimer's Disease. This software queries a list of genes in the Kyoto Encyclopedia of Genes and Genomes' R API to determine which metabolic pathways appeared most often, which gene played a role in the most amount of record pathways, and how the number of genes encoding a pathway related to protein and RNA production. By applying this software to Alzheimer's disease-related genes, this paper identifies the metabolic pathway that these genes contribute to the most, and by extension, which pathways may play a role in metabolic pathology.

1 Introduction

Alzheimer's Disease, or AD, is defined by the presence of Beta-amyloid containing plaques and tau-containing neurofibrillary tangles, which cause both amnestic and non-amnestic impairments modified by other neurodegenerative and cerebrovascular conditions. Impairments arise from the complex interplay between the loss of synapse homeostasis, and disrupted endosome/lysosome clearance pathways, all modulated by the incidence of the precursors, aggregate forms and post-translational products of amyloid-beta and tau [1]. Key risk factors for interplay include genetic mutations to APP, PSEN1, PSEN2, or having an APOEe4 allele. Environmental factors and comorbidities also impact risk, but epigenetic factors only minimally increased risk compared to age alone [2].

AD pathology typically emerges when tau-containing neurofibrillary tangles accumulate and migrate downstream from the entorhinal cortex. If the brain has pre-existing AB deposits in these downstream regions, overt and covert destructive mechanisms begin [1].Understanding interplay is central to AD treatment and poses a large and growing challenge to public health given the enormous personal and economic losses each diagnosis confers [3]. Work is still ongoing into the biologic mechanisms of AD, and alternate psychological and biomarker methods are used to identify AD in a patient.

1.1 Problem Statement

A recent study from the Stanford University School of Medicine identified neurofibrillary tangle aggregation-prone cell states by conducting a transcriptomeprofiling of soma in a human AD patient. Using a model-based analysis of single cell transcriptomics (MAST), 293 genes demonstrated transcriptomic signatures of tau pathology within and across neuronal subtypes. 63 of these genes were associated with the up-regulation of synaptic vesicle cycling [4]. An examination of MAST genes with pathway databases and data analysis techniques may allow us to find new relationships of how tau encoding genes direct metabolic pathways. This paper seeks to identify the metabolic pathway with the most genes from the tau pathology contributing to it.

Figure 1: This software used Rstudio as its R IDE.

1.2 Solution Approach

To find the most contributed pathway, a software package needs to acquire a list of every metabolic pathway that a gene contributes to. This will require a database containing gene and metabolic pathway data, and a data frame for examining details of the contributing genes. These pathways then need an ongoing tally of the most contributed pathway, and a dataframe to contain data about the pathway. Additionally the software needs to call an image of the desired pathway. As an exploratory goal, the software will also create data visualizations of a variety of details within the gene and pathway dataframes to discover unexpected relationships between the data. The software is written in R, due to its effectiveness in prototyping data pipeline tools, API availability, and visualization tools.

1.3 Data Set

KEGG (Kyoto Encyclopedia of Genes and Genomes) is a knowledge base for systemic analysis of genes function, established and curated by Kyoto University. The knowledge base conatins two databases useful to the software, the GENES and PATHWAY databases. The GENES database sequenced genomes for humans and other animals, and a database identifier tags (HSA values) for connecting known genes to their associated pathways, and some basic information about the gene, its amino acid sequence and DNA sequence. The PATHWAY database contains metabolic pathway data, including standardized diagrams documenting the relationships between the genes, RNA, protein, and metabolome [5]. By accessing this data set, we can associate tau correlated genes with metabolic pathways.

KEGG supports an application programming interface (KEGG API) to access its integrated databases from code. The API uses a REST-style API (CITE). In R, the KEGG API is called "KEGGREST" and is a package to provide a client interface with the KEGG server [6]. Further documentation can be found at https://bioconductor.org/packages/release/bioc/html/KEGGREST.html.

2 Methods

You can find the code and example dataset at https://github.com/gketronDS/KeggPathwaySelector. The code structure in R to support the data pipeline requires a number of supporting libraries, along with KEGGREST to help structure and transform the data into a visualize data frame. In order to determine the most contributed pathway, the relevant genes are taken from a CSV file found in the supporting documentation of the Standford study [4]. We then use the KEGG API to return the unique HSA id's of the genes. These values allow us to query the KEGG server to return data on the pathways and genes. This data is formatted along with operations to visualize relationships in the data.

2.1 Libraries

Listing 1: Used Libraries

```
## ------setup, echo=FALSE----------------------------------------------------
library(knitr)
options(width=80)
library(readxl)
library(tidyverse)
library(ggplot2)
library(ggthemes)
library(plotly)
library(KEGGREST)
library(png)
print("est runtime: 8min 30s")
```

"kniter" allows markup language integration. "readxl" allows us to bring the excel data into R. "tidyverse" is a collection of packages to help with data science. "ggplot2", "ggthemes", and "plotly" all help with data visualization. KEGGREST allows us to connect with the KEGG server. "png" allows the display of images pulled from the KEGG server.

2.2 List of Genes

The list of genes used in this code can be found here.

Listing 2: Accessing List of Genes

```
## ------Accessing List of Genes----------------------------------------------
##Read from Excel
REST_genes<-read_excel("1-s2.0-S0896627322006006-mmc3.xlsx",sheet="REST (shared)")
excel_sheets("1-s2.0-S0896627322006006-mmc3.xlsx")
##Only take genes the study identified that impact Alzheimer's Disease.
Deg_REST<-REST_genes[(REST_genes$type=="DEG"),]
Deg_REST<- Deg_REST %>%
  select(gene_names)
Deg_LIST<- list(Deg_REST)
```

This section grabs the data from the excel sheet, takes only the genes with tau pathology, and puts the names of the genes into a list.

2.3 HSA Values

Listing 3: Get HSA values for Genes

```
## ------Get HSA values for Genes----------------------------------------------
##Turn list into characters for API to search
Deg_GENE <- Deg_LIST[[1]][[1]]
out_GENE<- list()
genehsa = list()
##Search for everything about the gene in KEGG
#gene_test <- keggFind("genes", Deg_GENE[[140]])
##Format result to get hsa values
#gene_test <- as.data.frame(gene_test) # Duplicate example data
#gene_test$row_names <- row.names(gene_test) # Apply row.names function
#genehsa <- gene_test$row_names[[1]]
for (x in 1:length(Deg_GENE)) {
  print(x / 293*100)
  print("% Finished - Finding HSA Values")
  gene_test <- keggFind("genes", Deg_GENE[[x]])
  ##Format result to get hsa values
```

```
  gene_test <- as.data.frame(gene_test) # Duplicate example data
  if(dim(gene_test)[1] != 0){
    gene_test$row_names <- row.names(gene_test) # Apply row.names function
    for (i in 1:length(gene_test[[2]])){
      strwrk <- gene_test[[2]][i]
      strwrk<-strsplit(strwrk, split = ":")
      if (strwrk[[1]][1]=="hsa"){
        out_GENE[[length(out_GENE)+1]]<-gene_test[[2]][i]
      }
    }
    #for (y in 1:length(genehsa)){
     # out_GENE[[length(out_GENE)+1]] <- genehsa[[y]]
    #}
  }
  else{
    genehsa <- "hsa:0"
    out_GENE[[length(out_GENE)+1]] <- genehsa
  }
}
## add out-GENE as a column in Deg_REST and remove hsa: 0
out_GENE <- unique(out_GENE)
in_GENE <- out_GENE[out_GENE != "hsa:0"]
print("HSA Values Recieved")
```

DegGENE breaks down DEG list to contain only strings of the gene names in a list. outGENE and genehsa are placeholder values to hold the output of the for loop. The for loop runs for the length of degGENE. The KEGGREST operation keggfind checks the "GENES" database in KEGG based on the string. These get added to a dataframe and checked if the dataframe is empty. If it is not empty, the index of the gene contains the hsa value we want. We use row.names to bring the index into the dataframe. If this index is labeled "hsa", the index is taken as a string and cut before the colon to isolate the unique hsa number. Some genes have multiple hsa values, so this is checked for all of the values taken from that gene name in KEGG. This increases computational complexity to $O(n^2)$, but since the hsa values always come first a future function could cut off the loop the moment a non-hsa string appears. The unique string is added to outGENE. If there is no info returned back from KEGG, genesha gives outGENE an hsa value of zero. Last, inGENE only takes unique copies of the hsa values.

2.4 Pathway Dataframe

Listing 4: Get Pathways from Genes

```
## ------Get Pathways from Genes-----------------------------------------------------------
## This section allows us to assign genes to pathways for later comparison.
in_GENE3 <- t(in_GENE)
pathquery2<-as.data.frame(matrix(in_GENE3, ncol=664, byrow = TRUE))
track_path<- list()
gene_track_id = list()
for (x in 1:length(pathquery2)){
  print(x / 664*100)
  print("% Finished - Finding Path Values")
  hold<-keggLink("pathway",in_GENE[x])
  for (y in 1:length(hold)){
    if (length(hold[y])!=0){
      if (!is.na(hold[y])){
        track_path[[length(track_path)+1]]<-hold[[y]]
        gene_track_id[[length(gene_track_id)+1]]<-in_GENE[x]
      }
    }
```

```
  }
}
track_path_1 <- as.data.frame(matrix(track_path))
gene_track_set <- as.data.frame(matrix(gene_track_id))
## Path_occurance counts how often each pathway comes up, as column "n".
path_occurance<-track_path_1 %>%
  count(V1)
#pathwayIN <- list() #Old attempt at track_path
pathwayID <- list() #Name of the pathway as a KEGG data structure.
pathwayNAME <- list() #Name in text
pathwayDRUGS <- list() #List of drugs that effect the path
pathwayDRUGL <- list() # # of drugs which effect the path
pathwayCOMPS <- list() # Compounds that effect the path
pathwayCOMPL <- list() # number of compounds that effect the path
pathwayREL <- list() # List of related paths
pathwayRELL <- list() # Number of related paths
pathwayREFL <- list() # Number of references in KEGG
pathwayPROT <- list() # List of proteins the pathway makes
pathwayGENEL <- list() # number of unique proteins the pathway makes
for (x in 1:length(path_occurance[[1]])){
  print(x / 273*100)
  print("% Finished - Finding Path Data")
  pathway_test <- keggGet(path_occurance[[1]][x])
  pathwayID[[length(pathwayID)+1]] <- pathway_test[[1]][["PATHWAY_MAP"]]
  pathwayNAME[[length(pathwayNAME)+1]] <-pathway_test[[1]][["NAME"]]
  #pathwayDRUGS[[length(pathwayDRUGS)+1]] <- pathway_test[[1]][["DRUG"]]
  pathwayDRUGL[[length(pathwayDRUGL)+1]] <- length(pathway_test[[1]][["DRUG"]])
  ##pathwayCOMPS[[length(pathwayCOMPS)+1]] <- pathway_test[[1]][["COMPOUND"]]
  pathwayCOMPL[[length(pathwayCOMPL)+1]] <- length(pathway_test[[1]][["COMPOUND"]])
  #pathwayREL[[length(pathwayREL)+1]] <- pathway_test[[1]][["REL_PATHWAY"]]
  pathwayRELL[[length(pathwayRELL)+1]] <- length(pathway_test[[1]][["REL_PATHWAY"]])
  pathwayREFL[[length(pathwayREFL)+1]] <- length(pathway_test[[1]][["REFERENCE"]])
  pathwayPROT[[length(pathwayPROT)+1]]<-pathway_test[[1]][["GENE"]]
  pathwayGENEL[[length(pathwayGENEL)+1]]<-length(pathway_test[[1]][["GENE"]])
}
pathWAY <- mutate(path_occurance, ID = pathwayID, NAME = pathwayNAME,
                  DRUGL = pathwayDRUGL, COMPL = pathwayCOMPL,
                  RELL = pathwayRELL, REF = pathwayREFL,
                  PROT = pathwayGENEL)
pathWAY <- as.data.frame(lapply(pathWAY, unlist))
pathWAY <- pathWAY[order(-pathWAY$n),]
```

The first for loop takes a transposed dataframe of HSA values to create a list of pathways with the associated gene hsa value. This list will allow us to count which pathway is contributed to by the most genes.

The second for loop takes the list of unique pathways and gathers data for each pathway using the keggGet function. We look at the pathway map, name, number of associated drugs, number of associated compounds, number of related pathways, number papers refering to the pathway, the proteins the pathway involves and the number of proteins. This columns are mutated together, and formatted to sort by the number of contributing genes.

2.5 Gene Dataframe

Listing 5: Get Pathways from Genes

```
## ------Get Genes from Genes----------------------------------------------------------
in_GENE2 <- t(in_GENE)
gene_query<-as.data.frame(matrix(in_GENE2, ncol=664, byrow = TRUE))
gene_NAME <- list() # Name of gene
gene_CDS <- list() #Hsa value of gene as a number
gene_path <- list() # list of pathways the gene is a part of
gene_PATHS <- list()# # of pathways the gene is a part of
gene_POS <- list() # gene position in the genome
gene_AA <- list() # amino acid sequence the gene encodes
gene_AL <- list() # Length of amino acid sequence
gene_DA <- list() # DNA sequence of the gene
gene_DL <- list() # Length of dna sequence
gene_ORTH <- list() #orthology reference number for later use
for (x in 1:length(gene_query)){
  print(x/664*100)
  print("% Finished - Finding Gene Data")
  gene_test2 <- keggGet(gene_query[[x]])
  gene_NAME[[length(gene_NAME)+1]] <- gene_test2[[1]][["SYMBOL"]]
  gene_CDS[[length(gene_CDS)+1]] <- gene_test2[[1]][["ENTRY"]]
  #gene_path[[length(gene_path)+1]] <- gene_test2[[1]][["PATHWAY"]]
  gene_PATHS[[length(gene_PATHS)+1]] <- length(gene_test2[[1]][["PATHWAY"]])
  strwrk<-gene_test2[[1]][["POSITION"]]
  strwrk<-strsplit(strwrk, split = ":")
  if (strwrk[[1]][1]=="X"){
    strwrk<- as.numeric(0)
  }
  strwrk<-as.numeric(strwrk[[1]][1])
  gene_POS[[length(gene_POS)+1]] <- strwrk
  gene_AA[[length(gene_AA)+1]] <- gene_test2[[1]][["AASEQ"]]
  #gene_AL[[length(gene_AL)+1]] <- gene_test2[[1]][["AASEQ"]]@ranges@width
  gene_DA[[length(gene_DA)+1]] <- gene_test2[[1]][["NTSEQ"]]
  #gene_DL[[length(gene_DL)+1]] <- gene_test2[[1]][["NTSEQ"]]@ranges@width
  gene_ORTH[[length(gene_ORTH)+1]] <-gene_test2[[1]][["ORTHOLOGY"]]
}
for (x in 1:length(gene_AA)){
  gene_AL[[length(gene_AL)+1]] <- gene_AA[[x]]@ranges@width
}
for (x in 1:length(gene_DA)){
  gene_DL[[length(gene_DL)+1]] <- gene_DA[[x]]@ranges@width
}
gene_query <- gene_query<-as.data.frame(matrix(gene_query, nrow=664))
geneWAY <- mutate(gene_query, NAME = gene_NAME, CDS = gene_CDS, PATHS = gene_PATHS,
                  POS = gene_POS, DL = gene_DL)
geneWAY <- as.data.frame(lapply(geneWAY, unlist))
geneWAY <- geneWAY[order(-geneWAY$PATHS),]
```

This section also used keggGet to query data on the genes. We gather the data for the gene's name, CDS value, the number of pathways the gene contributes to, the chromosome the gene is found on, the gene's amino acid sequence, the gene's DNA sequence, and the gene's orthology reference number. Last, we format the genes to sort the dataframe by genes with the most pathways they support.

	V1	n	ID	NAME	DRUGL	COMPL	RELL	REF	PROT
hsa03010	path:hsa03010	59	Ribosome	Ribosome - Homo sapiens (human)	0	0	0	7	334
hsa01100	path:hsa01100	49	Metabolic pathways	Metabolic pathways - Homo sapiens (human)	2	0	84	0	0
hsa05171	path:hsa05171	45	Coronavirus disease - COVID-19	Coronavirus disease - COVID-19 - Homo sapiens (hu…	6	10	16	35	464
hsa05022	path:hsa05022	36	Pathways of neurodegeneration - multiple diseases	Pathways of neurodegeneration - multiple diseases - …	6	32	12	28	952
hsa05132	path:hsa05132	35	Salmonella infection	Salmonella infection - Homo sapiens (human)	0	5	5	14	498
hsa05012	path:hsa05012	31	Parkinson disease	Parkinson disease - Homo sapiens (human)	38	26	9	38	532
hsa05014	path:hsa05014	29	Amyotrophic lateral sclerosis	Amyotrophic lateral sclerosis - Homo sapiens (human)	6	14	13	47	728
hsa05010	path:hsa05010	27	Alzheimer disease	Alzheimer disease - Homo sapiens (human)	40	19	9	39	768
hsa04723	path:hsa04723	26	Retrograde endocannabinoid signaling	Retrograde endocannabinoid signaling - Homo sapie…	10	19	5	22	296

Figure 2: First 9 results of the pathways.

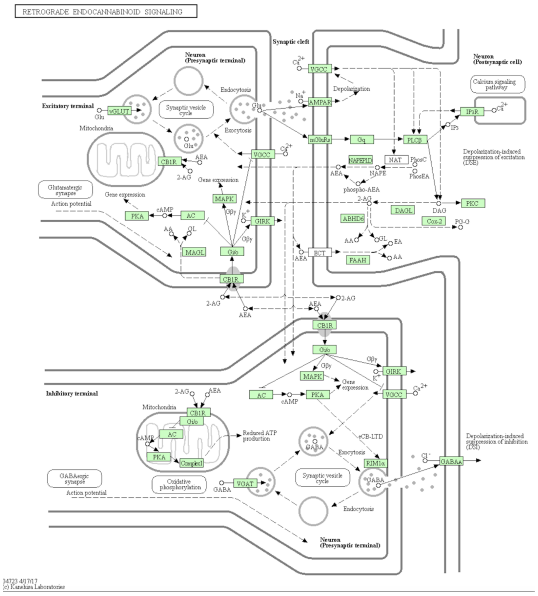

Figure 3: Retrograde endocannabinoid signaling pathway.

3 Results

With two dataframes for the pathways and genes, we can visualize the relationship between different columns of the two data frames.

3.1 Pathway

Figure 2 shows the first 9 results of the pathway query. the ribosome is the most contributed to pathway with 59 contributing genes, but the ribosome itself is not a pathway. The second, "metabolic pathways" contains all metabolic pathways. The first isolated metabolic pathway is Covid-19 infection pathway with 45 contributing genes. The first neuron-based metabolic pathway not related to disease is "retrograde endocannabinoid signaling", with 26 genes contributing. We use code to grab an image of the pathway from KEGG.

Listing 6: Picture of neuron-based pathway with most genes

```
strwrk <- pathWAY[[1]][[9]]
strwrk<-strsplit(strwrk, split = ":")
temp <- strwrk[[1]][2]
png<-keggGet(temp,"image")
t <- tempfile()
writePNG(png,t)
if (interactive()) browseURL(t)
```

Figure 3 shows the pathway, and which genes contribute to interneuronal synapse signaling.

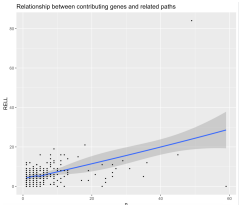

Figure 4: Relationship between N and Related paths.

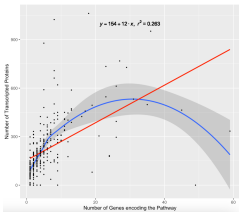

Figure 5: Relationship between N and number of proteins.

Notably, in Figure 5, participating genes in a pathway show a U-shaped relationship with the number of produced proteins. At a lower number of genes, the relationship is proportional, but as the number of encoding genes grow, the transcribed proteins decrease. The linear regression only has an Rsquared value of 0.265, supporting the U-shaped relationship hypothesis. The best fit line and graph was generated with the following code.

Listing 7: Linear Model Equation Function

```
lm_eqn <- function(df){
  m <- lm(PROT ~ n, df);
  eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(r)^2~"="~r2,
                   list(a = format(unname(coef(m)[1]), digits = 2),
                        b = format(unname(coef(m)[2]), digits = 2),
                        r2 = format(summary(m)$r.squared, digits = 3)))
  as.character(as.expression(eq));
}
ggplot(pathWAY, aes(x=n,y=PROT)) +
  geom_smooth()+
  geom_smooth(method = "lm",se=FALSE, color = "Red" )+
  geom_text(x = 30, y = 1000, label = lm_eqn(pathWAY), parse = TRUE)+
  geom_point(size=0.5)+
  xlab("Number of Genes encoding the Pathway") +
  ylab("Number of Transcripted Proteins")
  ggtitle("Relationship Between Encoding Genes and Proteins") +
  theme_bw()
```

The other graphs were generated witht the following code:

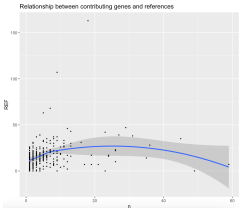

Figure 6: Relationship between N and references.

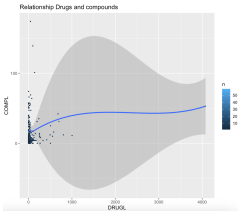

Figure 7: Relationship drugs and compounds.

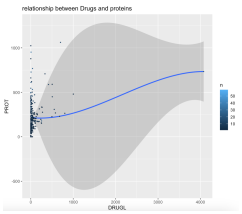

Figure 8: Relationship between drugs and proteins.

Listing 8: Pathway visualization.

```
##Relationship between N and Related paths
ggplot(pathWAY, aes(x=n,y=RELL)) +
  geom_smooth()+
  geom_point(size=0.5)+
   ggtitle("Relationship between contributing genes and related paths")
  #Relationship between N and references
ggplot(pathWAY, aes(x=n,y=REF)) +
  geom_smooth()+
  geom_point(size=0.5)+
   ggtitle("Relationship between contributing genes and references")
#Relationship Drugs and compounds
ggplot(pathWAY, aes(x=DRUGL,y=COMPL,color=n)) +
  geom_smooth()+
  geom_point(size=0.5)+
   ggtitle("Relationship Drugs and compounds")
#relationship between Drugs and proteins
ggplot(pathWAY, aes(x=DRUGL,y=PROT,color=n)) +
  geom_smooth()+
  geom_point(size=0.5)+
   ggtitle("relationship between Drugs and proteins")
#relationship between compounds and proteins
ggplot(pathWAY, aes(x=COMPL,y=PROT,color=n)) +
  geom_smooth()+
  geom_point(size=0.5)+
   ggtitle("relationship between compounds and proteins")
```

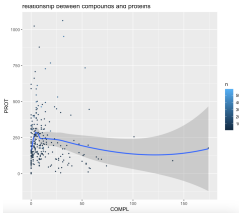

Figure 9: Relationship between compounds and proteins.

	V1	NAME	CDS	PATHS	POS	DL
14	hsa:5567	PRKACB, CAFD2, PKA_C-beta, PKACB	5567	70	1	1056
261	hsa:5879	RAC1, MIG5, MRD48, Rac-1, TC-25, p21-Rac1	5879	52	7	579
81	hsa:4792	NFKBIA, EDAID2, IKBA, MAD-3, NFKBI	4792	49	14	954
405	hsa:801	CALM1, CALML2, CAM2, CAM3, CAMB, CAMC, CAMI, ...	801	43	14	450
144	hsa:2771	GNAI2, GIP, GNAI2B, HG1C, H_LUCA15.1, H_LUCA16.1	2771	41	3	1068
263	hsa:54205	CYCS, CYC, HCS, THC4	54205	34	7	318
195	hsa:815	CAMK2A, CAMKA, CaMKIINalpha, CaMKIIalpha, MRD5...	815	33	5	1437
142	hsa:1499	CTNNB1, CTNNB, EVR7, MRD19, NEDSDV, armadillo	1499	31	3	2346
555	hsa:71	ACTG1, ACT, ACTG, DFNA20, DFNA26, HEL-176	71	30	17	1128
317	hsa:5728	PTEN, 10q23del, BZS, CWS1, DEC, GLM2, MHAM, MMA...	5728	28	10	1212
2	hsa:2782	GNB1, HG2A, MDS, MRD42	2782	20	1	1023
500	hsa:6416	MAP2K4, JNKK, JNKK1, MAPKK4, MEK4, MKK4, PRKMK...	6416	20	17	1200
258	hsa:5058	PAK1, IDDMSSD, PAKalpha, alpha-PAK, p65-PAK	5058	19	11	1638
44	hsa:481	ATP1B1, ATP1B	481	18	1	912
78	hsa:6777	STAT5B, GHISID2, STAT5	6777	18	17	2364
228	hsa:718	C3, AHUS5, ARMD9, ASP, C3a, C3b, CPAMD1, HEL-S-...	718	17	19	4992
476	hsa:3455	IFNAR2, IFN-R, IFN-alpha-REC, IFNABR, IFNARB, IMD45	3455	17	21	1548
283	hsa:5516	PPP2CB, PP2Abeta, PP2CB	5516	16	8	930
208	hsa:3326	HSP90AB1, D6S182, HSP84, HSP90B, HSPC2, HSPCB	3326	15	6	2175
397	hsa:8660	IRS2, IRS-2	8660	15	13	4017
43	hsa:6391	SDHC, CYB560, CYBL, PGL3, QPS1, SDH3	6391	14	1	510

Figure 10: Gene dataframe head.

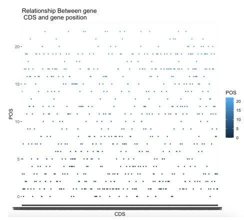

Figure 11: Relationship between gene CDS and gene position.

3.2 Gene Results

Figure 10 shows a portion of the gene dataframe. The following figures show the relationship between each of the columns, but not relationships stand out for the purposes of this project. The following code shows the format each comparison was made:

Listing 9: Gene visualization.

```
ggplot(geneWAY, aes(x=CDS, y=POS, color=POS)) +
  geom_smooth()+
  geom_point(size=0.5)+
  ggtitle("Relationship Between gene \n CDS and gene position")
  ##Relationship Between CDS and Gene Paths
ggplot(geneWAY, aes(x=CDS, y=PATHS, color=POS)) +
  geom_smooth()+
  geom_point(size=0.5)+
  ggtitle("Relationship Between CDS and Gene Paths")
##Relationship Between CDS and DNA Length
ggplot(geneWAY, aes(x=CDS, y=DL, color=POS)) +
  geom_smooth()+
  geom_point(size=0.5) +
  ggtitle("Relationship Between CDS and DNA Length")
##Relationship between Gene postion and gene paths
ggplot(geneWAY, aes(x=POS, y=PATHS, color=POS)) +
  geom_smooth()+
  geom_point(size=0.5) +
  ggtitle("Relationship between Gene postion and gene paths")
##relationship between Gene position and DNA length
ggplot(geneWAY, aes(x=POS, y=DL, color=POS)) +
  geom_smooth()+
  geom_point(size=0.5) +
  ggtitle("relationship between Gene position and DNA length")
##relationship between Gene paths and DNA length
ggplot(geneWAY, aes(x=PATHS, y=DL, color=POS)) +
  geom_smooth()+
  geom_point(size=0.5) +
  ggtitle("relationship between Gene paths and DNA length")
```

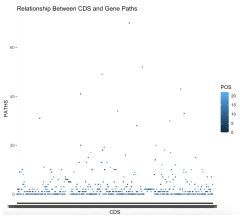

Figure 12: Relationship between CDS and Gene Paths.

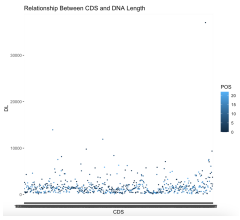

Figure 13: Relationship between CDS and DNA length.

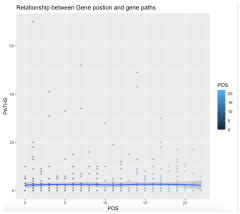

Figure 14: Relationship between Gene postion and gene paths.

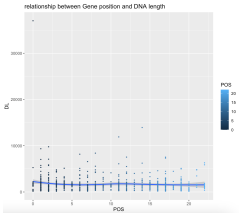

Figure 15: Relationship between Gene position and DNA length.

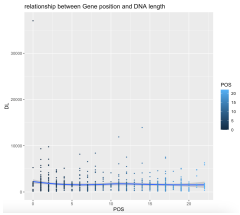

Figure 16: relationship between Gene paths and DNA length.

4 Analysis

Ribosome metabolic pathways were the most contributed pathway with 59 unique tau-correlated genes. This is expected since the original paper examined the transcriptome of neuronal soma. The transcriptome describes the protein production in the ribosomes. The Stanford paper identified synaptic vesicle cycling as possible pathways, and the pathways retrieved from KEGG also identify several synaptic metabolic pathways affected by tau-correlated genes, such as neuroactive ligand-receptor interaction [4].

5 Discussion

To make this study useful to future research avenues, we need to examine other pathways than the most contributed to pathway. Further investigation into the relationship between Covid-19 metabolism and taupathy, as well as possible metabolic mechanisms of retrograde endocannabinoid signaling, may contribute to understanding of how multi-omic changes can contribute to the AD brain. This paper is to demonstrated R software developed to congregate relevant multiomic data to identify unexpected metabolic pathways which may impact Alzheimer's Disease. By applying this software to tau-correlated genes, this paper identified Covid-19 and two neurotransmitter signaling pathways as previously unexamined in relation to tau-correlated genes. These pathways may play a role in the emergence of taupathy and warrant further investigation.

References

[1] D. S. Knopman, H. Amieva, R. C. Petersen, G. Chételat, D. M. Holtzman, B. T. Hyman, R. A. Nixon, and D. T. Jones, "Alzheimer disease," *Nature reviews Disease primers*, vol. 7, no. 1, pp. 1–21, 2021.

[2] M. Thambisetty, Y. An, and T. Tanaka, "Alzheimer's disease risk genes and the age-at-onset phenotype," *Neurobiology of aging*, vol. 34, no. 11, pp. 2696–e1, 2013.

[3] Y.-T. Wu, A. S. Beiser, M. Breteler, L. Fratiglioni, C. Helmer, H. C. Hendrie, H. Honda, M. A. Ikram, K. M. Langa, A. Lobo, *et al.*, "The changing prevalence and incidence of dementia over time—current evidence," *Nature Reviews Neurology*, vol. 13, no. 6, pp. 327–339, 2017.

[4] M. Otero-Garcia, S. U. Mahajani, D. Wakhloo, W. Tang, Y.-Q. Xue, S. Morabito, J. Pan, J. Oberhauser, A. E. Madira, T. Shakouri, *et al.*, "Molecular signatures underlying neurofibrillary tangle susceptibility in alzheimer's disease," *Neuron*, vol. 110, no. 18, pp. 2929–2948, 2022.

[5] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic acids research*, vol. 28, no. 1, pp. 27–30, 2000.

[6] D. Tenenbaum and B. Maintainer, "Keggrest: Client-side rest access to the kyoto encyclopedia of genes and genomes (kegg), r package version 1.34. 0; 2021."