DA5020 Final Project

Annie Bryant

2019-12-01

GitHub Project Information

All of the files referenced here are located in this GitHub repository: https://github.com/anniegbryant/DA5020_Final_Project. This includes the R source code, Rmd file used to generate this report, R environment data, and .csv input and output.

Project Overview

For my final project, I chose to curate a database of protein-specific information acquired from the UniProt Knowledge Base[1]. I wanted the R script to read in a list of UniProt IDs from a CSV file and output R and SQL dataframes containing relevant information corresponding to each UniProt entry, including molecular function, biological process, Reactome pathway, and disease involvement. This involves querying the UniProt webpage for each ID using rvest[2], cleaning the data within R using dplyr[3], and organizing the data into a SQL database with normalized tables using sqldf[4].

My motivation for this project comes from my work as a Research Technician in the lab of Dr. Brad Hyman at Massachusetts General Hospital. I joined a project over the summer in which plasma samples were collected from cognitively impaired patients at two or three time points over several years to measure protein expression. Specifically, expression levels for 414 different proteins were measured using a highly-sensitive transcription-based amplification technology from Olink Proteomics, yielding normalized relative protein expression values for each plasma sample. In total, for 414 proteins and 123 patients with plasma from multiple time points (2 or 3), this data amounts to 145,314 data points. One predominant goal of this project is to identify biomarkers that either predict or correlate with cognitive decline, since both protein expression and cognitive status were measured at each visit for each patient.

Multiple groups in the lab are analyzing this resulting dataset with other various goals, focusing on different subsets of the 414 proteins assayed. My group is particularly interested in the dynamics of cerebral vasculature with AD pathogenesis and progression[5]. As such, we have focused on a smaller subset of 21 vasculature-related proteins. I came across the R package Time-course Gene Set Analysis (TcGSA) package which is designed to analyze longitudinal RNA-Seq data by grouping individual genes into gene sets, *a priori*, which are known to share common biological functions and/or expression patterns[6]. While I wasn't able to get this package to work with my protein data, it inspired me to approach our longitudinal protein expression data analysis from a similar perspective.

I realized it would be tremendously helpful to have functional information about each protein in addition

to the expression data, in order to potentially identify overarching trends in protein networks.Hopefully, the SQL database created in this project will provide further insight into the relationship between protein expression change and cognitive decline going forward.

About the UniProt Data Involved

In this longitudinal study, expression levels were measured for 414 different proteins. One particularly useful collection of information I used comes from the **Gene Ontology**[7,8], which is an online consortium integrating structural and functional information about genes and gene products from numerous sources. I used UniProt's Retrieve/ID Mapping tool to download the following information about each of these proteins:

- **Gene Ontology (GO) Biological Process**: Broader biological processes achieved via several molecular activities, from DNA repair to lipid transporter activity.
- **Gene Ontology Cellular Component**: Intracellular component(s) in which a gene product executes a particular function – for example, ribosome or Golgi apparatus.
- **Gene Ontology Molecular Function**: Activities performed at the molecular level by one or more gene products, including transporter activity and Toll-like receptor binding.
- **Tissue specificity**: Organ(s) or organ system(s) in which the protein is expressed throughout the body.
- **Function overview**: Higher-level information about the general function(s) of the protein.
- **KEGG ID**: ID linking the UniProt entry to the corrresponding entry in the Kyoto Encyclopedia of Genes and Genomes (KEGG)[9,10,11]

To hone in on specific information I wanted, I also used rvest to scrape the following information about each protein:

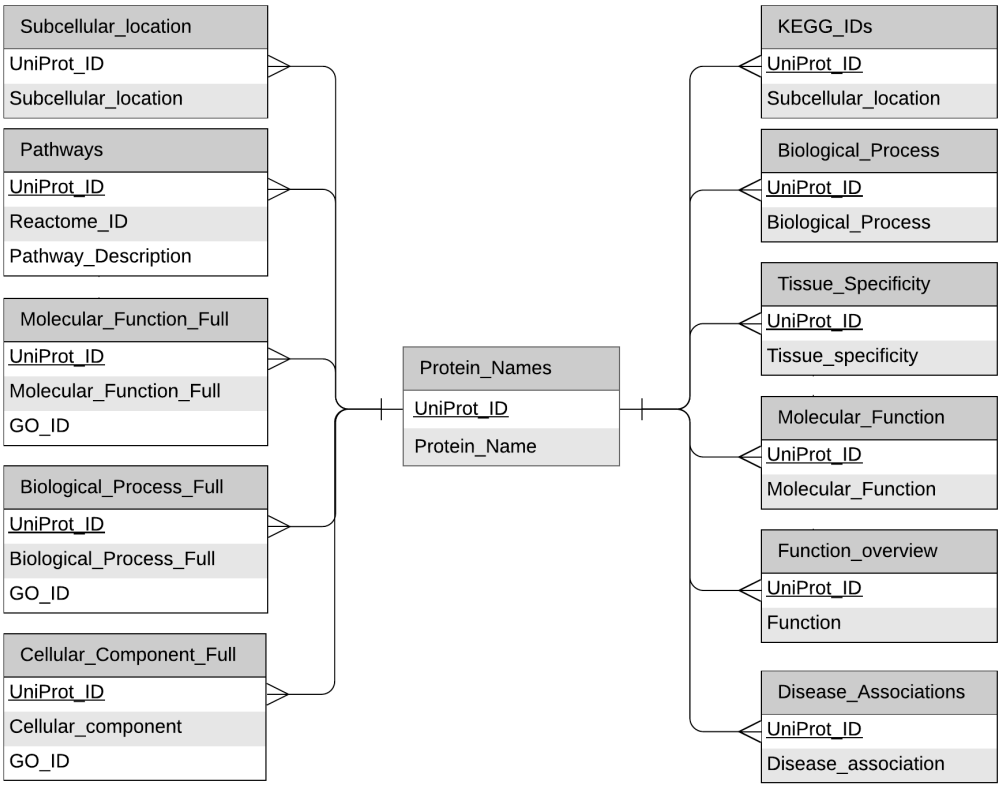
- **Biological Process keyword**: One or two primary GO biological processes.
- **Molecular Function keyword**: One or two primary GO molecular functions.
- **Associated diseases**: Any disease(s) associated with genetic variations in the protein.
- **Subcellular location**: The location and the topology of the mature protein in the cell.
- **Reactome Pathway**: The ID and description associated with a Reactome Pathway, an expansive collection of biological pathways and processes.[12]

Method Selection

I downloaded some data directly from the UniProt batch download tool, and any data I couldn't download in CSV format, I extracted via web scraping. For the web scraping, I chose to use rvest over other scraping tools we covered this semester (e.g. import.io) since I wanted to access 414 different webpages programmatically, an application for which rvest is well-suited. By adding an 0.5s delay between webpage queries, I avoided bombarding the UniProt server without exponentially increasing the code runtime. I opted to load each UniProt webpage and read its HTML contents just once, and then apply helper functions to extract the desired information (e.g. GO keyword, subcellular location) from each page once loaded. Each helper function takes in a UniProt page upon which read_html() has been called, and extracts the pertinent information to save into a dataframe.

I prefer to work with the dataframes in dplyr before adding them to SQL data tables. Therefore, once all of the UniProt data was extracted via CSV download or web scrape, I used dplyr to clean the data. The biggest issue with the preliminary dataset was that many cells contained multiple values, separated by delimiters such as commas, periods, or semi-colon. I chose to split these values into separate rows, yielding two- or three-column dataframes, with the UniProt ID in one column and the data value(s) in the other column(s). This meant my dataframes were largely in long format, which I anticipate will be more conducive for my downstream analysis involving grouping proteins by function or pathway. Once the data was cleaned, I used sqldf to add the dataframes to the SQLite connection which can then be queried in RStudio with dbGetQuery() commands.

SQL was very conducive for storing and accessing this data since it is inherently tabular in nature, and the different tables are not necessarily directly related, aside from all containing UniProt_IDs. SQL databases can be easily queried with R and the output is conveniently formatted as a dataframe, which will be very useful for generating figures with ggplot2 and for statistical analysis. I created an Entity-Relationship (ER) diagram to demonstrate how the protein names table links to all the other tables:



Issues Encountered

I went through several iterations of my R code. My first approach was to use self-contained functions to download the HTML contents of each UniProt page for each value I wanted to extract, instead of downloading the content once and passing in helper functions. For example, I originally wrote the following two functions that separately extracted the GO Keywords and Reactome Pathways from UniProt pages that were downloaded twice, which is redunant and takes far longer:

```
# Extract GO Keywords
extract_keywords <- function(id) {
  #Sleep for 0.5s between requests to not overload Uniprot server
  Sys.sleep(0.5)

  #Create temporary dataframe to store this page's data
  page <- data.frame(X1="NA", X2="NA", Uniprot_ID=id)
```

```
  #Try the given URL and return NA if there is a 404 error
  try(
    page <- sprintf(uniprot_url, URLencode(id)) %>%
      read_html() %>%
      #Keywords are contained in the first databaseTable
      html_node(".databaseTable") %>%
      #Convert to table (dataframe)
      html_table() %>%
      #Add Uniprot ID
      mutate(Uniprot_ID = id),
    silent=TRUE
  )
  #Append to dataframe
  all_keywords <- rbind(all_keywords, page)
}

# Extract Reactome Pathway
extract_pathway <- function(id) {
  #Sleep for 0.5s between requests to not overload Uniprot server
  Sys.sleep(0.5)

  #Create temporary dataframe to store this page's data
  path <- data.frame(Uniprot_ID=id, .="NA")

  try(
    path <- sprintf(uniprot_url, URLencode(id)) %>%
      read_html() %>%
      html_nodes(".databaseTable :contains(ReactomeiR)") %>%
      html_text() %>%
      .[2] %>%
      as.data.frame(.) %>%
      mutate(Uniprot_ID=id),
```

```
    silent=TRUE
  )
  rc_path <- rbind(rc_path, path)
}
```

Once I realized this redundancy and instead wrote individual helper functions and one larger function to download the HTML content, it was easy to tweak the helper functions to get exactly the format of the content I wanted. It was also easier to download all the content in one go, and then once the data was downloaded and stored in dataframes, to clean and organize the data using dplyr afterward.

Summary of Data Collected

In total, I have compiled eleven distinct data tables that encompass protein function, cellular location, and protein pathways for the 414 proteins in our study. More specifically, I have amassed the following distinct counts of values across the respective tables:

- Biological Process, Keywords: 81
- Biological Process, Full: 2825
- Cellular Compartment, Full: 335
- Disease Associations: 200
- Function overview: 1763
- Molecular Function, Keywords: 67
- Molecular Function, Full: 580
- Subcellular Location: 41
- Reactome Pathway: 583
- Tissue Specificity: 638
- KEGG IDs: 414

Future Implementations

As described, the purpose for this project is to obtain more information about the proteins studied in the longitudinal Alzheimer's Disease plasma study so that we can examine trends among protein pathways and

across functional domains. While I cannot share any of the actual data or analysis for this project due to confidentiality, I am excited about how these datasets will facilitate new discoveries for our project going forward. I have thus far only focused on acquiring and organizing the data, as per the project guidelines, but the next phase of project implementation will be to perform statistical analyses (e.g. T-Tests, Logistic Regression, PCA) with biological process and Reactome pathway as categorical variables.

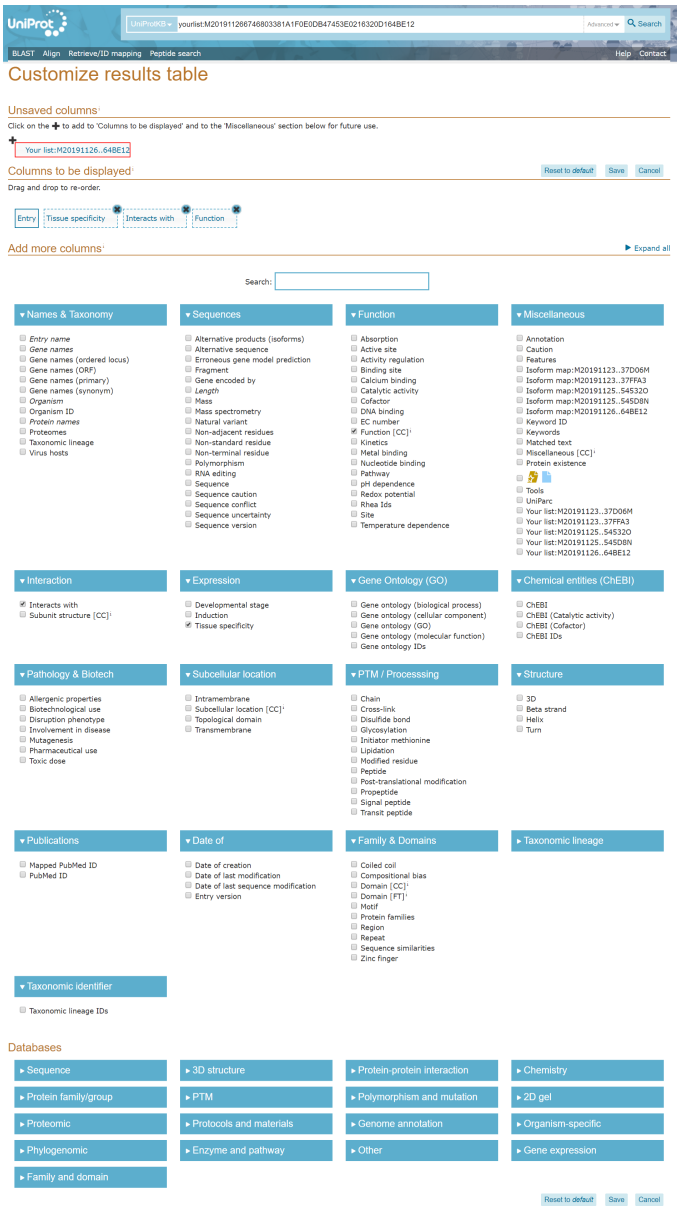
References

1. UniProt Consortium. (2018). UniProt: a worldwide hub of protein knowledge. Nucleic acids research, 47(D1), D506-D515.
2. Hadley Wickham (2019). rvest: Easily Harvest (Scrape) Web Pages. R package version 0.3.5. https://CRAN.R-project.org/package=rvest
3. Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2018). dplyr: A Grammar of Data Manipulation. R package version 0.7.6. https://CRAN.R-project.org/package=dplyr
4. G. Grothendieck (2017). sqldf: Manipulate R Data Frames Using SQL. R package version 0.4-11. https://CRAN.R-project.org/package=sqldf
5. Bennett, R. E., Robbins, A. B., Hu, M., Cao, X., Betensky, R. A., Clark, T., ... & Hyman, B. T. (2018). Tau induces blood vessel abnormalities and angiogenesis-related gene expression in P301L transgenic mice and human Alzheimer's disease. Proceedings of the National Academy of Sciences, 115(6), E1289-E1298.
6. Hejblum BP, Skinner J, and Thiebaut R (2015) Time-Course Gene Set Analysis for Longitudinal Gene Expression Data. PLoS Comput Biol 11(6): e1004310.<doi:10.1371/journal.pcbi.1004310
7. Ashburner et al. Gene ontology: tool for the unification of biology. Nat Genet. May 2000;25(1):25-9.
8. The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. Nucleic Acids Res. Jan 2019;47(D1):D330-D338.
9. Kanehisa, M. and Goto, S.; KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 28, 27-30 (2000).
10. Kanehisa, M., Sato, Y., Furumichi, M., Morishima, K., and Tanabe, M.; New approach for understanding genome variations in KEGG. Nucleic Acids Res. 47, D590-D595 (2019).
11. Kanehisa, M; Toward understanding the origin and evolution of cellular organisms. Protein Sci. (2019)
12. Fabregat, A., Jupe, S., Matthews, L., Sidiropoulos, K., Gillespie, M., Garapati, P., ... & Milacic, M. (2017). The reactome pathway knowledgebase. Nucleic acids research, 46(D1), D649-D655.

Appendix: Methods and Figures

Methods

I first explored UniProt's Retrieve/ID Mapping tool (https://www.uniprot.org/uploadlists/) to see what information I could download for a given list of UniProt IDs. The tool allows a user to enter a list of UniProt IDs or upload a file containing IDs, after which the user can select which column(s) they wish to download corresponding to each UniProt ID entry.



I opted to download the Gene Ontology (GO) Biological Process, GO Cellular Component, GO Molecular Function, Function overview, Reactome ID, and KEGG ID. This downloaded file is available in the GitHub project as UniProt_download.csv.

```
UP_df <- read.csv("UniProt_download.csv", stringsAsFactors = F)
colnames(UP_df)[1] <- "UniProt_ID"
```

Here is an excerpt of this dataset:

UniProt_ID	GO_BP	GO_CC	GO_MF	Tissue_specificity	Function	KEGG_ID
O00175	cell-cell signa	extracellular s	CCR3 chemokine	Activated monoc	Chemotactic for	hsa:6369;
O00182	cellular respon	collagen-contai	carbohydrate bi	Peripheral bloo	Binds galactosi	hsa:3965;
O00300	apoptotic proce	extracellular m	cytokine activi	Highly expresse	Acts as decoy r	hsa:4982;
O00308	cellular protei	cytoplasm [GO:0	RNA polymerase	Detected in hea	E3 ubiquitin-pr	hsa:11060;
O00533	adult locomotor	apical part of	protease bindin	Expressed in th	Extracellular m	hsa:10752;

I then normalized this dataset by breaking it up into several subsets, each in long format with one value per row and (potentially) multiple rows per UniProt ID.

First, I created a **Tissue_specificity** tibble to store UniProt IDs and the corresponding tissue(s) in which the protein is typically present. Some proteins are expressed in multiple sets of tissue systems, the names of which were all concatenated in one value. To simplify for downstream analysis, I split the string strings by the [.] delimiter and stored separate entries in different rows, such that one UniProt ID may be listed in several rows corresponding to different tissues.

```
Tissue_specificity <- UP_df %>%
  select(UniProt_ID, Tissue_specificity) %>%
  mutate(Tissue_specificity = str_replace_all(Tissue_specificity, "[(]PubMed:.*[)]", "")) %>%
  filter(str_detect(Tissue_specificity, "ECO", negate=T),
         Tissue_specificity != "") %>%
  na.omit()
```

UniProt_ID	Tissue_specificity
O00175	Activated monocytes and activated T lymphocytes.
O00182	Peripheral blood leukocytes and lymphatic tissues
O00182	Expressed in lung, liver, breast and kidney with higher levels in tumor endothelial cells than no...
O00182	Expressed in trophoblast cells in decidua and placenta in pregnancy (at protein level)
O00182	Isoform 2 is the most abundant isoform expressed in endothelial cells
O00182	Upon endothelial cell activation isoform 2 expression decreases while expression of isoform 3 and...
O00182	Isoform 4 decreases in pathological pregnancy
O00300	Highly expressed in adult lung, heart, kidney, liver, spleen, thymus, prostate, ovary, small inte...
O00300	Detected at very low levels in brain, placenta and skeletal muscle
O00300	Highly expressed in fetal kidney, liver and lung.

I created a **Function_overview** tibble to store each sentence in the protein's general function overview as a separate row.

```
# Table for Function_overview
Function_overview <- UP_df %>%
  select(UniProt_ID, Function) %>%
  mutate(Function = str_split(Function, "[.] ")) %>%
  unnest(Function) %>%
  mutate(Function = str_replace_all(Function, "[(]PubMed:.*[)]", "")) %>%
  filter(Function != "") %>%
  filter(str_detect(Function, "ECO", negate=T),
         Function != "")
```

UniProt_ID	Function
O00175	Chemotactic for resting T-lymphocytes, and eosinophils
O00175	Has lower chemotactic activity for neutrophils but none for monocytes and activated lymphocytes
O00175	Is a strong suppressor of colony formation by a multipotential hematopoietic progenitor cell line
O00175	Binds to CCR3.
O00182	Binds galactosides

Lastly, I created a table linking UniProt IDs with the corresponding KEGG ID so I can later extract information from the Kyoto Encyclopedia of Genes and Genomes (KEGG, https://www.genome.jp/kegg/).

```
# Table for KEGG IDs
UniProt_KEGG <- UP_df %>%
  select(UniProt_ID, KEGG_ID)
```

UniProt_ID	KEGG_ID
O00175	hsa:6369;
O00182	hsa:3965;
O00300	hsa:4982;
O00308	hsa:11060;
O00533	hsa:10752;

I created separate dataframes for the BP, CC, and MF. As with other dataframes, I split multiple values from one string into multiple rows per UniProt ID.

Biological Process:

```
# Table for all Gene Ontology Biological Processes
GO_BP_Full <- UP_df %>%
  select(UniProt_ID, GO_BP) %>%
  mutate(GO_BP = str_split(GO_BP, "; ")) %>%
  unnest(GO_BP) %>%
  filter(GO_BP != "") %>%
  separate(GO_BP, into=c("Biological_process", "GO_ID"), sep=" \\[") %>%
  mutate(GO_ID = str_replace(GO_ID, "\\]", ""))
```

UniProt_ID	Biological_process	GO_ID
O00175	cell-cell signaling	GO:0007267
O00175	cellular response to interferon-gamma	GO:0071346
O00175	cellular response to interleukin-1	GO:0071347
O00175	cellular response to tumor necrosis factor	GO:0071356
O00175	chemokine-mediated signaling pathway	GO:0070098

Cellular Components:

```
# Table for all Gene Ontology Cellular Components
GO_CC_Full <- UP_df %>%
  select(UniProt_ID, GO_CC) %>%
```

```
  mutate(GO_CC = str_split(GO_CC, "; ")) %>%
  unnest(GO_CC) %>%
  filter(GO_CC != "") %>%
  separate(GO_CC, into=c("Cellular_component", "GO_ID"), sep=" \\[") %>%
  mutate(GO_ID = str_replace(GO_ID, "\\]", ""))
```

UniProt_ID	Cellular_component	GO_ID
O00175	extracellular space	GO:0005615
O00182	collagen-containing extracellular matrix	GO:0062023
O00182	cytoplasm	GO:0005737
O00182	cytosol	GO:0005829
O00182	extracellular space	GO:0005615

Molecular Functions:

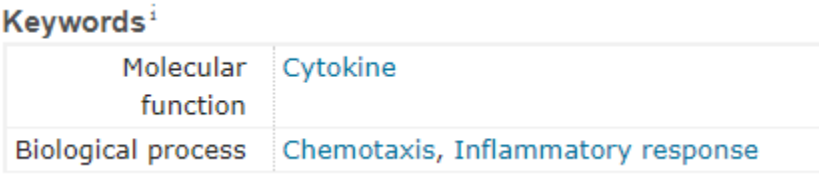
```
# Table for all Gene Ontology Molecular Functions
GO_MF_Full <- UP_df %>%
  select(UniProt_ID, GO_MF) %>%
  mutate(GO_MF = str_split(GO_MF, "; ")) %>%
  unnest(GO_MF) %>%
  filter(GO_MF != "") %>%
  separate(GO_MF, into=c("Molecular_function", "GO_ID"), sep=" \\[") %>%
  mutate(GO_ID = str_replace(GO_ID, "\\]", ""))
```

UniProt_ID	Molecular_function	GO_ID
O00175	CCR3 chemokine receptor binding	GO:0031728
O00175	CCR chemokine receptor binding	GO:0048020
O00175	chemokine activity	GO:0008009
O00175	receptor ligand activity	GO:0048018
O00182	carbohydrate binding	GO:0030246

This dataset contains exhaustive information about the functions, interactions, and localized expression of each protein in the list. However, given the large number of proteins in our study (n=414), I thought it would

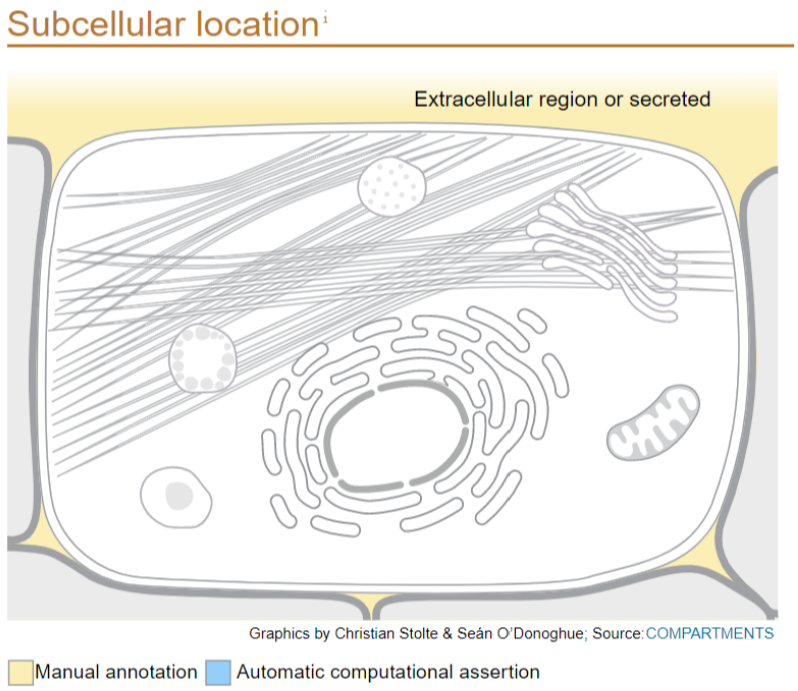
be beneficial to acquire more succinct information about primary functions and locations of the proteins. This would enable a simpler first-pass analysis to identify relevant protein functions and pathways, which I could then examine in greater detail. I visited the UniProt page for the first protein in the list, with ID O00175, to identify potential information that could serve this purpose.

I noticed that there is a "Keywords" section, with a succinct list of one or two primary Biological Processes and Molecular Functions per protein:

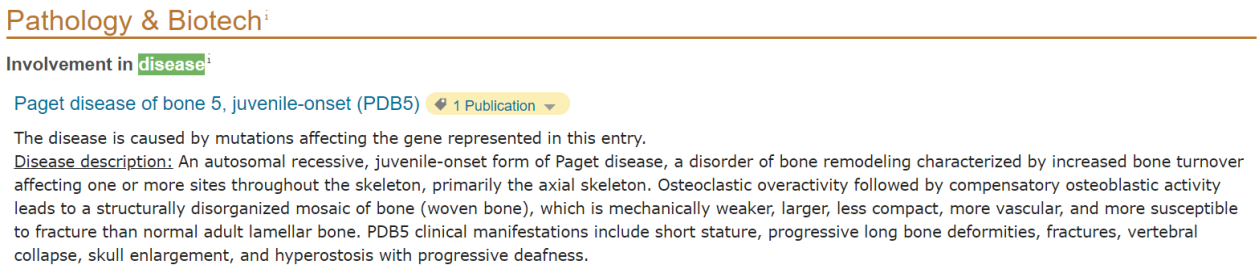


These keywords are manually selected by UniProt database curators from the UniProtKB to reflect the primary one or two biological processes and molecular functions in which a particular protein participates.

There is also a Keywords sections for Subcellular location:



The UniProt page also includes any noted disease associations with the given protein:



My next step would thus be to scrape the Keywords, Subcellular Location, and any Disease Associations for each protein in the list. I loaded a CSV file containing the protein names and UniProt IDs included in the longitudinal study.

```
# Load UniProt IDs and protein names
proteins <- read.csv("Protein_UniProt_List.csv")


# Create matrix of UniProt_IDs
UniProt_IDs <- as.matrix(proteins$Uniprot_ID)
```

The base URL for any UniProt protein query is https://www.uniprot.org/uniprot/.

```
uniprot_url <- "https://www.uniprot.org/uniprot/%s"
```

To reduce computation time and minimize queries sent to UniProt, I chose to load each webpage and read its contents just once, and then apply helper functions to extract the desired information from each page once loaded. Each helper function below takes in a UniProt page upon which read_html() has been called, and extracts the pertinent information to save into a dataframe.

GO Keywords:

```
# Function to extract keywords from UniProt page for a given UniProt ID
extract_keywords <- function(page) {
  page %>%
      #Keywords are contained in the first databaseTable
      html_node(".databaseTable") %>%
      #Convert to table (dataframe)
      html_table()
}
```

Disease Associations:

```
# Function to extract diseases from UniProt page for a given UniProt ID
extract_diseases <- function(page) {
  page %>%
    html_node(".diseaseAnnotation") %>%
    html_nodes(xpath="//a[contains(@href, '/diseases/')]") %>%
    html_text() %>%
    as.data.frame()
}
```

Subcellular location:

```
# Function to extract subcellular location
extract_location <- function(page) {
  page %>%
    html_nodes(xpath="//*[@class='namespace-uniprot']/
                main[@id='content']/
                div[@class='main-aside']/
                div[@class='content entry_view_content up_entry swissprot']/
                div[@id='subcellular_location']/
                span/a") %>%
    html_text() %>%
    as.data.frame()
}
```

Reactome Pathway:

```
# Function to extract pathway
extract_pathway <- function(page) {
  page %>%
    html_nodes(".databaseTable :contains(ReactomeiR)") %>%
    html_text() %>%
    # Only keep the second text entry
```

```
    .[2] %>%
    as.data.frame()
}
```

Initialize empty dataframes to append results from helper functions:

```
# Initialize empty dataframes
keyword_df <- data.frame(X1=character(0), X2=character(0),
                         UniProt_ID=character(0))
disease_df <- location_df <- pathway_df <- data.frame(.=character(0),
                                                      UniProt_ID=character(0))
```

Extract_all is the function which will load a UniProt URL, read and parse the associated html content, and call each of the helper functions to extract all of the desired information.

```
# Given a UniProt ID, apply all the above functions to extract relevant info
extract_all <- function(id) {
  Sys.sleep(0.5)

  #Try the given URL and return NA if there is a 404 error
  try(
    page <- sprintf(uniprot_url, URLencode(id)) %>%
      read_html(),
    silent=T
  )

  # Keywords
  kw <- extract_keywords(page) %>%
    mutate(UniProt_ID = id) %>%
    distinct()
  keyword_df <<- rbind(keyword_df, kw)

  # Associated diseases
  disease <- extract_diseases(page) %>%
```

```
    mutate(UniProt_ID = id) %>%
    distinct()
  disease_df <<- rbind(disease_df, disease)

  # Subcellular location
  location <- extract_location(page) %>%
    mutate(UniProt_ID = id) %>%
    distinct()
  location_df <<- rbind(location_df, location)

  # Pathway
  pathway <- extract_pathway(page) %>%
    mutate(UniProt_ID = id) %>%
    distinct()
  pathway_df <<- rbind(pathway_df, pathway)
}
```

I then apply extract_all to each row in the 414x1 matrix containing the UniProt IDs.

```
# Calling invisible() to suppress console output
invisible(apply(UniProt_IDs, 1, extract_all))
```

Once the data is collected in individual dataframes, I then cleaned them using dplyr. I first renamed the columns in location_df and disease_df.

```
# Rename DF columnes
colnames(location_df) <- c("Subcellular_location", "UniProt_ID")
colnames(disease_df) <- c("Disease_association", "UniProt_ID")
```

Then I reorganized the keywords dataframe into separate Biological Process and Molecular Function dataframes.

```r
# Create dataframe to hold Biological Process and Molecular Function
BP_MF <- keyword_df %>%
  # Only interested in Biological Process or
  # Molecular Function keywords
  filter(str_detect(X1,
                    "<p>|Ligand|Reactome|SABIO-RK|SignaLink|SIGNOR|MEROPS|Error",
                    negate=T)) %>%
  spread(key=X1, value=X2)
```

To better organize rows with multiple values in one column, I split BP_MF into two dataframes, one for BP and one for MF.

```r
# Create one dataframe for biological processes
BP <- BP_MF %>%
  select(UniProt_ID, `Biological process`) %>%
  rename(Biological_Process = `Biological process`) %>%
  mutate(Biological_Process = str_split(Biological_Process, ", ")) %>%
  unnest(Biological_Process) %>%
  na.omit()

# Create one dataframe for molecular functions
MF <- BP_MF %>%
  select(UniProt_ID, `Molecular function`) %>%
  rename(Molecular_Function = `Molecular function`) %>%
  mutate(Molecular_Function = str_split(Molecular_Function, ", ")) %>%
  unnest(Molecular_Function) %>%
  na.omit()
```

The pathways dataframe contains multiple values in the Path column, all as one string. I split them by the "R-HSA-" delimiter and store distinct R-HSA pathway entries as separate rows.

```r
# Many UniProt proteins are mapped to several Reactome Pathways,
# This code splits multiple pathway entries into distinct rows
colnames(pathway_df) <- c("Path", "UniProt_ID")
```

```r
pathway_df <- pathway_df %>%
  rowwise() %>%
  mutate(Path = str_replace(Path, "Reactomei", "")) %>%
  # Split by the R-HSA- string
  mutate(Path= str_split(Path, "R-HSA-", n=Inf)) %>%

  # Un-list
  unnest(Path) %>%

  # Remove empty strings that reflect non-existent pathways
  filter(Path != "") %>%

  # Split into R-HSA IDs and pathway descriptions
  mutate(Path = str_split(Path, " ", n=2)) %>%
  purrr::quietly(unnest_wider)(Path) %>%

  # only keep Results of purrr::quietly()
  .[[1]] %>%
  rowwise() %>%
  mutate(`...1` = paste("R-HSA-", `...1`, sep="", collapse=""))

colnames(pathway_df) <- c("Reactome_ID", "Pathway_Description", "UniProt_ID")
```

Here are snippets of the cleaned web-scraped dataframes:

Biological Process

UniProt_ID	Biological_Process
O00175	Chemotaxis
O00175	Inflammatory response
O00182	Chemotaxis
O00182	Immunity
O00300	Apoptosis

Molecular Function

UniProt_ID	Molecular_Function
O00175	Cytokine
O00300	Receptor
O00308	Transferase
O00533	Developmental protein
O14625	Cytokine

Disease Associations

UniProt_ID	Disease_association
O00300	Paget disease of bone 5, juvenile-onset (PDB5)
O14788	Osteopetrosis, autosomal recessive 2 (OPTB2)
O15169	Hepatocellular carcinoma (HCC)
O15169	Caudal duplication anomaly (CADUA)
O95630	Microcephaly-capillary malformation syndrome (MICCAP)

Subcellular Location

UniProt_ID	Subcellular_location
O00175	Secreted
O00182	Cytoplasm
O00182	Nucleus
O00182	Secreted
O00300	Secreted

Reactome Pathways

UniProt_ID	Reactome_ID	Pathway_Description
O00182	R-HSA-451927	Interleukin-2 family signaling
O00300	R-HSA-5669034	TNFs bind their physiological receptors
O00308	R-HSA-8948751	Regulation of PTEN stability and activity
O00308	R-HSA-9013507	NOTCH3 Activation and Transmission of Signal to the Nucleus
O00533	R-HSA-447041	CHL1 interactions

Each of these cleaned dataframes is available in the GitHub project as a .csv file in the CSV Files folder.

Lastly, I used sqldf to add the dataframes to a SQL database.

```
# Connect to SQLite
db <- dbConnect(SQLite(), dbname="UniProt.sqlite")

dbWriteTable(conn = db, name = "Biological_Process", value=BP,
             row.names=FALSE, header=TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Biological_Process_Full", value=GO_BP_Full,
             row.names=F, header=T, overwrite=T)

dbWriteTable(conn = db, name = "Cellular_Component_Full", value=GO_CC_Full,
             row.names=F, header=T, overwrite=T)

dbWriteTable(conn = db, name = "Disease_Associations", value=disease_df,
             row.names=FALSE, header=TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Function_Overview", value=Function_overview,
             row.names=FALSE, header=TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Molecular_Function", value=MF,
             row.names=FALSE, header=TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Molecular_Function_Full", value=GO_MF_Full,
             row.names=F, header=T, overwrite=T)

dbWriteTable(conn = db, name = "Protein_Names", value = proteins,
             row.names = FALSE, header = TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Pathways", value=pathway_df,
```

```
             row.names=FALSE, header=TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Subcellular_Location", value=location_df,
             row.names=FALSE, header=TRUE, overwrite=T)

dbWriteTable(conn = db, name = "Tissue_Specificity", value=Tissue_specificity,
             row.names=F, header=T, overwrite=T)

dbWriteTable(conn = db, name = "KEGG_IDs", value=UniProt_KEGG,
             row.names=F, header=T, overwrite=T)
```

To confirm the tables were successfully added to the SQL database, I can call dbListTables() on the database:

```
dbListTables(db)
```

```
##  [1] "Biological_Process"      "Biological_Process_Full"
##  [3] "Cellular_Component_Full" "Disease_Associations"
##  [5] "Function_Overview"       "KEGG_IDs"
##  [7] "Molecular_Function"      "Molecular_Function_Full"
##  [9] "Pathways"                "Protein_Names"
## [11] "Subcellular_Location"    "Tissue_Specificity"
```

Results

To demonstrate the efficacy of accessing data from the SQL database, I queried the most frequent values from some of the data tables.

Biological Process Keywords:

```
BP_Count <- dbGetQuery(db, "SELECT Biological_Process, COUNT(*) AS 'Count'
           FROM Biological_Process
           GROUP BY Biological_Process
           ORDER BY COUNT(*) DESC")
```

Biological_Process	Count
Cell adhesion	48
Immunity	45
Inflammatory response	33
Apoptosis	31
Chemotaxis	29
Differentiation	27
Host-virus interaction	27
Adaptive immunity	23
Angiogenesis	23
Innate immunity	20

Biological Process, Full:

```
BP_Full_Count <- dbGetQuery(db, "SELECT Biological_process, COUNT(*) AS 'Count'
                            FROM Biological_Process_Full
                            GROUP BY Biological_process
                            ORDER BY COUNT(*) DESC")
```

Biological_process	Count
signal transduction	86
cytokine-mediated signaling pathway	72
immune response	65
inflammatory response	62
positive regulation of cell population proliferation	60
cell adhesion	47
cell-cell signaling	47
neutrophil degranulation	45
positive regulation of ERK1 and ERK2 cascade	42
negative regulation of apoptotic process	39

Molecular Function Keywords:

```
MF_Count <- dbGetQuery(db, "SELECT Molecular_Function, COUNT(*) as 'Count'
           FROM Molecular_Function
           GROUP BY Molecular_Function
           ORDER BY COUNT(*) DESC")
```

Molecular_Function	Count
Receptor	85
Cytokine	74
Hydrolase	56
Protease	38
Growth factor	37
Developmental protein	29
Transferase	24
Heparin-binding	20
Metalloprotease	16
Serine protease	15

Molecular Function, Full:

```
MF_Full_Count <- dbGetQuery(db, "SELECT Molecular_function, COUNT(*) as 'Count'
           FROM Molecular_Function_Full
           GROUP BY Molecular_function
           ORDER BY COUNT(*) DESC")
```

Molecular_function	Count
identical protein binding	59
cytokine activity	51
protein homodimerization activity	49
signaling receptor binding	44
growth factor activity	42
calcium ion binding	40
heparin binding	34
metal ion binding	32
chemokine activity	31
zinc ion binding	29

Disease Association:

```
DS_Count <- dbGetQuery(db, "SELECT Disease_association, COUNT(*) as 'Count'
            FROM Disease_Associations
            GROUP BY Disease_association
            ORDER BY COUNT(*) DESC")
```

Disease_association	Count
Psoriatic arthritis (PSORAS)	3
Hepatocellular carcinoma (HCC)	2
Ischemic stroke (ISCHSTR)	2
Adams-Oliver syndrome 5 (AOS5)	1
Allergic rhinitis (ALRH)	1

Function Overview:

```
FO_Count <- dbGetQuery(db, "SELECT Function, COUNT(*) as 'Count'
            FROM Function_Overview
            GROUP BY Function
            ORDER BY COUNT(*) DESC")
```

Function	Count
IGF-binding proteins prolong the half-life of the IGFs and have been shown to either inhibit or stimulate the growth promoting effects of the IGFs on cell culture	4
One of the major HIV-suppressive factors produced by CD8+ T-cells	3
Reversible hydration of carbon dioxide	3
They alter the interaction of IGFs with their cell surface receptors	3
They preferentially interact with themselves in a homophilic manner in connecting cells; cadherins may thus contribute to the sorting of heterogeneous cell types	3
This protein can bind heparin	3
Activates procollagenase.	2
Activation of PLCG1 leads to the production of the cellular signaling molecules diacylglycerol and inositol 1,4,5-trisphosphate	2
Activities are controlled by presence or absence of small cytoplasmic adapter proteins, SH2D1A/SAP and/or SH2D1B/EAT-2	2
After birth, activates or inhibits angiogenesis, depending on the context	2

Subcellular Location:

```
LC_Count <- dbGetQuery(db, "SELECT Subcellular_location AS 'Subcellular Location',
            COUNT(*) as 'Count'
            FROM Subcellular_Location
            GROUP BY Subcellular_location
            ORDER BY COUNT(*) DESC")
```

Subcellular Location	Count
Secreted	242
Membrane	179
Cell membrane	124
Cytoplasm	65
Nucleus	39
Extracellular matrix	31
Endoplasmic reticulum	17
Cell junction	16
Cell projection	15
Golgi apparatus	15

Cellular Compartment, Full:

```
CC_Full_Count <- dbGetQuery(db, "SELECT Cellular_component, COUNT(*) as 'Count'
            FROM Cellular_Component_Full
            GROUP BY Cellular_component
            ORDER BY COUNT(*) DESC")
```

Cellular_component	Count
extracellular region	218
extracellular space	211
plasma membrane	176
extracellular exosome	125
cytosol	87
integral component of membrane	87
cell surface	85
cytoplasm	81
integral component of plasma membrane	65
external side of plasma membrane	64

Reactome Pathways:

```
PT_Count <- dbGetQuery(db, "SELECT Reactome_ID,
            Pathway_Description AS 'Pathway Description', COUNT(*) as 'Count'
            FROM Pathways
            GROUP BY Reactome_ID
            ORDER BY COUNT(*) DESC")
```

Reactome_ID	Pathway Description	Count
R-HSA-6798695	Neutrophil degranulation	45
R-HSA-6785807	Interleukin-4 and Interleukin-13 signaling	34
R-HSA-5673001	RAF/MAP kinase cascade	29
R-HSA-6783783	Interleukin-10 signaling	26
R-HSA-380108	Chemokine receptors bind chemokines	25
R-HSA-198933	Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell	24
R-HSA-381426	Regulation of Insulin-like Growth Factor (IGF) transport and uptake by Insulin-like Growth Factor Binding Proteins (IGFBPs)	22
R-HSA-418594	G alpha (i) signalling events	21
R-HSA-1257604	PIP3 activates AKT signaling	19
R-HSA-6811558	PI5P, PP2A and IER3 Regulate PI3K/AKT Signaling	19