Drug Repurposing Data Engineering Write Up

Table of Contents

[Introduction: 3](#_Toc71537143)

[Detail of Data Sources: 3](#_Toc71537144)

[Gene – Gene Dendrogram (GNBR): 3](#_Toc71537145)

[Description: 3](#_Toc71537146)

[Source: 3](#_Toc71537147)

[Process: 3](#_Toc71537148)

[Chemical – Gene Dendrogram (GNBR): 3](#_Toc71537149)

[Description: 3](#_Toc71537150)

[Source: 3](#_Toc71537151)

[Process: 3](#_Toc71537152)

[Gene – Disease Dendrogram (GNBR): 3](#_Toc71537153)

[Description: 3](#_Toc71537154)

[Source: 3](#_Toc71537155)

[Process: 3](#_Toc71537156)

[Chemical – Disease Dendrogram (GNBR): 3](#_Toc71537157)

[Description: 3](#_Toc71537158)

[Source: 3](#_Toc71537159)

[Process: 3](#_Toc71537160)

[GeneDiseaseUniprot: 3](#_Toc71537161)

[Description: 3](#_Toc71537162)

[Source: 3](#_Toc71537163)

[Process: 3](#_Toc71537164)

[GeneGeneUniprot: 3](#_Toc71537165)

[Description: 3](#_Toc71537166)

[Source: 3](#_Toc71537167)

[Process: 3](#_Toc71537168)

[ChemicalGeneUniprot: 3](#_Toc71537169)

[Description: 3](#_Toc71537170)

[Source: 3](#_Toc71537171)

[Process: 3](#_Toc71537172)

[DBSmallChemicalGene: 3](#_Toc71537173)

[Description: 3](#_Toc71537174)

[Source: 3](#_Toc71537175)

[Process: 3](#_Toc71537176)

[CGDrugBankTruth: 4](#_Toc71537177)

[Description: 4](#_Toc71537178)

[Source: 4](#_Toc71537179)

[Process: 4](#_Toc71537180)

[DBSmallChemicalDisease: 4](#_Toc71537181)

[Description: 4](#_Toc71537182)

[Source: 4](#_Toc71537183)

[Process: 4](#_Toc71537184)

[CDDrugBankTruth: 4](#_Toc71537185)

[Description: 4](#_Toc71537186)

[Source: 4](#_Toc71537187)

[Process: 4](#_Toc71537188)

# Introduction:

The data collection process was a two-fold approach. We were given five datasets from the FDA team (DrugBank, FDA Clinical Trials, STRING Protein Interactions, COVID Virus Host Proteins, Uniprot ID table) to integrate as a first pass into the data engineering process. The Drugbank data was the most difficult to work with, as it was a large XML database with thousands of attributes per drug that we needed to parse through. The data engineering team looked through these attributes and with the help of the FDA subject matter experts, selected specific attributes to pull out that would assist us the most in our analysis.

With the FDA curated resources prepared for eventual integration, the team began an investigatory and research process into other data we could possibly leverage. We searched the broader knowledge in the drug repurposing topic to see if there were other data sources that were used a lot in previous work. We kept coming across a drug/gene/disease knowledge graph database called the Global Network of Biomedical Relationships that was used in several journals pertaining to drug repurposing in the past. This knowledge graph, GNBR for short, is a meta-analysis of tens of thousands of research paper abstracts that describe specific relationships between drugs, genes (proteins), and diseases. This information was extracted via their proprietary text parser algorithm that would extract the specific gene, disease, or drug/chemical mentioned. The database was separated into 8 datasets, two for each entity relationship type, containing a dataset of entity characteristic ranks and a dataset of ground truth information. In the ranking datasets called “path”, there were several fields called “themes” that are rankings of specific characteristics of the type of relationship that these entities had together. Finally, in the ground truth datasets with the name “dependency”, the GNBR team went and curated six validated databases to find the IDs of the all the entities they extracted. All this information was vital and used in the data preparation and integration process. The tables were joined together using their unique “DependencyPath” as a key.

# Detail of Data Sources:

## Gene – Gene Dendrogram (GNBR):

### Description:

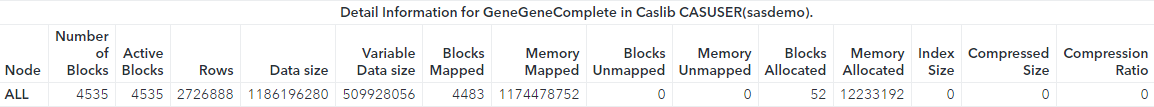
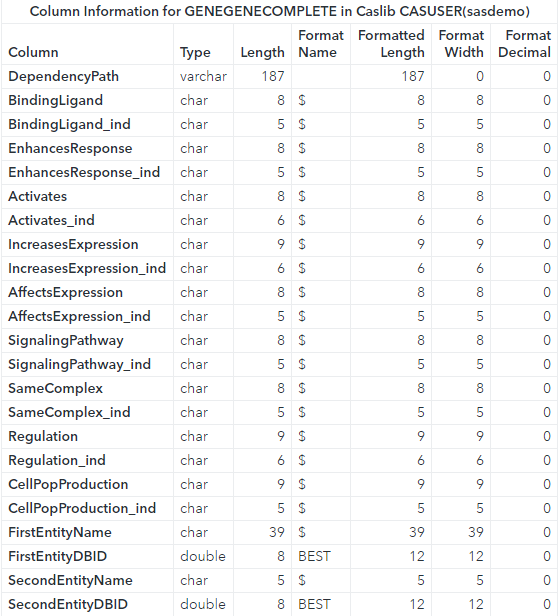
The first dendrogram to go through the data ingestion process was the Gene – Gene dataset. There wasn’t much preprocessing with this particular dataset besides the joining due to the dataset containing GeneIDs which were mappable back to DrubBank using the Uniprot dataset as a bridge table. This dataset contains information specific to known gene interactions and their characteristics.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

As of 9/24/2019 (Version 7)

### Process:

data mycas.GeneGeneDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneGeneDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath Sentence;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.GeneGenePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneGenePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;

proc fedsql sessref=casauto;

create table casuser.GeneGeneComplete {options replace=true copies=0} as

select GeneGenePath.\*, GeneGeneDependency.FirstEntityName, GeneGeneDependency.FirstEntityDBID,

GeneGeneDependency.SecondEntityName, GeneGeneDependency.SecondEntityDBID

from casuser.GeneGenePath, casuser.GeneGeneDependency

where GeneGenePath.DependencyPath = GeneGeneDependency.DependencyPath;

quit;

## Chemical – Gene Dendrogram (GNBR):

### Description:

The second dendrogram needed a bit more work to cleanse the dataset. There was an issue with the GeneIDs that were extracted, some fields had several GeneIDs and some had a TaxonomyID for gene’s analyzed that weren’t from humans. These discrepancies needed to be dealt with in order for the dataset to be ready for joining. We were able to parse out the TaxonomyID and, after discussion with our subject matter experts at the FDA, decided to just keep the first instance of the GeneID for usage in joining.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

As of 9/24/2019 (Version 7)

### Process:

data mycas.ChemicalGeneDependency(replace=yes copies=0);

length Dependency\_Path varchar(200);

set GNBR.ChemicalGeneDependency;

Dependency\_Path = lowcase(DependencyPath);

drop DependencyPath;

rename Dependency\_Path = DependencyPath;

run;

data mycas.ChemicalGenePath(replace=yes copies=0);

length Dependency\_Path varchar(200);

set GNBR.ChemicalGenePath;

Dependency\_Path = lowcase(DependencyPath);

drop DependencyPath;

rename Dependency\_Path = DependencyPath;

run;

/\*Once the datasets are in CAS with the standardized DependencyPath, its time to investigate the data and start parsing out the GeneIds from the Dependency datasets\*/

data mycas.ChemicalGeneDependency(replace=yes copies=0);

set mycas.ChemicalGeneDependency;

if find(SecondEntityDBID, 'Tax') ge 1 then CleanFlag = 1;

else CleanFlag = 0;

run;

/\*Extract Taxonomy ID from the SecondEntityDBID, once it's extracted we can remove the delimiters to just get the ID, also leaving blanks for DBIDs without Taxonomy\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGeneDependency;  
 if CleanFlag = 1 then   
 do;  
 Taxonomy=substr(SecondEntityDBID, index(SecondEntityDBID, ":"));  
 Taxonomy=compress(Taxonomy,":)");   
 end;  
 else Taxonomy=.;  
run;  
  
/\*The next step is to remove the taxonomy IDs from the original SecondEntityDBID so that we can use these numbers to join with DrugBank\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGeneDependency;  
 if CleanFlag = 1 then   
 do;  
 SecondEntityDBID= scan(SecondEntityDBID,1, '(');  
 end;  
 else SecondEntityDBID=SecondEntityDBID;  
run;  
  
/\*Parse out two IDs in field\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGenedependency;  
 if Cleanflag = 0 then  
 do;  
 ID2=scan(SecondEntityDBID, 2,";");  
 ID3=scan(SecondEntityDBID, 3, ";");  
 end;  
 else if CleanFlag = 1 then  
 do;  
 ID2=.;  
 ID3=.;  
 end;  
 SecondEntityDBID=scan(SecondEntityDBID, 1, ";");   
run;  
  
/\*proc sort data=mycas.ChemicalGeneDependency;  
 by descending ID2;  
run;\*/  
  
/\*Now that we have the Taxonomy extracted and the DB IDs cleaned its time to convert data types to prepare for joining and drop unneeded variables\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 length SecondEntityDBID\_ 8;  
 length Taxonomy\_ 8;  
 length ID2\_ 8;  
 length ID3\_ 8;  
 set mycas.ChemicalGeneDependency;  
 SecondEntityDBID\_ = SecondEntityDBID;  
 Taxonomy\_ = Taxonomy;  
 ID2\_ = ID2;  
 ID3\_ = ID3;  
 drop SecondEntityDBID Taxonomy SecondDBID CleanFlag else ID2 ID3;  
 rename SecondEntityDBID\_ = SecondEntityDBID;  
 rename Taxonomy\_ = Taxonomy;  
 rename ID2\_ = ID2;  
 rename ID3\_ = ID3;  
run;

## Gene – Disease Dendrogram (GNBR):

### Description:

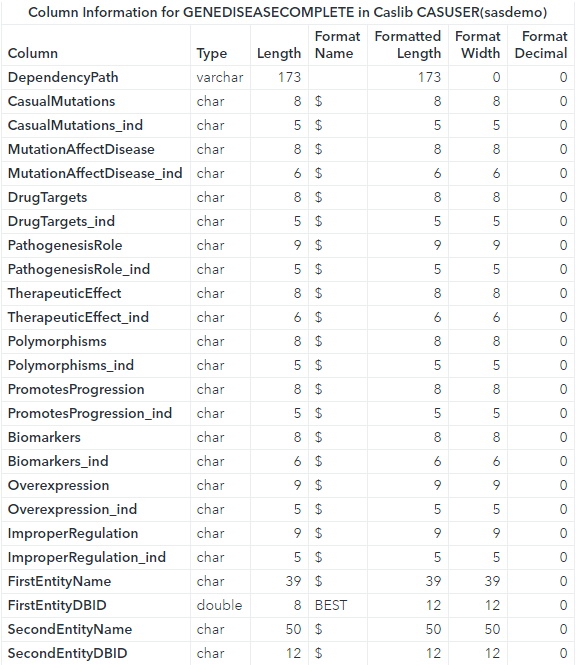
The third dendrogram to go through the data preparation process were the Gene – Disease datasets. This one also did not require a ton of data cleansing work. We had two usable IDs to integrate with the DrugBank data. The first was the GeneIDs used by the other datasets but there was also MESHIDs, which is a unique ID for a specific disease. Depending on the modeling approach we leverage both IDs in integration.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

As of 9/24/2019 (Version 7)

### Process:



data mycas.GeneDiseaseDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneDiseaseDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath Sentence;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.GeneDiseasePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneDiseasePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;

/\*Join newly added GNBR CAS data together to create completed datasets of the Dendrograms Gene-Gene Gene-Disease Gene-Chemical Chemical-Disease\*/  
proc fedsql sessref=casauto;  
 create table casuser.GeneDiseaseComplete {options replace=true copies=0} as  
 select GeneDiseasePath.\*, GeneDiseaseDependency.FirstEntityName, GeneDiseaseDependency.FirstEntityDBID,   
 GeneDiseaseDependency.SecondEntityName, GeneDiseaseDependency.SecondEntityDBID  
 from casuser.GeneDiseasePath, casuser.GeneDiseaseDependency   
 where GeneDiseasePath.DependencyPath = GeneDiseaseDependency.DependencyPath;  
quit;

## Chemical – Disease Dendrogram (GNBR):

### Description:

The final dataset exported from the GNBR website for use was the chemical – disease dendrogram. This one we found much more difficult to join in with the DrugBank data. While we could use the disease MESHID, this wouldn’t be the most efficient join. There was also a parsing error on the GNBR side with extracting drug/chemical names from the ground truth abstracts. The first step was to first join the “Dependency” dataset with the “Path” dataset. Once complete, we then made sure to include some ground truth information like the raw sentence which would eventually be used for chemical/drug name extraction. This table by itself wasn’t used much but was eventually leveraged to enhance the modelers created tables from the other datasets.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

As of 9/24/2019 (Version 7)

### Process:

data mycas.ChemicalDiseaseDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.ChemicalDiseaseDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.ChemicalDiseasePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.ChemicalDiseasePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;

/\*Join newly added GNBR CAS data together to create completed datasets of the Dendrograms Gene-Gene Gene-Disease Gene-Chemical Chemical-Disease\*/  
proc fedsql sessref=casauto;  
 create table casuser.ChemicalDiseaseComplete {options replace=true copies=0} as  
 select ChemicalDiseasePath.\*, ChemicalDiseaseDependency.FirstEntityName, ChemicalDiseaseDependency.FirstEntityDBID,   
 ChemicalDiseaseDependency.SecondEntityName, ChemicalDiseaseDependency.SecondEntityDBID, ChemicalDiseaseDependency.Sentence  
 from casuser.ChemicalDiseasePath, casuser.ChemicalDiseaseDependency   
 where ChemicalDiseasePath.DependencyPath = ChemicalDiseaseDependency.DependencyPath;  
quit;

## GeneDiseaseUniprot:

### Description:

The created GeneDiseaseComplete dataset, which contains all of the GNBR information from the Gene – Disease perspective, needed to be joined with the Uniprot data in order to be integrated with the other FDA datasets. We joined it with the Uniprot ID table which contains mappable IDs for genes/proteins in different formats. Once we got a uniport ID for each gene in the GNBR dataset, this dataset is now ready to be joining with the other FDA datasets based on the modelers needs.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

<https://ftp.uniprot.org/pub/databases/uniprot/knowledgebase/docs/sec%5Fac.txt>

As of 4/14/2021

### Process:

/\*Load in Uniprot CSV of GeneIDs\*/  
proc casutil;  
 load file="/opt/sas/viya/config/data/cas/default/public/FDA/UniProtGeneIDs.csv"   
 outcaslib="Casuser" casout="UniprotGeneID";  
run;

/\*The final join is a bit more simple, only one GeneID from this Dendrogram, so we will do a join on that\*/  
proc fedsql sessref=casauto;  
 create table Public.GeneDiseaseUniprot {options replace=true copies=0} as  
 select GeneDiseaseComplete.\*, UniprotGeneID.GeneID, UniprotGeneID.UniProtID   
 from casuser.GeneDiseaseComplete left outer join casuser.UniprotGeneID  
 on (GeneDiseaseComplete.FirstEntityDBID = UniprotGeneID.GeneID);  
quit;

## GeneGeneUniprot:

### Description:

The created GeneGeneComplete dataset, which contains all of the GNBR information from the Gene – Gene perspective, needed to be joined with the Uniprot data in order to be integrated with the other FDA datasets. We joined it with the Uniprot ID table which contains mappable IDs for genes/proteins in different formats. We decided on using the first instance of the GeneID for the joins, however in this dataset there are two instances of GeneIDs for each entity relationship. Once we got a uniport ID for each gene in the GNBR dataset, this dataset is now ready to be joining with the other FDA datasets based on the modelers needs.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

<https://ftp.uniprot.org/pub/databases/uniprot/knowledgebase/docs/sec%5Fac.txt>

As of 4/14/2021

### Process:

/\*Load in Uniprot CSV of GeneIDs\*/  
proc casutil;  
 load file="/opt/sas/viya/config/data/cas/default/public/FDA/UniProtGeneIDs.csv"   
 outcaslib="Casuser" casout="UniprotGeneID";  
run;

/\*This first join we are using the first GeneID extracted, we could use the second GeneID since its a Gene-Gene relationship but that would be up to the team\*/  
proc fedsql sessref=casauto;  
 create table public.GeneGeneUniprot {options replace=true copies=0} as  
 select GeneGeneComplete.\*, UniprotGeneID.GeneID, UniprotGeneID.UniProtID  
 from casuser.GeneGeneComplete left outer join casuser.UniprotGeneID  
 on (GeneGeneComplete.SecondEntityDBID = UniprotGeneID.GeneID);  
quit;

## ChemicalGeneUniprot:

### Description:

The created ChemicalGeneComplete dataset, which contains all of the GNBR information from the Chemical - Gene perspective, needed to be joined with the Uniprot data in order to be integrated with the other FDA datasets. We joined it with the Uniprot ID table which contains mappable IDs for genes/proteins in different formats. Once we got a uniport ID for each gene in the GNBR dataset, this dataset is now ready to be joining with the other FDA datasets based on the modelers needs.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

<https://ftp.uniprot.org/pub/databases/uniprot/knowledgebase/docs/sec%5Fac.txt>

As of 4/14/2021

### Process:

/\*Load in Uniprot CSV of GeneIDs\*/  
proc casutil;  
 load file="/opt/sas/viya/config/data/cas/default/public/FDA/UniProtGeneIDs.csv"   
 outcaslib="Casuser" casout="UniprotGeneID";  
run;

/\*We are joing this data on the first GeneID extracted, some instances there were multiple IDs but we are going off of the first one for now\*/  
proc fedsql sessref=casauto;  
 create table public.ChemicalGeneUniprot {options replace=true copies=0} as  
 select ChemicalGeneComplete.\*, UniprotGeneID.GeneID, UniprotGeneID.UniProtID   
 from casuser.ChemicalGeneComplete left outer join casuser.UniprotGeneID  
 on (ChemicalGeneComplete.SecondEntityDBID = UniprotGeneID.GeneID);  
quit;

## DBSmallChemicalGene:

### Description:

The next few datasets described here are joins done from the GNBR data with some curated datasets prepared for modeling, the first of these was combining the DrugBank curated data with the Chemical Gene dendrogram. These were done by joining the tables on the extracted drug name from the GNBR dataset (referred to as chemical in GNBR), with the drug names in the DrugBank data. To deal with the parsing error we had to perform several string manipulation techniques using only the first letter of the drug extracted and the ground truth sentence from GNBR. We lost a lot of data in this process but were still able to create some fairly large datasets. The reason we couldn’t leverage SAS text analytics procedures was due to the nature of the data and the lack of patterns in how drugs/chemicals are written and described in the raw sentences.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

Combined with tables created from Ryan Carr’s DrugBank work

As of 4/30/2021

### Process:

/\*Repeat the same process for Chemical - Gene Dendrogram\*/  
data casuser.CGNormalizedEntityLength;  
 set casuser.Chemicalgenecomplete;  
 ChemicalLocation = index(Sentence, FirstEntityName);  
run;  
  
data casuser.CGNormalizedEntityLength;  
 set casuser.CGNormalizedEntityLength;  
 Chemical\_Name\_Full = substr(Sentence, ChemicalLocation, 70);  
 Chemical\_Name = scan(Chemical\_Name\_Full, 1, '');  
 drop ChemicalLocation;  
run;  
  
/\*Lets count the amount of times we were only able to pull the number or letter\*/  
data casuser.CGDrugChemicalExtracts;  
 set casuser.CGNormalizedEntityLength;  
 if length(Chemical\_Name) =< 1 then flag=1;  
 else flag=0;  
run;

/\*We can try several merges on the different DrugBank sets (MDS\_100Drugs\_V1, Truth, DBSmall\_Truth)\*/  
/\*DBSmall\_Truth to start for Chemical Gene\*/  
proc fedsql sessref=casauto;  
 create table casuser.DBSmallChemicalGene {options replace=true copies=0} as  
 select CGDrugChemicalExtracts.\*, DBSmallM\_Truth.\*  
 from casuser.CGDrugChemicalExtracts left outer join public.DBSmallM\_Truth  
 on (upcase(CGDrugChemicalExtracts.Chemical\_Name) = upcase(DBSmallM\_Truth.name));  
quit;  
  
data casuser.DBSmallChemicalGene;  
 set casuser.DBSmallChemicalGene;  
 where name is not missing;  
 drop Chemical\_Name\_Full name Sentence;  
run; /\*88661 Records\*/

## CGDrugBankTruth:

### Description:

The next dataset we are merging with the Chemical – Gene dendrogram was the FDA clinical trials data called the “Truth” dataset. This dataset was combined with our DrugBank data to provide the truth set for our models. We created this dataset to enhance the drug/protein information in the known truth dataset from the FDA.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

Combined with tables created from Ryan Carr’s DrugBank work

As of 4/30/2021

### Process:

/\*Repeat the same process for Chemical - Gene Dendrogram\*/  
data casuser.CGNormalizedEntityLength;  
 set casuser.Chemicalgenecomplete;  
 ChemicalLocation = index(Sentence, FirstEntityName);  
run;  
  
data casuser.CGNormalizedEntityLength;  
 set casuser.CGNormalizedEntityLength;  
 Chemical\_Name\_Full = substr(Sentence, ChemicalLocation, 70);  
 Chemical\_Name = scan(Chemical\_Name\_Full, 1, '');  
 drop ChemicalLocation;  
run;  
  
/\*Lets count the amount of times we were only able to pull the number or letter\*/  
data casuser.CGDrugChemicalExtracts;  
 set casuser.CGNormalizedEntityLength;  
 if length(Chemical\_Name) =< 1 then flag=1;  
 else flag=0;  
run;

/\*Next on to try is merging on the truth dataset for Chemical Gene\*/  
proc fedsql sessref=casauto;  
 create table casuser.CGDrugBankTruth {options replace=true copies=0} as  
 select CGDrugChemicalExtracts.\*, Truth.\*  
 from casuser.CGDrugChemicalExtracts left outer join public.Truth  
 on (upcase(CGDrugChemicalExtracts.Chemical\_Name) = upcase(Truth.iName));  
quit;  
  
data casuser.CGDrugBankTruth;  
 set casuser.CGDrugBankTruth;  
 where iName is not missing;  
 drop Chemical\_Name\_Full iName Sentence;  
run;/\*20688\*/

## DBSmallChemicalDisease:

### Description:

We then conducted a similar process for the Chemical – Disease dendrogram, we first joined the GNBR data with the DrugBank drug dataset. This was done by joining the tables on the extracted drug name from the GNBR dataset (referred to as chemical in GNBR), with the drug names in the DrugBank data. To deal with the parsing error we had to perform several string manipulation techniques using only the first letter of the drug extracted and the ground truth sentence from GNBR.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

Combined with tables created from Ryan Carr’s DrugBank work

As of 4/30/2021

### Process:

/\*Use First Entity Location, take the last two numbers seperated by the comma and minus them (that gets the length of the word), next we find the first word that begins with  
the extracted entity name and parse out everything after that first letter with the length coming from the entity location information\*/  
data casuser.CDNormalizedEntityLength;  
 set casuser.Chemicaldiseasecomplete;  
 ChemicalLocation = index(Sentence, FirstEntityName);  
run;  
  
data casuser.CDNormalizedEntityLength;  
 set casuser.CDNormalizedEntityLength;  
 Chemical\_Name\_Full = substr(Sentence, ChemicalLocation, 70);  
 Chemical\_Name = scan(Chemical\_Name\_Full, 1, '');  
 drop ChemicalLocation;  
run;  
  
/\*Lets count the amount of times we were only able to pull the number or letter\*/  
data casuser.CDDrugChemicalExtracts;  
 set casuser.CDNormalizedEntityLength;  
 if length(Chemical\_Name) =< 1 then flag=1;  
 else flag=0;  
run;

/\*DBSmall\_Truth\*/  
proc fedsql sessref=casauto;  
 create table casuser.DBSmallChemicalDisease {options replace=true copies=0} as  
 select CDDrugChemicalExtracts.\*, DBSmallM\_Truth.\*  
 from casuser.CDDrugChemicalExtracts left outer join public.DBSmallM\_Truth  
 on (upcase(CDDrugChemicalExtracts.Chemical\_Name) = upcase(DBSmallM\_Truth.name));  
quit;  
  
data casuser.DBSmallChemicalDisease;  
 set casuser.DBSmallChemicalDisease;  
 where name is not missing;  
 drop Chemical\_Name\_Full name Sentence;  
run;/\*62119\*/

## CDDrugBankTruth:

### Description:

The final dataset we are margining with the Chemical – Disease dendrogram was the FDA clinical trials data called the “Truth” dataset. This dataset was combined with our DrugBank data to provide to truth set for our models. We created this dataset to enhance the drug/disease information in the known truth dataset from the FDA.

### Source:

<https://zenodo.org/record/3459420#.YJrH5bVKg2w>

Combined with tables created from Ryan Carr’s DrugBank work

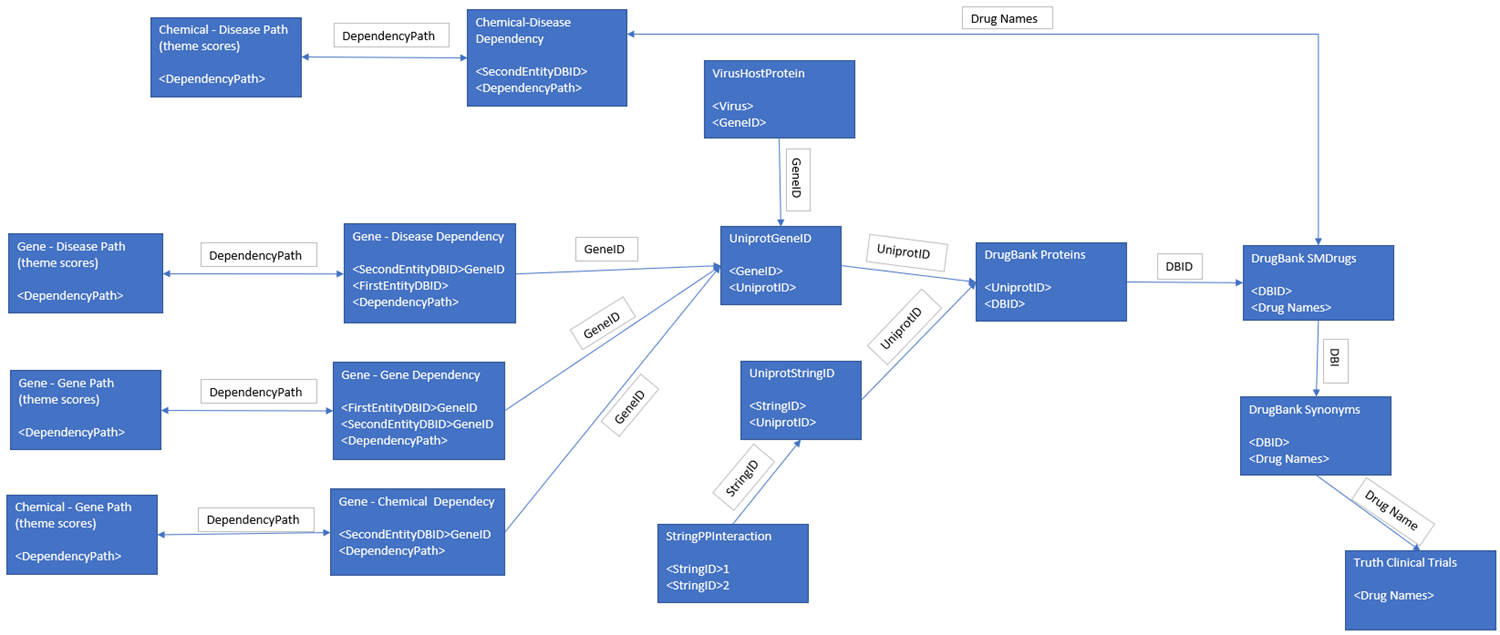
As of 4/30/2021

### Process:

/\*Use First Entity Location, take the last two numbers seperated by the comma and minus them (that gets the length of the word), next we find the first word that begins with  
the extracted entity name and parse out everything after that first letter with the length coming from the entity location information\*/  
data casuser.CDNormalizedEntityLength;  
 set casuser.Chemicaldiseasecomplete;  
 ChemicalLocation = index(Sentence, FirstEntityName);  
run;  
  
data casuser.CDNormalizedEntityLength;  
 set casuser.CDNormalizedEntityLength;  
 Chemical\_Name\_Full = substr(Sentence, ChemicalLocation, 70);  
 Chemical\_Name = scan(Chemical\_Name\_Full, 1, '');  
 drop ChemicalLocation;  
run;  
  
/\*Lets count the amount of times we were only able to pull the number or letter\*/  
data casuser.CDDrugChemicalExtracts;  
 set casuser.CDNormalizedEntityLength;  
 if length(Chemical\_Name) =< 1 then flag=1;  
 else flag=0;  
run;

/\*Truth\*/  
proc fedsql sessref=casauto;  
 create table casuser.CDDrugBankTruth {options replace=true copies=0} as  
 select CDDrugChemicalExtracts.\*, Truth.\*  
 from casuser.CDDrugChemicalExtracts left outer join public.Truth  
 on (upcase(CDDrugChemicalExtracts.Chemical\_Name) = upcase(Truth.iName));  
quit;  
  
data casuser.CDDrugBankTruth;  
 set casuser.CDDrugBankTruth;  
 where iName is not missing;  
 drop Chemical\_Name\_Full iName Sentence;  
run;/\*20708\*/

# Entity Relationship Diagram:



# Additional Steps/Actions:

The first major hurdle we had to handle was dealing with the overall size of the data. These dendrograms were all millions of rows and were fairly long. On top of this we were using a SAS demo environment for our work which wasn’t sized appropriately to deal with these large amounts of data. In a production use case, we would size out an appropriate server instance to accommodate the large data loads. However, we were able to use a few techniques leveraging SAS’ new architecture, Cloud Analytical Services, the handle these data size issues. The first technique was to set copies=0, by default CAS will make a copy of the dataset to add a level on redundancy to the data. But since our data doesn’t need redundancy, we can overwrite that setting. The next method was to use PROC FEDSQL instead of PROC SQL, the difference between the two procedures is that PROC FEDSQL leverages the new Viya CAS distributed architecture to handle large data whereas PROC SQL leverages the old SAS compute server. Finally, we set compress=true when passing data back and forth and doing large joins, this will eliminate any added white space in our column data.

In terms of next steps, the goals from the data perspective are to automate and be flexible in adjusting to the needs of the modelers. With respect to automation, there is a need to expand this framework beyond just COVID but being able to apply this to future pandemics, this means we need to have a consistent up to date data model that can be ideally updated with one job run rather than manually uploading and updating new datasets every time. In terms of the FDA data, while there may not be an opportunity to automate the internal FDA data sources, we can certainly pull data from DrugBank updates and we have scripts that achieve this task. Also, there is no real issue of having old versions of data as the FDA gets continual updates on drug development. Dealing with the GNBR data has proven to be much more difficult in terms of the longevity of the use of the knowledge graph. The GNBR data has seemed to be updated on a cadence of about 6-8 months but the last update hasn’t been since December 2019, so in terms of the continual usage in the future we need to figure out if the GNBR team plans on adding additional updates to the knowledge graph. Additionally, using API procedures like PROC HTTP, to automatically pull data from the GNBR website, from SAS needs to be investigated further as the possible usage of those functions still needs to be tested.

# Appendix:

**Procedures Used:**

* Proc Import – <https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/proc/n1qn5sclnu2l9dn1w61ifw8wqhts.htm>
* Cas session – <https://go.documentation.sas.com/doc/en/vdmmlcdc/8.1/casl/n0akw8thj5057on1db39sn55stn5.htm>
* Proc casutil – <https://go.documentation.sas.com/doc/en/vdmmlcdc/8.1/casref/p0p9kn6ouww8xfn1c402guo4uxmh.htm>
* Proc Fedsql – <https://go.documentation.sas.com/doc/en/vdmmlcdc/8.1/proc/p0aq8wvz0nw5mvn1atw6gqqotdlu.htm>
* Proc contents – <https://go.documentation.sas.com/doc/en/vdmmlcdc/8.1/proc/n1hqa4dk5tay0an15nrys1iwr5o2.htm>

**Journal and Research References:**

* <https://zenodo.org/record/3459420#.YDfQuWhKg2x>​
* <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004216#sec014>​
* <https://academic.oup.com/bioinformatics/article/34/15/2614/4911883#119219742>​
* <https://academic.oup.com/bioinformatics/article/34/15/2614/4911883#119219725>

**Full Script:**

/\*Pull in Data sources from the server\*/  
/\*In this current version, files need to be manually uploaded to the server from the GNBR website and then brought in via proc import\*/  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-i-chemical-disease-path-theme-distributions.txt"  
 out = work.ChemicalDiseasePath  
 dbms = dlm  
 replace;  
 getnames=no;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-i-chemical-gene-path-theme-distributions.txt"  
 out = work.ChemicalGenePath  
 dbms = dlm  
 replace;  
 getnames=no;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-i-gene-disease-path-theme-distributions.txt"  
 out = work.GeneDiseasePath  
 dbms = dlm  
 replace;  
 getnames=no;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-i-gene-gene-path-theme-distributions.txt"  
 out = work.GeneGenePath  
 dbms = dlm  
 replace;  
 getnames=no;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-chemical-disease-sorted-with-themes.txt"  
 out = work.ChemicalDiseaseDependencyThemes  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-chemical-disease-sorted.txt"  
 out = work.ChemicalDiseaseDependency  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-chemical-gene-sorted-with-themes.txt"  
 out = work.ChemicalGeneDependencyThemes  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-chemical-gene-sorted.txt"  
 out = work.ChemicalGeneDependency  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-gene-gene-sorted-with-themes.txt"  
 out = work.GeneGeneDependencyThemes  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-gene-gene-sorted.txt"  
 out = work.GeneGeneDependency  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
  
proc import datafile = "/opt/sas/viya/config/data/cas/default/public/GNBR/part-ii-dependency-paths-gene-disease-sorted.txt"  
 out = work.GeneDiseaseDependency  
 dbms = dlm  
 replace;  
 Getnames = No;  
 delimiter = '09'x;  
run;  
  
/\*Relabel Data from original coding in data sources\*/  
/\*Rename variables for Dependency Paths Part-ii to fit the documentation\*/  
data work.ChemicalDiseaseDependency;  
 set work.ChemicalDiseaseDependency;  
 rename var1=PubmedID var2=SentenceNumber var3=FirstEntityName var4=FirstEntityLocation var5=SecondEntityName var6=SecondEntityLocation var7=FirstEntityRaw   
 var8=SecondEntityRaw var9=FirstEntityDBID var10=SecondEntityDBID var11=FirstEntityType var12=SecondEntityType var13=DependencyPath var14=Sentence;  
run;  
  
data work.ChemicalGeneDependency;  
 set work.ChemicalGeneDependency;  
 rename var1=PubmedID var2=SentenceNumber var3=FirstEntityName var4=FirstEntityLocation var5=SecondEntityName var6=SecondEntityLocation var7=FirstEntityRaw   
 var8=SecondEntityRaw var9=FirstEntityDBID var10=SecondEntityDBID var11=FirstEntityType var12=SecondEntityType var13=DependencyPath var14=Sentence;  
run;  
  
data work.GeneGeneDependency;  
 set work.GeneGeneDependency;  
 rename var1=PubmedID var2=SentenceNumber var3=FirstEntityName var4=FirstEntityLocation var5=SecondEntityName var6=SecondEntityLocation var7=FirstEntityRaw   
 var8=SecondEntityRaw var9=FirstEntityDBID var10=SecondEntityDBID var11=FirstEntityType var12=SecondEntityType var13=DependencyPath var14=Sentence;  
run;  
  
data work.GeneDiseaseDependency;  
 set work.GeneDiseaseDependency;  
 rename var1=PubmedID var2=SentenceNumber var3=FirstEntityName var4=FirstEntityLocation var5=SecondEntityName var6=SecondEntityLocation var7=FirstEntityRaw   
 var8=SecondEntityRaw var9=FirstEntityDBID var10=SecondEntityDBID var11=FirstEntityType var12=SecondEntityType var13=DependencyPath var14=Sentence;  
run;  
  
/\*Rename variables for connections from Part-i data\*/  
data work.ChemicalDiseasePath;  
 set work.ChemicalDiseasePath;  
 rename var4=InhibitsCellGrowth var5=CellGrowth\_ind var12=PathogenesisRole var13=Pathogenesis\_ind var14=Biomarkers var15=Biomarkers\_ind var10=Alleviates   
 var11=Alleviates\_ind var1=DependencyPath var8=Prevents var9=Prevents\_ind var6=SideEffects var7=SideEffects\_ind var2=Treatment var3=Treatment\_ind;  
 If \_N\_ = 1 then delete;  
run;  
  
data work.ChemicalGenePath;  
 set work.ChemicalGenePath;  
 rename var1=DependencyPath var2=Agonism var3=Agonism\_ind var4=Antagonism var5=Antagonism\_ind var6=BindingLigand var7=BindingLigand\_ind  
 var8=IncreaseExpression var9=IncreaseExpression\_ind var10=DecreaseExpression var11=DecreaseExpression\_ind var12=AffectsExpression var13=AffectsExpression\_ind   
 var14=Inhibits var15=Inhibits\_ind var16=Transport var17=Transport\_ind var18=Metabolism var19=Metabolism\_ind var20=EnzymeActivity var21=EnzymeActivity\_ind;  
 If \_N\_ = 1 then delete;  
run;  
  
data work.GeneDiseasePath;  
 set work.GeneDiseasePath;  
 rename var1=DependencyPath var2=CasualMutations var3=CasualMutations\_ind var4=MutationAffectDisease var5=MutationAffectDisease\_ind var6=DrugTargets var7=DrugTargets\_ind   
 var8=PathogenesisRole var9=PathogenesisRole\_ind var10=TherapeuticEffect var11=TherapeuticEffect\_ind var12=Polymorphisms var13=Polymorphisms\_ind   
 var14=PromotesProgression var15=PromotesProgression\_ind var16=Biomarkers var17=Biomarkers\_ind var18=Overexpression var19=Overexpression\_ind var20=ImproperRegulation var21=ImproperRegulation\_ind;  
 If \_N\_ = 1 then delete;  
run;  
  
data work.GeneGenePath;  
 set work.GeneGenePath;  
 rename var1=DependencyPath var2=BindingLigand var3=BindingLigand\_ind var4=EnhancesResponse var5=EnhancesResponse\_ind var6=Activates var7=Activates\_ind   
 var8=IncreasesExpression var9=IncreasesExpression\_ind var10=AffectsExpression var11=AffectsExpression\_ind var12=SignalingPathway var13=SignalingPathway\_ind   
 var14=SameComplex var15=SameComplex\_ind var16=Regulation var17=Regulation\_ind var18=CellPopProduction var19=CellPopProduction\_ind;  
 If \_N\_ = 1 then delete;  
run;  
  
/\*Load up caslibs and connect to CAS\*/  
/\*Create CAS session and set mycas to active CASLib\*/  
cas casauto sessopts=(caslib='casuser');  
libname mycas cas readtransfersize=1000; /\*Used for just our race image to deal with memory issues\*/  
  
  
/\*caslib GNBR datasource=(srctype="path") path="/opt/sas/viya/config/data/cas/default/public/GNBR/" global;\*/  
  
proc casutil;  
 load data=work.ChemicalDiseasePath outcaslib="mycas"  
 casout="Chemical\_Disease\_Path" replace;  
run;  
proc casutil;  
 load data=work.ChemicalGenePath outcaslib="mycas"  
 casout="Chemical\_Gene\_Path" replace;  
run;  
proc casutil;  
 load data=work.GeneDiseasePath outcaslib="mycas"  
 casout="Gene\_Disease\_Path" replace;  
run;  
proc casutil;  
 load data=work.GeneGenePath outcaslib="mycas"  
 casout="Gene\_Gene\_Path" replace;  
run;  
proc casutil;  
 load data=work.GeneGeneDependency outcaslib="mycas"  
 casout="Gene\_Gene\_Dependency" replace;  
run;  
  
proc casutil;  
 load data=work.ChemicalGeneDependency outcaslib="mycas"  
 casout="Chemical\_Gene\_Dependency" replace;  
run;  
  
proc casutil;  
 load data=work.ChemicalDiseaseDependency outcaslib="mycas"  
 casout="Chemical\_Disease\_Dependency" replace;  
run;  
proc casutil;  
 load data=work.GeneDiseaseDependency outcaslib="mycas"  
 casout="Gene\_Disease\_Dependency" replace;  
run;  
  
/\*Clear Work Directory For Space\*/  
proc datasets library=work kill nolist;  
/\*DONE WITH THE IMPORT STEP\*/  
/\*Transfer Data from SAS compute engine to CAS, change dependency path variable, dropping the raw sentance, and making 0 in memory copies to save space\*/  
/\*Need to also convert all dependency paths to lowercase so we can preform the necessary joins\*/  
/\*This assumes there is a GNBR library already created\*/  
data mycas.GeneGeneDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneGeneDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath Sentence;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.GeneGenePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneGenePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.GeneDiseaseDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneDiseaseDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath Sentence;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.GeneDiseasePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.GeneDiseasePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.ChemicalDiseaseDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.ChemicalDiseaseDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.ChemicalDiseasePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.ChemicalDiseasePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.ChemicalGeneDependency;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
data mycas.ChemicalGenePath(replace=yes copies=0);  
 length Dependency\_Path varchar(200);  
 set GNBR.ChemicalGenePath;  
 Dependency\_Path = lowcase(DependencyPath);  
 drop DependencyPath;  
 rename Dependency\_Path = DependencyPath;  
run;  
  
/\*Once the datasets are in CAS with the standardized DependencyPath, its time to investigate the data and start parsing out the GeneIds from the Dependency datasets\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGeneDependency;  
 if find(SecondEntityDBID, 'Tax') ge 1 then CleanFlag = 1;  
 else CleanFlag = 0;  
run;  
/\*Next steps: Pull out anything after 'Tax' starts and create its own field of taxonomy with the full string, so we can still have information around species type,   
after this we can move onto the GNBR Integration to pull together GNBR datasets\*/  
  
/\*Extract Taxonomy ID from the SecondEntityDBID, once it's extracted we can remove the delimiters to just get the ID, also leaving blanks for DBIDs without Taxonomy\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGeneDependency;  
 if CleanFlag = 1 then   
 do;  
 Taxonomy=substr(SecondEntityDBID, index(SecondEntityDBID, ":"));  
 Taxonomy=compress(Taxonomy,":)");   
 end;  
 else Taxonomy=.;  
run;  
  
/\*The next step is to remove the taxonomy IDs from the original SecondEntityDBID so that we can use these numbers to join with DrugBank\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGeneDependency;  
 if CleanFlag = 1 then   
 do;  
 SecondEntityDBID= scan(SecondEntityDBID,1, '(');  
 end;  
 else SecondEntityDBID=SecondEntityDBID;  
run;  
  
/\*Parse out two IDs in field\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 set mycas.ChemicalGenedependency;  
 if Cleanflag = 0 then  
 do;  
 ID2=scan(SecondEntityDBID, 2,";");  
 ID3=scan(SecondEntityDBID, 3, ";");  
 end;  
 else if CleanFlag = 1 then  
 do;  
 ID2=.;  
 ID3=.;  
 end;  
 SecondEntityDBID=scan(SecondEntityDBID, 1, ";");   
run;  
/\*Now that we have the Taxonomy extracted and the DB IDs cleaned its time to convert data types to prepare for joining and drop unneeded variables\*/  
data mycas.ChemicalGeneDependency(replace=yes copies=0);  
 length SecondEntityDBID\_ 8;  
 length Taxonomy\_ 8;  
 length ID2\_ 8;  
 length ID3\_ 8;  
 set mycas.ChemicalGeneDependency;  
 SecondEntityDBID\_ = SecondEntityDBID;  
 Taxonomy\_ = Taxonomy;  
 ID2\_ = ID2;  
 ID3\_ = ID3;  
 drop SecondEntityDBID Taxonomy SecondDBID CleanFlag else ID2 ID3;  
 rename SecondEntityDBID\_ = SecondEntityDBID;  
 rename Taxonomy\_ = Taxonomy;  
 rename ID2\_ = ID2;  
 rename ID3\_ = ID3;  
run;  
  
/\*Make sure the run this code before doing any joins, this will bring your data from mycas to casuser so you can do the FedSQL joins\*/  
cas casauto sessopts=(Timeout=9700 LOCALE="en\_US" caslib=PUBLIC);  
caslib \_all\_ assign;  
  
/\*Join newly added GNBR CAS data together to create completed datasets of the Dendrograms Gene-Gene Gene-Disease Gene-Chemical Chemical-Disease\*/  
proc fedsql sessref=casauto;  
 create table casuser.GeneGeneComplete {options replace=true copies=0} as  
 select GeneGenePath.\*, GeneGeneDependency.FirstEntityName, GeneGeneDependency.FirstEntityDBID,   
 GeneGeneDependency.SecondEntityName, GeneGeneDependency.SecondEntityDBID  
 from casuser.GeneGenePath, casuser.GeneGeneDependency   
 where GeneGenePath.DependencyPath = GeneGeneDependency.DependencyPath;  
quit;  
  
/\*Join newly added GNBR CAS data together to create completed datasets of the Dendrograms Gene-Gene Gene-Disease Gene-Chemical Chemical-Disease\*/  
proc fedsql sessref=casauto;  
 create table casuser.GeneDiseaseComplete {options replace=true copies=0} as  
 select GeneDiseasePath.\*, GeneDiseaseDependency.FirstEntityName, GeneDiseaseDependency.FirstEntityDBID,   
 GeneDiseaseDependency.SecondEntityName, GeneDiseaseDependency.SecondEntityDBID  
 from casuser.GeneDiseasePath, casuser.GeneDiseaseDependency   
 where GeneDiseasePath.DependencyPath = GeneDiseaseDependency.DependencyPath;  
quit;  
  
/\*Join newly added GNBR CAS data together to create completed datasets of the Dendrograms Gene-Gene Gene-Disease Gene-Chemical Chemical-Disease\*/  
proc fedsql sessref=casauto;  
 create table casuser.ChemicalGeneComplete {options replace=true copies=0} as  
 select ChemicalGenePath.\*, ChemicalGeneDependency.FirstEntityName, ChemicalGeneDependency.FirstEntityDBID,   
 ChemicalGeneDependency.SecondEntityName, ChemicalGeneDependency.SecondEntityDBID, ChemicalGeneDependency.Taxonomy, ChemicalGeneDependency.Sentence  
 from casuser.ChemicalGenePath, casuser.ChemicalGeneDependency   
 where ChemicalGenePath.DependencyPath = ChemicalGeneDependency.DependencyPath;  
quit;  
  
/\*Join newly added GNBR CAS data together to create completed datasets of the Dendrograms Gene-Gene Gene-Disease Gene-Chemical Chemical-Disease\*/  
proc fedsql sessref=casauto;  
 create table casuser.ChemicalDiseaseComplete {options replace=true copies=0} as  
 select ChemicalDiseasePath.\*, ChemicalDiseaseDependency.FirstEntityName, ChemicalDiseaseDependency.FirstEntityDBID,   
 ChemicalDiseaseDependency.SecondEntityName, ChemicalDiseaseDependency.SecondEntityDBID, ChemicalDiseaseDependency.Sentence  
 from casuser.ChemicalDiseasePath, casuser.ChemicalDiseaseDependency   
 where ChemicalDiseasePath.DependencyPath = ChemicalDiseaseDependency.DependencyPath;  
quit;  
  
/\*Load in Uniprot CSV of GeneIDs\*/  
proc casutil;  
 load file="/opt/sas/viya/config/data/cas/default/public/FDA/UniProtGeneIDs.csv"   
 outcaslib="Casuser" casout="UniprotGeneID";  
run;  
  
/\*This first join we are using the first GeneID extracted, we could use the second GeneID since its a Gene-Gene relationship but that would be up to the team\*/  
proc fedsql sessref=casauto;  
 create table casuser.GeneGeneUniprot {options replace=true copies=0} as  
 select GeneGeneComplete.\*, UniprotGeneID.GeneID, UniprotGeneID.UniProtID  
 from casuser.GeneGeneComplete left outer join casuser.UniprotGeneID  
 on (GeneGeneComplete.SecondEntityDBID = UniprotGeneID.GeneID);  
quit;   
  
/\*We are joing this data on the first GeneID extracted, some instances there were multiple IDs but we are going off of the first one for now\*/  
proc fedsql sessref=casauto;  
 create table casuser.ChemicalGeneUniprot {options replace=true copies=0} as  
 select ChemicalGeneComplete.\*, UniprotGeneID.GeneID, UniprotGeneID.UniProtID   
 from casuser.ChemicalGeneComplete left outer join casuser.UniprotGeneID  
 on (ChemicalGeneComplete.SecondEntityDBID = UniprotGeneID.GeneID);  
quit;   
  
/\*The final join is a bit more simple, only one GeneID from this Dendrogram, so we will do a join on that\*/  
proc fedsql sessref=casauto;  
 create table casuser.GeneDiseaseUniprot {options replace=true copies=0} as  
 select GeneDiseaseComplete.\*, UniprotGeneID.GeneID, UniprotGeneID.UniProtID   
 from casuser.GeneDiseaseComplete left outer join casuser.UniprotGeneID  
 on (GeneDiseaseComplete.FirstEntityDBID = UniprotGeneID.GeneID);  
quit;   
  
/\*Now we can drop the unneeded tables in the casuser library to add space for new tables\*/  
proc casutil;  
 droptable casdata="ChemicalDiseaseDependency" incaslib="Casuser";  
 droptable casdata="ChemicalDiseasePath" incaslib="Casuser";  
 droptable casdata="ChemicalGeneDependency" incaslib="Casuser";  
 droptable casdata="ChemicalGenePath" incaslib="Casuser";  
 droptable casdata="GeneDiseaseDependency" incaslib="Casuser";  
 droptable casdata="GeneDiseasePath" incaslib="Casuser";  
 droptable casdata="GeneGeneDependency" incaslib="Casuser";  
 droptable casdata="GeneGenePath" incaslib="Casuser";  
run;  
   
/\*Use First Entity Location, take the last two numbers seperated by the comma and minus them (that gets the length of the word), next we find the first word that begins with  
the extracted entity name and parse out everything after that first letter with the length coming from the entity location information\*/  
data casuser.CDNormalizedEntityLength;  
 set casuser.Chemicaldiseasecomplete;  
 ChemicalLocation = index(Sentence, FirstEntityName);  
run;  
  
data casuser.CDNormalizedEntityLength;  
 set casuser.CDNormalizedEntityLength;  
 Chemical\_Name\_Full = substr(Sentence, ChemicalLocation, 70);  
 Chemical\_Name = scan(Chemical\_Name\_Full, 1, '');  
 drop ChemicalLocation;  
run;  
  
/\*Lets count the amount of times we were only able to pull the number or letter\*/  
data casuser.CDDrugChemicalExtracts;  
 set casuser.CDNormalizedEntityLength;  
 if length(Chemical\_Name) =< 1 then flag=1;  
 else flag=0;  
run;  
  
/\*Repeat the same process for Chemical - Gene Dendrogram\*/  
data casuser.CGNormalizedEntityLength;  
 set casuser.Chemicalgenecomplete;  
 ChemicalLocation = index(Sentence, FirstEntityName);  
run;  
  
data casuser.CGNormalizedEntityLength;  
 set casuser.CGNormalizedEntityLength;  
 Chemical\_Name\_Full = substr(Sentence, ChemicalLocation, 70);  
 Chemical\_Name = scan(Chemical\_Name\_Full, 1, '');  
 drop ChemicalLocation;  
run;  
  
/\*Lets count the amount of times we were only able to pull the number or letter\*/  
data casuser.CGDrugChemicalExtracts;  
 set casuser.CGNormalizedEntityLength;  
 if length(Chemical\_Name) =< 1 then flag=1;  
 else flag=0;  
run;  
  
proc casutil;  
 droptable casdata="CDNormalizedEntityLength" incaslib="Casuser";  
 droptable casdata="CGNormalizedEntityLength" incaslib="Casuser";  
run;  
  
/\*We can try several merges on the different DrugBank sets (MDS\_100Drugs\_V1, Truth, DBSmall\_Truth)\*/  
/\*DBSmall\_Truth to start for Chemical Gene\*/  
proc fedsql sessref=casauto;  
 create table casuser.DBSmallChemicalGene {options replace=true copies=0} as  
 select CGDrugChemicalExtracts.\*, DBSmallM\_Truth.\*  
 from casuser.CGDrugChemicalExtracts left outer join public.DBSmallM\_Truth  
 on (upcase(CGDrugChemicalExtracts.Chemical\_Name) = upcase(DBSmallM\_Truth.name));  
quit;  
  
data casuser.DBSmallChemicalGene;  
 set casuser.DBSmallChemicalGene;  
 where name is not missing;  
 drop Chemical\_Name\_Full name Sentence;  
run; /\*88661 Records\*/  
  
/\*Next on to try is merging on the truth dataset for Chemical Gene\*/  
proc fedsql sessref=casauto;  
 create table casuser.CGDrugBankTruth {options replace=true copies=0} as  
 select CGDrugChemicalExtracts.\*, Truth.\*  
 from casuser.CGDrugChemicalExtracts left outer join public.Truth  
 on (upcase(CGDrugChemicalExtracts.Chemical\_Name) = upcase(Truth.iName));  
quit;  
  
data casuser.CGDrugBankTruth;  
 set casuser.CGDrugBankTruth;  
 where iName is not missing;  
 drop Chemical\_Name\_Full iName Sentence;  
run;/\*20688\*/  
  
  
/\*Finally we will try the MDS\_100Drugs\_V1 dataset for Chemical Gene\*/  
proc fedsql sessref=casauto;  
 create table casuser.CGDrugBankMDS {options replace=true copies=0} as  
 select CGDrugChemicalExtracts.\*, MDS\_100Drugs\_V1.\*  
 from casuser.CGDrugChemicalExtracts left outer join public.MDS\_100Drugs\_V1  
 on (upcase(CGDrugChemicalExtracts.Chemical\_Name) = upcase(MDS\_100Drugs\_V1.name));  
quit;  
  
data casuser.CGDrugBankMDS;  
 set casuser.CGDrugBankMDS;  
 where name is not missing;  
 drop Chemical\_Name\_Full name Sentence;  
run;/\*11638\*/  
  
/\*DBSmall\_Truth\*/  
proc fedsql sessref=casauto;  
 create table casuser.DBSmallChemicalDisease {options replace=true copies=0} as  
 select CDDrugChemicalExtracts.\*, DBSmallM\_Truth.\*  
 from casuser.CDDrugChemicalExtracts left outer join public.DBSmallM\_Truth  
 on (upcase(CDDrugChemicalExtracts.Chemical\_Name) = upcase(DBSmallM\_Truth.name));  
quit;  
  
data casuser.DBSmallChemicalDisease;  
 set casuser.DBSmallChemicalDisease;  
 where name is not missing;  
 drop Chemical\_Name\_Full name Sentence;  
run;/\*62119\*/  
  
  
/\*Truth\*/  
proc fedsql sessref=casauto;  
 create table casuser.CDDrugBankTruth {options replace=true copies=0} as  
 select CDDrugChemicalExtracts.\*, Truth.\*  
 from casuser.CDDrugChemicalExtracts left outer join public.Truth  
 on (upcase(CDDrugChemicalExtracts.Chemical\_Name) = upcase(Truth.iName));  
quit;  
  
data casuser.CDDrugBankTruth;  
 set casuser.CDDrugBankTruth;  
 where iName is not missing;  
 drop Chemical\_Name\_Full iName Sentence;  
run;/\*20708\*/  
  
/\*MDS\_100Drugs\_V1\*/  
proc fedsql sessref=casauto;  
 create table casuser.CDDrugBankMDS {options replace=true copies=0} as  
 select CDDrugChemicalExtracts.Chemical\_Name, MDS\_100Drugs\_V1.\*  
 from casuser.CDDrugChemicalExtracts left outer join public.MDS\_100Drugs\_V1  
 on (upcase(CDDrugChemicalExtracts.Chemical\_Name) = upcase(MDS\_100Drugs\_V1.name));  
quit;  
  
data casuser.CDDrugBankMDS;  
 set casuser.CDDrugBankMDS;  
 where name is not missing;  
 drop Chemical\_Name\_Full name Sentence;  
run;/\*9183\*/  
  
/\*Finally Promote completely tables to be used by the team\*/  
/\*Only will drop tables after first run, this can be used if data is being updated\*/  
  
/\*proc casutil;  
 droptable casdata="DBSmallChemicalGene" incaslib="CASUSER";  
 droptable casdata="CGDrugBankTruth" incaslib="CASUSER";  
 droptable casdata="DBSmallChemicalDisease" incaslib="CASUSER";  
 droptable casdata="CDDrugBankTruth" incaslib="CASUSER";  
 droptable casdata="GeneGeneUniprot" incaslib="CASUSER";  
 droptable casdata="ChemicalGeneUniprot" incaslib="CASUSER";  
 droptable casdata="GeneDiseaseUniprot" incaslib="CASUSER";  
quit;\*/  
  
proc casutil outcaslib="public";  
 promote casdata="DBSmallChemicalGene" incaslib="CASUSER";  
 promote casdata="CGDrugBankTruth" incaslib="CASUSER";  
 promote casdata="DBSmallChemicalDisease" incaslib="CASUSER";  
 promote casdata="CDDrugBankTruth" incaslib="CASUSER";  
 promote casdata="GeneGeneUniprot" incaslib="CASUSER";  
 promote casdata="ChemicalGeneUniprot" incaslib="CASUSER";  
 promote casdata="GeneDiseaseUniprot" incaslib="CASUSER";  
quit;