Final Project

BioE 131/231 Fall 2019

Overview

Congratulations, all! You have made it to the final project. From now until the end of the semester, you will analyze a set of genomes using the skills you have learned thus far and explore new types of analysis. These genomes come from the Salmonella reference collection (https://people.ucalgary.ca/~kesander/Kit_14A.html (https://people.ucalgary.ca/~kesander/Kit_14A.html)), which were isolated from different sources many years ago, but have not been fully analyzed. We are interested in learning more about these strains and their pathogenicity.

The goal of this part of class is to let you work on a real, unpublished dataset, applying your own ideas of how best to conduct the analysis. Ideas for projects, deliverables, and due dates are listed below on a week-by-week basis.

Timeline

Week 1: Break into groups, obtain data, assemble genome, annotate genome.
Week 2: Propose your own analysis.
Week 3: No lab (Thanksgiving). Work on your own analysis.
Week 4: Group presentations during lab. Final report due.

Background

Genome sequencing and assembly are common techniques in biology. To obtain the sequence of a long genome, DNA must be chopped into small pieces that can be read by a sequencer. These short reads must then be stitched back together to form a complete genome. Often, the genome cannot be fully assembled because there are multiple equally plausible ways of stitching the reads together. Ideally, each chromosome is assembled into a single, long sequence. In practice, chromosomes are often assembled into multiple "contigs," or contiguous sequences. A genome assembly is generally considered complete only when all (or nearly all) the sequences are accounted for. Otherwise, it is considered a draft genome.

In a previous lab, you filtered human reads from a bacterial genome by alignment using Bowtie2. In this lab, you will continue your analysis, albeit with a different set of reads. First, you must take the reads and combine them into a complete genome. Next week, you will analyze the contents of your genome.

Week 1

First off, find a group of four people (max five per group). Please let your GSI know who is in your new group so they can keep track. Your GSI will assign you a set of reads for the final project. Each group will be given a different set of reads.

Next, please assemble your genome into contigs using SPAdes as we did in lab 8/9 using the -1 and -2 flags for paired end reads. Remember to run your assembly using -t 1 and -m 16 to limit your CPU and memory usage (save some for everyone else). If those limits are too low, let your GSI know. This will take a while, so be sure to run it in tmux.

When you're done, calculate assembly statistics and plot a histogram of contig lengths from your genome assembly in iPython, along with the N50. Summarize your results from SeqMatch (do we have the strain we expect?) and RAST/BASys annotations. (If annotation isn't ready in time, submit this in Week 2). Your report is due the following Wednesday at 11:59 AM.

Running SPAdes

Ran

```
ls /bigdata/FinalProject_data
ls /bigdata/FinalProject_data/190724_SARA_Genomes
```

to validate the file location and file names.

Prepare spades command and run from `/bigdata/FinalProject_groups/Group_5/assembly` :

```
spades.py -o . -1 /bigdata/FinalProject_data/190724_SARA_Genomes/SA
RA_5_S28_L004_R1_001.fastq.gz -2 /bigdata/FinalProject_data/190724_
SARA_Genomes/SARA_5_S28_L004_R2_001.fastq.gz -t 1
```

-o is destination directory (. for present directory)
-1 is location of reads 1
-2 is location of reads 2
-t 1 reserves only 1 core for the process

Run assembly-stats

Prepare Command, run from `bigdata/FinalProject_groups/Group_5/assembly` :

```
assembly-stats ./contigs.fasta ./scaffolds.fasta
```

Output:

```
stats for ./contigs.fasta
sum = 4960322, n = 153, ave = 32420.41, largest = 449208
N50 = 194186, n = 9
N60 = 130292, n = 13
N70 = 109439, n = 17
N80 = 87947, n = 22
N90 = 51460, n = 29
N100 = 56, n = 153
N_count = 0
Gaps = 0
-----------------------------------------------------------------
stats for ./scaffolds.fasta
sum = 4960972, n = 146, ave = 33979.26, largest = 449208
N50 = 223794, n = 8
N60 = 159283, n = 10
N70 = 143384, n = 13
N80 = 89581, n = 18
N90 = 56692, n = 24
N100 = 56, n = 146
N_count = 700
Gaps = 7
```

Plot Contig Coverage Histogram

Imports

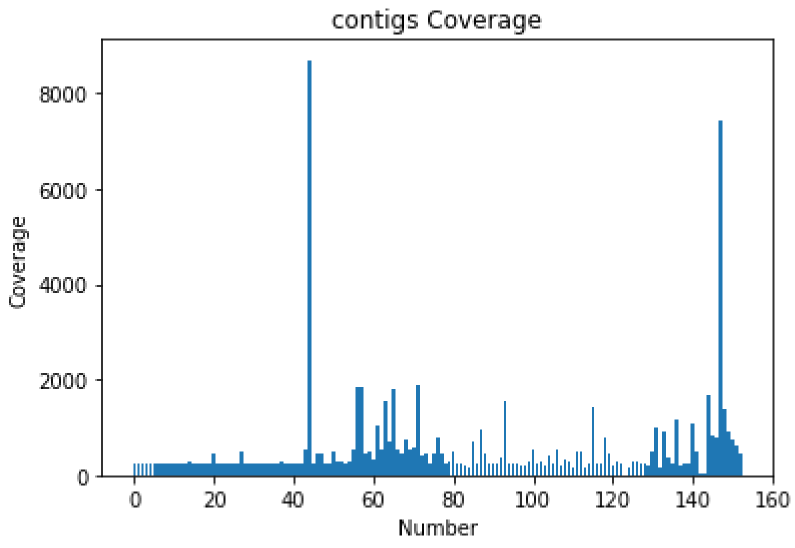
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from Bio import SeqIO
```

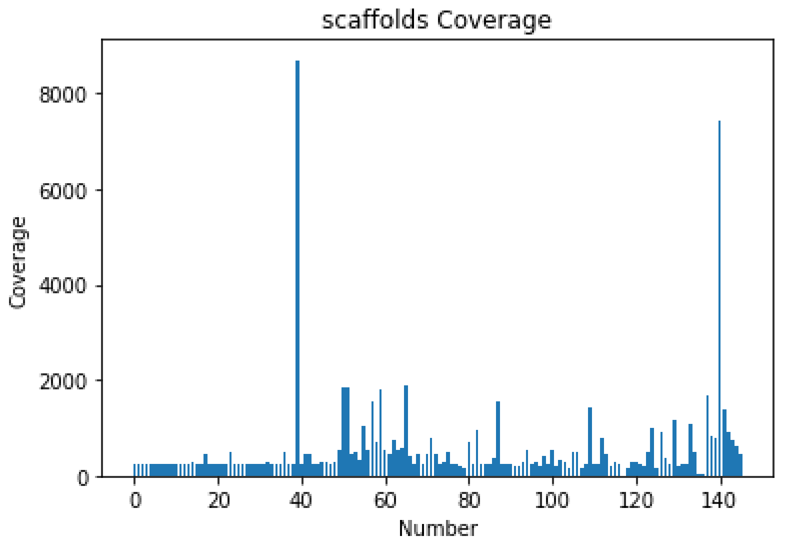
Extract Coverage Data

In [2]:
```
coverage_datas = []
directory = "/bigdata/FinalProject_groups/Group_5/assembly/"
file_names = ["contigs", "scaffolds"]
for file_name in file_names:
    coverage_data = []
    with open(directory + file_name + ".fasta", "r") as handle:
        for record in SeqIO.parse(handle, "fasta"):
            header = record.id
            index = int(header.find("_cov_") + len("_cov_"))
            coverage = float(header[index:])
            coverage_data.append(coverage)
    coverage_datas.append(coverage_data)
```

Plot Coverage Data Histogram

```
In [3]: for i in range(len(coverage_datas)):
            coverage_data = coverage_datas[i]
            plt.title(file_names[i] + " Coverage")
            plt.bar(range(len(coverage_data)), coverage_data)
            plt.xlabel("Number")
            plt.ylabel("Coverage")
            plt.show()
        #     plt.savefig(file_names[i] + "_coverage")
```

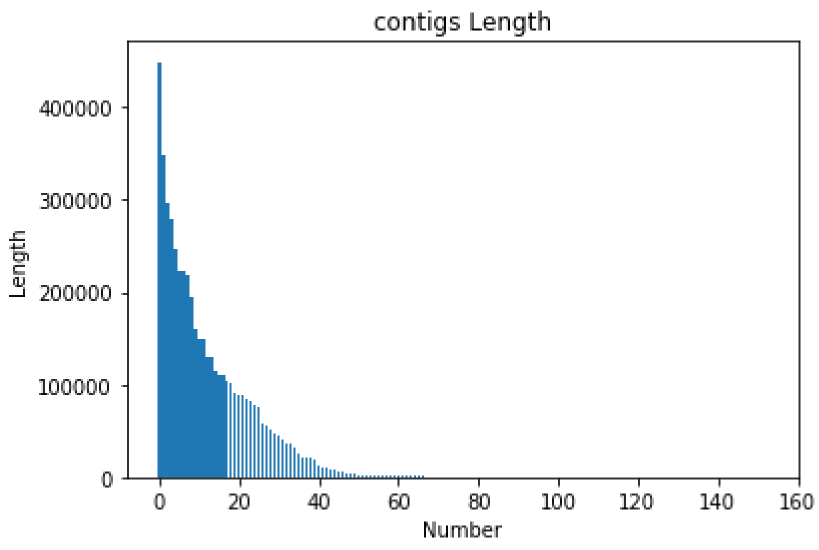


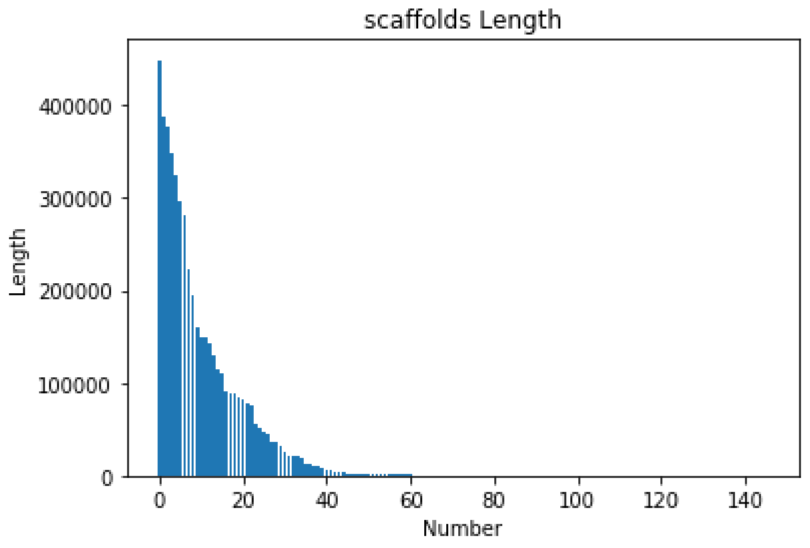


Plot Contig Length Histogram

In [4]:
```
length_datas = []
directory = "/bigdata/FinalProject_groups/Group_5/assembly/"
file_names = ["contigs", "scaffolds"]
for file_name in file_names:
    length_data = []
    with open(directory + file_name + ".fasta", "r") as handle:
        for record in SeqIO.parse(handle, "fasta"):
            header = record.id
            start_index = int(header.find("_length_") + len("_length_"))
            end_index = int(header.find("_cov_") + len("_cov_"))
            coverage = float(header[start_index:end_index-5])
            length_data.append(coverage)
    length_datas.append(length_data)
```

```
In [5]: for i in range(len(length_datas)):
            length_data = length_datas[i]
            plt.title(file_names[i] + " Length")
            plt.bar(range(len(length_data)), length_data)
            plt.xlabel("Number")
            plt.ylabel("Length")
            plt.show()
        #     plt.savefig(file_names[i] + "_length.png")
```





Identify the taxon from which your genome originated

We moved the HMM database to `/bigdata/FinalProject_groups/Group_5/rna_hmm3/HMM3` :

```
ls /bigdata/FinalProject_groups/Group_5/rna_hmm3/HMM3

arc_lsu.hmm  arc_ssu.hmm  arc_tsu.hmm  bac_lsu.hmm  bac_ssu.hmm  ba
c_tsu.hmm
```

run `rna_hmm.py` from `/bigdata/FinalProject_groups/Group_5`

```
rna_hmm3.py -i /bigdata/FinalProject_groups/Group_5/assembly/contig
s.fasta -o ./rna_hmm3_o -L /bigdata/FinalProject_groups/Group_5/rna
_hmm3/HMM3
```

output file: `rna_hmm3_o`

Made a copy and deleted all lines other than 16S_rRNA in text editor. Filename: `rna_hmm3_16`

Extract nucleic acid sequences from `/bigdata/FinalProject_groups/Group_5` Note: didn't have write-permission to assembly, so ran

```
cp assembly/contigs.fasta .
```

```
bedtools getfasta -fi ./contigs.fasta -bed ./rna_hmm3_16 -fo ./nucl
eic_acids
```

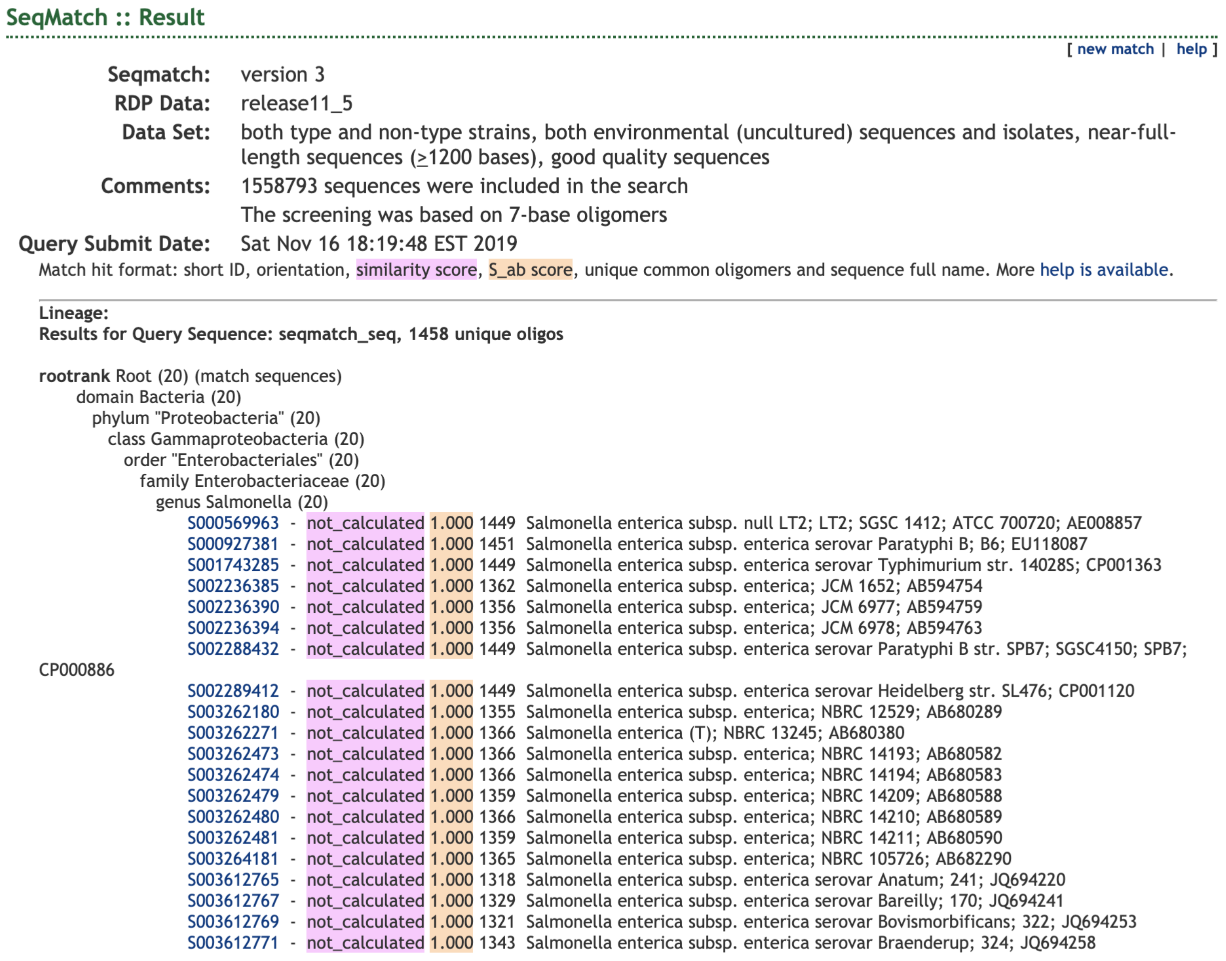
Output sequence in next cell.

Ran SeqMatch on `contigs.fasta`

```
domain Bacteria (20)
  phylum "Proteobacteria" (20)
    class Gammaproteobacteria (20)
      order Enterobacteriales (20)
        family Enterobacteriaceae (20)
          genus Salmonella (20)
```

```
>NODE_57_length_1736_cov_1821.819155:45-1598
AAGGTAAGGAGGTGATCCAACCGCAGGTTCCCCTACGGTTACCTTGTTACGACTTCACCCCAGTCAT
GAATCACAAAGTGGTAAGCGCCCTCCCGAAGGTTAAGCTACCTACTTCTTTTGCAACCCACTCCCAT
GGTGTGACGGGCGGTGTGTACAAGGCCCGGGAACGTATTCACCGTGGCATTCTGATCCACGATTACT
AGCGATTCCGACTTCATGGAGTCGAGTTGCAGACTCCAATCCGGACTACGACGCACTTTATGAGGTC
CGCTTGCTCTCGCGAGGTCGCTTCTCTTTGTATGCGCCATTGTAGCACGTGTGTAGCCCTGGTCGTA
AGGGCCATGATGACTTGACGTCATCCCCACCTTCCTCCAGTTTATCACTGGCAGTCTCCTTTGAGTT
CCCGACCTAATCGCTGGCAACAAAGGATAAGGGTTGCGCTCGTTGCGGGACTTAACCCAACATTTCA
CAACACGAGCTGACGACAGCCATGCAGCACCTGTCTCACAGTTCCCGAAGGCACCAATCCATCTCTG
GAAAGTTCTGTGGATGTCAAGACCAGGTAAGGTTCTTCGCGTTGCATCGAATTAAACCACATGCTCC
ACCGCTTGTGCGGGCCCCCGTCAATTCATTTGAGTTTTAACCTTGCGGCCGTACTCCCCAGGCGGTC
TACTTAACGCGTTAGCTCCGGAAGCCACGCCTCAAGGGCACAACCTCCAAGTAGACATCGTTTACGG
CGTGGACTACCAGGGTATCTAATCCTGTTTGCTCCCCACGCTTTCGCACCTGAGCGTCAGTCTTTGT
CCAGGGGGCCGCCTTCGCCACCGGTATTCCTCCAGATCTCTACGCATTTCACCGCTACACCTGGAAT
TCTACCCCCCTCTACAAGACTCAAGCCTGCCAGTTTCGAATGCAGTTCCCAGGTTGAGCCCGGGGAT
TTCACATCCGACTTGACAGACCGCCTGCGTGCGCTTTACGCCCAGTAATTCCGATTAACGCTTGCAC
CCTCCGTATTACCGCGGCTGCTGGCACGGAGTTAGCCGGTGCTTCTTCTGCGGGTAACGTCAATTGC
TGCGGTTATTAACCACAACACCTTCCTCCCCGCTGAAAGTACTTTACAACCCGAAGGCCTTCTTCAT
ACACGCGGCATGGCTGCATCAGGCTTGCGCCCATTGTGCAATATTCCCCACTGCTGCCTCCCGTAGG
AGTCTGGACCGTGTCTCAGTTCCAGTGTGGCTGGTCATCCTCTCAGACCAGCTAGGGATCGTCGCCT
TGGTGAGCCGTTACCTCACCAACAAGCTAATCCCATCTGGGCACATCTGATGGCAAGAGGCCCGAAG
GTCCCCCTCTTTGGTCTTGCGACGTTATGCGGTATTAGCCACCGTTTCCAGTAGTTATCCCCCTCCA
TCAGGCAGTTTCCCAGACATTACTCACCCGTCCGCCACTCGTCAGCGAAGCAGCAAGCTGCTTCCTG
```

```
TTACCGTTCGACTTGCATGTGTTAGGCCTGCCGCCAGCGTTCAATCTGAGCCATGATCAAACTCTTC
AATTTAAAAGTT
```



SeqMatch :: Result

[new match | help]

```
          Seqmatch:  version 3
          RDP Data:  release11_5
          Data Set:  both type and non-type strains, both environmental (uncultured) sequences and isolates, near-full-
                     length sequences (≥1200 bases), good quality sequences
          Comments:  1558793 sequences were included in the search
                     The screening was based on 7-base oligomers
Query Submit Date:   Sat Nov 16 18:19:48 EST 2019
Match hit format: short ID, orientation, similarity score, S_ab score, unique common oligomers and sequence full name. More help is available.

Lineage:
Results for Query Sequence: seqmatch_seq, 1458 unique oligos

rootrank Root (20) (match sequences)
    domain Bacteria (20)
      phylum "Proteobacteria" (20)
        class Gammaproteobacteria (20)
          order "Enterobacteriales" (20)
            family Enterobacteriaceae (20)
              genus Salmonella (20)
                S000569963  -  not_calculated 1.000 1449  Salmonella enterica subsp. null LT2; LT2; SGSC 1412; ATCC 700720; AE008857
                S000927381  -  not_calculated 1.000 1451  Salmonella enterica subsp. enterica serovar Paratyphi B; B6; EU118087
                S001743285  -  not_calculated 1.000 1449  Salmonella enterica subsp. enterica serovar Typhimurium str. 14028S; CP001363
                S002236385  -  not_calculated 1.000 1362  Salmonella enterica subsp. enterica; JCM 1652; AB594754
                S002236390  -  not_calculated 1.000 1356  Salmonella enterica subsp. enterica; JCM 6977; AB594759
                S002236394  -  not_calculated 1.000 1356  Salmonella enterica subsp. enterica; JCM 6978; AB594763
                S002288432  -  not_calculated 1.000 1449  Salmonella enterica subsp. enterica serovar Paratyphi B str. SPB7; SGSC4150; SPB7;
CP000886
                S002289412  -  not_calculated 1.000 1449  Salmonella enterica subsp. enterica serovar Heidelberg str. SL476; CP001120
                S003262180  -  not_calculated 1.000 1355  Salmonella enterica subsp. enterica; NBRC 12529; AB680289
                S003262271  -  not_calculated 1.000 1366  Salmonella enterica (T); NBRC 13245; AB680380
                S003262473  -  not_calculated 1.000 1366  Salmonella enterica subsp. enterica; NBRC 14193; AB680582
                S003262474  -  not_calculated 1.000 1366  Salmonella enterica subsp. enterica; NBRC 14194; AB680583
                S003262479  -  not_calculated 1.000 1359  Salmonella enterica subsp. enterica; NBRC 14209; AB680588
                S003262480  -  not_calculated 1.000 1366  Salmonella enterica subsp. enterica; NBRC 14210; AB680589
                S003262481  -  not_calculated 1.000 1359  Salmonella enterica subsp. enterica; NBRC 14211; AB680590
                S003264181  -  not_calculated 1.000 1365  Salmonella enterica subsp. enterica; NBRC 105726; AB682290
                S003612765  -  not_calculated 1.000 1318  Salmonella enterica subsp. enterica serovar Anatum; 241; JQ694220
                S003612767  -  not_calculated 1.000 1329  Salmonella enterica subsp. enterica serovar Bareilly; 170; JQ694241
                S003612769  -  not_calculated 1.000 1321  Salmonella enterica subsp. enterica serovar Bovismorbificans; 322; JQ694253
                S003612771  -  not_calculated 1.000 1343  Salmonella enterica subsp. enterica serovar Braenderup; 324; JQ694258
```

Genome annotation

RAST

```
Login: Qube5
Password: w5mAvPg4
```

Ran RAST annotation with contigs.fasta

Job Status: [http://rast.theseed.org/FIG/rast.cgi?page=JobDetails&job=797079 (http://rast.theseed.org/FIG/rast.cgi?page=JobDetails&job=797079)](http://rast.theseed.org/FIG/rast.cgi?page=JobDetails&job=797079)

Week 2

By now, you should have an assembly and annotation of your genome. While you are waiting for your annotations, you can start brainstorming ideas for your own analysis. Before you leave today, make sure your GSI approves of your project. You will have two weeks (including Thanksgiving) to work on it before your final presentation.

A list of project ideas will be uploaded to bCourses shortly if it hasn't been already.

Upload a report summarizing the findings of your annotations and a description of the project you intend to work on by the following Wednesday at 11:59 AM.

Imports

In [6]:
```
import pandas as pd
from collections import Counter
```

Extract Data

In [7]:
```
# Extract genbank data and put into dictionary for recovery later
directory = "/bigdata/FinalProject_groups/Group_5/Groups/"
groups = [
    1, 3, 4, 5, 6, 7,
    8, 9, 10, 11, 12,
    14, 15
]
genbank = {}
for group in groups:
    file_name = directory + "Group" + str(group) + "/Group" + str(group) +
    for seq_record in SeqIO.parse(file_name, "genbank"):
        organism = seq_record.annotations["organism"]
        if len(seq_record.features) > 1:
            for feature in seq_record.features[1:]:
                name = feature.qualifiers["db_xref"][0][5:]
                if feature.type != 'CDS':
                    continue
                protein_sequence = feature.qualifiers["translation"][0]
                protein_name = feature.qualifiers["product"][0]
                genbank[name] = [protein_sequence, protein_name, organism]
```

In [8]:

```
# Parse mp3 output data
columns = ["Group", "Sr._No.", "Sequence_Name", "Type_of_Pfam_domains",
           "HMM_Prediction", "SVM_Score",
           "SVM_prediction", "Hybrid_Prediction", "Assignment",
           "Sequence", "Product", "Organism"]
data = []
for group in groups:
    file_name = directory + "Group" + str(group) + "/Group" + str(group) +
    with open(file_name, "r") as handle:
        lines = handle.readlines()[1:]
        for line in lines:
            if len(line) == 1:
                continue
            l = line.split('\t')
            if len(l) == 8:
                for i in range(len(l)):
                    l[i] = l[i].strip()
                l[4] = float(l[4])
                l[0] = int(float(l[0]))
                # add in genbank data
                l.extend(genbank[l[1]])
                # add group number field
                l.insert(0, group)
                data.append(l)
```

Create Dataframe

We chose to use pandas dataframe for our database because it is easy to implement. In the future we could implement a SQL relational database which would help link our entries.

In [9]:

```
df = pd.DataFrame(data=data, columns=columns)
```

Save to pickle for faster loading without input files

In [10]:

```
df.to_pickle("./salmonella_database.pkl")
```

Load from Pickle

In [11]:

```
df = pd.read_pickle("./salmonella_database.pkl")
```

In [12]:
```python
display(df)
```

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_Sc
0	1	1	fig|6666666.498498.peg.1337	Uncalssified protein	----------	-0.0908
1	1	2	fig|6666666.498498.peg.1338	Excl. Non-pathogenic	Non-Pathogenic	0.4953
2	1	3	fig|6666666.498498.peg.1339	Excl. Non-pathogenic	Non-Pathogenic	0.0295
3	1	4	fig|6666666.498498.peg.1340	Uncalssified protein	----------	-0.6233
4	1	5	fig|6666666.498498.peg.1341	Uncalssified protein	----------	1.3877

Query Examples

In [13]:
```python
qHS = df.query('Assignment == "HS"')
display(qHS)
```

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_Sc
5	1	6	fig|6666666.498498.peg.1342	Excl. Non-pathogenic	Non-Pathogenic	-1.1394
6	1	7	fig|6666666.498498.peg.1343	Excl. Non-pathogenic	Non-Pathogenic	-1.1550
8	1	9	fig|6666666.498498.peg.1345	Excl. Non-pathogenic	Non-Pathogenic	-0.5731
11	1	12	fig|6666666.498498.peg.1348	Exclusive Pathogenic	Pathogenic	1.6284
12	1	13	fig|6666666.498498.peg.1349	Exclusive Pathogenic	Pathogenic	2.0774

In [14]:
```
qMult = df.query('HMM_Prediction == "Pathogenic" and SVM_prediction == "Pat
display(qMult)
```

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_Sc
11	1	12	fig|6666666.498498.peg.1348	Exclusive Pathogenic	Pathogenic	1.628
12	1	13	fig|6666666.498498.peg.1349	Exclusive Pathogenic	Pathogenic	2.077
13	1	14	fig|6666666.498498.peg.1350	Exclusive Pathogenic	Pathogenic	1.611
14	1	15	fig|6666666.498498.peg.1351	Exclusive Pathogenic	Pathogenic	1.228
15	1	16	fig|6666666.498498.peg.1352	Exclusive Pathogenic	Pathogenic	3.980

In [15]:
```
qComp = df.query('HMM_Prediction == "Pathogenic" and SVM_Score >= 2.5 and G
display(qComp)
```

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_Sc
50502	12	237	fig|6666666.498491.peg.1440	Exclusive Pathogenic	Pathogenic	3.870
50506	12	241	fig|6666666.498491.peg.1444	Exclusive Pathogenic	Pathogenic	2.551
50663	12	398	fig|6666666.498491.peg.1601	Exclusive Pathogenic	Pathogenic	3.147
50825	12	560	fig|6666666.498491.peg.1763	Exclusive Pathogenic	Pathogenic	3.914
51045	12	780	fig|6666666.498491.peg.2600	Exclusive Pathogenic	Pathogenic	2.787

In [16]:
```
qSeq = df.query('Sequence_Name == "fig|6666666.495479.peg.4413"')
display(qSeq)
```

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_S
20105	5	5049	fig|6666666.495479.peg.4413	Excl. Non-pathogenic	Non-Pathogenic	-0.39

In [17]:
```
dSubSeq = df.query('Sequence_Name.str.contains("6666666.498491")', engine='
display(dSubSeq)
```

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_Sc
50266	12	1	fig|6666666.498491.peg.1204	Uncalssified protein	----------	-0.1132
50267	12	2	fig|6666666.498491.peg.1205	Uncalssified protein	----------	2.6635
50268	12	3	fig|6666666.498491.peg.1206	Uncalssified protein	----------	0.2336
50269	12	4	fig|6666666.498491.peg.1207	Excl. Non-pathogenic	Non-Pathogenic	-1.2236
50270	12	5	fig|6666666.498491.peg.1208	Excl. Non-pathogenic	Non-Pathogenic	0.1807

In [18]:
```
dProt = df.query('Product.str.contains("Bactoprenol glucosyl transferase")'
display(dProt)
print(len(dProt))
```

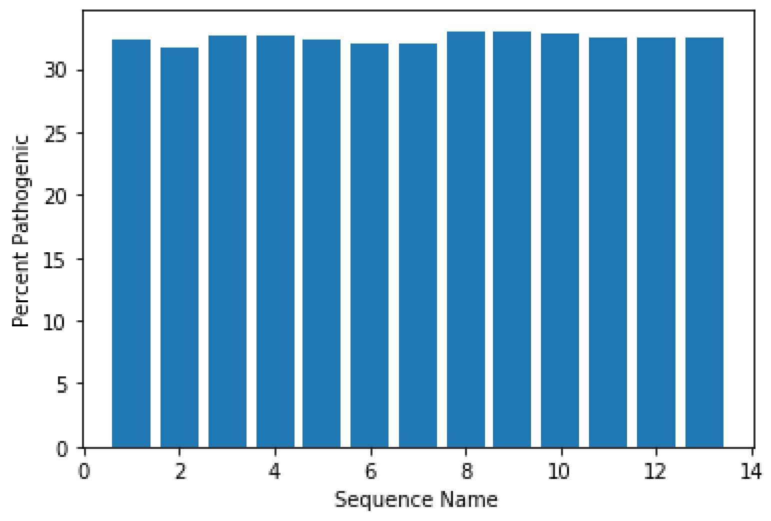
	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_Sc
244	1	245	fig|6666666.498498.peg.1581	Excl. Non-pathogenic	Non-Pathogenic	-0.8654
2671	1	2672	fig|6666666.498498.peg.241	Excl. Non-pathogenic	Non-Pathogenic	-0.4135
5265	3	197	fig|6666666.498495.peg.1459	Excl. Non-pathogenic	Non-Pathogenic	-0.8654
7773	3	2705	fig|6666666.498495.peg.417	Excl. Non-pathogenic	Non-Pathogenic	-0.4135
9890	3	4822	fig|6666666.498495.peg.4286	Excl. Non-pathogenic	Non-Pathogenic	-0.3968

Graph Percent of Pathogenic genes in each Group

```
In [19]: percent_pathogenic = {}
         sequence_name = {}
         for group in df.Group.unique():
             group_proteins = df.query(f'Group == {group}')
             sequence_name = group_proteins.Sequence_Name
             pathogenic_proteins = group_proteins.query('Hybrid_Prediction == "Pathc
             percent_pathogenic[group] = len(pathogenic_proteins) / len(group_protei
```

```
In [20]: fig, ax = plt.subplots()
         ax.bar(np.arange(1, len(percent_pathogenic) + 1), np.fromiter(percent_pathc
         ax.set_xlabel('Sequence Name')
         ax.set_ylabel('Percent Pathogenic')
```

```
Out[20]: Text(0, 0.5, 'Percent Pathogenic')
```



Analyze percentage of pathogenic genes across all groups

```
In [21]: vals = np.array(list(percent_pathogenic.values()))
         print("Minimum", np.min(vals), "Group:", np.argmin(vals))
         print("Maximum", np.max(vals), "Group:", np.argmax(vals))
         print("Average", np.average(vals))
```

```
Minimum 0.3165156507413509 Group: 1
Maximum 0.3301568393885249 Group: 7
Average 0.3246301530546951
```

Analyze number of occurrences of each sequence across all groups

```
In [22]: df.Sequence.count()
```

```
Out[22]: 65173
```

In [23]:
```
num_strains = list(df.groupby('Sequence').count().Organism)
num_strains.sort()
counts = dict(Counter(num_strains))
counts
```

Out[23]:
```
{1: 3607,
 2: 1288,
 3: 171,
 4: 122,
 5: 202,
 6: 82,
 7: 95,
 8: 141,
 9: 157,
 10: 377,
 11: 676,
 12: 735,
 13: 2551,
 14: 1,
 18: 1,
 19: 2,
 22: 1}
```

In [24]:
```
# The most common sequence
df.groupby('Sequence').count()[['Organism']].query('Organism == 22', engine
```

Out[24]:

	Organism
Sequence	
MSKEKFERTKPHVNVGTIGHVDH	22

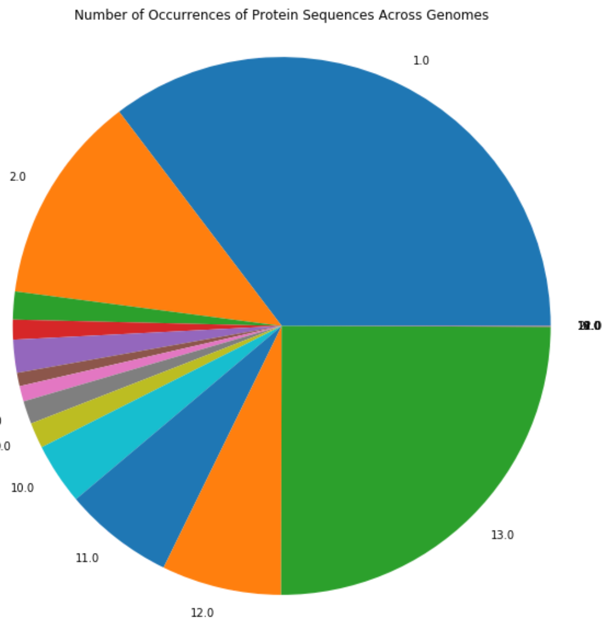
In [25]:
```
df.query("Sequence == 'MSKEKFERTKPHVNVGTIGHVDH'")
```

Out[25]:

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_S
4700	1	4701	fig|6666666.498498.peg.3059	Uncalssified protein	----------	-2.48
4949	1	4950	fig|6666666.498498.peg.3584	Uncalssified protein	----------	-2.48
9571	3	4503	fig|6666666.498495.peg.2987	Uncalssified protein	----------	-2.48
9840	3	4772	fig|6666666.498495.peg.3557	Uncalssified protein	----------	-2.48
14639	4	4715	fig|6666666.498497.peg.2975	Uncalssified protein	----------	-2.48
14959	4	5035	fig|6666666.498497.peg.3826	Uncalssified protein	----------	-2.48
19774	5	4718	fig|6666666.495479.peg.3073	Uncalssified protein	----------	-2.48
20052	5	4996	fig|6666666.495479.peg.3651	Uncalssified protein	----------	-2.48
24836	6	4701	fig|6666666.498498.peg.3059	Uncalssified protein	----------	-2.48
25085	6	4950	fig|6666666.498498.peg.3584	Uncalssified protein	----------	-2.48
29653	7	4449	fig|6666666.498492.peg.2844	Uncalssified protein	----------	-2.48
30092	7	4888	fig|6666666.498492.peg.3860	Uncalssified protein	----------	-2.48
34757	8	4586	fig|6666666.498494.peg.2349	Uncalssified protein	----------	-2.48
34939	8	4768	fig|6666666.498494.peg.2945	Uncalssified protein	----------	-2.48
39509	9	4534	fig|6666666.498496.peg.3103	Uncalssified protein	----------	-2.48
39928	9	4953	fig|6666666.498496.peg.4037	Uncalssified protein	----------	-2.48

	Group	Sr._No.	Sequence_Name	Type_of_Pfam_domains	HMM_Prediction	SVM_S
44860	10	4848	fig|6666666.498501.peg.3250	Uncalssified protein	----------	-2.48
45116	10	5104	fig|6666666.498501.peg.4061	Uncalssified protein	----------	-2.48
49846	11	4670	fig|6666666.498502.peg.3265	Uncalssified protein	----------	-2.48
50131	11	4955	fig|6666666.498502.peg.4105	Uncalssified protein	----------	-2.48
64827	15	4596	fig|6666666.498500.peg.3001	Uncalssified protein	----------	-2.48
65087	15	4856	fig|6666666.498500.peg.3535	Uncalssified protein	----------	-2.48

In [26]:
```
fig, ax = plt.subplots(figsize=(20, 10))
explode = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1)
plt.pie([float(v) for v in counts.values()], labels=[float(k) for k in coun
           autopct=None)
ax.axis('equal')
plt.title("Number of Occurrences of Protein Sequences Across Genomes")
plt.show()
```



In [27]:
```
print("Unique:", 3607 / sum(counts.values()))
print("Shared:", 1 - 3607 / sum(counts.values()))
```

```
Unique: 0.3533157018317171
Shared: 0.6466842981682829
```

Analyze number of occurrences of each pathongenic sequence across all groups

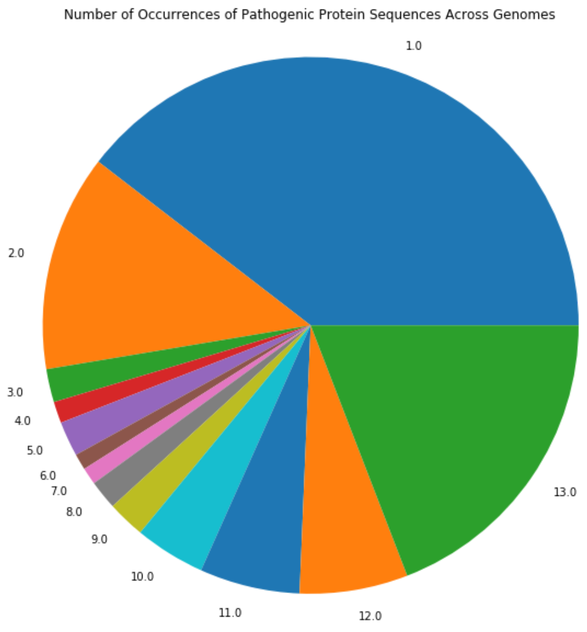
```
In [28]: pathogenic = df.query("Hybrid_Prediction == 'Pathogenic'", engine='python')
         pathogenic.Sequence.count()
```

```
Out[28]: 21161
```

```
In [29]: num_strains = list(pathogenic.groupby('Sequence').count().Organism)
         num_strains.sort()
         counts = dict(Counter(num_strains))
         counts
```

```
Out[29]: {1: 1466,
          2: 484,
          3: 74,
          4: 48,
          5: 78,
          6: 36,
          7: 38,
          8: 65,
          9: 84,
          10: 158,
          11: 224,
          12: 242,
          13: 709}
```

```
In [30]: fig, ax = plt.subplots(figsize=(20, 10))
         explode = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1)
         plt.pie([float(v) for v in counts.values()], labels=[float(k) for k in coun
                     autopct=None)
         ax.axis('equal')
         plt.title("Number of Occurrences of Pathogenic Protein Sequences Across Gen
         plt.show()
```



```
In [31]: print("Unique:", 1466 / sum(counts.values()))
         print("Shared:", 1 - 1466 / sum(counts.values()))
         print("13 Occurrences:", 709 / sum(counts.values()))
```

```
Unique: 0.39557474365893147
Shared: 0.6044252563410686
13 Occurrences: 0.19131138694009714
```

Presentations

Prepare a ten minute PowerPoint presentation describing all of the results of your genome assembly and analysis. Everyone in your group should speak during the presentation. You will have 5 minutes for questions at the end. Summarize the results of your assembly (e.g., N50, contig length histogram).

Summarize the results of your annotations.
Which analysis project did you choose?
What were some of the issues you ran into?
What were your results?
If you had more time, what additional experiments and analyses would you perform?

https://docs.google.com/presentation/d/1RbJZauYtNAMGt-8NderWXZFYrPkHX5TcpzVry1Rf-8M/edit#slide=id.g78e2f7005c_0_139 (https://docs.google.com/presentation/d/1RbJZauYtNAMGt-8NderWXZFYrPkHX5TcpzVry1Rf-8M/edit#slide=id.g78e2f7005c_0_139)

Written report

Submit a report (up to 5 pages, not included figures) along with your final presentation. This report should summarize your assembly results, your annotations, the methods that you used for your original analysis, and the findings of your analysis. Please be clear about the question you are trying to answer, how your chosen method will help you answer it, and any potential limitations of your results

https://docs.google.com/document/d/14i9ybUDlcupi-CD59QiB8MZmO2S0zLC6iamq70-43xA/edit?usp=sharing (https://docs.google.com/document/d/14i9ybUDlcupi-CD59QiB8MZmO2S0zLC6iamq70-43xA/edit?usp=sharing)