BT1501 计算生物学 2024-2025-2 Project 1.

冠状病毒的主蛋白酶序列和结构分析 冠状病毒是人类健康的一大威胁。冠装病毒完成复制周期需要主蛋白酶（main protease）来切割连在一起的多蛋白（polyproteins）。

1）请使用 NCBI 数据库找到所有基因组已经测序的冠状病毒科的病毒（Taxon: 11118）。请提交 Excel 文件 1, 给出病毒的 Name 和 Taxon ID。

1. 通过https://www.ncbi.nlm.nih.gov/datasets/genome/?taxon=11118访问所有已测序的冠状病毒科的病毒，下载到本地。
2. 用''将上一步文件转为xlsx format, 直接在excel中删除重复项目。文件包含 `Assembly Accession` `Assembly Name`、`Organism Name` 和 `Taxon ID` 的Excel 文件 1。共**487**个。

2）对每种已测序的冠状病毒，获取它们的主蛋白酶的蛋白序列。请提交 Excel 文件 2，给出每种病毒主蛋白酶的序列（fasta 格式，header 请注明蛋白的 accession （如果有的话），病毒 Name 和 Taxon ID）。也可以直接提交 fasta 序列（所有 fasta 放入一个文件）。因为一些测序的冠状病毒未必做了翻译和注释，所以可能需要自己从哪些未做注释的物种中把主蛋白酶找出来（可以使用tBLASTn， 或者 nhmmer）。

1. **通过tBLASTn用R1AB_SARS2的nsp5蛋白比对得到上述487蛋白相似的核酸序列，接着通过python脚本批量翻译这些序列**

   将上一步487个病毒的accession号提取出来，通过accessionID提取所有基因组：

   `datasets download genome accession --inputfile accessions.txt  --filename genomes.zip`

   将所有基因组（fna格式）放到一个文件夹下

   `(BT1051) liweihang@Mac-3 ncbi_dataset % mkdir -p ~/BT1051/Project/genomes/all_fna`

   `(BT1051) liweihang@Mac-3 ncbi_dataset % mv ~/BT1051/Project/genomes/ncbi_dataset/data/*/*/*.fna ~/BT1051/Project/genomes/all_fna/`

   将这些.fna格式的文件合并为一个文件

   `cat *.fna > total.fna`

   建立blast本地数据库：

   ```
   makeblastdb -in /Users/liweihang/BT1051/Project/total.fna \
               -dbtype nucl \
               -out /Users/liweihang/BT1051/Project/Virus/Virusgenome
   ```

   将已知的nsp5蛋白序列和数据库进行序列比对：

   ```
   tblastn -query /Users/liweihang/BT1051/Project/Virus/nsp5.fna \
           -db /Users/liweihang/BT1051/Project/Virus/Virusgenome \
           -out /Users/liweihang/BT1051/Project/Virus/tblastn_results.txt \
           -outfmt 6
   ```

   提取比对后的第2，9，10列，第2列为AccessionID，第9，10列为序列的起点和终点，再通过script提取序列，翻译

   `awk '{print $2, $9, $10}' /Users/liweihang/BT1051/Project/Virus/tblastn_results.txt > regions.txt`

```python
from Bio import SeqIO

# 输入文件
fasta_file = "/Users/liweihang/BT1051/Project/Virus/total.fna"
regions_file = "/Users/liweihang/BT1051/Project/Virus/regions.txt"
output_file = "/Users/liweihang/BT1051/Project/Virus/extracted_sequences.fasta"

# 读取 regions.txt
with open(regions_file, "r") as f:
    regions = [line.strip().split() for line in f]

# 提取序列
with open(output_file, "w") as out:
    for record in SeqIO.parse(fasta_file, "fasta"):
        for region in regions:
            accession_id, start, end = region
            if record.id == accession_id or record.id.startswith(accession_id + "."):
                start = int(start) - 1  # 转换为 0-based 索引
                end = int(end)
                subseq = record.seq[start:end]
                out.write(f">{record.id}_{start+1}-{end}\n{subseq}\n")
print('OK!')
```

```python
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord

# 输入文件
input_file = "/Users/liweihang/BT1051/Project/Virus/extracted_sequences.fasta"
output_file = "/Users/liweihang/BT1051/Project/Virus/translated_proteins.fasta"

# 翻译核酸序列为蛋白质序列
translated_records = []
for record in SeqIO.parse(input_file, "fasta"):
    # 翻译序列
    protein_seq = record.seq.translate()
    # 创建新的 SeqRecord 对象
    protein_record = SeqRecord(protein_seq, id=record.id, description="")
    translated_records.append(protein_record)

# 保存翻译后的蛋白质序列
SeqIO.write(translated_records, output_file, "fasta")
```

2.之后找到Assembly Accession和GenBank nucleotide accessions对应关系，把序列并到第一问的excel，生成Excel2即可

GCA_031270255.1	ASM3127025v1	Wenzhou Suncus murinus alphacoronavirus 1	2877474	MZ328303.1	AGLKKILQPTGVVEHCVVKVSYGGLTLNGLWLGNNVYCPRHVIADD
GCA_031174035.1	ASM3117403v1	Quail deltacoronavirus	2249315	MH532440.1	AGIKILLHPSGVVEQCVVSVVYNGSALNGIWLNNIVYCPRHVIGKYR
GCA_031171235.1	ASM3117123v1	Quail coronavirus UAE-HKU30	2078581	LC364345.1	AGIKILLHPSGVVERCIVSVVYNGSALNGIWLNNVVYCPRHVIGKYR
GCA_031173895.1	ASM3117389v1	Sparrow deltacoronavirus	2219863	MG812376.1	AGIKILLHPSGVVERCIVSVVYNGSALNGIWLKNVVYCPRHVIGKYR
GCA_002889935.1	ASM288993v1	unidentified human coronavirus	694448	N/A	N/A
GCA_023124525.1	ASM2312452v1	Pacific salmon nidovirus	2587487	MK611985.1	SALRKFATPSGDIEKHICMVSTAETMLTGLIYERTLYCPRHIVGSHI
GCA_031118375.1	ASM3111837v1	SARS coronavirus Sin3408	267385	AY559083.1	SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDTVYCPRHVICT
GCA_031120055.1	ASM3112005v1	SARS coronavirus GZ0401	281976	AY568539.1	SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDTVYCPRHVICT
GCA_031118515.1	ASM3111851v1	SARS coronavirus Sin3408L	267399	AY559097.1	SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDTVYCPRHVICT
GCA_031167765.1	ASM3116776v1	Porcine enteric alphacoronavirus	2018513	MH697599.1	AGLKKMAQPSGLVEPCVVRVSYGNTVLNGVWLDDKVYCPRHVLA

Note：结果如上，有一个ID未能找到相似序列。

3）对所有找到的主蛋白酶进行聚类处理。可以使用cdhit，按85% seq id 来聚类， 总共可以分多少类？请提交聚类结果文件 3。

Command：

```
cd-hit -i proteins.fasta -o clustered_proteins -c 0.85 -n 5 -M 16000 -T 8
```

参数说明：

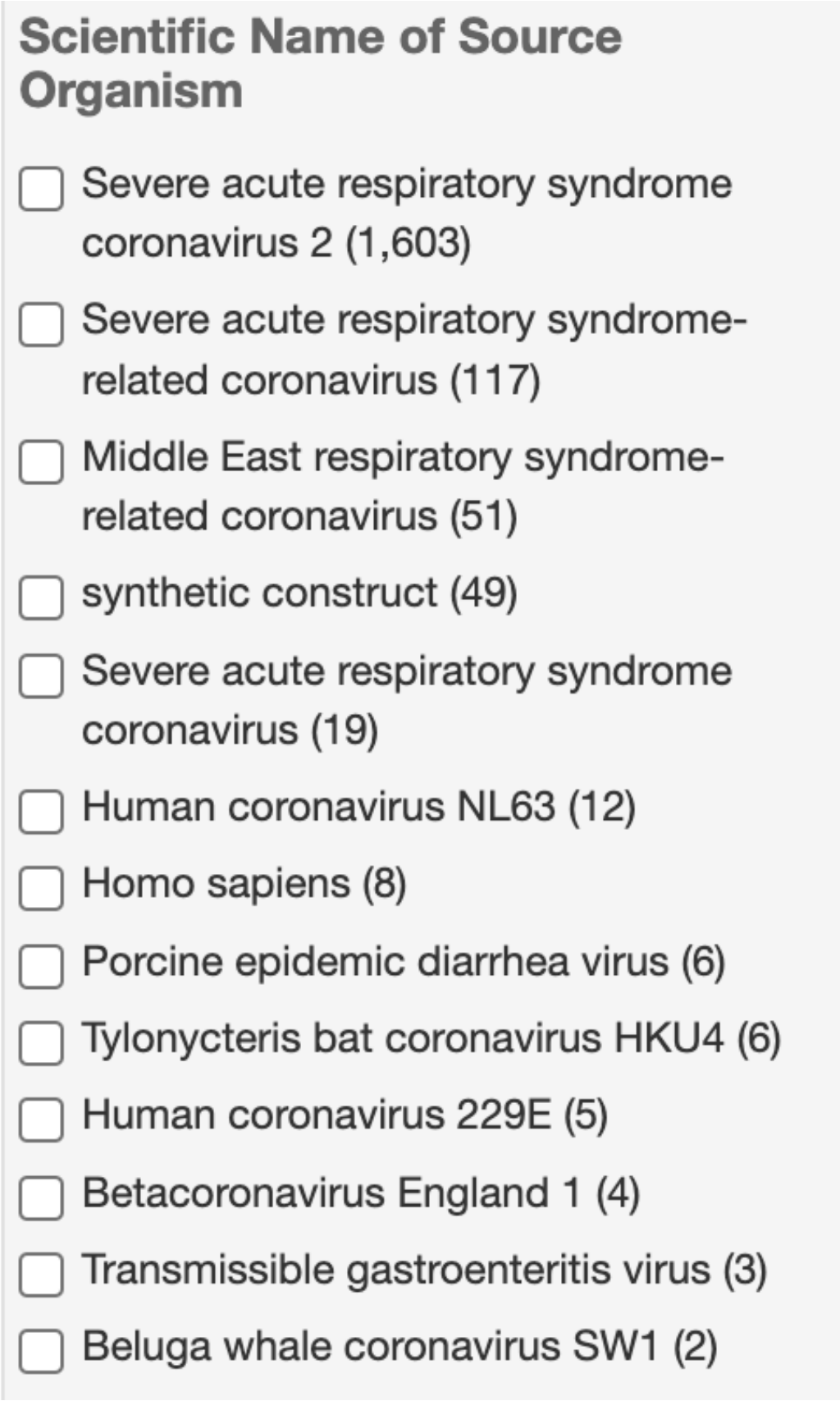
- `-i proteins.fasta`：输入文件。
- `-o clustered_proteins`：输出文件前缀。
- `-c 0.85`：设置85%的序列相似性阈值。
- `-n 5`：使用5个字母的单词长度。
- `-M 16000`：内存限制为16000MB。
- `-T 8`：使用8个CPU线程。（源于deepseek）

生成文件`clustered_proteins.clstr`，可以点进去看到病毒被分成了**57**类。

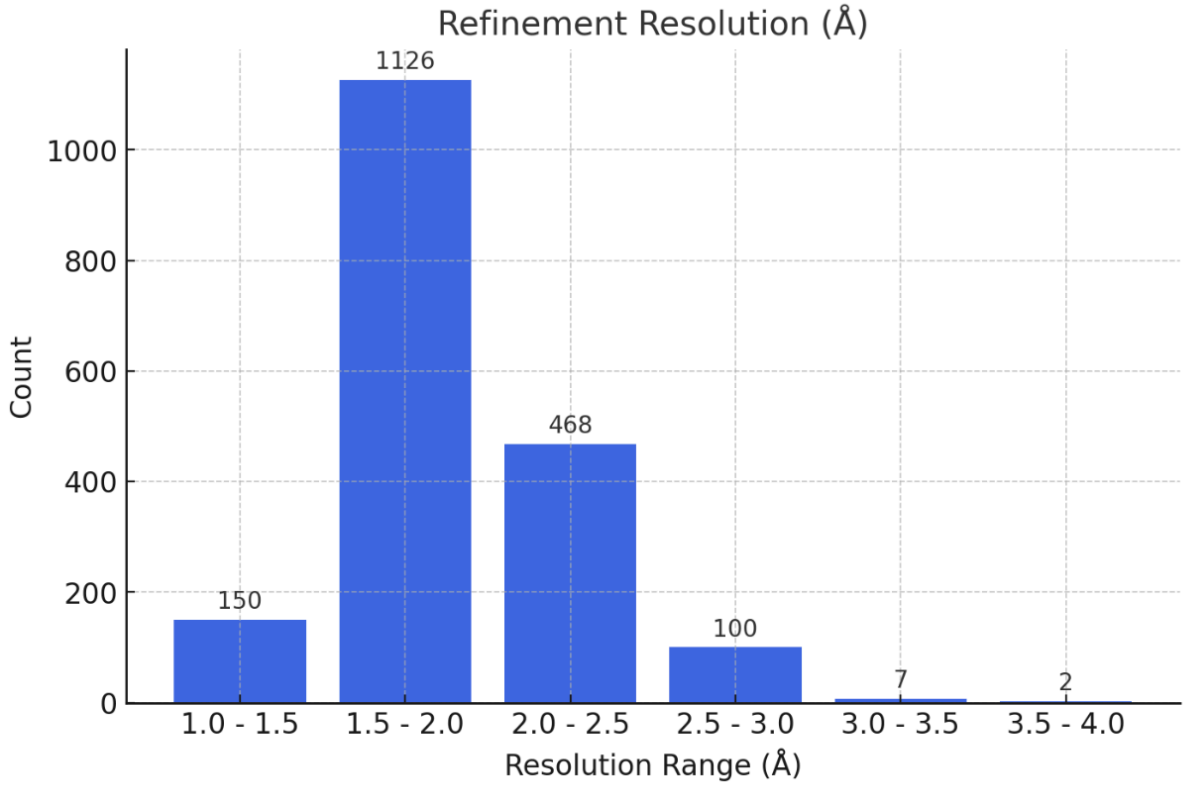
4）这些主蛋白酶中，哪些物种的主蛋白酶有结构？总共有多少个结构？这些结 构的分辨率（resolution）和 R-free 的分布是怎样的？它们是由多少个课题组 解析的？解析主蛋白酶数量位居前五的课题组分别是哪些？有多少个结构是 主蛋白酶和药物分子结合的复合物（注意排除溶剂分子）？ 请提交文件 4，内容是主蛋白酶 accession+病毒 Name+Taxon ID + PDB IDs。

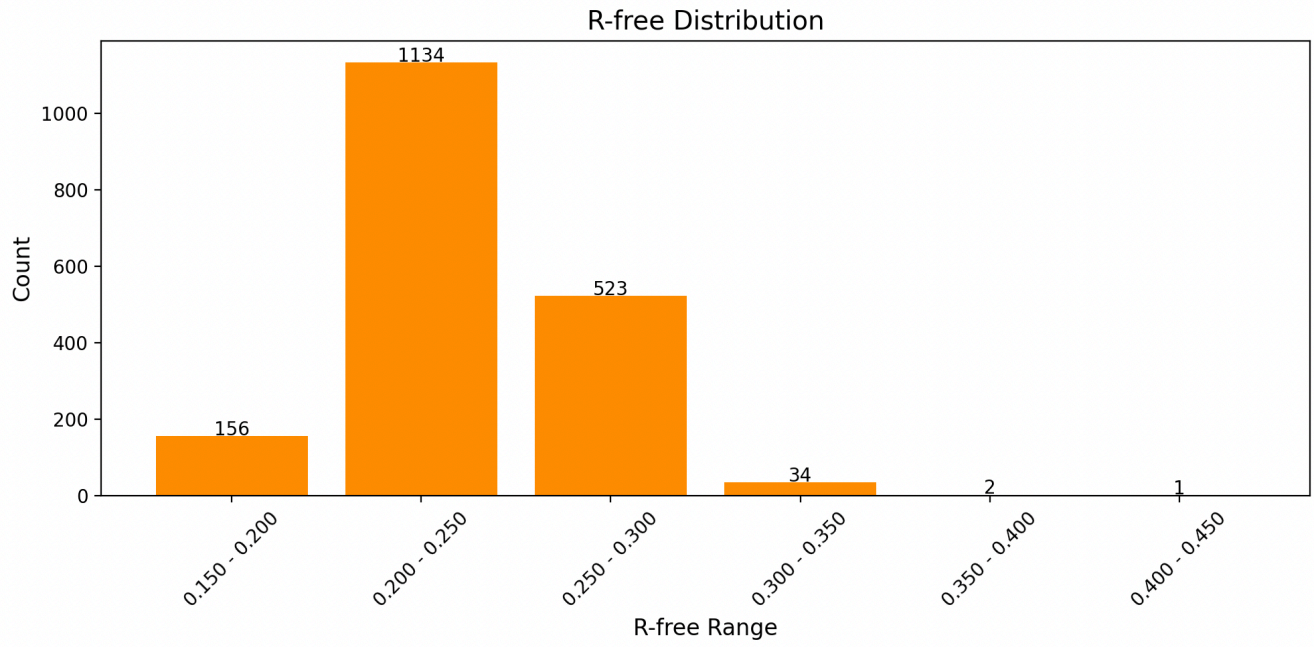
1. 访问PDB，在`Advanced Search Query Builder`的 `Sequence Similarity`中输入选用的比对序列（UniprotID：P0C6Y1）`E-value Cutoff=0.1`一共得到1852个结果`Tabular Report`选择想要的信息（Structure Author, Taxonomy ID, Ligand ID...）

   可以看到总共有**1,852** Structures，物种如下（并不完全展示，可以在PDB页面看到更完整的物种统计）



2. 以JSON格式下载下图这些信息，对resolution和R-free做数据可视化处理。





3. 统计**Structure Author**的最后一位（一般为通讯作者），以此来代表课题组，并统计筛选出解析主蛋白酶数量位居前五的课题组。

```
import json
from collections import Counter

# 读取JSON文件
file_path = "rcsb_pdb_custom_report_20250307044526.json"
with open(file_path, "r") as f:
    data = json.load(f)

# 提取每个 identifier 下的最后一个 audit_author
```

```
last_authors = []

for entry in data:
    authors = entry["data"].get("audit_author", [])
    if authors:
        last_authors.append(authors[-1]["name"])

# 统计出现次数最多的前五个作者
author_counts = Counter(last_authors)
top_5_authors = author_counts.most_common(5)

# 统计总共有多少个不同的最后一个作者
unique_last_authors_count = len(set(last_authors))

# 输出结果
print("Top 5 most frequent last authors:")
for author, count in top_5_authors:
    print(f"{author}: {count}")

print(f"Total number of unique last authors: {unique_last_authors_count}")
```

Top 5 most frequent last authors:
von Delft, F.: 591
Li, J.: 71
Groutas, W.C.: 60
Hilgenfeld, R.: 59
Liu, W.R.: 48
Total number of unique last authors: 199

4. 访问https://yanglab.qd.sdu.edu.cn/Q-BioLiP/Download/ ， 下载“The list of small molecules that are commonly used in protein structure determination can be downloaded”为默认溶剂分子，以这个列表来去重、统计主蛋白酶和药物分子结合的复合物结构。

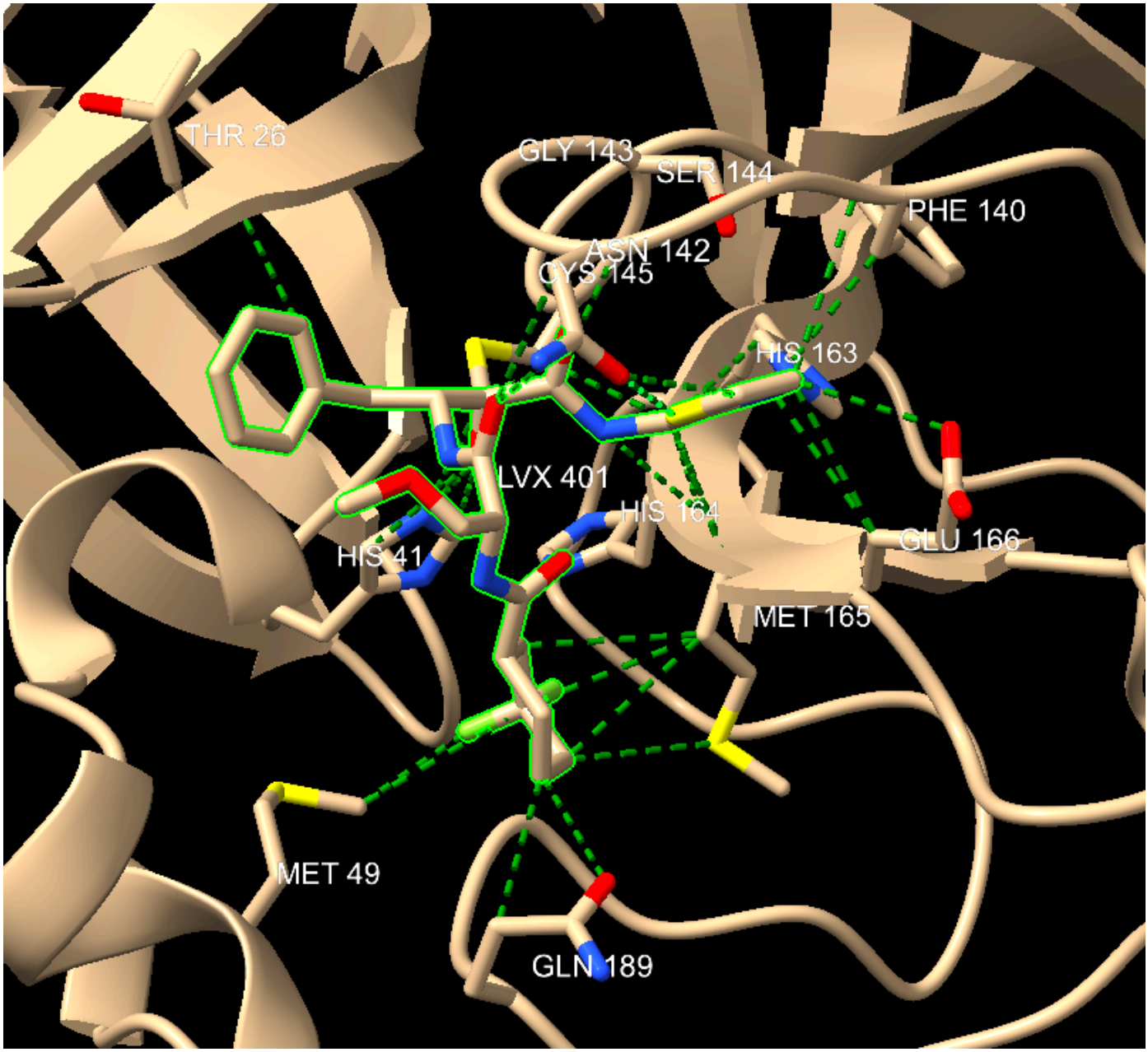
```
import json
from collections import Counter

# 读取JSON文件
pdb_file_path = "rcsb_pdb_custom_report_20250307044526.json"
ligand_file_path = "/Users/liweihang/BT1051/Project/solutionligand.txt"

# 读取ligand列表
with open(ligand_file_path, "r") as f:
    known_ligands = set(line.strip() for line in f)

# 读取PDB JSON数据
with open(pdb_file_path, "r") as f:
    data = json.load(f)

# 统计复合物结构数量
```

```
complex_count = 0

def is_complex_structure(nonpolymer_entities):
    """判断结构是否包含非txt文件中的chem_comp"""
    if not nonpolymer_entities:  # 处理None情况
        return False

    for entity in nonpolymer_entities:
        chem_comp = entity.get("nonpolymer_comp", {}).get("chem_comp", {}).get("id")
        if chem_comp and chem_comp not in known_ligands:
            return True
    return False

for entry in data:
    nonpolymer_entities = entry["data"].get("nonpolymer_entities", []) or []  # 确保非None
    if is_complex_structure(nonpolymer_entities):
        complex_count += 1

# 输出复合物结构统计结果
print(f"Total number of complex structures: {complex_count} out of {len(data)}")
```

Total number of complex structures: 1503 out of 1852

5. 文件4在习题课时已经与助教沟通，将taxonomyID和第一问、第二问结果合并即可，这里不在赘述。

5）选择一个高分辨率的你认为具有代表性的主蛋白酶的结构，使用结构可视化 软件（如 UCSF ChimeraX, PyMOL）进行观察。找出构成配体结合口袋的残基，把配体结合口袋存成图片文件，图片中请标注 PDB ID 以及口袋残基。 请提交图片文件 5。



Command：

```
open 8hhu
```

```
[hide](help:user/commands/show.html#hide) **solvent**
```

```
[select](help:user/commands/select.html) **:LVX**
```

```
[ui tool show](help:user/commands/ui.html#tool-show) **Contacts**
```

```
[contacts](help:user/commands/clashes.html) **sel ignoreHiddenModels true**
```

```
[label](help:user/commands/label.html) **@@display**
```

图片已经存储为“图片文件5”

PDB ID: 8HHU

Ligand：LVX

口袋残基：如图所示：
THR26,HIS41,MET49,PHE140,ASN142,GLY143,SER144,HIS163,HIS164,MET165,GLU166

6）对 2）中得到的主蛋白酶序列进行多序列比对（可以使用 mafft）。在多序列比对结果中，把 5）中找到的残基所在列标注出来。请尝试根据这些标注的列对主蛋白酶序列进行重新聚类，可以使用 cdhit，按 85% seq id 来聚类。请提交 聚类结果文件 6。得到的结果和 3）一样吗？这样聚类对药物开发有什么好处?

1. 使用MAFFT进行多序列比对，先生成PDB ID: 8HH对应序列的fasta文件(8HHU.fa)，用其作为模板来比对，保证氨基酸残基不变：

```
mafft --keeplength --add translated_proteins.fasta 8HHU.fa >
aligned_with_template.fasta
```

2. 将这些位置的氨基酸提取出来，用cdhit，按 85% seq id 来聚类。

```
from Bio import AlignIO

# 读取比对后的 FASTA 文件
alignment = AlignIO.read("aligned_with_template.fasta", "fasta")

# 定义感兴趣的残基位置（1-based）
residue_positions = [26, 41, 49, 140, 142, 143, 144, 163, 164, 165, 166]

# 提取模板序列（8HHU）中的列号
template_seq = str(alignment[0].seq)
columns_to_extract = []
current_pos = 0
for i, char in enumerate(template_seq):
    if char != '-':
        current_pos += 1
    if current_pos in residue_positions:
        columns_to_extract.append(i)

# 提取每个序列中对应列的残基
with open("extracted_residues.fasta", "w") as output_file:
    for record in alignment:
        extracted_seq = "".join([record.seq[i] for i in columns_to_extract])
        output_file.write(f">{record.id}\n{extracted_seq}\n")
```

```
cd-hit -i extracted_residues.fasta -o clustered_residues -c 0.85 -n 5
```

查看文件：

```
189     11aa, >MZ081382.1_9986-109... at 100.00%
190     11aa, >MZ328294.1_9952-108... at 100.00%
191     11aa, >AY545917.1_9949-108... at 100.00%
192     11aa, >MZ293757.1_10353-11... at 90.91%
>Cluster 1
0       11aa, >AF353511.1_9288-101... *
1       11aa, >EU420139.1_9600-105... at 90.91%
2       11aa, >KF430219.1_9203-101... at 90.91%
3       11aa, >KJ473809.1_8659-956... at 90.91%
4       11aa, >MH938449.1_8834-973... at 90.91%
5       11aa, >MK472068.1_8466-937... at 90.91%
6       11aa, >MK472070.1_9196-101... at 90.91%
7       11aa, >MK720944.1_8729-963... at 90.91%
8       11aa, >KJ473797.1_9601-105... at 90.91%
9       11aa, >KY799179.1_9203-101... at 90.91%
10      11aa, >MH938448.1_8810-971... at 90.91%
11      11aa, >MH938450.1_8851-975... at 90.91%
12      11aa, >MK720945.1_9084-998... at 90.91%
13      11aa, >MK211373.1_9112-100... at 90.91%
14      11aa, >MN611518.1_9587-104... at 90.91%
15      11aa, >MZ293749.1_9103-100... at 90.91%
16      11aa, >MZ293734.1_9287-101... at 100.00%
17      11aa, >OM030318.1_9204-101... at 90.91%
18      11aa, >OP715780.1_9399-103... at 90.91%
>Cluster 2
0       11aa, >AF304460.1_9188-100... *
1       11aa, >AY567487.2_9104-100... at 100.00%
2       11aa, >EF203064.1_9120-100... at 100.00%
3       11aa, >KT368907.1_9203-101... at 100.00%
4       11aa, >KJ473808.1_8925-983... at 100.00%
5       11aa, >KJ473806.1_8909-981... at 90.91%
6       11aa, >KY073745.1_9222-101... at 100.00%
7       11aa, >AY518894.1_9090-999... at 100.00%
8       11aa, >JQ410000.1_9203-101... at 100.00%
9       11aa, >KY073747.1_9150-100... at 100.00%
10      11aa, >KT253324.1_9203-101... at 100.00%
11      11aa, >MH697599.1_9122-100... at 100.00%
12      11aa, >MF167434.1_9120-100... at 100.00%
13      11aa, >MT039232.1_9138-100... at 100.00%
14      11aa, >MK977618.1_9123-100... at 100.00%
15      11aa, >MF094685.1_9130-100... at 100.00%
16      11aa, >MN611517.1_9208-101... at 100.00%
17      11aa, >MN611522.1_8958-986... at 100.00%
18      11aa, >MZ328299.1_9040-994... at 90.91%
19      11aa, >MZ328298.1_8896-980... at 90.91%
20      11aa, >MZ293736.1_9268-101... at 100.00%
>Cluster 3
0       11aa, >AY597011.2_10208-11... *
1       11aa, >AF391541.1_9949-108... at 100.00%
2       11aa, >EF065513.1_9538-104... at 90.91%
3       11aa, >FJ938068.1_10164-11... at 100.00%
4       11aa, >JN874559.1_10127-11... at 100.00%
5       11aa, >KM349742.1_10092-11... at 100.00%
6       11aa, >KU762338.1_9676-105... at 90.91%
7       11aa, >AY700211.1_10210-11... at 100.00%
8       11aa, >AY585228.1_9948-108... at 100.00%
9       11aa, >KY370046.1_10054-10... at 90.91%
10      11aa, >MG693168.1_9402-103... at 90.91%
11      11aa, >EJ884687.1_10189-11... at 100.00%
```

显然和得到的结果和3）不一样，这些序列被分成了24类，相比上一问被分成的种类大大减少。原因是：

- 3）问：**多序列比对**：展示了所有序列的全局相似性，特别是关键残基的保守性。
- 6）问：**聚类分析**：基于特定残基位置的序列相似性，将序列分组，展示局部相似性模式

聚类对药物开发的好处

基于关键残基位置的聚类分析对药物开发有以下几方面的好处：

（1）识别关键残基的保守性，指导药物设计

- 通过聚类，可以快速识别保守的残基（如 THR26、HIS41 等），这些残基通常是蛋白质功能或结构的关键位点，可能是药物设计的潜在靶点，可以针对这些残基设计广谱抑制剂。
- 聚类可以将序列分为不同的亚群，每个亚群可能代表不同的功能或结构特征；某些亚群可能在关键残基位置有独特的突变，这可能与药物的选择性或耐药性相关。

（2）预测耐药性

- 这些位点是先通过解析蛋白和ligand结构后再进一步分析的，这些ligand可能就是某些药物，通过分析聚类中关键残基的变异模式，可以预测哪些突变可能导致药物耐药性。如果某些突变在聚类中频繁出现，可能表明这些突变与耐药性相关。

（3）优化实验设计

- 聚类可以帮助选择代表性的序列进行实验验证，减少实验工作量。可以选择每个聚类的代表性序列进行酶活性测定或药物筛选。

7）应对未来 COVID-X 爆发的策略之一开发广谱抗冠状病毒的药物，主蛋白酶 是冠状病毒内部的蛋白，和冠状病毒表面的刺突蛋白（spike）相比，受到的 进化压力要小得多，所以序列上相当保守，这为开发广谱抗冠状病毒药物提供了良好条件。如果你是相关决策人，根据以上的调查研究，你认为需要开发多少类冠状病毒主蛋白酶抑制剂就可以解决未来 COVID-X 爆发的问题。 请提交文件 7 来阐述理由。

从聚类文件6来看，Nsp5与候选药物结合的氨基酸可以被分成24类，因此只需要针对这些不同的类别开发24类冠状病毒主蛋白酶抑制剂即可。