```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt

from collections import defaultdict

import os
```

Introduction

In the following exploratory data analysis, the original pdb-secondary-structure dataset is compared to the updated datasets in Protein Secondary Structure - 2022 dataset.

```
# Load the various datasets

# original dataset from 2018
ss_2018 = pd.read_csv('/kaggle/input/protein-secondary-structure/2018-06-06-pdb-intersect-pisces.csv')

# these datasets were found to be only updated through mid-2020 - variables names were changed to reflect this
# updated data with 25% identity and 2.0 Angstrom cutoffs
ss_2020_25_20 = pd.read_csv('/kaggle/input/protein-secondary-structure-2022/2022-08-06-pdb-intersect-pisces_pc25_r2.0.csv')

# updated data with 25% identity and 2.5 Angstrom cutoffs
ss_2020_25_25 = pd.read_csv('/kaggle/input/protein-secondary-structure-2022/2022-08-06-pdb-intersect-pisces_pc25_r2.5.csv')

# updated data with 30% identity and 2.5 Angstrom cutoffs
ss_2020_30_25 = pd.read_csv('/kaggle/input/protein-secondary-structure-2022/2022-08-06-pdb-intersect-pisces_pc30_r2.5.csv')

# update datasets through end of 2022
ss_2022_25_20 = pd.read_csv('/kaggle/input/protein-secondary-structure-2022/2022-12-17-pdb-intersect-pisces_pc25_r2.0.csv')
ss_2022_25_25 = pd.read_csv('/kaggle/input/protein-secondary-structure-2022/2022-12-17-pdb-intersect-pisces_pc25_r2.5.csv')
ss_2022_30_25 = pd.read_csv('/kaggle/input/protein-secondary-structure-2022/2022-12-17-pdb-intersect-pisces_pc30_r2.5.csv')
```

Number of Sequences

When the updates were made, the culling file used had changed to only have sequences of length 40 and higher, where the original culling file used had sequences of length 20 and higher. This reduced the overall number of sequences available. To account for this and provide more sequences and secondary structures the percent identity and resolution cutoffs were relaxed.

The quality of the data should not be affected and the number of sequences increased substantially

```
tbl_data = [['ss_2018', '25%', '2.0 Angstrom', len(ss_2018)],
            ['ss_2020_25_20', '25%', '2.0 Angstrom',
len(ss_2020_25_20)],
            ['ss_2020_25_25', '25%', '2.5 Angstrom',
len(ss_2020_25_25)],
            ['ss_2020_30_20', '30%', '2.5 Angstrom',
len(ss_2020_30_25)],
            ['ss_2022_25_20', '25%', '2.0 Angstrom',
len(ss_2022_25_20)],
            ['ss_2022_25_25', '25%', '2.5 Angstrom',
len(ss_2022_25_25)],
            ['ss_2022_30_20', '30%', '2.5 Angstrom',
len(ss_2022_30_25)]]
pd.DataFrame(tbl_data, columns = ['Dataset', 'Percent Identity
Cutoff',
                                  'Resolution Cutoff', 'Number of
Sequences'])
```

Distribution of Sequence Lengths

As mentioned above, when keeping the same cutoff criteria as the original dataset from 2018, the updated file had fewer overall sequences and was slightly shifted toward longer sequences. Adjusting the cutoff criteria allowed for expansion of the dataset to ~47% more overall sequences. Also, sequences within the 50 to 500 amino acid range are substantially increased in the ss_2022_30_25 dataset which had the most permissive cutoff criteria of 30% sequence identity and 2.5 Angstrom resolution (bottom-left).

```
fig, axs = plt.subplots(3, 3, sharex = True, sharey = True, figsize =
(20, 12))

bins = range(25, 1000, 25) # bins are truncated at 1000 as there are
very few sequences beyond this length

axs[0, 0].hist(ss_2018['len'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[0, 0].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 0].title.set_text('ss_2018')

axs[1, 0].axis('off')
axs[2, 0].axis('off')

axs[0, 1].hist(ss_2020_25_20['len_x'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[0, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 1].title.set_text('ss_2020_25_20')
```

```
axs[1, 1].hist(ss_2020_25_25['len_x'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[1, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[1, 1].title.set_text('ss_2020_25_25')

axs[2, 1].hist(ss_2020_30_25['len_x'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[2, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[2, 1].title.set_text('ss_2020_30_25')

axs[0, 2].hist(ss_2022_25_20['len_x'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[0, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 2].title.set_text('ss_2022_25_20')

axs[1, 2].hist(ss_2022_25_25['len_x'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[1, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[1, 2].title.set_text('ss_2022_25_25')

axs[2, 2].hist(ss_2022_30_25['len_x'], bins = bins, zorder = 3,
                        edgecolor = 'black', linewidth = 1.0)
axs[2, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[2, 2].title.set_text('ss_2022_30_25')

fig.show()
```

Secondary Structure Distibutions

The bar plots, below, show the numbers of each secondary structure type for SST-8 and SST-3 categories. The bars are color coded by their SST-3 groupings in order to illustrate which SST-8 types are collected into SST-3 types. The one letter abbreviations for SST-8 and SST-3 are:

SST-8 Type	Description	SST-3 Type	Description	Color
B	β-bridge	E	Sheets	Yellow
E	β-strand	E		
G	3-helix	H	Helices	Red
H	α-helix	H		
I	π-helix	H		
C	Coil	C	Irregular or extended	Blue
S	Bend	C		
T	Turn	C		

The largest dataset (2022-12-17-pdb-intersect-pisces_pc30_r2.5.csv) provides nearly 800,000 amino acids in sheets and over 1 million amino acids in helices or irregular strucutres (bends, turns, and coils).

```
# set up storage for all files, both SST categoreis and the types of
SST for each category
SS_counts = {'ss_2018': {'SST-8': defaultdict(lambda: 0), 'SST-3':
defaultdict(lambda: 0) },
             'ss_2020_25_20': {'SST-8': defaultdict(lambda: 0), 'SST-
3': defaultdict(lambda: 0) },
             'ss_2020_25_25': {'SST-8': defaultdict(lambda: 0), 'SST-
3': defaultdict(lambda: 0) },
             'ss_2020_30_25': {'SST-8': defaultdict(lambda: 0), 'SST-
3': defaultdict(lambda: 0) },
             'ss_2022_25_20': {'SST-8': defaultdict(lambda: 0), 'SST-
3': defaultdict(lambda: 0) },
             'ss_2022_25_25': {'SST-8': defaultdict(lambda: 0), 'SST-
3': defaultdict(lambda: 0) },
             'ss_2022_30_25': {'SST-8': defaultdict(lambda: 0), 'SST-
3': defaultdict(lambda: 0) }}

# count the types for each dataset
for seq in ss_2018['sst8']:
    for ss in set(seq):
        SS_counts['ss_2018']['SST-8'][ss] += seq.count(ss)

for seq in ss_2018['sst3']:
    for ss in set(seq):
        SS_counts['ss_2018']['SST-3'][ss] += seq.count(ss)

for seq in ss_2020_25_20['sst8']:
    for ss in set(seq):
        SS_counts['ss_2020_25_20']['SST-8'][ss] += seq.count(ss)

for seq in ss_2020_25_20['sst3']:
    for ss in set(seq):
        SS_counts['ss_2020_25_20']['SST-3'][ss] += seq.count(ss)

for seq in ss_2020_25_25['sst8']:
    for ss in set(seq):
        SS_counts['ss_2020_25_25']['SST-8'][ss] += seq.count(ss)

for seq in ss_2020_25_25['sst3']:
    for ss in set(seq):
        SS_counts['ss_2020_25_25']['SST-3'][ss] += seq.count(ss)

for seq in ss_2020_30_25['sst8']:
    for ss in set(seq):
        SS_counts['ss_2020_30_25']['SST-8'][ss] += seq.count(ss)

for seq in ss_2020_30_25['sst3']:
    for ss in set(seq):
        SS_counts['ss_2020_30_25']['SST-3'][ss] += seq.count(ss)
```

```
# updated data from end of 2022
for seq in ss_2022_25_20['sst8']:
    for ss in set(seq):
        SS_counts['ss_2022_25_20']['SST-8'][ss] += seq.count(ss)

for seq in ss_2022_25_20['sst3']:
    for ss in set(seq):
        SS_counts['ss_2022_25_20']['SST-3'][ss] += seq.count(ss)

for seq in ss_2022_25_25['sst8']:
    for ss in set(seq):
        SS_counts['ss_2022_25_25']['SST-8'][ss] += seq.count(ss)

for seq in ss_2022_25_25['sst3']:
    for ss in set(seq):
        SS_counts['ss_2022_25_25']['SST-3'][ss] += seq.count(ss)

for seq in ss_2022_30_25['sst8']:
    for ss in set(seq):
        SS_counts['ss_2022_30_25']['SST-8'][ss] += seq.count(ss)

for seq in ss_2022_30_25['sst3']:
    for ss in set(seq):
        SS_counts['ss_2022_30_25']['SST-3'][ss] += seq.count(ss)

# plot a comparison across datasets

# define order for ss types
ss8_types = ['B', 'E', 'G', 'H', 'I', 'C', 'S', 'T']
ss3_types = ['E', 'H', 'C']

sst8_colors = ['gold', 'gold', 'crimson', 'crimson', 'crimson',
'navy', 'navy', 'navy']
sst3_colors = ['gold', 'crimson', 'navy']

fig, axs = plt.subplots(6, 3, sharey = 'all', figsize = (15, 25))

# SST-8 comparisons

axs[0, 0].bar(range(8), height = [SS_counts['ss_2018']['SST-8'][ss]
for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
axs[0, 0].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 0].title.set_text('ss_2018')

axs[0, 1].bar(range(8), height = [SS_counts['ss_2020_25_20']['SST-8']
[ss] for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
```

```
axs[0, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 1].title.set_text('ss_2020_25_20')

axs[0, 2].bar(range(8), height = [SS_counts['ss_2022_25_20']['SST-8']
[ss] for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
axs[0, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 2].title.set_text('ss_2022_25_20')

axs[1, 0].axis('off')

axs[1, 1].bar(range(8), height = [SS_counts['ss_2020_25_25']['SST-8']
[ss] for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
axs[1, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[1, 1].title.set_text('ss_2020_25_25')

axs[1, 2].bar(range(8), height = [SS_counts['ss_2022_25_25']['SST-8']
[ss] for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
axs[1, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[1, 2].title.set_text('ss_2022_25_25')

axs[2, 0].axis('off')

axs[2, 1].bar(range(8), height = [SS_counts['ss_2020_30_25']['SST-8']
[ss] for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
axs[2, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[2, 1].title.set_text('ss_2020_25_25')

axs[2, 2].bar(range(8), height = [SS_counts['ss_2022_30_25']['SST-8']
[ss] for ss in ss8_types],
              tick_label = ss8_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst8_colors)
axs[2, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[2, 2].title.set_text('ss_2022_25_25')

# SST-3 comparisons

axs[3, 0].bar(range(3), height = [SS_counts['ss_2018']['SST-3'][ss]
for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[3, 0].grid(axis = 'y', which = 'both', zorder = 0)
axs[3, 0].title.set_text('ss_2018')
```

```
axs[3, 1].bar(range(3), height = [SS_counts['ss_2022_25_20']['SST-3']
[ss] for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[3, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[3, 1].title.set_text('ss_2022_25_20')

axs[3, 2].bar(range(3), height = [SS_counts['ss_2022_25_20']['SST-3']
[ss] for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[3, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[3, 2].title.set_text('ss_2022_25_20')

axs[4, 0].axis('off')

axs[4, 1].bar(range(3), height = [SS_counts['ss_2020_25_25']['SST-3']
[ss] for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[4, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[4, 1].title.set_text('ss_2020_25_25')

axs[4, 2].bar(range(3), height = [SS_counts['ss_2022_25_25']['SST-3']
[ss] for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[4, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[4, 2].title.set_text('ss_2022_25_25')

axs[5, 0].axis('off')

axs[5, 1].bar(range(3), height = [SS_counts['ss_2020_30_25']['SST-3']
[ss] for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[5, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[5, 1].title.set_text('ss_2020_30_25')

axs[5, 2].bar(range(3), height = [SS_counts['ss_2022_30_25']['SST-3']
[ss] for ss in ss3_types],
              tick_label = ss3_types, edgecolor = 'black', width =
0.75, zorder = 3, color = sst3_colors)
axs[5, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[5, 2].title.set_text('ss_2022_30_25')

fig.show()
```

Amino Acid Representation

Looking at the actual amino acid distributions in the file (ignoring those sequences with non-standard amino acids), the histograms show a similar distribution as those for the secondary structures. The dataset with similar constraints as the 2018 original loses some content while the datasets with relaxed constraints show substantial increases in actual numbers of amino acids. Interestingly, the table shows that across all three datasets, the relative proportions of amino acids are very well conserved.

```
# set up storage for all files
AA_counts = {'ss_2018': defaultdict(lambda: 0),
             'ss_2020_25_20': defaultdict(lambda: 0),
             'ss_2020_25_25': defaultdict(lambda: 0),
             'ss_2020_30_25': defaultdict(lambda: 0),
             'ss_2022_25_20': defaultdict(lambda: 0),
             'ss_2022_25_25': defaultdict(lambda: 0),
             'ss_2022_30_25': defaultdict(lambda: 0)}

# count the types for each dataset
for (seq, nonstd) in zip(ss_2018['seq'], ss_2018['has_nonstd_aa']):
    if not nonstd:
        for aa in set(seq):
            if aa != '*':
                AA_counts['ss_2018'][aa] += seq.count(aa)

for (seq, nonstd) in zip(ss_2020_25_20['seq'],
ss_2020_25_20['has_nonstd_aa']):
    if not nonstd:
        for aa in set(seq):
            if aa != '*':
                AA_counts['ss_2020_25_20'][aa] += seq.count(aa)

for (seq, nonstd) in zip(ss_2020_25_25['seq'],
ss_2020_25_25['has_nonstd_aa']):
    if not nonstd:
        for aa in set(seq):
            if aa != '*':
                AA_counts['ss_2020_25_25'][aa] += seq.count(aa)

for (seq, nonstd) in zip(ss_2020_30_25['seq'],
ss_2020_30_25['has_nonstd_aa']):
    if not nonstd:
        try:
            for aa in set(seq):
                if aa != '*':
                    AA_counts['ss_2020_30_25'][aa] += seq.count(aa)
        except:
            pass

for (seq, nonstd) in zip(ss_2022_25_20['seq'],
```

```
ss_2022_25_20['has_nonstd_aa']):
    if not nonstd:
        for aa in set(seq):
            if aa != '*':
                AA_counts['ss_2022_25_20'][aa] += seq.count(aa)

for (seq, nonstd) in zip(ss_2022_25_25['seq'],
ss_2022_25_25['has_nonstd_aa']):
    if not nonstd:
        for aa in set(seq):
            if aa != '*':
                AA_counts['ss_2022_25_25'][aa] += seq.count(aa)

for (seq, nonstd) in zip(ss_2022_30_25['seq'],
ss_2022_30_25['has_nonstd_aa']):
    if not nonstd:
        for aa in set(seq):
            if aa != '*':
                AA_counts['ss_2022_30_25'][aa] += seq.count(aa)

[sum(AA_counts[d].values()) for d in AA_counts.keys()]

# order the amino acids by decreasing total abundance
total_aa = [sum([AA_counts[d][aa] for d in AA_counts.keys()]) for aa
in AA_counts['ss_2018'].keys() ]
temp = sorted(total_aa, reverse = True)
order = [total_aa.index(v) for v in temp]
aa_order = [list(AA_counts['ss_2018'].keys())[i] for i in order]

# plot a comparison across datasets
fig, axs = plt.subplots(3, 3, sharey = True, figsize = (20, 20))

axs[0, 0].bar(range(20), height = [AA_counts['ss_2018'][aa] for aa in
aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
axs[0, 0].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 0].title.set_text('ss_2018')

axs[0, 1].bar(range(20), height = [AA_counts['ss_2020_25_20'][aa] for
aa in aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
axs[0, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 1].title.set_text('ss_2020_25_20')

axs[0, 2].bar(range(20), height = [AA_counts['ss_2022_25_20'][aa] for
aa in aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
```

```
axs[0, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[0, 2].title.set_text('ss_2022_25_20')

axs[1, 0].axis('off')

axs[1, 1].bar(range(20), height = [AA_counts['ss_2020_25_25'][aa] for
aa in aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
axs[1, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[1, 1].title.set_text('ss_2020_25_25')

axs[1, 2].bar(range(20), height = [AA_counts['ss_2022_25_25'][aa] for
aa in aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
axs[1, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[1, 2].title.set_text('ss_2022_25_25')

axs[2, 0].axis('off')

axs[2, 1].bar(range(20), height = [AA_counts['ss_2020_30_25'][aa] for
aa in aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
axs[2, 1].grid(axis = 'y', which = 'both', zorder = 0)
axs[2, 1].title.set_text('ss_2020_30_25')

axs[2, 2].bar(range(20), height = [AA_counts['ss_2022_30_25'][aa] for
aa in aa_order],
              tick_label = aa_order, edgecolor = 'black', width =
0.75, zorder = 3)
axs[2, 2].grid(axis = 'y', which = 'both', zorder = 0)
axs[2, 2].title.set_text('ss_2022_30_25')

fig.show()

# show proportion of each amino acid in a table
tbl_data = {'Amino Acid': aa_order,
            'ss_2018': [ round(AA_counts['ss_2018'][aa] /
sum(AA_counts['ss_2018'].values()), 3) for aa in aa_order],
            'ss_2020_25_20': [ round(AA_counts['ss_2020_25_20'][aa] /
sum(AA_counts['ss_2020_25_20'].values()), 3) for aa in aa_order],
            'ss_2020_25_25': [ round(AA_counts['ss_2020_25_25'][aa] /
sum(AA_counts['ss_2020_25_25'].values()), 3) for aa in aa_order],
            'ss_2020_30_25': [ round(AA_counts['ss_2020_30_25'][aa] /
sum(AA_counts['ss_2020_30_25'].values()), 3) for aa in aa_order],
            'ss_2022_25_20': [ round(AA_counts['ss_2022_25_20'][aa] /
sum(AA_counts['ss_2022_25_20'].values()), 3) for aa in aa_order],
            'ss_2022_25_25': [ round(AA_counts['ss_2022_25_25'][aa] /
```

```
sum(AA_counts['ss_2022_25_25'].values()), 3) for aa in aa_order],
            'ss_2022_30_25': [ round(AA_counts['ss_2022_30_25'][aa] /
sum(AA_counts['ss_2022_30_25'].values()), 3) for aa in aa_order]}
pd.DataFrame(tbl_data)
```