



Analyzing the Protein-Protein Interaction Network

Fall 22, SBE3031 - Advanced Topics in Medical Informatics

Ibrahim Mohamed Youssef, PhD

Assistant Professor

Abdelrahman Ali Farouk

Mahmoud Yaser Salman

Nevein Mohamed Ayman

Omar Saad EL-Gharbawy



February 6, 2023

Table of Contents

- Introduction.....	3
- Methods.....	3
- Conclusion	14
- References.....	15
- Members Contribution	16



- Introduction

The protein-protein interaction network plays a critical role in various biological processes, such as signaling pathways and networks. Protein-protein interactions (PPIs) are physical contacts of high specificity established between two or more protein molecules, proteins do not function in isolation; instead, it is their interactions with one another and with other molecules that mediate metabolic and signaling pathways, cellular processes, and organismal systems, protein interaction data is incredibly important. It describes the interplay between the biomolecules encoded by genes. It allows us to understand the complexities of cellular function and even predict potential therapeutics. In this research paper, we will focus on analyzing these interactions. We will utilize computational techniques to map and study the network, providing valuable insights into the organization and regulation of these interactions.

The interactome used in this study represents a directed network of PPIs and was obtained from the PathLinker database. Using the NetworkX python package, we will construct a graph from the interactome and perform various analyses on the network, including finding the shortest path between two proteins, identifying directly connected proteins, ranking proteins based on their degree, converting UniProt IDs to gene names, and converting the graph to an unweighted graph.

- Methods

Used Packages:

- 1- **NetworkX** is a powerful Python package used to create, manipulate, and analyze complex network structures. It provides a variety of algorithms and data structures to support the efficient handling of graph-based data, making it an ideal tool for the study of protein-protein interactions.
- 2- **UniProt** is a freely accessible database of protein sequence and functional information, many entries being derived from genome sequencing projects. It is maintained by the UniProt consortium, which consists of several European bioinformatics organizations and a foundation. The proteome identifier (UPID) is the unique identifier assigned to the protein set that constitutes the proteome. It consists of the characters 'UP' followed by 9 digits, is stable across releases, and can therefore be used to cite a UniProt proteome.
- 3- **Pandas** is Python's widely used data manipulation and analysis library. It provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. We used it for data exploration.



- 4- **NumPy** is a powerful library for scientific computing in Python. It provides support for arrays, which are essential for numerical computing and data analysis, as well as mathematical functions and tools to manipulate arrays efficiently. We used it in constructing an adjacency matrix.
- 5- **Matplotlib and Seaborn** are widely used Python data visualization libraries that allow for creating interactive, publication-quality graphs and charts. We used it for visualizing all our figures.
- 6- **StringIO** from the **io** library allows for the creation of a virtual file-like object from a string. We used it for reading **UniProt** gene information from a given protein.

Steps:

- **Reading Text File Data.**

- Unzipped the data file and read the data using python.
- Extracting proteins' tails, heads, and edge weights.

- **Data Exploration with Pandas.**

```
In 9 1 print('Network Observations')
2 print('-'*100)
3 print(f' Our network has {len(weighted_edges_df['Tail'].unique())} unique tails: {weighted_edges_df['Tail'].unique()}')
4 print('-'*100)
5 print(f' Our network has {len(weighted_edges_df['Head'].unique())} unique heads: {weighted_edges_df['Head'].unique()}')
6 print('-'*100)

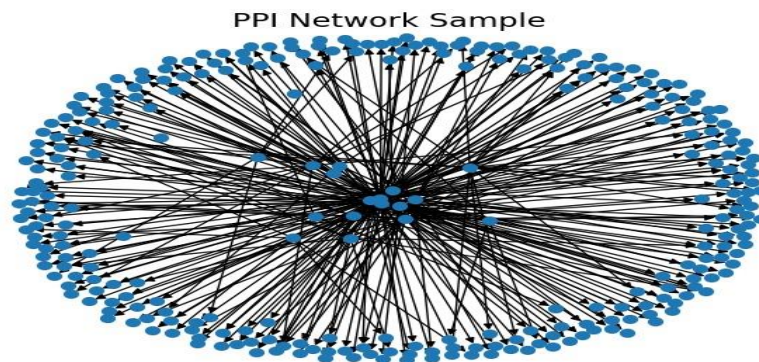
v Network Observations
.....
Our network has 17082 unique tails: ['Q8TBF5' 'Q8TBF4' 'Q5M1Z7' ... 'Q7Z739' 'Q17R08' 'Q3LF05']
.....
Our network has 17095 unique heads: ['Q9UKB1' 'Q15717' 'P68865' ... 'Q088S8' 'Q8N4T0' 'Q8IXL9']
.....
```

- As the number of heads is very close to the number of tails, we may assume that the network is balanced. and this assumption can lead to several insights:
 - Network stability: A balanced PPI network is likely to be more stable and less prone to disruption than an imbalanced network.
 - Symmetrical interactions: The equal number of unique tails and heads suggests that the interactions in the network are symmetrical and that proteins are likely to be interacting with each other in a reciprocal manner.
 - Functionality: Proteins in a balanced PPI network are likely to be playing *similar roles* in the network, regardless of their specific interactions.



- Redundancy: A balanced PPI network is likely to have a high degree of redundancy, with many proteins having similar interactions and functional roles.

- **Construct a graph (biological network) from the above interactome.**



- By using NetworkX package we constructed a graph by taking a sample of 300 edge out of 612,516

- **Given two proteins, list the acyclic shortest path(s) between these two nodes in a text file.**

```
In 12 1 weighted_shortest_paths= ([p for p in nx.all_shortest_paths(DG, source='P20933', target='Q15303', weight='weight')])
      2 weighted_shortest_paths

Out 12 1 [['P20933', 'Q9Y3A3', 'Q68CZ1', 'P04637', 'Q15303'],
          ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P04637', 'Q15303'],
          ['P20933', 'Q9Y3A3', 'Q15259', 'P22681', 'Q15303'],
          ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P22681', 'Q15303'],
          ['P20933', 'Q9Y3A3', 'Q96ST8', 'P46108', 'Q15303'],
          ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P46108', 'Q15303']]
      2
      3
      4
      5
      6
      7
      8
      9
     10
     11
     12
     13
     14
     15
     16
     17
     18
     19
     20
     21
     22
     23
     24
     25
     26
     27
     28
     29
     30
     31
     32
     33
     34
     35
     36
     37
     38
     39
     40
     41
     42
     43
     44
     45
     46
     47
     48
     49
     50
     51
     52
     53
     54
     55
     56
     57
     58
     59
     60
     61
     62
     63
     64
     65
     66
     67
     68
     69
     70
     71
     72
     73
     74
     75
     76
     77
     78
     79
     80
     81
     82
     83
     84
     85
     86
     87
     88
     89
     90
     91
     92
     93
     94
     95
     96
     97
     98
     99
    100
    101
    102
    103
    104
    105
    106
    107
    108
    109
    110
    111
    112
    113
    114
    115
    116
    117
    118
    119
    120
    121
    122
    123
    124
    125
    126
    127
    128
    129
    130
    131
    132
    133
    134
    135
    136
    137
    138
    139
    140
    141
    142
    143
    144
    145
    146
    147
    148
    149
    150
    151
    152
    153
    154
    155
    156
    157
    158
    159
    160
    161
    162
    163
    164
    165
    166
    167
    168
    169
    170
    171
    172
    173
    174
    175
    176
    177
    178
    179
    180
    181
    182
    183
    184
    185
    186
    187
    188
    189
    190
    191
    192
    193
    194
    195
    196
    197
    198
    199
    200
    201
    202
    203
    204
    205
    206
    207
    208
    209
    210
    211
    212
    213
    214
    215
    216
    217
    218
    219
    220
    221
    222
    223
    224
    225
    226
    227
    228
    229
    230
    231
    232
    233
    234
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    250
    251
    252
    253
    254
    255
    256
    257
    258
    259
    260
    261
    262
    263
    264
    265
    266
    267
    268
    269
    270
    271
    272
    273
    274
    275
    276
    277
    278
    279
    280
    281
    282
    283
    284
    285
    286
    287
    288
    289
    290
    291
    292
    293
    294
    295
    296
    297
    298
    299
    300
    301
    302
    303
    304
    305
    306
    307
    308
    309
    310
    311
    312
    313
    314
    315
    316
    317
    318
    319
    320
    321
    322
    323
    324
    325
    326
    327
    328
    329
    330
    331
    332
    333
    334
    335
    336
    337
    338
    339
    340
    341
    342
    343
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368
    369
    370
    371
    372
    373
    374
    375
    376
    377
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811

```



○ Provide the total path score.

■ Weighted graphs:

- Weights are costs: the path with the minimum total weight of its edges (the minimum cost), and it is the path with the minimum energy consumption in the cell.
- Weights are probabilities of interaction: the path with the maximum total weight of its edges (the most probable path), and it is the path with the highest interaction rate.
- So, we conclude that the shortest path in our case is the path that is most likely to happen. Assuming that the edge weights are cost weights.

○ Provide the weight of each interaction in the path(s).

```
In 14 1 sub_network = nx.DiGraph()
2 all_paths_edges = []
3 print('The weight of each interaction in the path(s)')
4 print("-"*150)
5 for shortest_path in weighted_shortest_paths :
6     paths_edges = [tuple([shortest_path[i],shortest_path[i+1],nx.path_weight(DG,([shortest_path[i],shortest_path[i+1]]), weight='weight')) for i in range(len
7         (shortest_path)-1))]
8     print(paths_edges)
9     # add path edges to the new sub-network
10    sub_network.add_weighted_edges_from(paths_edges)
11    all_paths_edges.append(paths_edges)
12    print("-"*150)
```

✓ The weight of each interaction in the path(s)

```
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q68C21', 0.0333391), ('Q68C21', 'P04637', 0.0333391), ('P04637', 'Q15303', 0.311133)]
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q6ZU80', 0.0333391), ('Q6ZU80', 'P04637', 0.0333391), ('P04637', 'Q15303', 0.311133)]
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', '015259', 0.0333391), ('015259', 'P22681', 0.0333391), ('P22681', 'Q15303', 0.311133)]
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q6ZU80', 0.0333391), ('Q6ZU80', 'P22681', 0.0333391), ('P22681', 'Q15303', 0.311133)]
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q96ST8', 0.0333391), ('Q96ST8', 'P46108', 0.0333391), ('P46108', 'Q15303', 0.311133)]
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q6ZU80', 0.0333391), ('Q6ZU80', 'P46108', 0.0333391), ('P46108', 'Q15303', 0.311133)]
```

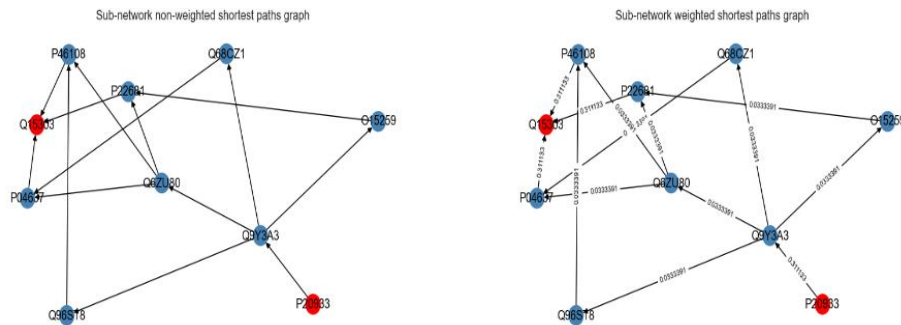
○ If more than one path, report all the paths.

```
In 15 1 for i, path in enumerate(weighted_shortest_paths):
2     print(f'path{i} is: {path}')

✓ path0 is: ['P20933', 'Q9Y3A3', 'Q68C21', 'P04637', 'Q15303']
path1 is: ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P04637', 'Q15303']
path2 is: ['P20933', 'Q9Y3A3', '015259', 'P22681', 'Q15303']
path3 is: ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P22681', 'Q15303']
path4 is: ['P20933', 'Q9Y3A3', 'Q96ST8', 'P46108', 'Q15303']
path5 is: ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P46108', 'Q15303']
```



- Use NetworkX and matplotlib to draw the sub-network formed by these shortest paths.



- **Given one protein, list all the directly connected proteins to it in a text file.**

- Report the degree (number of connections) of this protein in a separate line.

```
In 36 1 predecessors = [n for n in DG.predecessors(given_protein)]
      2 print(f'The In Degree for the given protein is {len(predecessors)} and the proteins that interacts with the given protein are:\n{predecessors}')
      3 in_degree = DG.in_degree(given_protein)

~ The In Degree for the given protein is 31 and the proteins that interacts with the given protein are:
['P62829', 'Q99689', 'Q14745', 'P27348', 'P30884', 'Q86VU5', 'Q14103', 'P04629', 'Q96LJ8', 'Q80839', 'P35557', 'P66510', 'Q13263', 'P07384', 'Q9H357',
'Q53H82', 'Q98ZF1', 'Q75439', 'Q60610', 'P62714', 'Q87866', 'P21980', 'Q9NR45', 'Q96KP4', 'P46379', 'P51648', 'Q9NV27', 'Q9H086', 'Q8K490', 'P33176',
'Q16719']

In 37 1 print(f'The total Degree for the given protein is {len(predecessors)+len(successors)} and the proteins that interacts with or the given protein interacts with
      2 are:\n{predecessors+successors}')
      3 all_degree = DG.degree(given_protein)

~ The total Degree for the given protein is 64 and the proteins that interacts with or the given protein interacts with are:
['P62829', 'Q99689', 'Q14745', 'P27348', 'P30884', 'Q86VU5', 'Q14103', 'P04629', 'Q96LJ8', 'Q80839', 'P35557', 'P66510', 'Q13263', 'P07384', 'Q9H357',
'Q53H82', 'Q98ZF1', 'Q75439', 'Q60610', 'P62714', 'Q87866', 'P21980', 'Q9NR45', 'Q96KP4', 'P46379', 'P51648', 'Q9NV27', 'Q9H086', 'Q8K490', 'P33176',
'Q16719', 'Q8K490', 'Q53H82', 'P51648', 'P04629', 'P46379', 'P16104', 'Q99689', 'P62714', 'Q9H357', 'Q16719', 'Q86VU5', 'Q96LJ8', 'Q07866', 'P27348',
'Q9NR45', 'P07384', 'P30884', 'Q75439', 'Q9H086', 'P33176', 'P62829', 'P35557', 'Q80839', 'Q53H82', 'Q14745', 'Q80839', 'Q13263', 'Q96KP4', 'Q9NV27',
'P66510', 'P21980', 'Q98ZF1', 'Q14103']
```





- **Conclusion:** when in-degree is close to out-degree that can result in stability of overall network topological, thus some studies have suggested that a more stable network may decrease the probability of disease infection by providing more robust connections between proteins, reducing the risk of network failure and disease spread. However, this is not a universal trend, and stability in the network topology may not always result in decreased disease infection. Further research is needed to fully understand the relationship between network stability, network topology, and disease infection.



- ```

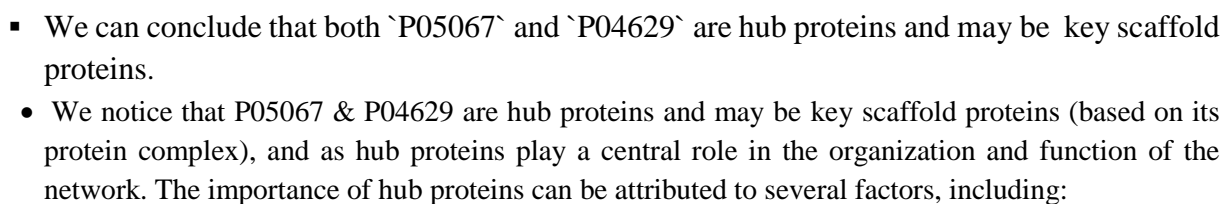
In 38: protein_predecessors_file = open(f"data\\{given_protein}.predecessors.txt", "w")
 1: protein_predecessors_file.write(f"{given_protein} in degree = {d} \n" Min_degree)
 2: protein_predecessors_file.write(f"\nTail\tHead\tEdge_weight\n")
 3: for tail in predecessors:
 4: protein_predecessors_file.write(f"{tail}\t{given_protein}\t{nx.path_weight(OG, (tail, given_protein), weight='weight')}\n")
 5: protein_predecessors_file.close()

In 39: # Saved text file
 1: weighted_edges_df[weighted_edges_df["Head"]==given_protein]

```
- Out 39:     Rows: 10/100
- |        | Tail   | Head   | Edge_weight |
|--------|--------|--------|-------------|
| 27817  | P62829 | Q5M1Z7 | 0.281461    |
| 31863  | Q99689 | Q5M1Z7 | 0.758086    |
| 34520  | O14745 | Q5M1Z7 | 0.281461    |
| 68461  | P27348 | Q5M1Z7 | 0.281461    |
| 122329 | P38084 | Q5M1Z7 | 0.281461    |
| 137890 | Q8AVU5 | Q5M1Z7 | 0.311133    |
| 159380 | Q14183 | Q5M1Z7 | 0.311133    |
| 166235 | P04629 | Q5M1Z7 | 0.311133    |
| 195937 | Q96L38 | Q5M1Z7 | 0.311133    |

| Tail   | Head   | Edge_weight |
|--------|--------|-------------|
| P62829 | Q5MIZ7 | 0.201461    |
| Q99689 | Q5MIZ7 | 0.75        |
| O14745 | Q5MIZ7 | 0.201461    |
| P27348 | Q5MIZ7 | 0.201461    |
| P30084 | Q5MIZ7 | 0.201461    |
| Q86VU5 | Q5MIZ7 | 0.311133    |

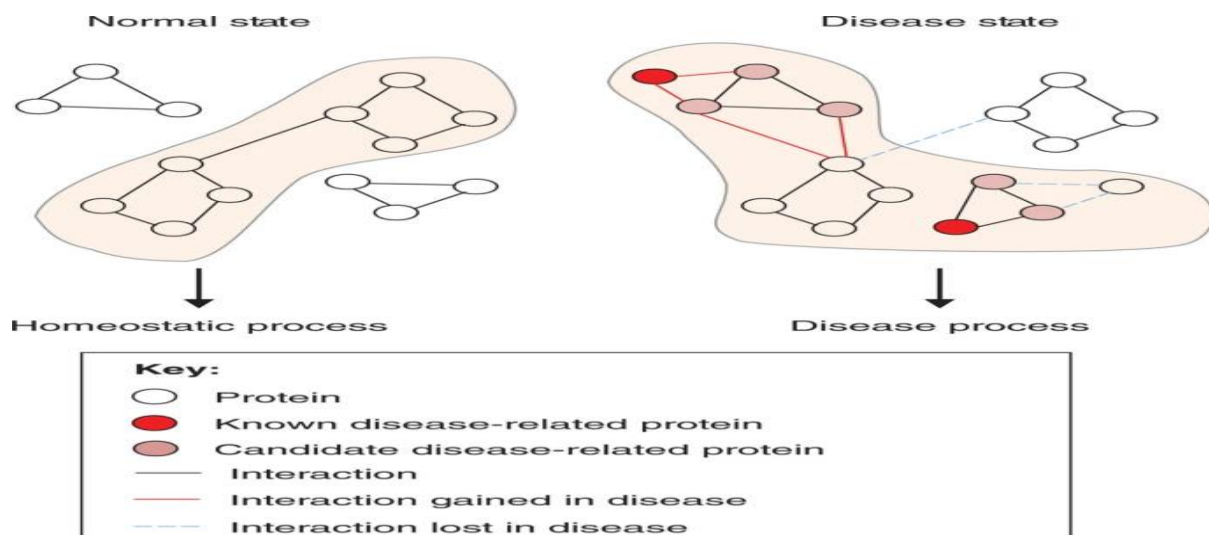
- Draw a histogram for the protein's degree.







- Modularity: Hub proteins can act as central connectors between different modules or functional units in the network, allowing for effective communication and coordination of cellular processes.
  - Robustness: Hub proteins can provide stability to the network by forming multiple interactions with other proteins, making the network less susceptible to failure or perturbations.
  - Functionality: Hub proteins are often involved in multiple pathways and functions, making them crucial for the overall function of the cell.
  - Evolution: Hub proteins tend to evolve slowly and are conserved across species, suggesting their importance in the evolution and stability of cellular processes.
  - Disease: Disruptions in the interactions of hub proteins can have a significant impact on cellular processes and have been implicated in various diseases, such as cancer.
- Diseases of interactions(PPI)
    - Many human diseases are the result of abnormal protein–protein interactions involving endogenous proteins, proteins from pathogens or both. The inhibition of these aberrant associations is of obvious clinical significance. Because of the diverse nature of protein–protein interactions, however, the successful design of therapeutics requires detailed knowledge of each system at a molecular and atomic level and new undesired protein interactions are the main causes of several diseases.
    - View of how **disease-related proteins** can drive disease processes by altering individual protein complexes and protein network dynamics.





- So, our subnetwork is robust, modular, & stable. and their hub proteins have some characteristics:
  - P05067
    - **Function**
      - Functions as a cell surface receptor and performs physiological functions on the surface of neurons relevant to neurite growth, neuronal adhesion and axonogenesis. Interaction between APP molecules on neighboring cells promotes synaptogenesis Involved in cell mobility and transcription regulation through protein-protein interactions. Can promote transcription activation through binding to APBB1-KAT5 and inhibits Notch signaling through interaction with Numb. Couples to apoptosis-inducing pathways such as those mediated by G(o) and JIP. Inhibits G(o) alpha ATPase activity (By similarity).
    - **Disease**
      - Alzheimer's disease 1 (AD1)
        - The disease is caused by variants affecting the gene represented in this entry.
      - Cerebral amyloid angiopathy, APP-related (CAA-APP)
        - The disease is caused by variants affecting the gene represented in this entry.
  - P04629
    - **Function**
      - Receptor tyrosine kinase is involved in the development and maturation of the central and peripheral nervous systems through the regulation of proliferation, differentiation, and survival of sympathetic and nervous neurons. High-affinity receptor for NGF which is its primary.
    - **Disease**
      - Congenital insensitivity to pain with anhidrosis (CIPA). The disease is caused by variants affecting the gene represented in this entry Characterized by congenital insensitivity to pain, anhidrosis (absence of sweating), absence of reaction to noxious stimuli, self-mutilating behavior, and intellectual disability.



- Rank these proteins from the highly connected to the least in a text file, where each line is a protein and its corresponding degree.

| Protein | Degree |
|---------|--------|
| P05067  | 4170   |
| P04629  | 3950   |
| Q15717  | 3597   |
| P05412  | 3052   |
| O14980  | 2462   |
| P01106  | 2429   |
| Q13618  | 2367   |
| Q9UBU9  | 2259   |
| P04637  | 2183   |

- **Provide a conversion map between the protein UniProt ID and its gene name.**

- You can be provided with one protein ID or a set of protein IDs, and then you need to get their corresponding gene names.

```
In [35]: 1 # Create a link to the UniProt web service
2 service = UniProt()
3 # Define an empty list to store the gene names
4 gene_names = []
5 # Get the list of UniProt IDs from the sub_proteins_degrees dataframe
6 UniProt_IDs = list(sub_proteins_degrees.iloc[:, 0]).copy()
7 # Loop through each UniProt ID in the list
8 for ID in UniProt_IDs:
9 # Send the query to UniProt and store the result
10 result = service.search(ID)
11 # Convert the result into a pandas dataframe using the tab separator
12 gene_df = pd.read_csv(StringIO(result), sep='\t')
13 # Append the gene name of the current ID to the gene_names list
14 gene_names.append(gene_df['Gene Names'])
15 # Print the UniProt IDs and the corresponding gene names
16 print('Proteins are\n {UniProt_IDs}\n Corresponding genes are {gene_names}')

> Out [00:00, ?it/s]\nOut [00:00, ?it/s]\nOut [00:00, ?it/s]\nOut [00:00, ?it/s]\nOut [00:00, ?it/s]\nOut [00:00, ?it/s]\n
>
> Proteins are
> ['P05067', 'P04629', 'Q15717', 'P05412', 'O14980', 'P01106', 'Q13618', 'Q9UBU9', 'P04637', 'P05533']
> Corresponding genes are ['APP', 'NTRK1', 'ELAVL1', 'JUN', 'XP01', 'MYC', 'CUL3', 'NFE1', 'TP53', 'EGFR']
```



- As we concluded before that we have two hub proteins, in this section we retrieved their corresponding genes.

```
In 51: # Make a link to the UniProt webservice
 service = UniProt()
 # Build a query string (try this a test)
 query_1 = "P05067"
 query_2 = "P04629"
 # Send the query to UniProt, and catch the search result in a variable
 result_1 = service.search(query_1)
 result_2 = service.search(query_2)
 # Create a dataframe from the first query result
 df_gene_1 = pd.read_csv(StringIO(result_1), sep='\t')
 # Create a dataframe from the second query result
 df_gene_2 = pd.read_csv(StringIO(result_2), sep='\t')
 # Concatenate both dataframes into a single dataframe
 df_genes = pd.concat([df_gene_1, df_gene_2])
 # Display the final dataframe
 df_genes

Out 51:
 Entry Entry Name Reviewed Protein names Gene Names Organism Length
0 P05067 A4_HUMAN reviewed Amyloid-beta precursor p. APP A4 AD1 Homo sapiens (Human) 776
0 P04629 NTRK1_HUMAN reviewed High affinity nerve grow. NTRK1 MTC TRK TRKA Homo sapiens (Human) 796
```

- Convert the above graph as an unweighted graph and save it using the adjacency matrix method.

```
In 38: import numpy as np
 data = weighted_shortest_paths.copy()
 unique_proteins = list(set([protein for sublist in data for protein in sublist]))
 adjacency_matrix = np.zeros((len(unique_proteins), len(unique_proteins)), dtype=int)
 for sublist in data:
 for i, protein1 in enumerate(sublist[:-1]):
 for j, protein2 in enumerate(sublist[i+1:], i+1):
 protein1_index = unique_proteins.index(protein1)
 protein2_index = unique_proteins.index(protein2)
 adjacency_matrix[protein1_index][protein2_index] + 1
 adjacency_matrix[protein2_index][protein1_index] + 1
 print(adjacency_matrix)

[[0 1 0 1 1 0 1 0 0 1]
 [1 0 1 1 1 1 1 1 1 1]
 [0 1 0 1 1 1 0 0 0 0]
 [1 1 0 1 1 1 1 1 1 1]
 [1 1 1 0 1 1 1 1 1 1]
 [0 1 1 1 0 1 0 0 0 0]
 [1 1 0 1 1 0 0 1 0 0]
 [0 1 0 1 1 0 0 0 1 0]
 [0 1 0 1 1 0 1 1 0 0]
 [1 1 0 1 1 0 0 0 0 0]]
```



- We constructed this adjacency matrix from the last network we created of shortest paths where “P20933” is the tail, “Q15303” is the head, and “0.6889442” is the weight cost.

**Adjacency Matrix with Unique Keys**

|        |        |        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| P46108 | 0      | 1      | 0      | 1      | 1      | 0      | 1      | 0      | 0      | 1      |
| Q9Y3A3 | 1      | 0      | 1      | 1      | 1      | 1      | 1      | 1      | 1      | 1      |
| O15259 | 0      | 1      | 0      | 1      | 1      | 1      | 0      | 0      | 0      | 0      |
| P20933 | 1      | 1      | 1      | 0      | 1      | 1      | 1      | 1      | 1      | 1      |
| Q15303 | 1      | 1      | 1      | 1      | 0      | 1      | 1      | 1      | 1      | 1      |
| P22681 | 0      | 1      | 1      | 1      | 1      | 0      | 1      | 0      | 0      | 0      |
| Q6ZU80 | 1      | 1      | 0      | 1      | 1      | 1      | 0      | 0      | 1      | 0      |
| Q68CZ1 | 0      | 1      | 0      | 1      | 1      | 0      | 0      | 0      | 1      | 0      |
| P04637 | 0      | 1      | 0      | 1      | 1      | 0      | 1      | 1      | 0      | 0      |
| Q96ST8 | 1      | 1      | 0      | 1      | 1      | 0      | 0      | 0      | 0      | 0      |
|        | P46108 | Q9Y3A3 | O15259 | P20933 | Q15303 | P22681 | Q6ZU80 | Q68CZ1 | P04637 | Q96ST8 |

- **Q9Y3A3** is the hub protein of this sub-network.



## - Conclusion

In conclusion, the analysis of protein-protein interactions is a critical step toward gaining a comprehensive understanding of biological systems and the formation of complex biological networks. This research paper utilized computational techniques to analyze the protein-protein interaction network, including utilizing the NetworkX python package, and UniProt database, and visualizing the results with matplotlib and seaborn. The tasks performed in this research paper provided valuable insights into the organization and regulation of protein-protein interactions, including finding the shortest path between two proteins, identifying directly connected proteins, ranking proteins based on their degree, converting UniProt IDs to gene names, and converting the graph to an unweighted graph. This study is a testament to the power of computational analysis in understanding the complexities of cellular function and predicting potential therapeutics.



## - References

- *Degree distribution. indegree and outdegree.* (2011, October). Retrieved February 2023, from [https://www.researchgate.net/figure/Degree-Distribution-Indegree-and-outdegree-distribution-averaged-over-5-time-points-The\\_fig2\\_51740396](https://www.researchgate.net/figure/Degree-Distribution-Indegree-and-outdegree-distribution-averaged-over-5-time-points-The_fig2_51740396)
- Protein-Protein Interaction(PPI). (2012, December). Retrieved February 2023, from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3531279/>
- Biological network and interactions effects. (2013, April). Retrieved February 2023, from <https://genomemedicine.biomedcentral.com/articles/10.1186/gm441>
- Using graph theory to analyze biological networks. (2011, April). Retrieved February 2023, from <https://biodatamining.biomedcentral.com/articles/10.1186/1756-0381-4-10>
- Controllability analysis of the directed human protein interaction network identifies disease genes and drug targets. (2016, April). Retrieved February 2023, from <https://www.pnas.org/doi/10.1073/pnas.1603992113>
- Jingyu Hou. (2017). Protein Function Prediction from Functional Connectivity. Retrieved February 2023, from <https://www.sciencedirect.com/topics/mathematics/protein-interaction-network>
- Diana Ekman, Sara Light, Åsa K Björklund, and Arne Elofsson. (2006, June). properties characterize the hub proteins of the protein-protein interaction network of *Saccharomyces cerevisiae*. Properties characterize the hub proteins of the protein-protein interaction network of *Saccharomyces cerevisiae*. Retrieved February 2023, from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1779539/#:~:text=Protein%20classification%20in%20the%20network&text=Hubs%20are%20defined%20in%20DIP,the%20rest%20are%20inte>
- Yutaka Hata & Junko Iida. (2009). Scaffold Protein. Retrieved February 2023, from [https://link.springer.com/referenceworkentry/10.1007/978-3-540-29678-2\\_5231#author-information](https://link.springer.com/referenceworkentry/10.1007/978-3-540-29678-2_5231#author-information)





## - Members Contribution

- Code Implementation
  - Mahmoud Yaser Salman
  - Omar Saad EL-Gharabawy
- Project Research
  - Abdelrahman Ali Farouk
  - Nevein Mohamed Ayman
- Report
  - Abdelrahman Ali Farouk
  - Mahmoud Yaser Salman
  - Nevein Mohamed Ayman
- Slides
  - Omar Saad El-Gharabawy

