

QuartPAC: Identifying mutational clusters while utilizing protein quaternary structural data

Gregory Ryslik
Genentech
gregory.ryslík@yale.edu

Yuwei Cheng
Yale University
yuwei.cheng@yale.edu

Hongyu Zhao
Yale University
hongyu.zhao@yale.edu

October 27, 2020

Abstract

The **QuartPAC** package is designed to identify mutated amino acid hotspots while accounting for protein quaternary structure. It is meant to work in conjunction with the **iPAC** [Ryslik and Zhao, 2012b], **GraphPAC** [Ryslik and Zhao, 2012a] and **SpacePAC** [Ryslik and Zhao, 2013] packages already available through Bioconductor. Specifically, the package takes as input the quaternary protein structure as well as the mutational data for each subunit of the assembly. It then maps the mutational data onto the protein and performs the algorithms described in **iPAC**, **GraphPAC** and **SpacePAC** to report the statistically significant clusters. By integrating the quaternary structure, **QuartPAC** may identify additional clusters that only become apparent when the different protein subunits are considered together.

1 Introduction

Recent advances in oncogenic pharmacology [Croce, 2008] have led to the creation of a variety of methods that attempt to identify mutational hotspots as these hotspots are often indicative of driver mutations [Wagner, 2007, Zhou et al., 2008, Ye et al., 2010]. Three recent methods, **iPAC**, **GraphPAC** and **SpacePAC** provide approaches to identify such hotspots while accounting for protein tertiary structure. While it has been shown that these mutations provide an improvement over linear clustering methods, [Ryslik et al., 2013, 2014b,a], they nevertheless consider only tertiary structure. **QuartPAC**, preprocesses the entire assembly structure in order to be able to accurately run these approaches on the quaternary protein unit. This allows for the identification of

additional mutational clusters that may otherwise be missed if only one protein subunit is considered at a time.

In order to run **QuartPAC**, four sources of data are required:

- The amino acid sequence of the protein which is obtained from the UniProt database (uniprot.org in FASTA format).
- The protein tertiary subunit information which is obtained from the .pdb file from PDB.org
- The quaternary structural information for the entire assembly which is obtained from the .pdb1 file from PDB.org
- The somatic mutation data which is obtained from the Catalogue of Somatic Mutations in Cancer (<http://cancer.sanger.ac.uk/cancergenome/projects/cosmic/>).

In order to map the mutations onto the protein quaternary structure, an alignment must be performed. For each uniprot within the assembly, mutational data must be provided. The data is in the format of $m \times n$ matrices for every subunit. A “1” in the (i, j) element indicates that residue j for individual i has a mutation while a “0” indicates no mutation. To be compatible with this software, please ensure that your mutation matrices have R column headings of $V1, V2, \dots, Vn$. Only missense mutations are currently supported, indels in the amino acid sequence are not. Sample mutational data are included in this package as textfiles in the *extdata* folder.

It is worth noting that there does not exist any one individual source to obtain mutational data. One common resource is the COSMIC database <http://cancer.sanger.ac.uk/cancergenome/projects/cosmic/>. The easiest way to obtain mutational data for many proteins is to load the the COSMIC database on a local sql server and then query the database for the protein of interest. It is important to restrict your query to whole gene screens or whole genome studies to prevent specific mutations from being selectively chosen (and thus violating the uniformity assumption that **iPAC**, **GraphPAC**, and **SpacePAC** rely upon).

Should you find a bug, or wish to contribute to the code base, please contact the author.

2 Identifying Clusters and Viewing the Remapping

The general principle of **QuartPAC** is that we preprocess the data into a format that can be recognized by **iPAC**, **GraphPAC** and **SpacePAC**. Most of this is automated and all that is needed is to point the algorithm to the mutational and structural data. **QuartPAC** will then reorganize the data, execute the cluster finding algorithms and report the results. The clusters are reported by

serial number. As each serial number is unique in the assembly, the user can then map each serial number to the exact atom of interest in the structure.

Below we run the algorithm with no multiple comparison adjustment. We do this to ensure that some clusters are found for each method. We also note that for **iPAC** and **GraphPAC**, if a multiple comparison adjustment is used and no clusters are found significant, the methods will show a null value. For **SpacePAC**, as there is no multiple comparison adjustment needed, the most significant clusters are always shown, regardless of the p-value. This behavior follows the functionality of the previous three packages, so users familiar with the tertiary algorithms will find the results directly comparable.

For more information on the output, please see the **iPAC**, **GraphPAC**, and **SpacePAC** packages as the output is similar. The main difference is that the amino acid numbers now refer to the serial numbers within the *.pdb1 file.

Code Example 1: Running QuartPAC.

```
> library(QuartPAC)
> #read the mutational data
> mutation_files <- list(
+ system.file("extdata","HFE_Q30201_MutationOutput.txt", package = "QuartPAC"),
+ system.file("extdata","B2M_P61769_MutationOutput.txt", package = "QuartPAC")
+ )
> uniprots <- list("Q30201","P61769")
> mutation.data <- getMutations(mutation_files = mutation_files, uniprots = uniprots)
> #read the pdb file
> pdb.location <- "https://files.rcsb.org/view/1A6Z.pdb"
> assembly.location <- "https://files.rcsb.org/download/1A6Z.pdb1"
> structural.data <- makeAlignedSuperStructure(pdb.location, assembly.location)
> #Perform Analysis
> #We use a very high alpha level here with no multiple comparison adjustment
> #to make sure that each method provides shows a result.
> #Lower alpha cut offs are typically used.
> quart_results <- quartCluster(mutation.data, structural.data, perform.ipac = "Y",
+                               perform.graphpac = "Y", perform.spacepac = "Y",
+                               create.map = "Y",alpha = .3,MultComp = "None",
+                               Graph.Title = "MDS Mapping to 1D Space",
+                               radii.vector = c(1:3))
```

We observe that the MDS remapping plot provided by **QuartPAC** is done automatically if the *create.map* parameter is set to “Y”. The plot is shown in Figure 1 below.

For the **GraphPAC** approach, the linear “Jump Plot” (see the **GraphPAC** package for more details and interpretation) has been implemented and is shown in Figure 2 below. Feel free to contact the author if you want to assist in porting other graphing functionality.

With regards to **SpacePAC**, as there is no remapping from 3D to 1D space, a plotting option that shows the protein in its folded state is presented in Section 4.

MDS Mapping to 1D Space

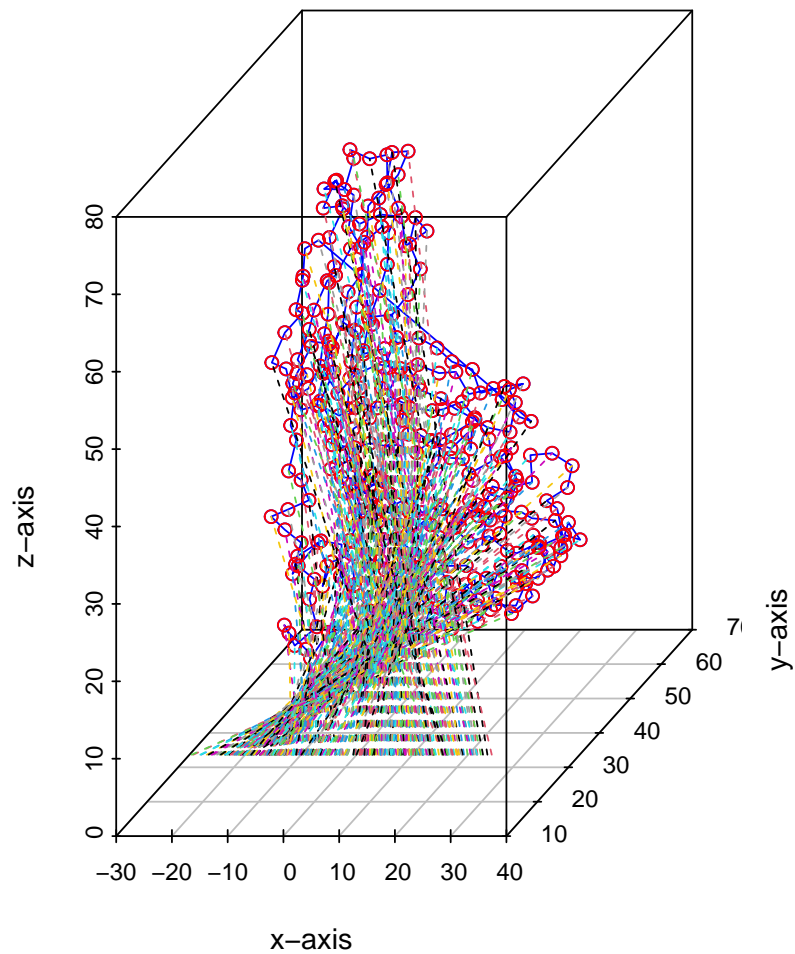


Figure 1: Remapping performed by iPAC.

Code Example 2: Plotting the GraphPAC candidate path.

```
> Plot.Protein.Linear(quart_results$graphpac$candidate.path, colCount = 10,  
+                    title = "Protein Reordering to 1D Space via GraphPAC")
```

Protein Reordering to 1D Space via GraphPAC

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 2 | 13 | 19 | 29 | 35 | 43 | 53 | 65 | 73 | 84 |
| 92 | 96 | 101 | 107 | 116 | 121 | 129 | 137 | 141 | 149 |
| 155 | 163 | 174 | 183 | 188 | 196 | 200 | 212 | 219 | 227 |
| 235 | 244 | 252 | 263 | 270 | 281 | 293 | 301 | 311 | 316 |
| 322 | 333 | 344 | 351 | 360 | 367 | 378 | 385 | 392 | 406 |
| 411 | 416 | 421 | 426 | 431 | 437 | 443 | 452 | 460 | 474 |
| 479 | 488 | 496 | 501 | 510 | 516 | 524 | 533 | 537 | 551 |
| 559 | 569 | 577 | 588 | 595 | 602 | 610 | 621 | 635 | 642 |
| 650 | 658 | 667 | 675 | 685 | 693 | 703 | 709 | 718 | 727 |
| 733 | 743 | 750 | 758 | 767 | 774 | 782 | 790 | 794 | 800 |
| 809 | 817 | 826 | 831 | 839 | 847 | 853 | 860 | 869 | 873 |
| 885 | 899 | 908 | 920 | 924 | 936 | 944 | 948 | 957 | 965 |
| 975 | 983 | 992 | 1003 | 1009 | 1016 | 1024 | 1031 | 1039 | 1047 |
| 1061 | 1072 | 1077 | 1082 | 1091 | 1098 | 1109 | 1114 | 1128 | 1135 |
| 1142 | 1151 | 1159 | 1168 | 1182 | 1191 | 1202 | 1212 | 1221 | 1229 |
| 1240 | 1245 | 1256 | 1265 | 1273 | 1284 | 1289 | 1301 | 1309 | 1318 |
| 1329 | 1337 | 1343 | 1350 | 1355 | 1364 | 1372 | 1381 | 1390 | 1398 |
| 1406 | 1415 | 1423 | 1427 | 1432 | 1436 | 1443 | 1451 | 1459 | 1468 |
| 1477 | 1484 | 1491 | 1498 | 1506 | 1513 | 1522 | 1529 | 1536 | 1546 |
| 1556 | 1563 | 1570 | 1576 | 1582 | 1589 | 1596 | 1603 | 1611 | 1622 |
| 1628 | 1639 | 1644 | 1652 | 1660 | 1672 | 1684 | 1691 | 1700 | 1708 |
| 1716 | 1723 | 1731 | 1740 | 1754 | 1762 | 1771 | 1779 | 1788 | 1797 |
| 1804 | 1812 | 1820 | 1825 | 1834 | 1843 | 1854 | 1863 | 1870 | 1879 |
| 1887 | 1894 | 1902 | 1909 | 1917 | 1921 | 1929 | 1933 | 1940 | 1952 |
| 1961 | 1965 | 1979 | 1987 | 1994 | 2002 | 2007 | 2014 | 2021 | 2028 |
| 2032 | 2041 | 2050 | 2059 | 2070 | 2082 | 2089 | 2095 | 2104 | 2111 |
| 2120 | 2130 | 2137 | 2141 | 2149 | 2157 | 2166 | 2173 | 2181 | 2189 |
| 2196 | 2204 | 2220 | 2228 | 2237 | 2248 | 2255 | 2262 | 2271 | 2279 |
| 2288 | 2295 | 2307 | 2313 | 2324 | 2334 | 2341 | 2346 | 2355 | 2363 |
| 2367 | 2376 | 2382 | 2390 | 2401 | 2409 | 2417 | 2423 | 2435 | 2442 |
| 2448 | 2452 | 2463 | 2473 | 2480 | 2486 | 2494 | 2502 | 2511 | 2518 |
| 2526 | 2534 | 2542 | 2551 | 2559 | 2563 | 2572 | 2583 | 2591 | 2600 |
| 2605 | 2612 | 2621 | 2631 | 2637 | 2645 | 2653 | 2659 | 2670 | 2676 |
| 2685 | 2693 | 2707 | 2713 | 2724 | 2736 | 2744 | 2752 | 2764 | 2776 |
| 2783 | 2792 | 2803 | 2810 | 2817 | 2824 | 2833 | 2838 | 2846 | 2855 |
| 2867 | 2872 | 2878 | 2889 | 2896 | 2904 | 2914 | 2921 | 2928 | 2936 |
| 2942 | 2951 | 2958 | 2967 | 2975 | 2982 | 2991 | 3005 | 3013 | 3024 |
| 3032 | | | | | | | | | |

Figure 2: Remapping performed by GraphPAC.

3 Using the Output

Now that we have the results, suppose that we wanted to visualize what the clusters are. For example, we see that the first cluster under the **SpacePAC** method for the optimal combination has two spheres. One sphere is centered at the atom with serial number 1265 and one sphere is centered at the atom with serial number 367.

To see where this matches we can query the *structural.data* list.

Code Example 3: Finding the residue of interest using the SpacePAC method.

```
> #look at the results for the optimal sphere combinations under the SpacePAC approach
> #For clarity we only look at columns 3 - 8 which show the sphere centers.
> quart_results$spacepac$optimal.sphere[,3:8]
```

| | Center1 | Center2 | Start1 | End1 | Start2 | End2 |
|----|---------|---------|--------|------|--------|------|
| 1 | 1265 | 367 | 1265 | 1265 | 367 | 367 |
| 2 | 2166 | 367 | 2166 | 2166 | 367 | 367 |
| 3 | 2295 | 367 | 2295 | 2295 | 367 | 367 |
| 4 | 2166 | 1265 | 2166 | 2166 | 1265 | 1265 |
| 5 | 2401 | 367 | 2401 | 2401 | 367 | 367 |
| 6 | 2295 | 1265 | 2295 | 2295 | 1265 | 1265 |
| 7 | 2583 | 367 | 2583 | 2583 | 367 | 367 |
| 8 | 2401 | 1265 | 2401 | 2401 | 1265 | 1265 |
| 9 | 2295 | 2166 | 2295 | 2295 | 2166 | 2166 |
| 10 | 2659 | 367 | 2659 | 2659 | 367 | 367 |
| 11 | 2583 | 1265 | 2583 | 2583 | 1265 | 1265 |
| 12 | 2401 | 2166 | 2401 | 2401 | 2166 | 2166 |
| 13 | 2846 | 367 | 2846 | 2846 | 367 | 367 |
| 14 | 2659 | 1265 | 2659 | 2659 | 1265 | 1265 |
| 15 | 2583 | 2166 | 2583 | 2583 | 2166 | 2166 |
| 16 | 2401 | 2295 | 2401 | 2401 | 2295 | 2295 |
| 17 | 2846 | 1265 | 2846 | 2846 | 1265 | 1265 |
| 18 | 2659 | 2166 | 2659 | 2659 | 2166 | 2166 |
| 19 | 2583 | 2295 | 2583 | 2583 | 2295 | 2295 |
| 20 | 2846 | 2166 | 2846 | 2846 | 2166 | 2166 |
| 21 | 2659 | 2295 | 2659 | 2659 | 2295 | 2295 |
| 22 | 2583 | 2401 | 2583 | 2583 | 2401 | 2401 |
| 23 | 2846 | 2295 | 2846 | 2846 | 2295 | 2295 |
| 24 | 2659 | 2401 | 2659 | 2659 | 2401 | 2401 |
| 25 | 2846 | 2401 | 2846 | 2846 | 2401 | 2401 |
| 26 | 2659 | 2583 | 2659 | 2659 | 2583 | 2583 |
| 27 | 2846 | 2583 | 2846 | 2846 | 2583 | 2583 |
| 28 | 2846 | 2659 | 2846 | 2846 | 2659 | 2659 |

```
> #Find the atom with serial number 1265
> required.row <- which(structural.data$aligned_structure$serial == 1265)
> #show the information for that atom
> structural.data$aligned_structure[required.row,]
```

| | recordName | serial | atom | altLoc | resName | chainID | resSeq | iCode | xCoord | yCoord |
|----|------------|--------|------|--------|---------|---------|--------|-------|--------|--------|
| 1: | ATOM | 1265 | CA | | ASN | A | 157 | | 5.743 | 52.485 |

```

      zCoord occupancy tempFactor element charge   UNP dbref protomer absPos
1: 13.919      1      42.28      C      Q30201      1      1      154
      canonical_pos
1:      179
>

```

Similarly, suppose you wanted to look at the **iPAC** results. The first cluster goes from serial 2583 and ends at 2846. To get all the residue information for that block, we can do the following:

Code Example 4: Finding the residue of interest using the iPAC method.

```

> #look at the results for the first cluster shown by the ipac method
> quart_results$ipac

      AA_in_Cluster serial_start serial_end number   p_value
V254           32      2583      2846      3 0.03093808
V172           22      2659      2846      2 0.04285346
V274           55      2401      2846      4 0.05244236
V254           52      2166      2583      4 0.07721776
V274           14      2295      2401      2 0.08306777
V254           37      2295      2583      3 0.09028823
V172           62      2166      2659      5 0.10970551
V274           29      2166      2401      3 0.12480796
V180          304       367      2846      8 0.12630378
V172          283       367      2659      7 0.16003397
V172           47      2295      2659      4 0.16898693
V278           68      2295      2846      5 0.19414336
V303           83      2166      2846      6 0.24825087
V180          222       367      2166      3 0.25497859

> #Find the atoms with serial numbers within the range of 2583 to 2846
> required.rows <- which(structural.data$aligned_structure$serial %in% (2583:2846))
> #show the information for those atoms
> structural.data$aligned_structure[required.rows,]

      recordName serial atom altLoc resName chainID resSeq iCode xCoord yCoord
1:      ATOM      2583  CA      ILE      B      46      12.885 19.924
2:      ATOM      2591  CA      GLU      B      47      12.538 18.636
3:      ATOM      2600  CA      LYS      B      48       9.370 18.878
4:      ATOM      2605  CA      VAL      B      49       9.120 22.638
5:      ATOM      2612  CA      GLU      B      50       7.189 24.889
6:      ATOM      2621  CA      HIS      B      51       7.867 28.493
7:      ATOM      2631  CA      SER      B      52       5.976 31.273
8:      ATOM      2637  CA      ASP      B      53       7.479 32.937
9:      ATOM      2645  CA      LEU      B      54      10.283 35.432
10:     ATOM      2653  CA      SER      B      55       8.837 38.923
11:     ATOM      2659  CA      PHE      B      56       9.783 42.469
12:     ATOM      2670  CA      SER      B      57       8.257 45.640
13:     ATOM      2676  CA      LYS      B      58       8.243 49.331
14:     ATOM      2685  CA      ASP      B      59      11.745 49.858

```

| | | | | | | | | |
|-----|------|------|----|-----|---|----|--------|--------|
| 15: | ATOM | 2693 | CA | TRP | B | 60 | 13.026 | 47.034 |
| 16: | ATOM | 2707 | CA | SER | B | 61 | 13.661 | 44.800 |
| 17: | ATOM | 2713 | CA | PHE | B | 62 | 12.750 | 41.139 |
| 18: | ATOM | 2724 | CA | TYR | B | 63 | 10.369 | 38.996 |
| 19: | ATOM | 2736 | CA | LEU | B | 64 | 9.841 | 35.241 |
| 20: | ATOM | 2744 | CA | LEU | B | 65 | 7.561 | 33.020 |
| 21: | ATOM | 2752 | CA | TYR | B | 66 | 8.522 | 29.543 |
| 22: | ATOM | 2764 | CA | TYR | B | 67 | 5.960 | 27.185 |
| 23: | ATOM | 2776 | CA | THR | B | 68 | 5.265 | 23.560 |
| 24: | ATOM | 2783 | CA | GLU | B | 69 | 2.275 | 21.616 |
| 25: | ATOM | 2792 | CA | PHE | B | 70 | 3.018 | 20.246 |
| 26: | ATOM | 2803 | CA | THR | B | 71 | 1.565 | 19.125 |
| 27: | ATOM | 2810 | CA | PRO | B | 72 | 3.166 | 20.559 |
| 28: | ATOM | 2817 | CA | THR | B | 73 | 3.655 | 18.871 |
| 29: | ATOM | 2824 | CA | GLU | B | 74 | 5.166 | 19.946 |
| 30: | ATOM | 2833 | CA | LYS | B | 75 | 8.589 | 18.374 |
| 31: | ATOM | 2838 | CA | ASP | B | 76 | 8.895 | 19.872 |
| 32: | ATOM | 2846 | CA | GLU | B | 77 | 10.881 | 23.063 |

| | recordName | serial | atom | altLoc | resName | chainID | resSeq | iCode | xCoord | yCoord | zCoord | occupancy | tempFactor | element | charge | UNP | dbref | protomer | absPos |
|-----|------------|--------|------|--------|---------|---------|--------|-------|--------|--------|--------|-----------|------------|---------|--------|-----|-------|----------|--------|
| 1: | | 45.964 | | 1 | | | 59.61 | | C | P61769 | 2 | 1 | 318 | | | | | | |
| 2: | | 42.427 | | 1 | | | 75.66 | | C | P61769 | 2 | 1 | 319 | | | | | | |
| 3: | | 40.348 | | 1 | | | 82.60 | | C | P61769 | 2 | 1 | 320 | | | | | | |
| 4: | | 40.986 | | 1 | | | 72.00 | | C | P61769 | 2 | 1 | 321 | | | | | | |
| 5: | | 38.587 | | 1 | | | 65.40 | | C | P61769 | 2 | 1 | 322 | | | | | | |
| 6: | | 37.552 | | 1 | | | 59.43 | | C | P61769 | 2 | 1 | 323 | | | | | | |
| 7: | | 35.791 | | 1 | | | 54.96 | | C | P61769 | 2 | 1 | 324 | | | | | | |
| 8: | | 32.724 | | 1 | | | 54.99 | | C | P61769 | 2 | 1 | 325 | | | | | | |
| 9: | | 33.143 | | 1 | | | 43.17 | | C | P61769 | 2 | 1 | 326 | | | | | | |
| 10: | | 32.873 | | 1 | | | 43.06 | | C | P61769 | 2 | 1 | 327 | | | | | | |
| 11: | | 33.932 | | 1 | | | 36.56 | | C | P61769 | 2 | 1 | 328 | | | | | | |
| 12: | | 35.371 | | 1 | | | 40.60 | | C | P61769 | 2 | 1 | 329 | | | | | | |
| 13: | | 34.385 | | 1 | | | 42.56 | | C | P61769 | 2 | 1 | 330 | | | | | | |
| 14: | | 35.795 | | 1 | | | 41.40 | | C | P61769 | 2 | 1 | 331 | | | | | | |
| 15: | | 33.576 | | 1 | | | 34.49 | | C | P61769 | 2 | 1 | 332 | | | | | | |
| 16: | | 36.581 | | 1 | | | 35.68 | | C | P61769 | 2 | 1 | 333 | | | | | | |
| 17: | | 36.218 | | 1 | | | 33.38 | | C | P61769 | 2 | 1 | 334 | | | | | | |
| 18: | | 38.234 | | 1 | | | 42.58 | | C | P61769 | 2 | 1 | 335 | | | | | | |
| 19: | | 38.323 | | 1 | | | 36.62 | | C | P61769 | 2 | 1 | 336 | | | | | | |
| 20: | | 40.370 | | 1 | | | 38.86 | | C | P61769 | 2 | 1 | 337 | | | | | | |
| 21: | | 41.587 | | 1 | | | 49.21 | | C | P61769 | 2 | 1 | 338 | | | | | | |
| 22: | | 43.115 | | 1 | | | 54.28 | | C | P61769 | 2 | 1 | 339 | | | | | | |
| 23: | | 44.155 | | 1 | | | 59.26 | | C | P61769 | 2 | 1 | 340 | | | | | | |
| 24: | | 45.397 | | 1 | | | 66.25 | | C | P61769 | 2 | 1 | 341 | | | | | | |
| 25: | | 48.852 | | 1 | | | 59.77 | | C | P61769 | 2 | 1 | 342 | | | | | | |
| 26: | | 52.163 | | 1 | | | 67.36 | | C | P61769 | 2 | 1 | 343 | | | | | | |
| 27: | | 55.294 | | 1 | | | 68.86 | | C | P61769 | 2 | 1 | 344 | | | | | | |
| 28: | | 58.669 | | 1 | | | 73.63 | | C | P61769 | 2 | 1 | 345 | | | | | | |
| 29: | | 61.984 | | 1 | | | 84.25 | | C | P61769 | 2 | 1 | 346 | | | | | | |
| 30: | | 61.405 | | 1 | | | 87.62 | | C | P61769 | 2 | 1 | 347 | | | | | | |


```

31: 57.911      1      77.81      C      P61769      2      1      348
32: 57.189      1      68.89      C      P61769      2      1      349
  zCoord occupancy tempFactor element charge  UNP dbref protomer absPos
  canonical_pos
1:          66
2:          67
3:          68
4:          69
5:          70
6:          71
7:          72
8:          73
9:          74
10:         75
11:         76
12:         77
13:         78
14:         79
15:         80
16:         81
17:         82
18:         83
19:         84
20:         85
21:         86
22:         87
23:         88
24:         89
25:         90
26:         91
27:         92
28:         93
29:         94
30:         95
31:         96
32:         97
  canonical_pos

```

As the **GraphPAC** results are in the same format as the **iPAC** results, the approach for identifying clusters in those atoms is identical as in the example above.

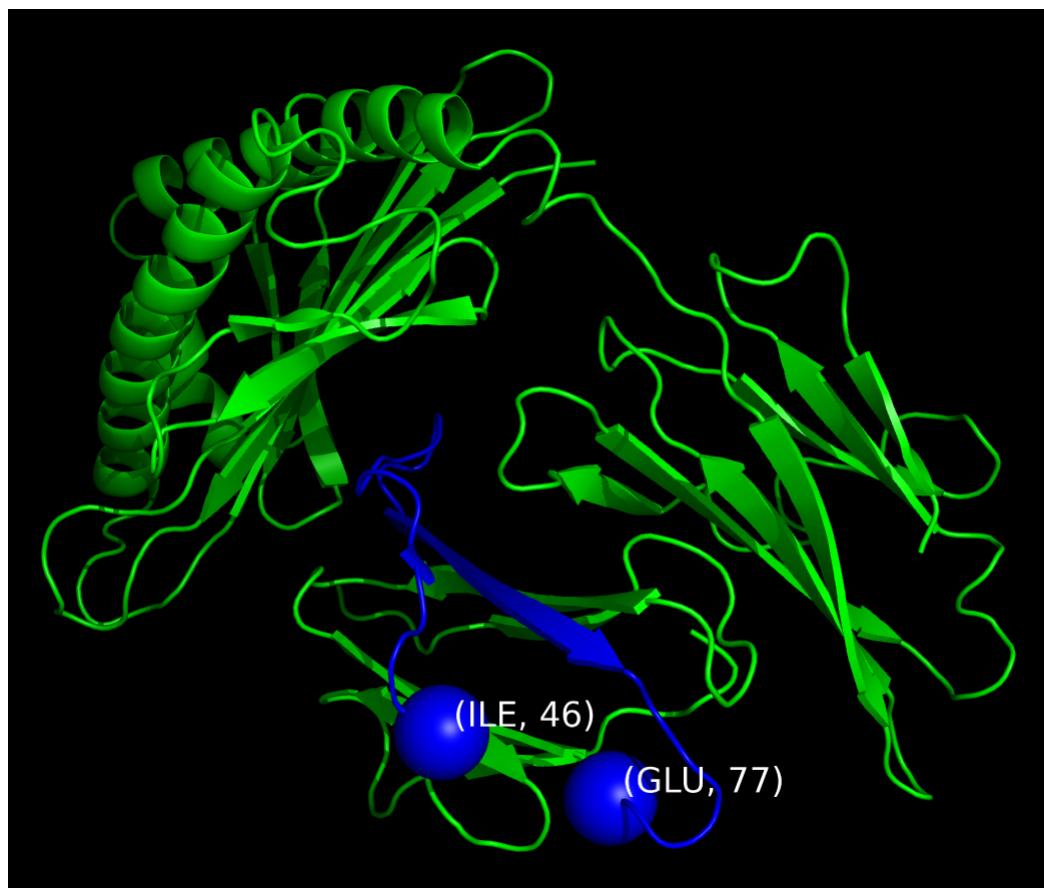
4 Visualizing the Results

Once you have the serial numbers of interest, you can then view the results in any pdb visualization application of your choice. One common option is to use the *PyMOL* software package [Schrödinger, LLC, 2010]. While it is not the purpose of this vignette to teach the reader *PyMOL* syntax, we present the following simplistic example and the resulting figure for reference. It will color

the first cluster outputted by the **iPAC** method, residues with serial numbers 2583-2846 in blue. The chain and resSeq information provided in Example 4 is used as below.

Code Example 5: PyMOL sample code

```
-----  
hide all  
show cartoon,  
show spheres, ///b/46/ca  
show spheres, ///b/77/ca  
color blue, ///b/46-77  
  
label c. B and n. CA and i. 46, "(%s, %s)" % (resn, resi)  
label c. B and n. CA and i. 77, "(%s, %s)" % (resn, resi)  
set label_position, (3,2,10)
```



References

- Carlo M Croce. Oncogenes and cancer. *The New England Journal of Medicine*, 358(5):502–511, January 2008. ISSN 1533-4406. doi: 10.1056/NEJMra072367. URL <http://www.ncbi.nlm.nih.gov/pubmed/18234754>. PMID: 18234754.
- Gregory Ryslik and Hongyu Zhao. *GraphPAC: Identification of Mutational Clusters in Proteins via a Graph Theoretical Approach.*, 2012a. R package version 1.6.0.
- Gregory Ryslik and Hongyu Zhao. *iPAC: Identification of Protein Amino acid Clustering*, 2012b. R package version 1.8.0.
- Gregory Ryslik and Hongyu Zhao. *SpacePAC: Identification of Mutational Clusters in 3D Protein Space via Simulation.*, 2013. R package version 1.2.0.
- Gregory A Ryslik, Yuwei Cheng, Kei-Hoi Cheung, Yorgo Modis, and Hongyu Zhao. Utilizing protein structure to identify non-random somatic mutations. *BMC Bioinformatics*, 14(1):190, 2013. ISSN 1471-2105. doi: 10.1186/1471-2105-14-190. URL <http://www.biomedcentral.com/1471-2105/14/190>.
- Gregory A Ryslik, Yuwei Cheng, Kei-Hoi Cheung, Robert D Bjornson, Daniel Zelterman, Yorgo Modis, and Hongyu Zhao. A spatial simulation approach to account for protein structure when identifying non-random somatic mutations. *BMC Bioinformatics*, 15(1):231, 2014a. ISSN 1471-2105. doi: 10.1186/1471-2105-15-231. URL <http://www.biomedcentral.com/1471-2105/15/231>.
- Gregory A Ryslik, Yuwei Cheng, Kei-Hoi Cheung, Yorgo Modis, and Hongyu Zhao. A graph theoretic approach to utilizing protein structure to identify non-random somatic mutations. *BMC Bioinformatics*, 15(1):86, 2014b. ISSN 1471-2105. doi: 10.1186/1471-2105-15-86. URL <http://www.biomedcentral.com/1471-2105/15/86>.
- Schrödinger, LLC. The PyMOL molecular graphics system, version 1.3r1. PyMOL, The PyMOL Molecular Graphics System, Version 1.3, Schrödinger, LLC., August 2010.
- A. Wagner. Rapid detection of positive selection in genes and genomes through variation clusters. *Genetics*, 176(4):2451–2463, August 2007. ISSN 0016-6731. doi: 10.1534/genetics.107.074732. URL <http://www.genetics.org/cgi/doi/10.1534/genetics.107.074732>.
- Jingjing Ye, Adam Pavlicek, Elizabeth A Lunney, Paul A Rejto, and Chi-Hse Teng. Statistical method on nonrandom clustering with application to somatic mutations in cancer. *BMC Bioinformatics*, 11(1):11, 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-11. URL <http://www.biomedcentral.com/1471-2105/11/11>.

Tong Zhou, Peter J. Enyeart, and Claus O. Wilke. Detecting clusters of mutations. *PLoS ONE*, 3(11):e3765, November 2008. ISSN 1932-6203. doi: 10.1371/journal.pone.0003765. URL <http://dx.plos.org/10.1371/journal.pone.0003765>.