Development of Software Package for Determining
Protein Titration Properties

By

Kaila Bennett, Amitoj Chopra, Jesse Johnson, Enrico Sagullo

Mentor: Dimitrios Morikis
Bioengineering 175-Senior Design
5/24/2010
Acknowledgements

We would like to thank Dr. Dimitrios Morikis for all his help and guidance through out this entire project, explaining the theoretical significance of the project. We would like to thank Mr. Chris Kieslich with is help in explaining how to code in R programming. We would also like to thank Dr. Jerome Schultz, and Mrs. Hung Xu for their help through out this project as well. This project would not been possible without all the people mentioned.
# Table of Contents

ACKNOWLEDGEMENTS .................................................................................................................. 2

ABSTRACT ...................................................................................................................................... 5

PROJECT OBJECTIVES .................................................................................................................. 6

BACKGROUND .................................................................................................................................. 7

I. Rationale of the Project .................................................................................................................. 7

II. Business Opportunity Prospects ................................................................................................ 10

PRIOR ART REVIEW ....................................................................................................................... 10

FUNCTIONAL AND PERFORMANCE SPECIFICATIONS ............................................................... 12

BLOCK DIAGRAM OF PROBLEM .................................................................................................... 13

EVOLUTION OF FINAL DESIGN .................................................................................................... 13

Calculating pKₐ values ...................................................................................................................... 13

Interaction Energies ......................................................................................................................... 14

Statistical Analysis for titratable groups and apparent pKₐ Calculation ........................................... 14

Titration Curves ............................................................................................................................... 15

DETAILED DESCRIPTION OF FINAL SOLUTION .......................................................................... 17

Titration Curves ............................................................................................................................... 17

GUI function in R ............................................................................................................................ 18

MATERIAL SELECTION .................................................................................................................. 19

METHOD OF PROTOTYPING DISCUSSION .................................................................................. 20

PERFORMANCE TESTING PROTOCOL DISCUSSION .................................................................. 22

RESULTS ........................................................................................................................................ 23
Abstract

Many biological processes are pH-dependent, which means that they are modulated by environmental charge changes. Important biological processes include, catalysis, binding, and molecular switching. Thus, the study of site-specific (at amino acid level) and global protein titration properties is important to elucidate function. Protein charge characteristics can be examined by determining protein ionization constants commonly referred to as pKa values. A pKa indicates the point in which there is an equal probability of ionization states for a single amino acid. Recently, many groups have tried to determine apparent (sum of individual amino acid residue pKa values) pKa values of a protein, by applying pKa algorithms to protein X-ray structure [1]. Indeed, these methods have proven to be beneficial in gaining insight into the overall charge characteristics of a protein, however, these methods have proven to lack accuracy.

A computational method was used for the calculation of pKa values, and generation of protein titration curves, so that 50% protonation could be visualized graphically. Furthermore, a computational method was used, for not only a high degree of accuracy, and efficiency in calculation, but also for user ease-ability. A software package was created using the Adaptive Poisson-Boltzmann Solver, and parse forcefield. Results showed a high degree of correlation between our values and experimentally determined values, as well as other existing software.

Results also suggest that run time could be greatly reduced by using either the linearized Poisson-Boltzmann equation or Coulomb’s Law. Development of a accurate and efficient pKa calculations package could aid in fully understanding the importance that charge plays in protein function.
Project Objectives

The purpose of our project will be to write a script using the programming language R, which will take any PDB file and calculate the desired thermodynamic properties along with electrostatic potentials, and the incorporation of other programs to create a local and portable convenience package for any to use. This convenience package will be divided into two phases corresponding to winter and spring quarters. The winter phase will consist of first learning the programming language R with the help of graduate student Chris Kieslich and Dr. Thomas Girke, who will be holding an introduction workshop the end of January. We will be working on the incorporation of APBS, which will allow us to calculate the intrinsic pKa values. These pKa values will be calculated much like they were in BIEN 135, by the use and understanding of the thermodynamic cycle. Upon the completion of the winter phase, we will continue with algorithms that will calculate different thermodynamic properties. At this time, we will be able to take the “divide and conquer” approach and each group member will be responsible for a particular algorithm. The Spring Phase will cover the statistical approximation for the calculation of apparent pKa values using the intrinsic values. We will approach the algorithm by either the method of clustering or the Monte Carlo method. To further continue in the convenience of the total package, we will create a Graphical User Interface that will be to print out titration curves for each and all ionizable amino acids as well as the overall protein titration curve. It will be able to generate curves and tables that will help in the analysis of protein stability. In the case of complexes, binding free energies will also be printed out in an easy to read data sheet. The scripts will then be optimized to reduce their size, also to increase speed and efficiency, and to create a pleasant and effective
Background

I. Rationale of the Project

Proteins, nucleic acids, and other biomacromolecules, function depends in large part on the pH of the environment [2, 3]. Located within a protein sequence, are functional groups that have the capability to extract and donate protons, which relies heavily on pH. Protein charge properties, more specifically, proton transfer, participate in local interactions, such as charge-charge, charge-dipole, dipole-dipole interactions and in solvation-desolvation effects. Furthermore local charge is responsible for stability, binding, catalysis, and conformational switching. Biological processes like catalysis, solely occur due to ionizable amino acids, (Glutamic acid, Aspartic acid, Arginine, Lysine, Histidine, and Tyrosine), ability to extract and donate protons [2]. Proton transfer, is a dynamic process, and allows for a protein to have different ionization states. Ionization states are determined by charged amino acids, as well as polar amino acids, and will either be basic (+1) or acidic (-1) in nature. Thus, it becomes pivotal to elucidate the ionization states for a protein. Protein ionization, is determined through quantification of ionization constants (Ka), and is commonly expressed as pKa = -log (Ka) [4]. For a single amino acid, the pKa is the point at which it is 50 percent ionized, or where the single amino acid is 50 percent in its charged state. The Henderson-Hasselbach equation shows a theoretical relationship between pH of the
environment, and pKa for a single amino acid residue [Equation 1], and also works to show algorithmically, that pKa is in fact, the pH where the concentration of the protonated state is equal to that of the deprotonated state [3].

\[ \text{pH} = \text{pKa} + \log \frac{[A^-]}{[HA]} \] \[..............[1] \]

The Henderson-Hasselbach equation, however can only account for an isolated single ionizable group or model pKa\(^0\). To determine a proteins overall protonation, a more robust method must be used. Apparent pKa denoted as pKa\(^{app}\), allows for determining overall protonation of a protein or biomacromolecule. However, apparent pKa’s are difficult to quantify because they are interdependent on neighboring residues charge. To bypass this problem a hypothetical or intermediate pH independent quantity called the intrinsic pKa (pKa\(^{intr}\)) first has to be determined. Intrinsic pKa’s are in large part quantified using a thermodynamic cycle, and are defined as the pKa value that a titratable group would have if all other titratable groups where neutralized and fixed in space [2]. Equation 2 shows the relationship between model pKa and intrinsic pKa, where Z takes -1 for acids and +1 for bases, and \( \Delta \Delta G^{\text{env}} \) is the interaction of background charge and its environment \( k_B \) is the Boltzmann constant and T is the temperature. It should be noted that \( \Delta \Delta G^{\text{env}} \) takes into account both desolvation energies and background energies. These energies define how the intermediate protein environment impacts the intrinsic pKa value

\[ \text{pKa}^{\text{intr}} = \text{pKa}^0 - \frac{Z \Delta \Delta G^{\text{env}}}{2.303 k_B T} \] \[..............[2] \]
Intrinsic pKa can then be used to quantify apparent, via the relationship determined in Equation 3. $\Delta G_{\text{inter}}$ can only be extracted using a statistical analysis, namely a clustering method, and is defined as site-site interactions.

\[
pKa_{\text{app}} = pKa_{\text{intr}} + \frac{\Delta G_{\text{inter}}}{2.303k_B T} \quad [3]
\]

Due to the large magnitude of possible ionizable groups in a biomacromolecule, a clustering method is used to efficiently determine the most energetically favorable state of a protein. Clustering is accomplished by separating ionizable groups into intra-cluster, which are treated exactly and inter-cluster treated approximately [1].

Calculations of protein pKa have been performed for at least the past 20 years [6], while many improvements have been made to accurately calculate a proteins pKa, no current method has complete correlation to experimental pKa values. Many groups have investigated the accuracy of computational methods to calculate pKa, developing pKa calculation packages, some notable tools, include pKD [10], pK tool [1], H++ [JC et al 2005]. Many present pKa calculation packages generate free energies by using a Poisson Boltzmann equation, these free energies or obtained by incorporating such software as DelPhi II [5,7]. However, pKa calculation packages that solely rely on obtaining free energies via a Poisson-Boltzman equation have proven to inaccurate, and computationally inefficient. Thus, we introduce a pKa calculation package that calculates free energies via APBS (Adaptive Poisson-Boltzmann equation) that instead uses the Linearized Poisson-Boltzmann equation (Equation 4). The use of APBS in our software package has shown to be computationally efficient [9]. Furthermore our results suggest a high degree of correlation with experimentally determined pKa values (for individual titration curves) and has also shown to have a high degree of correlation with other established pKa
calculation tools. Suggesting that a protein’s overall pKa titration as well as individual amino acid titrations can be accurately described by developing a pKa calculation tool that relies on using a Linearized Poisson-Boltzmann equation (LPBE).

\[
- \nabla \cdot \varepsilon(r) \nabla \varphi(r) + \varepsilon_0 \varepsilon(r) \kappa^2(r) \varphi(r) = \frac{4\pi e^2}{\varepsilon_0 k_B T} \sum_{i=1}^{F} Z_i \delta(r - r_i) \ldots \ldots \ldots \ldots [4]
\]

II. Business Opportunity Prospects

The product of this design project will not have any market value and will be placed on the Internet as freeware. Therefore there are not any prospects for a business opportunity, but the product will be free for the scientific community to use.

Prior Art Review

There are many software packages that work to calculate pKa titrations for proteins, because pKa have proven to be pivotal in understanding protein charge characteristics. Many established pKa calculation tools rely on a Poisson Boltzmann equation to calculate free energies. These pKa calculation packages call to such programs as UMDB [8], and DelPhi II [6,7], to computationally determine free energies. Some of the mostly widely used pKa calculation packages are briefly outlined below.

1. **pKa tool** – This software has many capabilities, and is a lot more advanced than the other two pKa calculation tools. The pKa tool was setup in conjecture with pKD server. pKa tool still calls to DelPhiII to calculate free energies. It allows the user to define both specific and special titratable groups [1]
2. **pKD** – Works to establish pKa values for mutated sequences. This server allows you to choose the range of pKa values [5,7, 10]. This software was specifically designed for computational modelers to see how mutations they introduce into there protein sequence, affects the pKa of all other amino acids, and also to see the change the mutation had on the protein overall charge. This software calculates free energies by calling to DelPhiII.

3. **H++** - Works to establish pKa values for ionizable groups within a protein or biomacromolecule. It establishes a pKa values by first asking the user to input a PDB file and it will then add charges and missing hydrogen’s to the file according to the pH the user defines. H++ uses an Amber forcefield and calculates free energies using a Poisson-Boltzmann equation [add reference -Onufriev]. This system is also a web based system.

All pKa calculation packages mentioned above provide us with rational for creating our pKa calculation package. All packages stated above use the same basic theory that we will use to create our package. We have reason to believe that the Linearized Poisson – Boltzmann equation, will provide more accurate free energies. Thus, our pKa calculation package relies on the Linearized Poisson - Boltzmann equation. APBS (Adaptive Poisson – Boltzmann Solver) will be used to calculate the free energy values. We also have reason to believe that calling to APBS will make the generation of free energies, intrinsic pKa, as well as apparent pKa, more computationally efficient. All packages mentioned above lay a foundation for us in the development of our pKa calculation package.
Functional and Performance Specifications

The design project will follow the functional specifications as desired by Dr. Morikis. Its main purpose will be to take any PDB file and generate titration curves and the statistical information needed for the analysis of protein stabilities. To do this, it will incorporate APBS and PBD2PQR in order to calculate free energies, which will be used to calculate the intrinsic pKa values. The software package will then use these values to further calculate apparent pKa values by taking the statistical approximations from the Monte Carlo method or Clustering for $n$ titratable groups. The software package will generate the corresponding titration curves for the overall protein and will be used for the stability analysis. The package also needs to be user friendly, so it will include a graphical user interface that will make it easy for anybody to use. The calculations mentioned will take a considerable amount of time, so performance of the written scripts is an important issue. The scripts written will be optimized to best minimize the time needed in order to complete calculations.
Evolution of Final Design

Calculating pKₐ values

We first calculated intrinsic pKa values for one titratable group. The written scripts took a PDB file from the protein database and cleaned up the file by taking out the header and other information that was not necessary for our calculations. The cleaned PDB file was then converted to a PQR file which added hydrogen’s and each atom’s charge. The file was run through APBS to calculate free energies corresponding to each state in the thermodynamic cycle.
The values received from the calculations were used as $\Delta G^{\text{env}}$ and with accordance to equation 2, we calculated the intrinsic pKa

**Interaction Energies**

The titratable residues found in a protein have an effect on each other’s state. These effects or interactions can be favorable or unfavorable, depending on the interaction charge. The type and magnitude of interactions affect the apparent value as well as the other mentioned forces. Therefore the next step in the design was to calculate the interaction energies for every possible interaction. The energy values were outputted in a specific format in order for statistical analysis calculations.

**Statistical Analysis for titratable groups and apparent pKₐ Calculation**

With the intrinsic pKa’s and the interaction energies between every titratable group of the protein, we were ready to calculate the apparent pKa’s. To do this we called a program called Hybrid that uses clustering to calculate the apparent pKa. The program asks for the cluster size and other parameters to generate data tables that will be discussed later.
The hybrid output allowed us to set up titration curves for the entire protein as well as each individual titratable group. The pH is on the x-axis and the charge is on the y-axis.
This curve then shows titratable parts of the ionizable groups and corresponding charges for each pH. It can and will be used for further analysis of stability of protein. Also, we additionally wrote scripts to print out the groups individually and not overlaid on each other.
Stability

Next would be to calculate stability of the protein, which also uses the output of hybrid. Stability curves look to identify where the $\Delta G$ is at a local minimum. It takes the derivative of the charge.

Detailed Description of Final Solution

Titration Curves

After calculating apparent pKa for all the individual residues on the protein we will then be able to calculate the curves. We first generate the general protein titration curve. This takes the protein curve for all the individuals added up to make one titration curve that is representative of the entire protein. The protein titration curve will have free energy as the y-axis and pKa as the x-axis. This curve will show stability and isoelectric point. (Show protein graph) Next will be the individual titration curves. These will be the individual amino acids graphed by their respectful
free energy and pKa. Every individual protein that has an ionizable group will have a different pKa and energy titration curve. Graphing will be the same as the total protein curve, with x-axis being pKa and the y-axis being free energy. *(show individual protein titration curve)*

**GUI function in R**

Scripts written in R are not user friendly and quite difficult to use for anyone else but the creators of the script. For that reason it was imperative that we create a GUI. This GUI system came through a package of R called traitr. It contains simple windows that have the ability to take in user information and then run it through the script in the background. The user will only see the opening screen and the final screen of the titration curves and data. The first window will be the opening screen that will have a data energy area for the user to type in which protein they would like to generate titration curves for. The protein codes need to be written in the four-letter code format that is used in the Protein Data Bank. The system will then call to the protein data bank, download the appropriate PDB file, and run it through our script to calculate energy and pKa.
The window shows three buttons: my data, my graph, and my individual, which correspond to the data calculated by scripts, the graphs generated from the data and the individual residue titration curves. The my data button will show the data which includes pKa’s and energies for the protein and for the individual amino acids. The protein titration will show the titration curve of the entire protein. The individual titration will show the titration curve of the individual proteins all tied put together in one graph with different colors. (both shown above) The my individual button will launch another window with a list of all the amino acids.

The user will be able to click on any one of these and be able to see the titration curve for that amino acid. (show individual graph)

Material Selection

The protein structures were downloaded from the Protein Data Bank (PDB) [12]. The specific structure used in our calculations was human CR2 [11]. The structure was then cleaned
of all solvent atoms and all ions deemed the result from crystallization methods. The computers used for the calculations were the laptops previously purchased before the start of the project.

**Method of Prototyping Discussion**

Our prototype will be designed to generate titration curves from Human CR2. To generate titration curves we begin with the understanding that within the protein are ionizable residues capable of extracting and releasing a proton. This process of extracting and releasing proton is known as a proton - transfer phenomena and is greatly dependent on the pH at which ionizable residue resides in. The pH can cause the residue to be in a neutral or charged state. To estimate the most probable state the residue will be at a given pH, we take the negative logarithmic of the disassociation constant corresponding to that residue. This value is called the pKa and represents when the residue will have 50% chance of being neutral or charged. However, this value is affected by more than just the pH of the environment, but other ionizable residues, solvent molecules, ions, and polar charged groups as well. To understand the complexity of calculating the pKa with all of these factors taken into account, or apparent pKa, we use the thermodynamic cycle outlined by Antosiewicz. The thermodynamic cycle allows us to determine the intrinsic pKa value mentioned earlier by taking into account the free energies associated with the states of the thermodynamic cycle. These states are the residue by itself or free protonated; the free, deprotonated; the residue within the protein, protonated; and the residue within the protein, deprotonated. Each of these states has its corresponding free energy value, which, is calculated using the Linearized Poisson-Boltzmann equation. APBS uses this equation and places the protein into a 3-D grid with defined user parameters and gives the corresponding free energy of a specific residue. The free energy is then used to elucidate the intrinsic pKa.
The intrinsic pKa is the first milestone the prototype program must complete and after doing so, it must measure the interactions between the ionizable residues. The interactions are based off of having two residues charged with holding the other ionizable groups in their neutral state. The perturbation of having two residues charged is then measured using Coulomb’s law by calculating the electrostatic potential between the charged atoms of each residue. The electrostatic potentials are then calculated for each of the many different interactions between ionizable residues.

Each ionizable residue can either be in the charged or neutral state. If a protein were to have 34 ionizable residues, we would have $2^{34}$ or 17,179,869,184 different possible states the protein could be in. To go through each individual state would be very inefficient and to greatly shorten this process of finding the most stable state, we employ the use of a clustering technique mentioned above.

The final destination of the prototype will be when it successfully calculates apparent pKa’s for every ionizable residues as well as the overall protein pKa. These values will be demonstrated by the generation of a titration curve that shows the charge of each residue and protein at a specific pH. From the titration curves we will then able to elucidate at which pH the protein is most stable and the residues that contribute to its stability. Only when the prototype has successfully taken a PDB file and generate the corresponding pKa, will the prototype program be completed.

Another aspect to the prototype program unrelated to its calculation’s but still a critical part of the specifications is the graphical user interface or GUI. This is an important part of the specifications because this is how the interaction between the prototype and the human user will
take place. The most crucial aspect of the GUI interaction is ease of use and must be designed so that anyone can be able to use.

**Performance Testing Protocol Discussion**

To test the prototype we needed to do debug the program quite a bit. Each script took a considerable amount of time to not only get working without errors but also generate accurate results. The protocol for each specific depended on that particular script and ranged from a few hours to a couple weeks.

To test for the accuracy for intrinsic pKa, we first needed to make sure our free energies calculated by APBS were in a considerable range. The range of free energies calculated from APBS should not exceed 50 kJ/mol because there is not that much energy available or generated within a titratable residue. However, when we first ran APBS we generated energy values > 500 kJ/mol. To obtain an accurate calculation we needed to edit the calculation parameters, most specifically the dielectric constants and also make sure the PQR files were converted correctly. Once we adjusted the parameters, we obtained accurate values within range as shown in the table below along with their resulting pKa’s.

<table>
<thead>
<tr>
<th>Titratable Group</th>
<th>Free Energy (ΔG) kJ/mol</th>
<th>Intrinsic pKa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glutamate</td>
<td>1.39</td>
<td>4.06</td>
</tr>
<tr>
<td>Arginine</td>
<td>5.29</td>
<td>12.93</td>
</tr>
<tr>
<td>Histidine</td>
<td>-10.58</td>
<td>7.93</td>
</tr>
</tbody>
</table>

Table 1: Energies for some titratable groups of 1LY2
The interaction energies also needed to be tested for accuracy. Once again the majority of the inaccuracy was the result of the dielectric constant used as a parameter. The interactions are based on Coulomb’s Law, where the point charges are the atoms that held the charge within each residue. The first level of screening values was the comparison between manual calculations to that of APBS’s own coulombic calculations. These values had over a 98% correlation. Since the manual calculations were very similar to that of APBS’, we decided to with manually calculated scripts for the greatest efficiency.

The last area of screening took place after Hybrid calculates apparent pKa’s. These pKa’s took into consideration of every interaction and free energy previously calculated which make them hard to point the source of inaccuracy. The easiest and most effective way of editing the pKa’s was again to change the dielectric point. These values including the intrinsic will be shown in the next section and discussed further.

**Results**

**Description of Codes**

a. List of how codes will follow below
Table 2: Shows the list of code that were generated below a description will be made about each code as well as each code will be shown

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Call APBS</td>
</tr>
<tr>
<td>2</td>
<td>Call APBS 2</td>
</tr>
<tr>
<td>3</td>
<td>Call APBS 3</td>
</tr>
<tr>
<td>4</td>
<td>Call APBS 4</td>
</tr>
<tr>
<td>5</td>
<td>Cat 2PQR</td>
</tr>
<tr>
<td>6</td>
<td>Neutral to Charge</td>
</tr>
<tr>
<td>7</td>
<td>APBS template</td>
</tr>
<tr>
<td>8</td>
<td>APBS template New</td>
</tr>
<tr>
<td>9</td>
<td>pKa</td>
</tr>
<tr>
<td>10</td>
<td>Mean</td>
</tr>
<tr>
<td>11</td>
<td>Intrinsic pKa</td>
</tr>
<tr>
<td>12</td>
<td>Intrinsic pKa 2</td>
</tr>
<tr>
<td>13</td>
<td>Our Sequence</td>
</tr>
<tr>
<td>14</td>
<td>Interaction Energy</td>
</tr>
<tr>
<td>15</td>
<td>Interaction Energy 2</td>
</tr>
<tr>
<td>16</td>
<td>Calculation Coulomb Function</td>
</tr>
<tr>
<td>17</td>
<td>Self - Energy</td>
</tr>
<tr>
<td>18</td>
<td>Interaction Coulomb</td>
</tr>
<tr>
<td>19</td>
<td>Plot Titration</td>
</tr>
<tr>
<td>20</td>
<td>Cat PDB</td>
</tr>
</tbody>
</table>

**Description of Codes**

1. Calls to APBS software to calculate free energies for each individual protein, these values are then generated into a out file which will be inputted for the calculations of intrinsic pKa

2. Calls to APBS software to calculate free energies for each individual protein, these values are then generated into a out file which will be inputted for the calculations of intrinsic pKa, this call APBS function using the thermodynamic cycle for APBS website. This call to APBS was done in efforts to calculate intrinsic pKa for the desolvation thermodynamic cycle.
3. Calls to APBS software to calculate free energies for each individual protein, these values are then generated into a out file which will be inputted for the calculations of intrinsic pKa. This call to APBS was done in efforts to calculate free energy when all amino acids

4. Calls to APBS software to calculate free energies for each individual protein, these values are then generated into a out file which will be inputted for the calculations of intrinsic pKa. This call to APBS was done in efforts to calculate interaction energies between two ionizable charges when all others are neutralized.

5. Cat two PQR was done in efforts to generate a PQR that only allowed the titratable amino acids of choice to be put into a file with all the other amino acids residues neutralized. Note later this code was modified to substract the charge of backbone to the titratable residue.

6. Neutral to charge was done so that our generated PQR files matched with the PARSE forcefield, so that electrostatic free energies could be calculated. This code made use of the R paste function.

7. APBS template was the file generated to input known parameters so that APBS could calculate desired free energy. The template parameters were constantly changed to obtain the right order of magnitude intrinsic pKa values. Eventually we found the right inputs son that our values correlated to other pKa calculation packages. Dielectric constants were finally set to be both 78.54 for both interior of protein and for solvent.

8. APBS template was the file generated to input known parameters so that APBS could calculate desired free energy. The template parameters were constantly changed to obtain the right order of magnitude intrinsic pKa values. This template was used to input to APBS for desolvation thermodynamic cycle.
9. pKa function was generated and implemented into intrinsic pKa code to take the free energies and calculate pKa values using pKa equation from, [4].

10. This function calculates the mean value for each individual amino acids, it takes the average value for each ionizable amino acids

11. Intrinsic Pka code incorporates all codes previously mentioned, to run intrinsic pKa calculations for each ionizable amino acids in protein of choice, the for loop accounts for the desolvation TC, and generates PQR and free energies based on that.

12. Intrinsic Pka code incorporates all codes previously mentioned, to run intrinsic pKa calculations for each ionizable amino acids in protein of choice, the for loop accounts for the desolvation TC, and generates PQR and free energies based on that. This is a updated and more independent code. It allows for PQR to generate charge locally. Also in completely incorporates all other needed codes.

13. Our sequence adds us in calculating interaction energies. It converts the PQR to the right PARSE forcefield naming scheme. Furthermore, it allows for a data frame to be generated with all ionizable amino acids, this data frame will then be put into the code that calculate coulombic interaction energies.

14. Interaction energy code was generated to discern the interaction between two titratable residues. The code was generated to calculate interaction via Coulombs law. This continues to run generating a lower triangular matrix with each titratable group interacting with all other titratable groups.
15. Interaction energy code was generated to discern the interaction between two titratable residues. The function was generated to calculate interaction via Coulomb's law. This continues to run generating a lower triangular matrix with each titratable group interacting with all other titratable groups. This was a modified version.

16. This function was simply set up to calculate to Coulombic interaction energy. This function was based on the APBS Coulombic function. This function was put into the for loop for Interaction energy code.

17. Self–Energy function is made so that, we can use a cluster method to generate apparent pKa values as well as titration curves. The cluster method will be done using hybrid. The function was defined using the Born energy equation.

18. This code was a more accurate version of the interaction energy code, in that it neutralized the amino acid backbone charge. This gave us highly correlated interaction energies with APBS. The backbone was even neutralized for the two titratable charge residue. This code also generated the right table to be imported into the hybrid system, which includes all interaction energies, model pKa, residue number, self energy, unit charge.

19. Plot titration simply plots the titration curves for HYBRID. It plots overall protein titration curve, and individual single amino acids titration curves.

20. This code converts a PDB to PQR, and will make two initial PQR files a neutral and charged case. This code will be inputted into intrinsic pKa code and these two PQR files will then be converted to account for all states of APBS thermodynamic cycle.
1. Call APBS

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Functions that calls apbs to calculate free energies
#for ionizable amino acid residues
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
call_apbs <- function(in_file)
{
  bdp_file <- "1ly2_neutral.pqr"
bp_file  <- "Bound_protonated.pqr"
fdp_file <- "Free_deprotonated.pqr"
fp_file  <- "Free_protonated.pqr"

  length <- 100
  width  <- 100
  height <- 100

in_file[35] <- paste("      cglen ",length,width,height, sep = "")
in_file[36] <- paste("      cglen ",length,width,height, sep = "")
in_file[60] <- paste("      cglen ",length,width,height, sep = "")
in_file[83] <- paste("      cglen ",length,width,height, sep = "")
in_file[84] <- paste("      cglen ",length,width,height, sep = "")
in_file[107] <- paste("      cglen ",length,width,height, sep = "")
in_file[108] <- paste("      cglen ",length,width,height, sep = "")
in_file[131] <- paste("      cglen ",length,width,height, sep = "")
in_file[154] <- paste("      cglen ",length,width,height, sep = "")
in_file[155] <- paste("      cglen ",length,width,height, sep = "")
in_file[178] <- paste("      cglen ",length,width,height, sep = "")

  con <- file("infile.in","w")
  writeLines(in_file,con,sep = "\n")
  close(con)

  TC <- system(paste("/apbs-1.2-mac-univ/bin/apbs", "infile.in",">",
```

**Code 1:** Call APBS function generated for General thermodynamic cycle
2. Call APBS 2

```r
#FUNCTIONS THAT CALLS APBS TO CALCULATE FREE ENERGIES
#FOR IONIZABLE AMINO ACID RESIDUES
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
call_apbs2 <- function(in_file) {

  bdp_file <- "1ly2_neutral.pqr"
  bp_file  <- "Bound_protonated.pqr"
  fdp_file <- "Free_deprotonated.pqr"
  fp_file  <- "Free_protonated.pqr"

  length <- 100
  width  <- 100
  height <- 100


  in_file[13] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[14] <- paste("      fglen ",length,width,height, sep = " ")
  in_file[37] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[38] <- paste("      fglen ",length,width,height, sep = " ")
  in_file[61] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[62] <- paste("      fglen ",length,width,height, sep = " ")
  in_file[85] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[86] <- paste("      fglen ",length,width,height, sep = " ")
  in_file[109] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[110] <- paste("      fglen ",length,width,height, sep = " ")
  in_file[133] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[134] <- paste("      fglen ",length,width,height, sep = " ")

  con <- file("infile.in","w")
  writeLines(in_file,con,sep = "\n")
  close(con)

  TC <- system(paste("/apbs-1.2-mac-univ/bin/apbs","infile.in",">outfile.txt",sep=" "))

  outfile <- readLines("outfile.txt")
```

**Code 2: Call APBS 2 function generated for Desolvation thermodynamic cycle**
3. Call APBS 3

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Functions that calls apbs to calculate free energies
#for ionizable amino acid residues
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

call_apbs3 <- function(in_file)
{

  bdp_file <- "1ly2_neutral.pqr"

  length <- 100
  width  <- 100
  height <- 100

  rico <- grep("cglen", in_file )
  ami  <- grep("fglen", in_file )

  in_file[rico] <- paste("      cglen ",length,width,height, sep = " ")
  in_file[ami] <- paste("      fglen ",length,width,height, sep = " ")

  con <- file("infile.in","w")
  writeLines(in_file,con,sep = "\n")
  close(con)

  TC <- system(paste( "/apbs-1.2-mac-univ/bin/apbs",
                      "neutral_template.in",">", "outfile.txt", sep = ")

  outfile <- readLines("outfile.txt")
  index <- grep("Global", outfile )
  str_energy <- strsplit(outfile[index[length(index)]] [,split = " "]
  char_number <- strsplit(str_energy[1],[split = " "]
  free_energy <- as.numeric(char_number[[2]][1])
  return(free_energy)
}
```

**Code 3: Call APBS 2 function generated for free energy for the neutral case**
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Functions that calls apbs to calculate free energies
#for ionizable amino acid residues
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

call_apbs4 <- function(in_file)
{

  bdp_file <- "Two_charge.pqr"
  length <- 100
  width  <- 100
  height <- 100
  rico <- grep("cglen", in_file )
  ami  <- grep("fglen", in_file )
  in_file[rico] <- paste("      cglen ",length,width,height, sep = ")
  in_file[ami] <- paste("      fglen ",length,width,height, sep = ")

  con <- file("infile1.in","w")
  writeln(in_file,con,sep = "\n")
  close(con)
  TC <- system(paste( "/apbs-1.2-mac-univ/bin/apbs", "infile1.in"">",
    "outfile.txt", sep = " "))
  outfile <- readLines("outfile.txt")
  index <- grep("Global", outfile )
  str_energy <- strsplit(outfile[index[length(index)]],split = " ")
  char_number <- strsplit(str_energy[[1]],split = " ")
  free_energy <- as.numeric(char_number[[2]][1])
  return(free_energy)
}

Code 4: Call APBS 2 function generated for calculation of free energies for charged case
source("cat_pdb.r")

# reads in our pqr files and these will be concatenated together
neutral_pqr <- read.pqr("clean.pqr")
charged_pqr <- read.pqr("test.pqr")

# parameters, extracts sequence of pqr file and creates a array
# of characters, k allows to extract length of pqr
k <- ( as.numeric(neutral_pqr$atom[1,"resno"]))
end_of_seq <- length(seq.pdb(neutral_pqr)) - 1
seq <- seq.pdb(neutral_pqr)

# runs a loop that will look for ionizable amino acids residues
# when it finds one it will create 4 pqr files to account for
# each state in the thermodynamic cycle, and writes to file
# this will have APBS incorporated into for loop, k counter
# used to keep a numerical value
for ( i in seq )
{
    if ( i == "R" | i == "K" | i == "H" | i == "C" | i == "Y"
    | i == "D" | i == "E" )
    {
        Before <- trim.pdb( neutral_pqr, atom.select(neutral_pqr,
        resno = 1:( k - 1 ) ) )
        Free_protonated <- trim.pdb( charged_pqr, atom.select
        (charged_pqr, resno = k )
        After <- trim.pdb( neutral_pqr, atom.select (neutral_pqr,
        resno = (k+1): end_of_seq )
        Free_deprotonated <- trim.pdb( neutral_pqr, atom.select
        (neutral_pqr, resno = k))
        write.pqr(Free_protonated, file = "Free_protonated.pqr")
        Before_FP <- cat_pdb( Before,
        Free_protonated )
        Total <- cat_pdb(Before_FP, After)
    }
}
6. Neutral to Charge

```r
# This function creates two specific pqr files a netural pqr file were all amino acids partial charges are neutralized, and the charged where charge for ionizable amino acids are added, this function will then be used to concatenate the two pqr files and create pqr files for all states in the thermodynamic cycle for all ionizable amino acids

Neu_Char_pdb <- function(pdb)
{
  x <- pdb
  ## for neutral pdb
  x$atom[atom.select(x, resid = "ASP")$atom, 4] <- sub("ASP", "ASH", x$atom [atom.select(x, resid = "ASP")$atom, 4])
  x$atom[atom.select(x, resid = "GLU")$atom, 4] <- sub("GLU", "GLH", x$atom [atom.select(x, resid = "GLU")$atom, 4])
  x$atom[atom.select(x, resid = "LYS")$atom, 4] <- sub("LYS", "LYN", x$atom [atom.select(x, resid = "LYS")$atom, 4])
  x$atom[atom.select(x, resid = "ARG")$atom, 4] <- sub("ARG", "AR0", x$atom [atom.select(x, resid = "ARG")$atom, 4])
  write.pdb(pdb = x, file = "1ly2_neutral")
  pdb <- x
  ## for charged pdb
  x$atom[atom.select(x, resid = "HIS")$atom, 4] <- sub("HIS", "HID", x$atom [atom.select(x, resid = "HIS")$atom, 4])
  x$atom[atom.select(x, resid = "CYS")$atom, 4] <- sub("CYS", "CYM", x$atom [atom.select(x, resid = "CYS")$atom, 4])
  x$atom[atom.select(x, resid = "TYR")$atom, 4] <- sub("TYR", "TYM", x$atom [atom.select(x, resid = "TYR")$atom, 4])
  write.pdb(pdb = x, file = "1ly2_charged")

  pdb_list <- list(pdb, x)
  return(pdb_list)
}
```

Code 6: Makes a neutral PQR file and a charged PQR
7. APBS Template

Code 7: Template file that sets the parameters for Call APBS code, it is based on general four state thermodynamic cycle
read
mol pqr bound_dp.pqr
mol pqr bound_p.pqr
mol pqr free_dp.pqr
mol pqr free_p.pqr
end
elec name bdp
  mg-auto
dime 129 129 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 1
  lpbe
  bcfl sdh
  pdie 20.0
  ion charge 1 conc 0.15 radius 2.0
  ion charge -1 conc 0.15 radius 2.0
  sdie 78.54
  srfm mol
  chgm spl2
  sdens 10.00
  srad 1.40
  swin 0.30
  temp 298.15
  calcenergy total
  calcforce no
end

elec name bp
  mg-auto
dime 129 129 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 2
  lpbe
  bcfl sdh
  pdie 20.0
  sdie 78.54
  ion charge 1 conc 0.15 radius 2.0
  ion charge -1 conc 0.15 radius 2.0

Code 7: cont.
8. APBS template new

read
  mol pqr bound_dp.pqr
  mol pqr bound_p.pqr
  mol pqr free_dp.pqr
  mol pqr free_p.pqr
end

elec name bdp
  mg-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 1
lpbe
bcfl sdh
pdie 20.0
ion charge 1 conc 0.15 radius 2.0
ion charge -1 conc 0.15 radius 2.0
sdie 78.54
srfm mol
chgm spl2
sdens 10.00
srad 1.40
swin 0.30
temp 298.15
calcenergy total
calcforce no
end

elec name bdp_vacum
  mg-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 1
lpbe
bcfl sdh
pdie 20.0
ion charge 1 conc 0.0 radius 2.0
ion charge -1 conc 0.0 radius 2.0
sdie 20.0
srfm mol

Code 8: Template file that sets the parameters for Call APBS code, it is based on desolvation thermodynamic cycle, that has 8 states
chgm spl2
sdens 10.00
srad 1.40
swin 0.30
temp 298.15
calcenergy total
calcforce no
der

elec name bp
mg-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 2
lpbe
bcfl sdh
pdie 20.0
sdie 78.54
ion charge 1 conc 0.15 radius 2.0
ion charge -1 conc 0.15 radius 2.0
srfm mol
chgm spl2
sdens 10.00
srad 1.40
swin 0.30
temp 298.15
calcenergy total
calcforce no
der

elec name bp_vacum
mg-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 2
lpbe
bcfl sdh
pdie 20.0
sdie 20.0
ion charge 1 conc 0.0 radius 2.0

Code 8: cont.
ion charge -1 conc 0.0 radius 2.0
srfm mol
chgm spl2
sdens 10.00
srad 1.40
swin 0.30
temp 298.15
calcenergy total
calcforce no
end
elec name fdp
  mg-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 3
lpbe
  bcfl sdh
  pdie 20.0
  sdie 78.54
ion charge 1 conc 0.15 radius 2.0
ion charge -1 conc 0.15 radius 2.0
srfm mol
  chgm spl2
  sdens 10.00
  srad 1.40
  swin 0.30
  temp 298.15
  calcenergy total
  calcforce no
end
elec name fdp_vacum
  mg-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 3
lpbe
  bcfl sdh
  pdie 20.0
  sdie 20.0

Code 8: cont.
ion charge 1 conc 0.0 radius 2.0
ion charge -1 conc 0.0 radius 2.0
srfm mol
cgm sp12
sdens 10.00
srad 1.40
swin 0.30
temp 298.15
calcenergy total
calcforce no
end
elec name fp
gm-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 4
lpbe
bcfl sdh
pdie 20.0
sdie 78.54
ion charge 1 conc 0.15 radius 2.0
ion charge -1 conc 0.15 radius 2.0
srfm mol
cgm sp12
sdens 10.00
srad 1.40
swin 0.30
temp 298.15
calcenergy total
calcforce no
end
elec name fp_vacum
gm-auto
dime 65 97 129
cglen 45.3322 54.9498 82.2633
fglen 45.3322 52.3234 68.3902
cgcent mol 1
fgcent mol 1
mol 4
lpbe
bcfl sdh
pdie 20.0

Code 8: cont.
9. pKa function

Code 8: cont.

```r
Code 9: Sets up a function to calculate pKa for each individual titratable residue based on their model pKa
```

```r
pKa <- function(AAdf)
{
  AAdf <- cbind(AAdf, "Pka"=as.numeric(6))
  AAdf[(AAdf[,1]=="E"),"Pka"] <- 4.3 - as.numeric(AAdf[(AAdf[,1]=="E"),3])/(2.303 * 0.008314 * 298)
  AAdf[(AAdf[,1]=="C"),"Pka"] <- 8.28 - as.numeric(AAdf[(AAdf[,1]=="C"),3])/(2.303 * 0.008314 * 298)
  AAdf[(AAdf[,1]=="K"),"Pka"] <- 10.5 - as.numeric(AAdf[(AAdf[,1]=="K"),3])/(2.303 * 0.008314 * 298)
  AAdf[(AAdf[,1]=="H"),"Pka"] <- 6.08 - as.numeric(AAdf[(AAdf[,1]=="H"),3])/(2.303 * 0.008314 * 298)
  AAdf[(AAdf[,1]=="R"),"Pka"] <- 12.0 - as.numeric(AAdf[(AAdf[,1]=="R"),3])/(2.303 * 0.008314 * 298)
  AAdf[(AAdf[,1]=="Y"),"Pka"] <- 10.1 - as.numeric(AAdf[(AAdf[,1]=="Y"),3])/(2.303 * 0.008314 * 298)
  return(AAdf)
}
```
10. Mean Function

```r
mean <- function(Pka)
{
    meandf <- NULL
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="E"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="E"],"Pka"])))
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="R"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="R"],"Pka"])))
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="K"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="K"],"Pka"])))
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="H"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="H"],"Pka"])))
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="C"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="C"],"Pka"])))
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="Y"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="Y"],"Pka"])))
    meandf <- rbind(meandf,"AvgG" = mean(as.numeric(Pka[[Pka[,1] =="D"],"delta_G"])), "Avg_Pka" = mean(as.numeric(Pka[[Pka[,1] =="D"],"Pka"])))
    return(meandf)
}
```

**Code 10: Mean function calculates the average of value for each amino acid residue**
11. Intrinsic pKa

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Cleans pdb file extracts line begin with atom
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LY2 <- readLines("1LY2.pdb")
index1 <- grep("ATOM ", LY2)
cat(x[y], sep = "\n", file = "clean_1LY2.pdb")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#sources all functions written
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
source("cat_pdb.r")
source("neutral_charged.r")
source("call_apbs.r")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#converts pdb files to pqr files using python connection
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
system("python /Users/senior_design/pdb2pqr-1.5/pdb2pqr.py --ff parse 1ly2_charged.pdb 1ly2_charged.pqr")
system("python /Users/senior_design/pdb2pqr-1.5/pdb2pqr.py --ff parse 1ly2_neutral.pdb 1ly2_neutral.pqr")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#reads in our pqr files and these will be concatenated together
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
neutral_pqr <- read.pqr("1ly2_charged.pqr")
charged_pqr <- read.pqr("1ly2_neutral.pqr")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#parameters, extracts sequence of pqr file and creates a array
#of characters, k allows to extract length of pqr
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
k <- (as.numeric(neutral_pqr$atom[1,"resno"]))
end_of_seq <- length(seq.pdb(neutral_pqr)) - 1
seq <- seq.pdb(neutral_pqr)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#runs a loop that will look for ionizable amino acids residues
```

**Code 11: Incorporates all relevant codes to calculate hypothetical value of intrinsic pKa**
# when it finds a one it will create 4 pqr files to account for
# each state in the thermodynamic cycle, and writes to file
# this will have APBS incorporated into for loop, k counter
# used to keep a numerical value
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for ( i in seq )
{
    if ( i == "R" | i == "K" | i == "H" | i == "C" | i == "Y"
        | i == "D" | i == "E" )
        {
            Before <- trim.pdb( neutral_pqr, atom.select(neutral_pqr,
            resno = 1:( k - 1 ) ) )
            Free_protonated <- trim.pdb( charged_pqr, atom.select
            (charged_pqr, resno = k ) )
            After <- trim.pdb( neutral_pqr, atom.select (neutral_pqr,
            resno = (k+1): end_of_seq ) )
            Free_deprotonated <- trim.pdb( neutral_pqr, atom.select
            (neutral_pqr, resno = k))
            write.pqr(Free_protonated, file =
            "Free_protonated.pqr")
            Before_FP <- cat_pdb( Before,
            Free_protonated )
            Total <- cat_pdb(Before_FP, After)
            write.pqr(Total, file =
            "Bound_Protonated.pqr")
            write.pqr(Free_deprotonated, file =
            "Free_deprotonated.pqr")
            bp <- read.pqr("Bound_Protonated.pqr")
            bdp <- read.pqr("1ly2_neutral.pqr")
            fp <- read.pqr("Free_protonated.pqr")
            fdp <- read.pqr("Free_deprotonated.pqr")
            call_apbs(bdp,bp,fdp,fp)
        }
    k <- k + 1
}

Code 11: cont.
12. Intrinsic pKa 2

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Cleans pdb file extracts line begin with atom
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#LY2 <- readLines("1LY2.pdb")
#index1 <- grep("ATOM ", LY2)
##cat(x[y], sep = "\n", file = "clean_1LY2.pdb")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Open the connection for the in file used by APBS
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

con <- file("apbs_template_new2.in", "r")
in_file <- readLines(con)
close(con)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#sources all functions written
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

source("cat_pdb.r")
source("neutral_charged.r")
source("call_apbs2.r")
source("our_seq.r")
source("Pka.r")
source("mean.r")
library(bio3d)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Loads pdb and sequences according to the naming scheme
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

LY2 <- read.pdb("/Users/kaila_bennett/Desktop/Senior_Design/1LY2.pdb")
Neu_Char_pdb(LY2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#converts pdb files to pqr files using python connection
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#system("python /Users/kaila_bennett/Desktop/Senior_Design/pdb2pqr-1.5/
pdb2pqr.py --ff parse 1ly2_charged.pdb 1ly2_charged.pqr")
#system("python /Users/kaila_bennett/Desktop/Senior_Design/pdb2pqr-1.5/
pdb2pqr.py --ff parse 1ly2_neutral.pdb 1ly2_neutral.pqr")

Code 12: Is a more modified version of code 11 and establishes the right pKa values
# reads in our pqr files and these will be concatenated together
neutral_pqr <- read.pqr("1ly2_1_neutral.pqr")
charged_pqr <- read.pqr("1ly2_charged.pqr")

# parameters, extracts sequence of pqr file and creates a array
# of characters, k allows to extract length of pqr
k <- (as.numeric(neutral_pqr$atom[1,"resno"]))
end_of_seq <- length(seq.pdb(neutral_pqr)) - 1
seq <- our_seq(LY2, end_of_seq)
AAdf <- NULL

# runs a loop that will look for ionizable amino acids residues
# when it finds a one it will create 4 pqr files to account for
# each state in the thermodynamic cycle, and writes to file
# this will have APBS incorporated into for loop, k counter
# used to keep a numerical value
# run time is about 30min.
for ( i in seq[1:15] )
{
    if ( i == "R" | i == "K" | i == "H" | i == "C" | i == "Y"
          | i == "D" | i == "E" )
        {
            Before <- trim.pdb( neutral_pqr, atom.select(neutral_pqr,
resno = 1:(k - 1)) )
            Free_protonated <- trim.pdb( charged_pqr, atom.select
(charged_pqr, resno = k))
            After <- trim.pdb( neutral_pqr, atom.select(neutral_pqr,
resno = (k+1):end_of_seq ))
            Free_deprotonated <- trim.pdb(neutral_pqr, atom.select
(neutral_pqr, resno = k))
write.pqr(Free_protonated, file =
"Free_protonated.pqr")
write.pqr(After, file =
"Free_deprotonated.pqr")
Before_FP <- cat.pdb( Before,
Free_protonated )
Total <- cat.pdb(Before_FP, After)
write.pqr(Total, file =
"Bound_Protonated.pqr")
}
write.pqr(Free_deprotonated, file =
"Free_deprotonated.pqr")
indice <- atom.select( neutral_pqr, resno =
k )
neutral_pqr$atom[indice$atom,11] <- "0.00"
charged_pqr$atom[indice$atom,11] <- "0.00"
write.pqr(neutral_pqr, file = "Protein_Neu_AA.pqr")
write.pqr(charged_pqr, file = "Protein_Chr_AA.pqr")
bp <- read.pqr("Bound_Protonated.pqr")
bdp <- read.pqr("1ly2_1_neutral.pqr")
fp <- read.pqr("Free_protonated.pqr")
fdp <- read.pqr("Free_deprotonated.pqr")
delta_G <- call_apbs2(in_file)
AAdf <- rbind(AAdf, c("Resid"=i,"Resno"=

k+1,"delta_G"=delta_G))
}
k <- k + 1
}
Pka <- NULL
Pka <- pKa(AAdf)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Mean values
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#pka values
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Code 12: Cont.
13. Our Sequence

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#function to rename our sequence so that it will
#recognized, and not return an X value
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

our_seq<- function(pdb, end_of_seq)
{
  df <- NULL

  for (i in 0:end_of_seq)
  {
    l <- FALSE
    x <- atom.select(pdb, resno = i)
    J <- pdb$atom[x$atom[1],"resid"]
    if (J == "ASP" | J == "AS0")
    {
      df <- rbind(df, "D")
      l <- TRUE
    }
    if (J == "GLU" | J == "GL0")
    {
      df <- rbind(df, "E")
      l <- TRUE
    }
    if (J == "HIS" | J == "HID")
    {
      df <- rbind(df, "H")
      l <- TRUE
    }
    if (J == "LYS" | J == "LYM")
    {
      df <- rbind(df, "K")
      l <- TRUE
    }
    if (J == "CYS" | J == "CYM")
    {
      df <- rbind(df, "C")
      l <- TRUE
    }
    if (J == "ARG" | J == "AR0")
    {
      df <- rbind(df, "R")
      l <- TRUE
    }
  }

  return(df)
}

Code 13: Code that changes our PQR files into the PARSE forcefield naming scheme, also helps to create a dataframe for Coulombic interaction energy
df <- rbind(df, "R")

l <- TRUE
}
if (J == "TYR" | J == "TYM")
{
  df <- rbind(df, "Y")
  l <- TRUE
}
if ( l == FALSE )
{
  df <- rbind(df,"X")
}

return(df)
14. Interaction energy

source("call_apbs3.r")
source("call_apbs4.r")
source("our_seq.R")
source("cat_pdb.r")

LY2 <- read.pdb("/Users/kaila_bennett/Desktop/Senior_Design/1LY2.pdb")

con <- file("neutral_template.in", "r")
close(con)

con <- file("Two_charge.in", "r")
close(con)

neutral_pqr <- read.pqr("1ly2_1_neutral.pqr")
charged_pqr <- read.pqr("1ly2_charged.pqr")
neutral_pqr$atom[,"o"] <- 0
k <- 1
end_of_seq <- length(seq.pdb(LY2)) - 1
AAdf <- NULL
Neu_NRG <- call_apbs3(neutral)
seq <- our_seq(LY2, end_of_seq)
Int_mat <- matrix(0, length(seq[,1]), length(seq[,1]))
rownames(Int_mat) <- paste(seq[,1],seq[,2],sep="_")
colnames(Int_mat) <- paste(seq[,1],seq[,2],sep="_")
for ( i in 1:(length(seq[1:6,2])-1) )
    for ( e in (i+1):(length(seq[2:7,2])) )
    {
        Before_FP <- cat_pdb( Before, Free_protonated )
        midway <- cat_pdb(Before_FP, between)
        almost <- cat_pdb(midway, Free_protonated2 )
        Total <- cat_pdb(almost, After)
        write.pqr(Total, file = "Two_charge.pqr")
        #delta_G <- call_apbs4(two_charge)
        Int_mat[i,e] <- Neu_NRG - delta_G
    }

Code 14: Calculates Coulombic Interaction for each titratable residue interacting with all other titratable residues
Code 15: Interaction Energy 2

```r
source("call_apbs3.r")
source("call_apbs4.r")
source("our_seq.R")
source("cat_pdb.r")
source("calc_coulomb_func.r")

LY2 <- read.pdb("/Users/kaila_bennett/Desktop/Senior_Design/1LY2.pdb")
con <- file("neutral_template.in", "r")
neutral <- readLines(con)
close(con)

con <- file("Two_charge.in", "r")
two_charge <- readLines(con)
close(con)

neutral_pqr <- read.pqr("1ly2_1_neutral.pqr")
charged_pqr <- read.pqr("1ly2_charged.pqr")
neutral_pqr$atom[, "o"] <- 0
k <- 1
end_of_seq <- length(seq.pdb(LY2)) - 1
AAdf <- NULL
# Neu_NRG <- call_apbs3(neutral)
seq <- our_seq(LY2, end_of_seq)
Int_mat <- matrix(0, length(seq[,1]), length(seq[,1]))
rownames(Int_mat) <- paste(seq[,1], seq[,2], sep="_")
colnames(Int_mat) <- paste(seq[,1], seq[,2], sep="_")
for (i in 1:length(seq[,2])-1)
{
  for (e in (i+1):(length(seq[,2])))
    {
      #Before <- trim.pdb(neutral_pqr, atom.select(neutral_pqr, resno = 0:(as.numeric(seq[i,2])-1))
      #residue <- atom.select(charged_pqr, resno = as.numeric(seq[i,2]))
      #backbone_index <- residue$atom[1:4]
      #charged_pqr$atom[backbone_index,"o"] <- 0.0000
      Free_protonated <- trim.pdb(charged_pqr, atom.select(charged_pqr, resno = as.numeric(seq[i,2])))
      #between <- trim.pdb(neutral_pqr, atom.select(neutral_pqr, resno = (as.numeric(seq[i,2]) + 1) : (as.numeric(seq[e,2]) - 1))
      Free_protonated2 <- trim.pdb(charged_pqr, atom.select(charged_pqr, resno = as.numeric(seq[e,2])))
      #residue2 <- atom.select(charged_pqr, resno = as.numeric(seq[e,2]))
      #backbone_index2 <- residue2$atom[1:4]
    }
}
```

Code 15: Interaction energy 2 is a modified version of Code 14
16. Calculation Coulombic Function

```
coulomb <- function(aa1, aa2, seq, i,e) {
  x1 <- as.numeric(aa1$atom[as.numeric(seq[i,3]), "x"] )
  x2 <- as.numeric(aa2$atom[as.numeric(seq[e,3]), "x"] )
  y1 <- as.numeric(aa1$atom[as.numeric(seq[i,3]), "y"] )
  y2 <- as.numeric(aa2$atom[as.numeric(seq[e,3]), "y"] )
  z1 <- as.numeric(aa1$atom[as.numeric(seq[i,3]), "z"] )
  z2 <- as.numeric(aa2$atom[as.numeric(seq[e,3]), "z"] )

  distance <- sqrt( (x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2 ) * 1E-10
  die <- 20
  premy <- 8.854E-12
  e <- 1.602E-19
  N_a <- 6.022E+23
  z <- 1
  K <- 12.56637061
  coulomb <- ( (N_a*(z*(e)^2)/(die*premy*K*distance)) )/1000
  return(coulomb)
}
```

Code 16: Calculation Coulombic function is a function written of APBS Coulombic function. It simply is implement into Interaction energy function to calculate interaction energy between titratable residues
17. Self-Energy function

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Functions that calls apbs to calculate free energies
#for ionizable amino acid residues
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

self_energy <- function(Free_protonated, i)
{
  residue <- atom.select (charged_pqr, resno = as.numeric(seq[i,2]) )
  backbone_index <- residue$atom[1:4]
  Free_protonated <- trim.pdb(charged_pqr,atom.select (charged_pqr,
     eleno = residue$atom[5:length(residue$atom)]) )
  write.pqr(Free_protonated, file = "coulomb.pqr")

  selfenergy <- system(paste( "/apbs-1.2-mac-univ/share/tools/manip/coulomb",
    "coulomb.pqr"," >", "outfile.txt", sep = " ")
  outfile <- readLines("outfile.txt")
  index <- grep("Total energy", outfile )
  str_energy <- strsplit(outfile[index[length(index)]],split = " ")
  char_number <- strsplit(str_energy[[1]],split = " ")
  free_energy <- as.numeric(char_number[[2]][1])/ 20
  return(free_energy)
}
```

Code 17: Is a function written to calculate the self energy for each titratable residue using a Born Equation
18. Interaction Coulombic

Code 18: This is a very important code in that it outputs our Intrinsic pKa, Self Energy, Coulombic energy, residue number and unit charge in the right format so that our output can be implemented into HYBRID so that we can calculate apparent pKa values based off model pKa’s. Also it generates titration curves for first the overall protein and titration curves for all individual amino acids. Tyrosines were eventually left out of our consideration because they went out of range.
19. Plot Titration

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#titration R
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
source("our_seq.R")
axis <- NULL
list <- readLines("testy.txt")
list2 <- readLines("jay.txt")
index <- grep("20.00",list2)
cat(list2[1:index[1]], sep = "\n", file = "mean.txt")
list3 <- read.table("mean.txt")
for( i in 1:(length(list)/(length(seq[,1])+1)) )
{
  sequence <- seq(i,length(list),by = length(seq[,1])+1)
  axis <- cbind(axis, as.numeric(list[sequence])  )
}
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#graph titration curve sequence
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
par(mfrow = c(2,1))
ph <- list3[,"V1"]
c <- list3[,"V2"]
plot(ph, c,main="Titration \n For Overall Protein", xlab = "pH", ylab = "Charge", type="o", col="blue", ylim=c(min(list3[,"V2"]),max(list3 [,"V2"])))
ph <- axis[,1]
Partial <- axis[,2]
plot(ph,Partial,main="Individual Ionizable Residues",xlab = "pH", ylab = "Partial Charge", type="o", col= "blue", xlim = c(-2,18), ylim=c(-1,1))
for ( i in 3:(length(list)/(length(seq[,1])+1)) )
{
  ph <- axis[,1]
  Partial <- axis[,i]
  lines(ph,Partial, type="o", col= i)
}
```

Code 19: Simply is code to generate plots in desired format. We made use of the lines function to plot multiple graphs.
20. CAT PDB

# Concatenates two pdb files
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```r
cat_pdb <- function(pdb1, pdb2)
{
  npdb <- pdb1
  npdb$atom <- rbind(npdb$atom, pdb2$atom)
  npdb$xyz <- c(npdb$xyz, pdb2$xyz)
  npdb$calpha <- c(npdb$calpha, pdb2$calpha)
  return(npdb)
}
```

Code 20: Very preliminary Code

### Intrinsic and Apparent pKa

a. Intrinsic pKa values

<table>
<thead>
<tr>
<th>Protein 1LY2</th>
<th>Intrinsic pKa (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residue</td>
<td></td>
</tr>
<tr>
<td>Arginine</td>
<td>12.62</td>
</tr>
<tr>
<td>Aspartic Acid</td>
<td>4.29</td>
</tr>
<tr>
<td>Cystine</td>
<td>N/A</td>
</tr>
<tr>
<td>Glutamic Acid</td>
<td>4.06</td>
</tr>
<tr>
<td>Histidine</td>
<td>7.93</td>
</tr>
<tr>
<td>Lysine</td>
<td>10.52</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>7.82</td>
</tr>
</tbody>
</table>

Table 3: shows intrinsic pKa values generated using above code, one should remember this a theoretical values and will be used to calculate the apparent pKa values.
b. Apparent pKa without Tyrosine’s

<table>
<thead>
<tr>
<th>Residue</th>
<th>Apparent pKa (Ours) Dielectric = 40</th>
<th>Apparent pKa (Dr. Morikis)</th>
<th>Apparent pKa (H++)</th>
<th>Apparent pKa (pKa Tool)</th>
<th>Standard Deviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-E</td>
<td>4.45</td>
<td>N/A</td>
<td>3.80</td>
<td>3.46</td>
<td>0.50</td>
</tr>
<tr>
<td>13-R</td>
<td>13.32</td>
<td>12.31</td>
<td>12.30</td>
<td>11.76</td>
<td>0.65</td>
</tr>
<tr>
<td>28-R</td>
<td>12.26</td>
<td>13.61</td>
<td>12.60</td>
<td>10.95</td>
<td>1.10</td>
</tr>
<tr>
<td>36-R</td>
<td>12.31</td>
<td>13.24</td>
<td>12.90</td>
<td>10.30</td>
<td>1.32</td>
</tr>
<tr>
<td>40-E</td>
<td>0.77</td>
<td>2.63</td>
<td>2.60</td>
<td>2.12</td>
<td>0.87</td>
</tr>
<tr>
<td>41-K</td>
<td>7.68</td>
<td>9.56</td>
<td>9.90</td>
<td>8.45</td>
<td>1.02</td>
</tr>
<tr>
<td>48-K</td>
<td>12.11</td>
<td>10.26</td>
<td>10.50</td>
<td>9.35</td>
<td>1.15</td>
</tr>
<tr>
<td>49-D</td>
<td>4.01</td>
<td>2.75</td>
<td>3.10</td>
<td>2.71</td>
<td>0.60</td>
</tr>
<tr>
<td>50-K</td>
<td>9.93</td>
<td>10.64</td>
<td>10.70</td>
<td>9.55</td>
<td>0.56</td>
</tr>
<tr>
<td>52-D</td>
<td>4.11</td>
<td>3.92</td>
<td>4.20</td>
<td>3.10</td>
<td>0.50</td>
</tr>
<tr>
<td>56-D</td>
<td>0.58</td>
<td>2.63</td>
<td>2.10</td>
<td>1.17</td>
<td>0.92</td>
</tr>
<tr>
<td>57-K</td>
<td>10.18</td>
<td>11.54</td>
<td>12.10</td>
<td>9.72</td>
<td>1.12</td>
</tr>
<tr>
<td>61-K</td>
<td>10.15</td>
<td>10.75</td>
<td>10.60</td>
<td>9.43</td>
<td>0.59</td>
</tr>
<tr>
<td>63-E</td>
<td>2.05</td>
<td>2.42</td>
<td>1.80</td>
<td>0.50</td>
<td>0.83</td>
</tr>
<tr>
<td>67-K</td>
<td>9.35</td>
<td>10.97</td>
<td>10.90</td>
<td>9.22</td>
<td>0.95</td>
</tr>
<tr>
<td>73-E</td>
<td>4.06</td>
<td>3.18</td>
<td>2.90</td>
<td>2.20</td>
<td>0.77</td>
</tr>
<tr>
<td>81-K</td>
<td>9.93</td>
<td>11.91</td>
<td>12.60</td>
<td>5.30</td>
<td>3.29</td>
</tr>
<tr>
<td>83-R</td>
<td>12.12</td>
<td>12.92</td>
<td>13.20</td>
<td>12.20</td>
<td>0.53</td>
</tr>
<tr>
<td>89-R</td>
<td>11.38</td>
<td>13.31</td>
<td>12.90</td>
<td>10.98</td>
<td>1.14</td>
</tr>
<tr>
<td>90-H</td>
<td>4.19</td>
<td>4.95</td>
<td>5.20</td>
<td>0.40</td>
<td>2.23</td>
</tr>
<tr>
<td>92-D</td>
<td>1.79</td>
<td>3.20</td>
<td>3.20</td>
<td>1.48</td>
<td>0.91</td>
</tr>
<tr>
<td>99-K</td>
<td>9.93</td>
<td>10.43</td>
<td>10.40</td>
<td>8.89</td>
<td>0.72</td>
</tr>
<tr>
<td>108-K</td>
<td>9.26</td>
<td>10.34</td>
<td>10.30</td>
<td>8.56</td>
<td>0.86</td>
</tr>
<tr>
<td>122-R</td>
<td>12.55</td>
<td>12.12</td>
<td>12.10</td>
<td>12.55</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 4: represents our apparent pKa values from Hybrid against other apparent pKa values from various other sources reporting values for protein 1LY2 (CR2)

c. Apparent pKa with Tyrosine’s
Titration Curves

Titration
For Overall Protein

Charge
-5 0 5 10

pH
-5 0 5 10 15 20

Individual Ionizable Residues

Partial Charge
-10 0 10

pH
-5 0 5 10 15 20

Figure: Titration curves generated from the Hybrid output in R
Figure: Shows a comparison of titration curves for individual amino acids residues
GUI

Figure: Represent the Main Gui, where the user will define the protein they wish to analyze and show output of analysis, so the user can see overall titration curves and individual titration curves, as well as intrinsic pKa values and apparent pKa values

Figure: Shows Gui so that the user can click on individual amino acids
Figure: Shows table generated from Gui for test case protein CR2 (1LY2). It shows Intrinsic, model and apparent pKa

<table>
<thead>
<tr>
<th>Group</th>
<th>Intrinsic pKa</th>
<th>pk(model)</th>
<th>pK(app)</th>
<th>pK(app) - pK(model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.382</td>
<td>4.4</td>
<td>4.548</td>
<td>0.168</td>
</tr>
<tr>
<td>2</td>
<td>-2.646</td>
<td>12</td>
<td>14.332</td>
<td>2.332</td>
</tr>
<tr>
<td>3</td>
<td>3.121</td>
<td>9.6</td>
<td>10.33</td>
<td>0.73</td>
</tr>
<tr>
<td>4</td>
<td>2.894</td>
<td>9.6</td>
<td>10.224</td>
<td>0.624</td>
</tr>
<tr>
<td>5</td>
<td>-1.548</td>
<td>12</td>
<td>14.287</td>
<td>2.287</td>
</tr>
<tr>
<td>6</td>
<td>7.914</td>
<td>9.6</td>
<td>12.124</td>
<td>2.524</td>
</tr>
<tr>
<td>7</td>
<td>-3.085</td>
<td>12</td>
<td>14.626</td>
<td>2.636</td>
</tr>
<tr>
<td>8</td>
<td>6.525</td>
<td>4.4</td>
<td>2.666</td>
<td>-1.754</td>
</tr>
<tr>
<td>9</td>
<td>4.468</td>
<td>10.4</td>
<td>8.703</td>
<td>-1.697</td>
</tr>
<tr>
<td>10</td>
<td>5.765</td>
<td>10.4</td>
<td>10.099</td>
<td>-0.301</td>
</tr>
<tr>
<td>11</td>
<td>-1.416</td>
<td>4</td>
<td>2.08</td>
<td>-1.52</td>
</tr>
<tr>
<td>12</td>
<td>5.774</td>
<td>10.4</td>
<td>9.969</td>
<td>-0.431</td>
</tr>
<tr>
<td>13</td>
<td>2.773</td>
<td>4</td>
<td>3.699</td>
<td>-0.301</td>
</tr>
<tr>
<td>14</td>
<td>2.704</td>
<td>4</td>
<td>1.295</td>
<td>-2.705</td>
</tr>
<tr>
<td>15</td>
<td>3.049</td>
<td>10.4</td>
<td>11.893</td>
<td>1.493</td>
</tr>
<tr>
<td>16</td>
<td>8.911</td>
<td>10.4</td>
<td>8.737</td>
<td>-1.663</td>
</tr>
<tr>
<td>17</td>
<td>2.991</td>
<td>4.4</td>
<td>1.884</td>
<td>-2.516</td>
</tr>
<tr>
<td>18</td>
<td>4.548</td>
<td>9.6</td>
<td>10.395</td>
<td>0.705</td>
</tr>
<tr>
<td>19</td>
<td>6.005</td>
<td>10.4</td>
<td>9.043</td>
<td>-3.257</td>
</tr>
<tr>
<td>20</td>
<td>4.390</td>
<td>9.6</td>
<td>10.13</td>
<td>0.53</td>
</tr>
<tr>
<td>21</td>
<td>2.061</td>
<td>4.4</td>
<td>2.650</td>
<td>-1.441</td>
</tr>
</tbody>
</table>
Discussion

Figure: Shows distances between interacting amino acids, and protein 1LY2 with titratable groups for easy visualization

Intrinsic pKa

The intrinsic pKa’s are dependent on the free energies calculated from the thermodynamic cycle. The Free energies are the difference between reference state, and charged and neutral state. To obtain reasonable free energy values, that to stay of the right order of magnitude, we implemented three different thermodynamic cycles. The thermodynamic cycle was finally implemented into all our codes, was the general cycle with 4 states. It turned out that to obtain reasonable free energy values, we simply needed to modulate the dielectric constants. For the free states we simply made the dielectric constant 78.54 for both solvent and solute, and for the bound state we made the dielectric constant 78.54 for the solvent and 40 for the solute. Intrinsic pKa, are a theoretical value that allows for the quantification of apparent pKa values as stated in the introduction. Being the intrinsic pKa’s are strictly hypothetical, they can not be
correlated to any empirical model. Therefore it is difficult to determine the accuracy of our pKa\textsuperscript{int}, thus we solely rely on our free energy values begin in range.

**Apparent pKa**

Apparent pKa values rely on the intermediate step of intrinsic pKa values, as well as the interaction energies between all ionizable groups. To determine interaction energies, we implemented a Coulombic function that was based off analytical Coulomb's Law. Once we had obtained the interaction energies with specific formatting that could be implemented into hybrid. The values we received from hybrid were initially out of the designated pH range, which intuitively was from zero to fourteen. However, the shifts in pKa due to the interaction energies and intrinsic pKa’s can be greater than the ionization of water. Due to the fact that our code includes Tyrosines and also due to favorable and unfavorable coulombic interactions, a pH range greater than the ionization of water is still accurate. Thus we expanded the pH range to -5-25, to include great shifts in pKa.

Referring back to our pKa\textsuperscript{app} (Table 3), we concluded that a few interactions exhibited strong favorable and unfavorable interactions due to their proximity. Namely, interactions between Lysine 81 and Tyrosine 88, and Glutamic acid 56 and Lysine 57 were the greatest. Below is a chart that depicts the interactions with distances under 5 angstroms, because of the cutoff value for strong interactions being 5 angstroms [4].

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E_40</td>
<td>K_57</td>
<td>4.98</td>
</tr>
<tr>
<td>D_56</td>
<td>K_57</td>
<td>3.57</td>
</tr>
<tr>
<td>K_67</td>
<td>Y_68</td>
<td>4.56</td>
</tr>
<tr>
<td>E_73</td>
<td>Y_88</td>
<td>4.62</td>
</tr>
<tr>
<td>K_81</td>
<td>Y_88</td>
<td>2.83</td>
</tr>
</tbody>
</table>
Graphical User Interface

The GUI is an important part to any program, especially if it is designed for others to use. Because of this, we needed to make a GUI that was easy to use. To do this we generated windows that will be able to run all scripts with just a click of a button. After the completion of the calculations, the user will be able to navigate through the calculated data with ease.

Financial Consideration of Design

Financial considerations for this project since all software used are open source; there are no monetary financial needs. All computers to design software were already purchased by the group individuals prior to beginning of the project. Since this program will be also be an open source, there are no financial considerations in the selling of the product. Only price to consider is an external hard drive to back up all code, script written.

External Hard Drive ..............................................................................................................$100.0

Conclusion

To be completed

Future Work

To further develop the prototype outlined in this report, it will be necessary to improve on its limitations and expand on its performance, time efficiency, and user experience. The prototype also did not meet all of its objectives. The performance issues of the prototype Improvements can include error boxes, help menus and a status bar. Script optimization will
always be an ongoing project cutting calculations that can take up to three hours with 32 titratable side chains.

Statement of Social Impact

As of today there are a few online programs that give us different protein properties that we are calculating. Our software package brings many of these calculations into a single program saving researchers time when looking for this type of protein quantification. Companies who could benefit from our program are Biotech companies that grow protein products. During the research phase of their production, our program can tell them at what environments their proteins will be at a desired functionality state. Other groups that could benefit from our software package are researching teams trying to further elucidate protein charge characteristics, to ultimately understand binding, and activity. This program could aid the field of rational drug design, to better design drugs and inhibitors, with catered physiochemical properties.
Appendix

I: List of Abbreviations

APBS = Adaptive Poisson-Boltzmann Solver

GUI = Graphical User Interface

PDB = Protein Data Bank

Coulombic Interaction

Intrinsic pKa = \( \text{pK}_{\text{int}} \)

Apparent pKa = \( \text{pK}_{\text{app}} \)

Model pKa = \( \text{pK}_0 \)

Linearized Poisson-Boltzmann Equation = LPBE

Poisson-Boltzmann Equation = PBE

Thermodynamic Cycle = TC

Free energy = \( \Delta \text{G} \)

Electrostatic potential = \( \Psi(r) \)

Dielectric constant = \( \varepsilon_r \)

Permittivity of free space = \( \varepsilon_0 \)
Boltzmann constant = $k_B$

Dissociation constant = $K_a$

Ion valence = $Z$

Unit charge = $e$

$\Delta\Delta G = \Delta\Delta G^{\text{Env}}$

Absolute Temperature = $T$

PARSE forcefield = Naming scheme of force field

Arginine = ARG or R

Aspartic Acid = ASP or D

Cysteine = CYS or C

Glutamic Acid = GLU or E

Histidine = HIS or H

Lysine = LYS or K

**II: Project Budget**

The only expenditure noted by our group was the purchase of an external hard drive to back up all data, scripts, and other necessary files. All programs like APBS and R programming were opened sourced software.
III: Individual Group Responsibilities

Amitoj Chopra and Enrico Sagullo generated the scripts to design a GUI and implement calculation scripts and data.

Kaila Bennett and Jesse Johnson designed and implemented theory and calculation scripts as well as input and output scripts, and helper functions.

IV: Acknowledgements

We would like to thank Dr. Dimitrios Morikis for all his help and guidance throughout this entire project, explaining theoretical significance of the project. We would like to thank Mr. Chris Kieslich with his help in explaining how to code in R programming. We would also like to thank Dr. Jerome Schultz, and Mrs. Hung Xu for their help throughout this project as well.
V: References

   *Protein Science.* **12**, 1894-1901. 2007


3. Antosiewicz, M.J. Protonation free energy levels in complex molecular systems.
   *Biopolymers.* **89**, 262-269. 2007


