Name: Carol Muriithi

Identification of Complementarity-determining region3(CDRH3) on the heavy chain using a logistic regression model

Methods

I used a logistic regression model to create a machine learning model to return a vector of predicted class labels for the predictor data in the matrixes we created. Logistic Regression is a machine learning algorithm which can be used for classification or regression problems.

Data was divided for training and testing purposes for the entire dataset. I created a subset file of 500 sequences from the entire dataset

Dividing of Testing and Training data	Number of rows
n_train(Training data)	500
n_test(Testing data)	500

Table1: Dividing of Testing and Training data

Next the subset files were saved under the names mouse_train and mouse_test and were read using tdfread on Matlab as shown below:

```
data = tdfread('mouse_train_500.txt','tab');
data1 = tdfread('mouse_test_500.txt','tab');
```

Cleaning the data

I then converted the amino acid sequences into ASCII numbering system for both the training and testing data and set all missing data to 0 as follows:

```
temp(isnan(temp)) = 0;    % set missing data to 0
```

This is because we had a lot of missing amino acids in our antibody sequences. This was done for both the training and testing data.

Conditions used to extract CDRH3 and CDRH1 regions and corresponding non-CDR regions

I used the following conditions to create a subset of an antibody sequence specifically for CDR1 and CDR3.

It was also important to create corresponding non-CDR regions for both subsets that would be assigned the 0 labels for the ytrain and ytest so as not to get errors while training your model.

The conditions to extract subsets for the CDRH1 and H3 regions used were those defined by Chothia and Kabat: I applied Kabat or Chothia numbering schemes to antibody sequences

Summary of CDR Conditions

	~Residue Start	Residue before	Residue After	Residue Length
CDRH1	26	Cys(C)	Trp (W)	10-12
CDRH2	15 after end of CDRH1	Leu-Glu-Trp-Ile-Gly	Lys/Arg-Leu/Ile	16-19
CDRH3	33 after end of CDRH2	Cys(C)	Trp-Gly-xxx-Gly	3-25

Table2: Numbering scheme used to extract features: Chothia/Abm/Kabat [1]

Selecting both non-CDR and CDR regions from each antibody sequence

```
%% select for nonCDR and CDR regions
istart=125;    % H22  %istart29    %125
iend=153;      % H36    %iend38    %153
```

Table 3:The following script was used to get a subset of the antibody sequence for each row. The subset included both CDR and non-CDR regions

The following loop was used to specifically assign labels for the CDRH3 region in the subset of the antibody sequence chosen above in every antibody sequence. The labels were 1 for ytrain for CDRH3 regions. Non-CDR regions would be automatically be assigned a 0. This was also done for ytest.

```
if data.H92(j) == double('C') && data.H93(j) == double('A') &&
 data.H94(j) == double('R')
   y_train(j) =1; %CDR-H3 condition 3
end
```

Note: The loop is part of the bigger loops shown in tables 5 and 6

Feature engineering

For feature engineering, I used the following features that are very specific for each of the 20 amino acids in nature that would be found in the antibody sequence.
The features used were as follows:

- Molecular Weight
- Solubility
- pI values
- Hydropathy: Hydrophillic or Hydrophobic
- Charge
- pKa values of the Side Group(R group): Every amino acid has different R group

I created a features table for every amino acid and saved them in an excel file that was labeled as follows as reference:

Name of Amino Acid	1 Letter Symbol	Molecular Weight	Solublity	pI	Hydropathy	Charge	pK(R Group)
Alanine	A	89.1	15.8	6	hydrophobic	N	6
Arginine	R	174.2	71.8	10.76	hydrophilic	+	10.76
Asparagine	N	132.12	2.4	5.41	hydrophilic	N	5.41
Aspartic acid	D	133.11	0.42	2.77	hydrophilic	-	2.77
Cysteine	C	121.16	freely	5.07	moderate	N	5.07
Glutamic acid	E	147.13	0.72	3.22	hydrophilic	-	3.22
Glutamine	Q	146.15	2.6	5.65	hydrophilic	N	5.65
Glycine	G	75.07	22.5	5.97	hydrophobic	N	5.97
Histidine	H	155.16	4.19	7.59	moderate	+	7.59
Isoleucine	I	131.18	3.36	6.02	hydrophobic	N	6.02
Leucine	L	131.18	2.37	5.98	hydrophobic	N	5.98
Lysine	K	146.19		9.74	hydrophilic	+	9.74
Methionine	M	149.21	5.14	5.74	moderate	N	5.74
Phenylalanine	F	165.19	2.7	5.48	hydrophobic	N	5.48
Proline	P	115.13	1.54	6.3	hydrophobic	N	6.3
Serine	S	105.09	36.2	5.68	hydrophilic	N	5.68
Threonine	T	119.12	freely	5.6	hydrophilic	N	5.6
Tryptophan	W	204.23	1.06	5.89	hydrophobic	N	5.89
Tyrosine	Y	181.19	0.038	5.66	hydrophobic	N	5.66
Valine	V	117.15	5.6	5.96	hydrophobic	N	5.96

Table: 3 and 4: Feature used for the amino acids (Raw)

Values assigned for hydropathy as follows:
Hydrophilic=1
Hydrophobic=2
Moderate=3
Values assigned for Charge were as follows:
Neutral Charge(N)=2
Positive Charge=3
Negative Charge=1
This was done so as to avoid too many zeros in my matrices. I then created a new features table with the new labels and read it in matlab as:
`features = readtable("Features_Table_New.xlsx"); and assigned the name [n_letters, n_features] in Matlab.`
For the second step of processing the features table it into an array.

The final table used for that was as follows:

	A	B	C	D	E	F	G	H
1	Name of Amino Acid	1 Letter Symbol	Molecular Weight	Solublity	pI	Hydropathy	Charge	pK(R Group)
2	Alanine	A	89.1	15.8	6	1	2	6
3	Arginine	R	174.2	71.8	10.76	2	3	10.76
4	Asparagine	N	132.12	2.4	5.41	2	2	5.41
5	Aspartic acid	D	133.11	0.42	2.77	2	1	2.77
6	Cysteine	C	121.16	10	5.07	3	2	5.07
7	Glutamic acid	E	147.13	0.72	3.22	2	1	3.22
8	Glutamine	Q	146.15	2.6	5.65	2	2	5.65
9	Glycine	G	75.07	22.5	5.97	1	2	5.97
10	Histidine	H	155.16	4.19	7.59	3	3	7.59
11	Isoleucine	I	131.18	3.36	6.02	1	2	6.02
12	Leucine	L	131.18	2.37	5.98	1	2	5.98
13	Lysine	K	146.19	10	9.74	2	3	9.74
14	Methionine	M	149.21	5.14	5.74	3	2	5.74
15	Phenylalanine	F	165.19	2.7	5.48	1	2	5.48
16	Proline	P	115.13	1.54	6.3	1	2	6.3
17	Serine	S	105.09	36.2	5.68	2	2	5.68
18	Threonine	T	119.12	9.59	5.6	2	2	5.6
19	Tryptophan	W	204.23	1.06	5.89	1	2	5.89
20	Tyrosine	Y	181.19	0.038	5.66	1	2	5.66
21	Valine	V	117.15	5.6	5.96	1	2	5.96

Table 4: Final Features table used for the amino acids with features names in column names.

The data was then further divided in ytest,ytrain,xtest and xtrain The following for loops were used to generate the xtrain and xtest and also assign labels to the ytrain and ytest.

```
for j = 1:n_train % loop of rows of data

     if data.H92(j) == double('C') && data.H93(j) == double('A') && data.H94(j) == double('R')
        %%if data.H103(j) == double('W') && data.H104(j) == double('G') && (data.H105(j) == double('Q')
        y_train(j) =1; %CDR-H3 condition 3
        %end
     end

         % fill x_train
       for k=1:n_letters  % loop of letters exctated from feature table
         for i=istart:iend  % loops between H22 and H36
           if data.(fields{i})(j) == letters(k)   % compare the data to the 20 letters of the amino aci
             idx = 0 ;
             for m = 1:n_features  % loops of features
                 %idx = (i-istart)*n_features + m;  % index of columns of x_train
                 idx = (m-1)*n_col + i - istart + 1;
                 x_train(j,idx) = features_num(k,m);  % assign x_train
             end
           end
         end
       end
end
x_train(1,1)   % check the values of x_train
%x_train(1,2)
```

Table 5: For loop that generates and assigns ytrain and xtrain

```
for j = 1:n_test

       %if data.H22(j) ==  double('C')  && data.H36(j) ==  double('W') % condition 1
          %y_test(j) = 1;    % CDR-H1
       %end
        if data.H92(j) == double('C') && data.H93(j) == double('A') && data.H94(j) == double('R')
        %if data.H103(j) == double('W') && data.H104(j) == double('G') && (data.H105(j) == double(
        y_test(j) =1; %CDR-H3 condition 3
        end
       % fill x_test
       for k=1:n_letters  % loop of letters exctated from feature table
         for i=istart:iend  % loops between H22 and H36
           if data.(fields{i})(j) == letters(k)   % compare the data to the 20 letters of the amin
             idx = 0 ;
             for m = 1:n_features  % loops of features
                 %idx = (i-istart)*n_features + m;  % index of columns of x_train
                 idx = (m-1)*n_col + i - istart + 1;
                 x_test(j,idx) = features_num(k,m);  % assign x_train
             end
           end
         end
       end

end
x_test(1,1)
```

Table6: For loop that selects for and assigns ytest and xtest

Results

After data preparation and feature engineering, logistic regression was then used to analyze and process and create a predicting model. The results of our model were as follows for post processing:

```
Model accuracy = 0.652
Model sensitivity = 0.000
Model specificity = 1.000
Model AUC = 0.500
>>
```

Table: Post Processing for Logistic Regression Model with 500 rows and 500 sequences for CDRH3

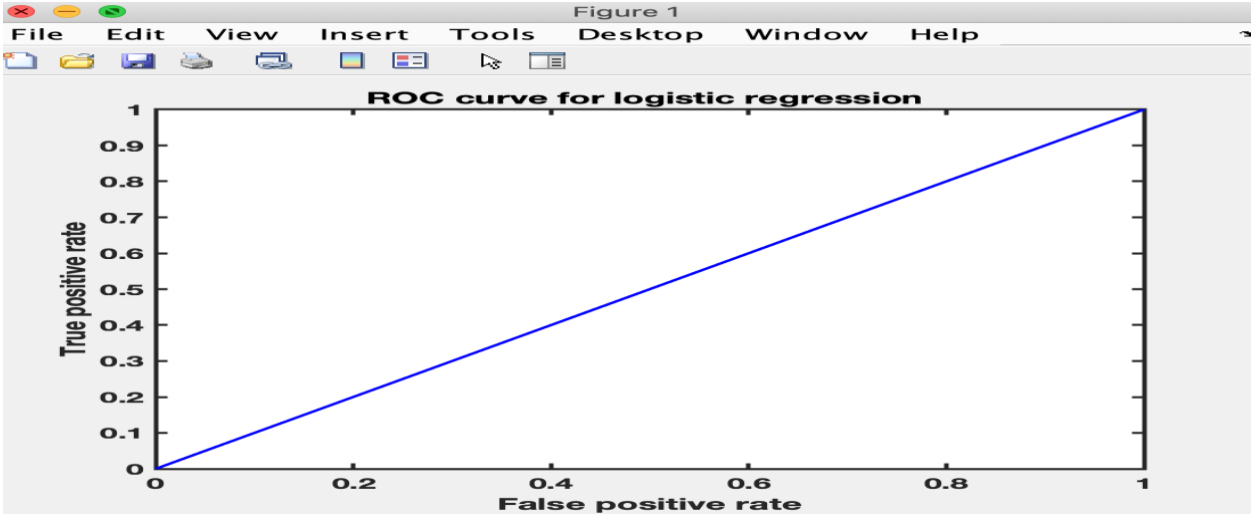

Figure 2: Plot showing area under the curve(AUC) for predicting CDRH3 using logistic regression

Discussion

Most of the sequence variation that is associated with antibodies and T cell receptors are found in the CDRs with CDR3 being the most variable. A drawback of the Kabat, Chothia, and IMGT numbering schemes that we used is that CDRs length variability only takes into account the most common loop lengths.

I used logistic regression to return a vector of predicted class labels for the predictor data in the matrixes we created. Our low sensitivity values from the trained logistic regression model for the CDRH3 regions could because of the trade-off between specificity and sensitivity of a model. In addition to this, more research needs to done on the non-CDR regions to better understand if they could also have a role in Ag binding. Overall, I am happy that I came up with a model that was somewhat accurate in predicting CDRH3 regions. I found it easier to work with CDRH3 as compared with CDH1 because it had more variability in between different antibody sequences as compared to CDRH1 which had very little variability between antibody sequences and this can be seen from both models results(also included CDH1 script in addition to CDRH3).

Citation/References

1. Yu, H. Analyzing antibody sequence for recombinant antibody experience. GenScript.20 March, 2015.
2. Culang,S.et al. The Structural Basis of Antibody-Antigen Recognition. Front Immuol. 8 Oct 2013, 4:302; doi: [10.3389/fimmu.2013.00302]
3. Group Members: Ahmed, Prisma, Zainab.
4. MATLAB
5. MATLAB SCRIPT: Carol Muriithi.