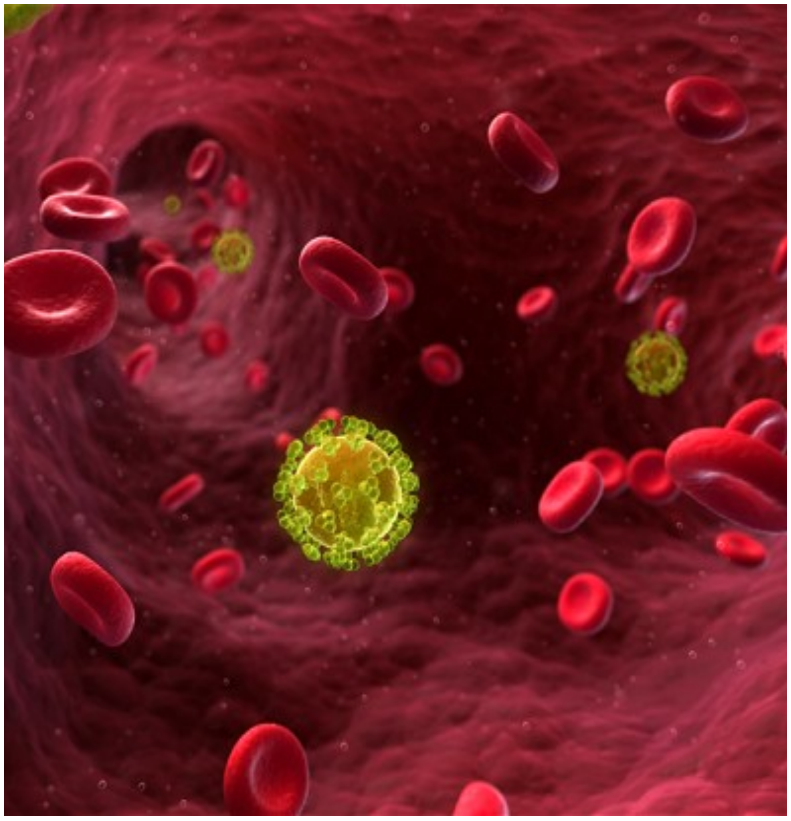


HIV Progression

A Statistical Analysis of the Improvement in HIV+ Patients after ART

Angel Garcia de la Garza

STAT 471: Modern Data Mining – The Wharton School
Dr. Linda Zhao
December 20th, 2014

Table of Contents

Summary

This project consists of the analysis of a cohort of 1000 HIV+ patients in order to statistically determine variables that are relevant to patient improvement as well as to create a model that can help classify the improvement of the patients after starting antiretroviral therapy. This dataset comes from Kaggle, a website that hosts several data prediction competitions. The dataset contains the information of 1000 patients who have undergone treatment under the same medication. As part of our variables we are given the patient ID (arbitrary), a binary response status, a viral load at the beginning of therapy (in log10 units), the CD4 count at the beginning of therapy, protease (PR) nucleotide sequences, and the reverse transcriptase (RT) nucleotide sequence.

The process of preparing and cleaning the data for the analysis was extensive. I first aligned all incomplete sequences so that we could look at the variability at each specific position in the sequence. I proceeded to break the sequences into codons (or 3-mers) and translate these into the respective amino acids they express. Next, I dealt with ambiguous/missing values in the data. I created a probabilistic process in order to replace these invalid entries with a valid value. I was able to achieve this using another database of the prevalence of mutation at each specific position in the HIV genetic sequence.

I was able to identify relevant variables for patient improvement using the Lasso. These results include the fact that patients who have a mutation on the 10th and 82nd position of the protease sequence tend to be less likely to improve under antiretroviral therapy. I was able to corroborate some of these findings in the literature and find that the 82nd position is highly correlated to HIV resistance to antiretroviral drugs. At last, I built a model for classification using logistic regression and splitting my data into training and test sets. I was able to achieve a test misclassification error of around 20%, well below the 50% threshold. As an addendum, I also compared this method to boosting and classification trees.

Description of the Problem

In 2013 alone, the World Health Organization estimates that around 1.5 million people died from HIV-related causes around the world. The WHO also states that there are around 35 million people living with HIV worldwide. While there is no cure for HIV infection, effective treatment with antiretroviral drugs can control the virus allowing patients to enjoy healthy and productive lives and has helped control this pandemic.

The Human Immunodeficiency Virus (HIV) targets the immune system and weakens its ability to respond to infections. As the virus progresses, people's defense system weakens and gradually becomes immunodeficient. This results in increased susceptibility to a wide range of infections that people with healthy immune systems can regularly fight off. The most advanced stage of HIV infection is the Acquire Immunodeficiency Syndrome (AIDS), which can take from 2 to 15 years to develop, depending on individual conditions.

The body can suppress the HIV virus with a combination of antiretroviral drug treatment (ART). ART does not cure HIV infection but helps maintain the virus's presence in the body to a minimum. The ART helps control the viral replication and allows an individual's immune system to strengthen and regain the capacity to fight off infections.

Antiretroviral therapy consists of a combination of at least three different antiretroviral drugs that aim to attack the virus at different points in its life cycle to inhibit replication. There are currently five different classes of HIV drugs. However, the most common include the Nucleotide Reverse Transcriptase Inhibitors (NRTIs) and Protease Inhibitors (PIs). For the purpose of this project, only the role of Protease Inhibitors in the treatment of an HIV infection will be explored.

Antiretroviral therapy is not perfect, and virus resistance to the drugs is common. When the HIV virus reproduces, it can create copies of itself that are imperfect. These mutations in the virus can lead to drug resistance and overall an increased difficulty to control HIV infection.

Description of the Data

The dataset was obtained from Kaggle and contains the information of 1000 HIV+ patients who underwent antiretroviral treatment with the same medication. All of the patients had only recently contracted HIV-1 and had never been treated for the virus before. As part of our variables we are given patient ID (arbitrary), a binary response status, viral load at the beginning of therapy (in log10 units), CD4 count at the beginning of therapy, protease (PR) nucleotide sequences, and reverse transcriptase (RT) nucleotide sequence. A more detailed explanation of the variables follows:

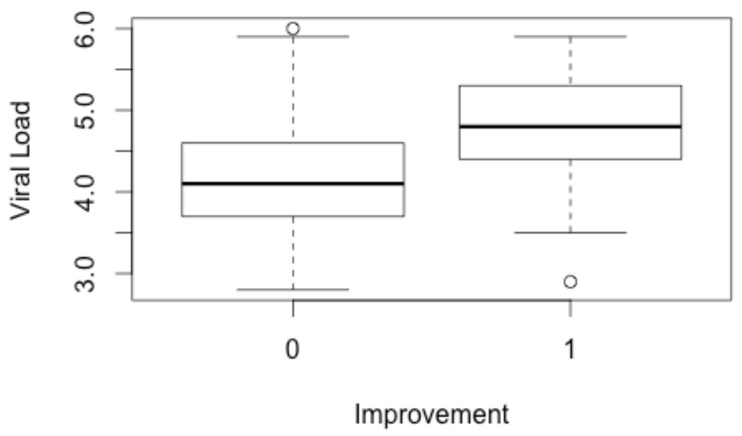
Response Variable (y)

The response variable is coded as "1" for patients who showed improvement after 16 weeks of therapy and "0" otherwise. We know that out of the 1000 that will be used to train our model 206 reacted positively to the antiretroviral drugs and showed improvement while 794 did not.

Viral Load (VL.t0)

The viral load, or viral burden, is a measurement of the amount of virus copies present in the patient's blood. It is also a measurement of the severity of a viral infection. This variable is used to determine the effectiveness of the treatment and improvement is defined as a 100-fold decrease in viral load for our response variable.

On the boxplot shown, we can observe that the cohort of patients that improved after ART follows a different distribution from the cohort of patients that didn't improve. The groups of patients that showed improvement were the ones with a higher average viral load, which means a more severe infection at the time of measurement.



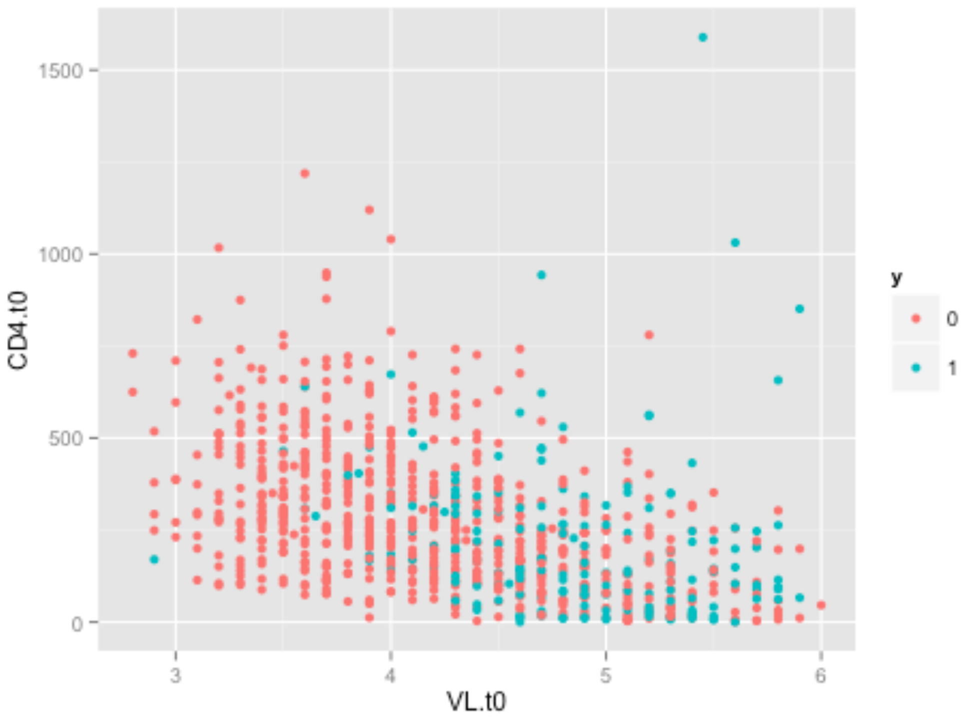
Cluster of Differentiation 4 Count (CD4.t0)

The cluster of differentiation (CD) 4 is a glycoprotein found on the surface of immune cells such as T helper cells. T helper cells are white blood cells and are essential to the immune system. CD4 T cells are in charge of sending signals to other type of immune killer cells about pathogens in the body. In short, they are in charge of alerting the body about a possible infection. HIV targets and destroys the CD4 cells in order to multiply inside the body. Therefore, a CD4 count is an important indicator of how well the immune system is working.

A quick summary of the distribution of the CD4 counts is shown below:

```
Min. 1st Qu.  Median    Mean  3rd Qu.    Max.
 0.0    132.8   249.0      279.6   383.2   1589.0
```

At a first glance, there seems to be no correlation between the response variable and the CD4 count. However, we can use our VL.t0 and CD4.t0 variables to visualize which patients are the ones that are improving in our current dataset.



This plot indicates that most of the patients that show improvement (in blue) are located in the lower right corner, which indicates high viral loads and small CD4 counts at the beginning of therapy. In other words, this indicates that the patients with the most severe infections and most impaired immune systems are the ones that are showing improvement to treatment.

Protease nucleotide Sequences (PR)

The next variable is the Protease nucleotide sequence. This variable contains a string of 297 characters or nucleotides that encode the PR protein. The HIV virus translates nucleotide sequences into amino acids and assembles them one after another in order to create proteins. During the replication by the HIV virus, the virus first replicates by creating a long string of joint proteins. The Protease is in charge of splitting these long strings, or polyproteins, in order to create functional components of the virus. The antiretroviral drugs target protease in order to interrupt the replication process of the virus and inhibit its replication.

I observed that we were missing the PR sequence for 80 instances and I have thus proceeded to exclude these from the analysis. In addition, I also observed 31 instances that have incomplete sequences with less than 297 nucleotides in them. In order to take full advantage of our data, I have decided to align these sequences and keep them as part of our dataset. I will give further explanation of this process in the next section.

Reverse Transcriptase Sequence (RT)

The second sequence is the one for the reverse transcriptase enzyme. The reverse transcriptase is in charge of transcribing the RNA of the virus into a complementary DNA sequence in order to copy the genetic material of the HIV. We observed a huge variability in the size of the RT sequences provided, and thus, we needed to align the sequences in order to be able to analyze variability at specific positions.

Data Cleaning and Preparation

The process of preparing and cleaning the data for the analysis was extensive. I first aligned all incomplete sequences so that we could look at the variability at each specific position in the sequence. I proceeded to break the sequences into codons (or 3-mers) and translated this into the respective amino acids they express. Next, I dealt with ambiguous/missing values in the data. I created a probabilistic process in order to replace these invalid entries with a valid value. Finally, I discarded all variables with no variability. A more detail description of my data cleaning process follows.

Alignment of Incomplete Sequences

I first began the process of cleaning our data by looking at both the PR and RT sequences. From the literature, we know that the complete HIV-1 PR gene has 297 nucleotides, and the RT gene has 1497. I identified all of the sequences that were incomplete and I proceed to align them. From the 920 patients remaining in our dataset, I identify that 31 had incomplete PR sequences. In contrast, almost all of the RT sequences are incomplete and there is a great variability in the length of the sequence across patients.

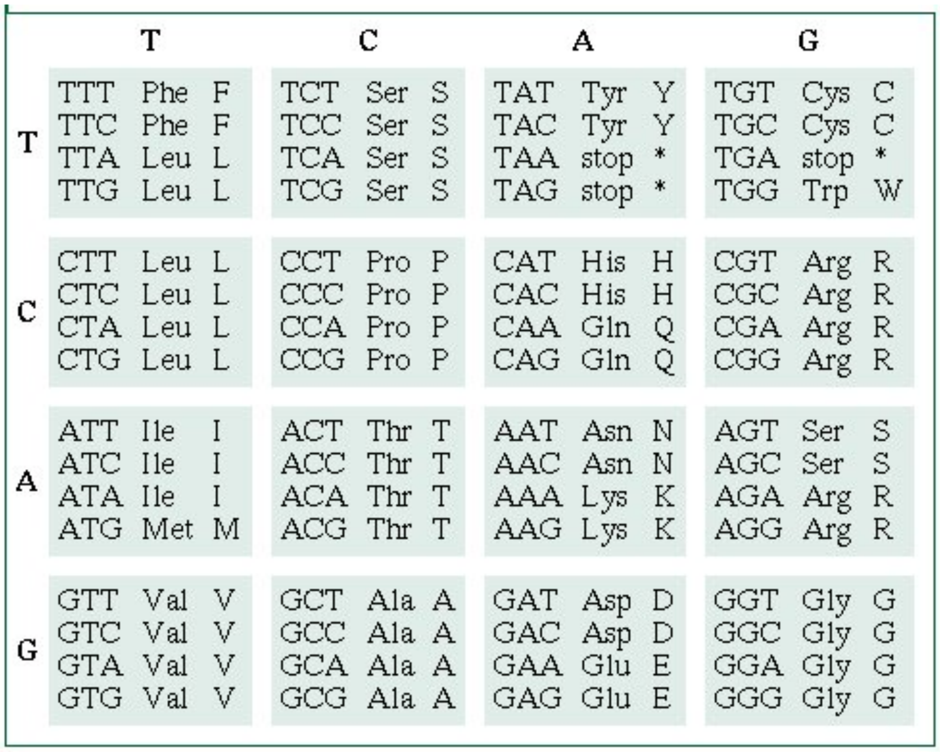
This variability in length in the RT sequence poses a great challenge to our analysis since it is computationally complicated to align a large number of sequences. As a result of this, I decided to concentrate my efforts on the PR sequence rather than RT sequence of the virus. Therefore, my analysis proceeds using only the PR sequence.

I proceed to align the incomplete PR sequences using BLAST, an algorithm that compares and aligns two or more genetic sequences. After having identified the correct positions of the incomplete sequences, compared to the complete gene, I filled the missing values with a dummy variable "X". The current training set reflects these changes. For the purpose of the next step, it is important to mention that the sequence was missing nucleotides in multiples of 3.

Translate Nucleotides into Amino Acids

The next step was to translate the current aligned nucleotide sequences into amino acid sequences. I started by grouping the nucleotides in the sequence by codons or (3-mers). Next I proceeded to find the corresponding amino acid expression for each specific codon. On the table shown, we can see the translation of each codon (set of three letters) to a specific amino acid directly to its right. There are 20 possible amino acids that can be coded from the genetic material of the virus and different combinations of nucleotides can lead to the same amino acid being expressed.

Codon-Amino Acid Translation Table



	T	C	A	G
T	TTT Phe F	TCT Ser S	TAT Tyr Y	TGT Cys C
	TTC Phe F	TCC Ser S	TAC Tyr Y	TGC Cys C
	TTA Leu L	TCA Ser S	TAA stop *	TGA stop *
	TTG Leu L	TCG Ser S	TAG stop *	TGG Trp W
C	CTT Leu L	CCT Pro P	CAT His H	CGT Arg R
	CTC Leu L	CCC Pro P	CAC His H	CGC Arg R
	CTA Leu L	CCA Pro P	CAA Gln Q	CGA Arg R
	CTG Leu L	CCG Pro P	CAG Gln Q	CGG Arg R
A	ATT Ile I	ACT Thr T	AAT Asn N	AGT Ser S
	ATC Ile I	ACC Thr T	AAC Asn N	AGC Ser S
	ATA Ile I	ACA Thr T	AAA Lys K	AGA Arg R
	ATG Met M	ACG Thr T	AAG Lys K	AGG Arg R
G	GTT Val V	GCT Ala A	GAT Asp D	GGT Gly G
	GTC Val V	GCC Ala A	GAC Asp D	GGC Gly G
	GTA Val V	GCA Ala A	GAA Glu E	GGA Gly G
	GTG Val V	GCG Ala A	GAG Glu E	GGG Gly G

I chose this approach to translate nucleotides into amino acids for two main reasons. The first is that this is how gene expression works. DNA and RNA sequences are translated into amino acids to form proteins, and biologically this makes the most sense. The second reason behind this choice is that it greatly helps us deal with the ambiguity of nucleotides.

Dealing with Ambiguous and Missing Values

There are four basic nucleotides that can be present in a sequence: Adenine, Cytosine, Guanine and Thymine. In biochemistry, the first letter in their name represents these four possible nucleotides. However, due to systematic error there is room for ambiguity. It is common that sequencing machines are not able to determine with certainty which of the four basic nucleic acids are present at a position in a sequence, so scientists have developed a representation for this ambiguity. This representation is shown next:

IUPAC Notation for Nucleotides

IUB code	Meaning
A	Adenosine
C	Cytidine
G	Guanine
T	Thymidine
R	G or A
Y	T or C
K	G or T
M	A or C
S	G or C
W	A or T
B	C, G or T
D	A, G or T
H	A, C or T
V	A, C or G
N	A, C, G or T (Any base)

The presence of ambiguity codes poses a challenge to the proper analysis of our data. First, it greatly increases the number of levels possible in each variable making our model more complicated. Second, if we choose to discard patients with ambiguous observations we lose around 80% of our instances. Therefore, it is crucial to address this ambiguity.

The first step I took in order to do this was to look for codons that only had an ambiguous code as the third nucleotide. This allowed me to identify codons that would translate to the same amino acid, regardless of the ambiguity in the third nucleotide. As an example, I identified TTY as a good candidate for this process. We know that Y can be translated to either T or C. Thus, the codon can be translated to TTC or TTT. However, when we look at the codon – amino acid table, we observe that both TTC and TTT translate to the amino acid "F". This process greatly helped reduce the number of ambiguous entries.

Still, there were ambiguous entries that can't be substituted using this methodology, so a second step was developed.

The second method starts with identifying all other ambiguous entries and creating a database of their possible amino acid expressions for future reference. Next, I identified all possible mutations that can be present in the HIV virus. The HIV Drug Resistance Database has a dataset of 52,000 patients who have been infected by the HIV-1 virus and have the PR sequence of the virus present in their bodies. This dataset gives us the wild type (or most common codon at each position) and the prevalence of mutations at any position.

We assume that our cohort of 920 patients is really similar to the 52,000 patients used in the prevalence dataset. We know that they are similar in the sense that they have been infected with HIV-1 (and not any other type of HIV) and none of the patients have received antiretroviral therapy before. This last fact is crucial since the HIV virus tends to mutate after treatment, so the prevalence of mutations is completely different for the pre-treatment and post-treatment cohorts. Consequently, we assume that the frequencies of mutations observed in the 52000 patients can be a direct measure of the probability of finding a mutation in any of our 920 patients.

I thus established a methodology to replace ambiguous codons in my training set. I first started by identifying an ambiguous codon in the training set and found the possible translations to amino acids. From this, we use the prevalence dataset and identify all possible amino acid mutations at that particular position. I continued by cross listing both the possible amino acids derived from the training dataset with the frequency dataset. Next, I randomly select an amino acid using the weighted probabilities from the prevalence dataset. Lastly, to deal with the missing values in the training data set, I simply randomly selected an amino acid out of all the possible mutations from the prevalence data set at each specific position in the sequence.

Further Data Preparation

After I replaced all ambiguous and missing values, I obtained a dataset with 920 instances and 102 variables. These variables include CD4.t0, VL.t0, binary response, and one variable for each specific position of the PR sequence (there's 297 nucleotides so we have 99 codons). To finalize preparing the data for the analysis, I discarded all of the variables that have no variability across observations (i.e. all the positions that have the same amino acids across all patients). I therefore ended up with a dataset with 920 instances and 80 variables (we discard 22 PR positions).

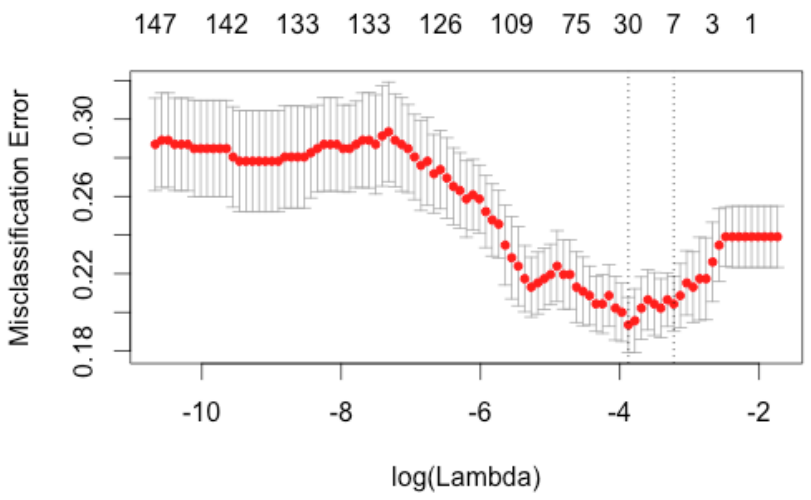
Model Description

After I had successfully prepared the data for analysis I continued to build the models. The first step I took was to identify a subset of variables that was related to the result. I did this by taking a random sample of 460 observations. Next, I analyzed the data to see if there are any observations that should be excluded. After that, I selected a new random sample of 460 to build my model and use the remaining observations as a test data set.

Logistic Regression Overview

The first step in building the logistics regression model was to be able to select a subset of variables. The current number of variables in the dataset is bigger than the number of observations (since each factor variable has many different levels), so I needed to exclude irrelevant variables in order to gain degrees of freedom. I began this process by taking a random sample of 460 observations out of our data set.

Next, I used the Lasso regression in order to select a subset of variables that was related to the result. I ran the Lasso using cross validation in order to determine the optimal lambda to minimize misclassification error. On the graph below, I observed the mean of the misclassification error as well as its standard deviation as a function of log(lambda).

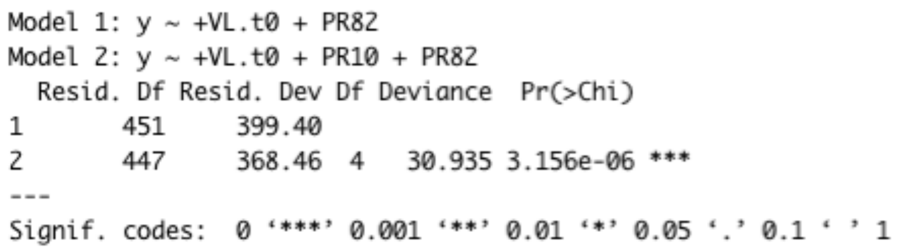


The first line represents the optimal level of lambda, and the number on top of the line represents the number of non-zeroed variables in the Lasso regression. Since I was trying to find the best subset of variables, I~~'ve~~ proceeded to choose the lambda in the second line, or lambda.1se. This lambda is able to produce a less complex model while maintaining misclassification within one standard error of the minimum. Attached is the output of the Lasso model with our lambda.1se:

(Intercept)	-6.51196160	PR12R	0.40193749
PR4T	-0.11168188	PR20R	-0.06502478
PR10I	-0.44755313	PR82V	0.26031621
PR10L	0.46467046	VL.t0	1.11775971

From this, we use the list of coefficients as a subset of variables that can be used in the logistic regression in order to build a classifier. I then proceeded to run a logistic regression with these variables. I first observed that some the levels in the factor variables are not statistically significant, so I cut all the variables that do not have any significant levels. Having done that, I have eliminated PR4, PR12 and PR20 from the model while keeping VL.t0, PR10 and PR82.

I proceeded to use a likelihood ratio test in order to compare models with and without the significant variables. By doing this, I also determined that I should keep VL.t0, PR10 and PR82. As an example, in the following output from the likelihood ratio test, we observe that the model with the 3 variables does fit significantly better than the model with only 2 variables, so I have chosen to keep the complex model.



```
Model 1: y ~ +VL.t0 + PR82
Model 2: y ~ +VL.t0 + PR10 + PR82
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1       451     399.40
2       447     368.46  4   30.935 3.156e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After determining the variables that will be used for our model, I proceeded to look at the data one last time in order to determine if there are any other observations that should be excluded from the dataset. I used the table function to find frequencies of each mutation and to identify that the PR82 variable has 4 levels that have one observation each, so I excluded them from the dataset in order to avoid over fitting the model.

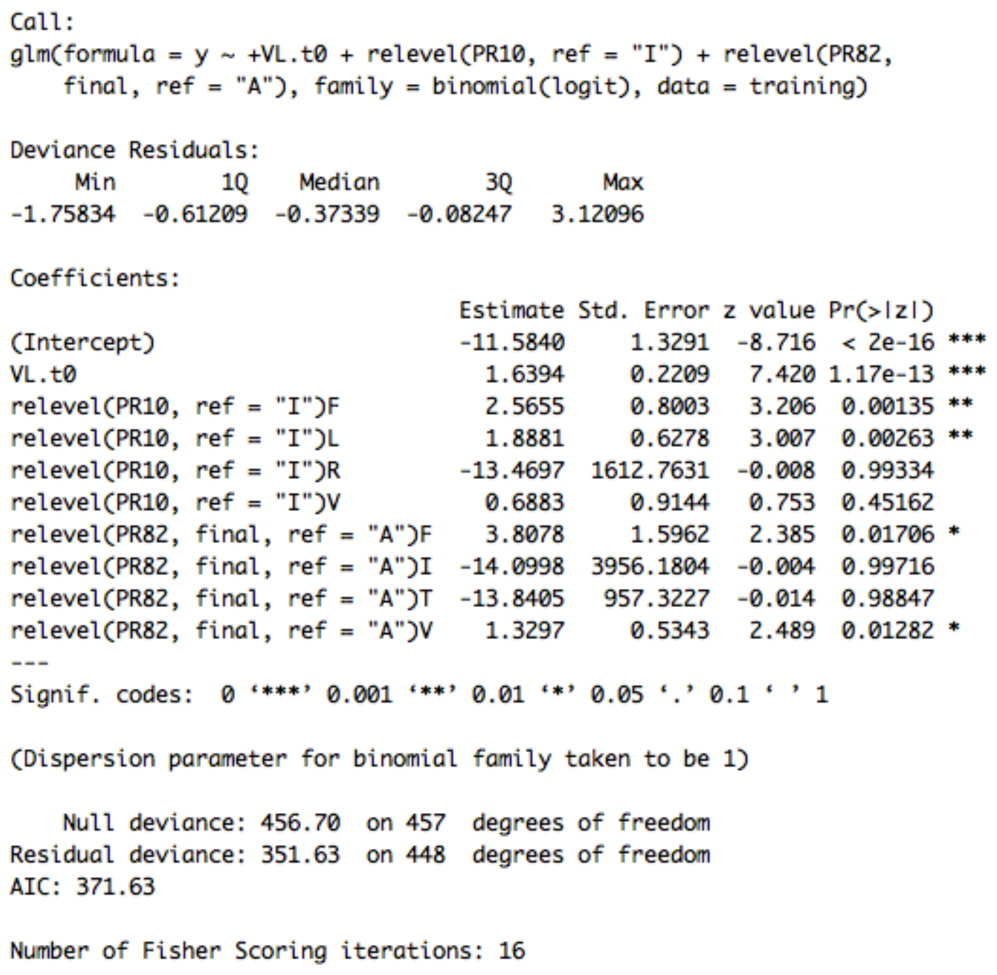
Logistic Regression Results

I proceeded to construct the model by selecting a new random sample of 458 out of the 916 observations in the dataset to use it as the training set. In order to better understand our output, I chose the most common amino acid for each position as a base level for the regression. The output for the model is shown below:

```
Call:
glm(formula = y ~ +VL.t0 + relevel(PR10, ref = "L") + relevel(PR82,
    ref = "V"), family = binomial(logit), data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.75834  -0.61209  -0.37339  -0.08247   3.12096

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -8.3662     1.0255  -8.158 3.40e-16 ***
VL.t0                        1.6394     0.2209   7.420 1.17e-13 ***
relevel(PR10, ref = "L")F    0.6774     0.5435   1.246  0.21261
relevel(PR10, ref = "L")I   -1.8881     0.6278  -3.007  0.00263 **
relevel(PR10, ref = "L")R  -15.3578  1612.7630  -0.010  0.99240
relevel(PR10, ref = "L")V   -1.1997     0.7057  -1.700  0.08911 .
relevel(PR82, ref = "V")A   -1.3297     0.5343  -2.489  0.01282 *
relevel(PR82, ref = "V")F    2.4780     1.5287   1.621  0.10501
relevel(PR82, ref = "V")I  -15.4295  3956.1803  -0.004  0.99689
relevel(PR82, ref = "V")T  -15.1702   957.3226  -0.016  0.98736
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 456.70  on 457  degrees of freedom
Residual deviance: 351.63  on 448  degrees of freedom
AIC: 371.63

Number of Fisher Scoring iterations: 16
```

From the output above, we can observe that VL.t0 is highly significant and an increase in viral load is consistent with improvement of patients. We also observe that PR10I is significant when compared to the base level L. PR10V is at the borderline of significance as well. I have chosen L as the base level for PR10 since this is the wild type, or the most common one. From this output, we can deduce that the patients who have amino acid I at the PR10 are less likely to improve compared to the patients who have the wild type in this position. We also observe that the patients who have mutation A in position 82 are less likely to improve compared to patients who have the wild type at this position.

More interestingly, we can change the contrast matrix of the regression in order to compare these levels against one another. I created a new model setting the mutation PR10I and PR82A as the new base levels. The output is as follows:



```
Call:
glm(formula = y ~ +VL.t0 + relevel(PR10, ref = "I") + relevel(PR82,
    final, ref = "A"), family = binomial(logit), data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.75834  -0.61209  -0.37339  -0.08247   3.12096

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                       -11.5840     1.3291  -8.716  < 2e-16 ***
VL.t0                               1.6394     0.2209   7.420 1.17e-13 ***
relevel(PR10, ref = "I")F           2.5655     0.8003   3.206  0.00135 **
relevel(PR10, ref = "I")L           1.8881     0.6278   3.007  0.00263 **
relevel(PR10, ref = "I")R         -13.4697  1612.7631  -0.008  0.99334
relevel(PR10, ref = "I")V           0.6883     0.9144   0.753  0.45162
relevel(PR82, final, ref = "A")F    3.8078     1.5962   2.385  0.01706 *
relevel(PR82, final, ref = "A")I  -14.0998  3956.1804  -0.004  0.99716
relevel(PR82, final, ref = "A")T  -13.8405   957.3227  -0.014  0.98847
relevel(PR82, final, ref = "A")V    1.3297     0.5343   2.489  0.01282 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 456.70  on 457  degrees of freedom
Residual deviance: 351.63  on 448  degrees of freedom
AIC: 371.63

Number of Fisher Scoring iterations: 16
```

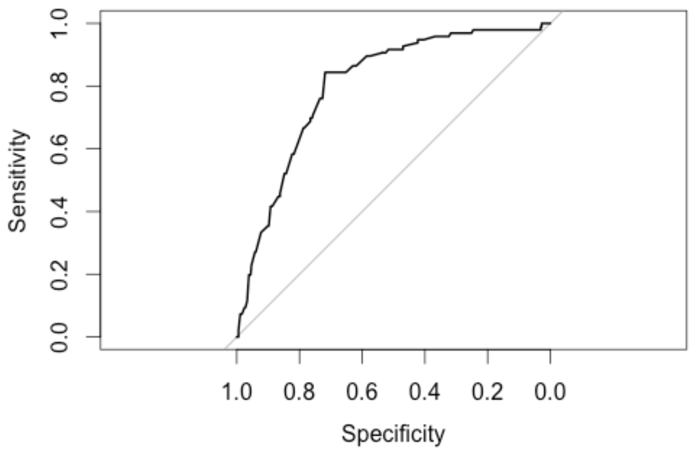
From this output, we can conclude that patients with the PR10F mutation are more likely to improve than those who have the “bad” PR10I mutation. We can also deduce that the patients who have the PR82F mutation are more likely to improve than ones with the “bad” PR82A. Ideally, I would have liked to create a hierarchy of mutations from least likely to improve to most likely to improve but, unfortunately, I don’t have significant enough results to do this. Next, I used the new model to classify our testing data and determine the test misclassification error. We obtain the following confusion matrix:

	Forecast No Improvement	Forecast Improvement
No Improvement	334	28
Improvement	64	32

The overall misclassification error from the model is 20.08%, way below the 50% threshold. We observe that we have a specificity of around 92% while the sensitivity to improvement is 33%. While we can use a different cost ratio in order to improve the sensitivity of the test, I have decided to proceed to use the model as is since it allows us to best identify the patients who will not improve in the current ART regime and should consider a different combination of antiretroviral drugs.

Finally, I have calculated the AUC and ROC of this model:

ROC of Logistic Regression Model



Area under the curve: 0.8034

Alternative Methods

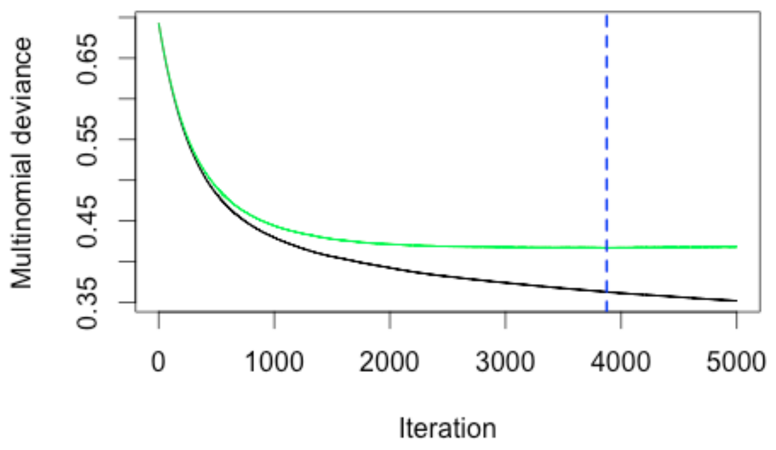
In this section, I will construct an alternative model to the logistic regression by using the boosting to classification trees. In order to make a valid comparison between these two methods I need to maintain consistency between the two models. Therefore, I will use the same training set of 458 observations that was used to build the logistic regression as well as the same test data set, to verify forecasting accuracy.

Overview of Boosting Method

Boosting is another method that can be used to develop and improve the results from a classification tree. As an overview, boosting consists of building several weak classifiers in order to create a single strong classifier. This process can be applied to classification trees, a process in which we partition the space of predictors and assign a value to each partition in order to classify observations.

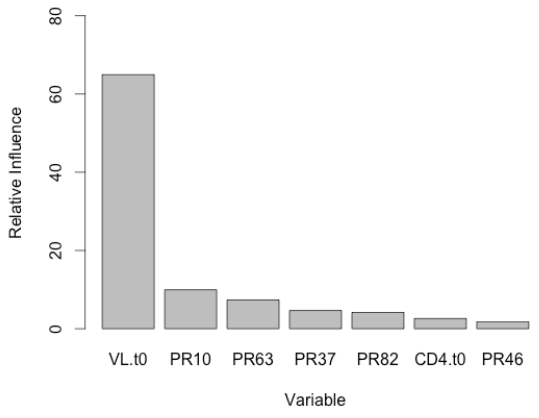
There is an important number of parameters in the GBM function that can be changed but the default parameters are recommended so I have proceeded to use them in order to create my alternative model. The only parameter that we could change is to find the best number of tree iterations in order to reduce the computation time. We know that after a certain point, adding trees to my model no longer improves the quality of my results.

I first ran my model with 5000 iterations and 10 cross validation folds and proceed to use the GBM performance function in order to estimate the optimal number of boosting iterations. The plot from the output of the GBM performance function tells us the number of iterations at which the deviance levels off and judging from the plot below we observe that the deviance levels off at around the 3800 iterations. In fact, the precise value is 3875 iterations.



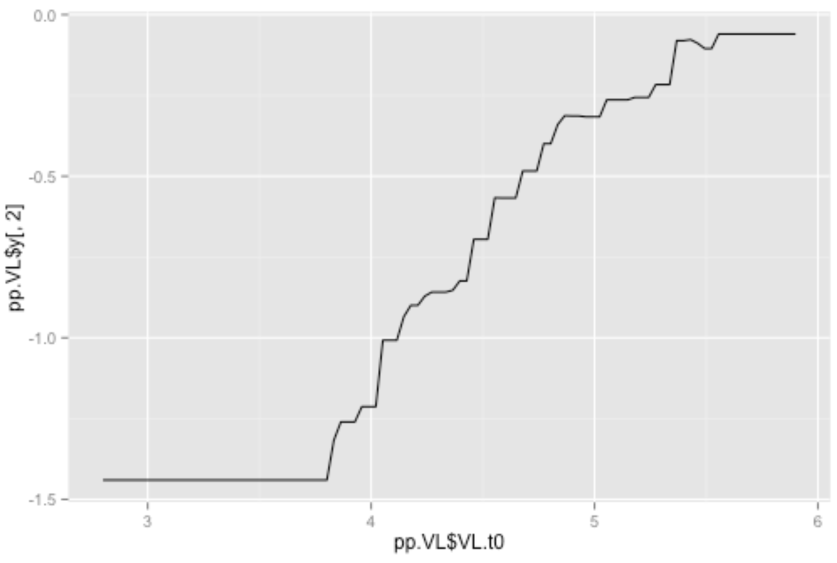
We proceed to change the "number of trees" parameter to 3875 iterations in order to minimize bias and avoid any further over fitting. Before we proceed to build our model, we analyze the relative influence of our variables in order to determine which ones contribute the most to the fit of the model. The following table shows the relative influence of the top variables. We know that any variable with a relative influence higher than 2 is significant so I have chosen to only present those ones.

Variable	Relative Influence
VL.t0	64.94
PR10	9.96
PR63	7.34
PR37	4.66
PR82	4.17
CD4.t0	2.60



Graphically, we can observe the relative influence of the variables in the following graph. The viral load at the beginning of treatment dominates in its contribution to the fit, accounting for almost 65% of the fit. We also observe that the PR10, PR63, PR37 and PR82 positions contribute to the fit. This is consistent with our findings from the Lasso Regression. However, it is important to note that we can't compare these findings directly with the Lasso regression since both the Lasso and boosting method deal with the data in different ways.

More interestingly, I proceeded to analyze the partial dependence plot of the viral load at the beginning of treatment compared to the binary response. We know that the partial dependence plot is a graphical representation of the marginal contribution of a variable to the class probability. From the graph below, we see that the probability of improvement increases with viral load, which is a finding that is consistent with our previous results. It is interesting to see that there is a flat region at the beginning of the graph in which there is no marginal increase probability of improvement as the viral load increases.

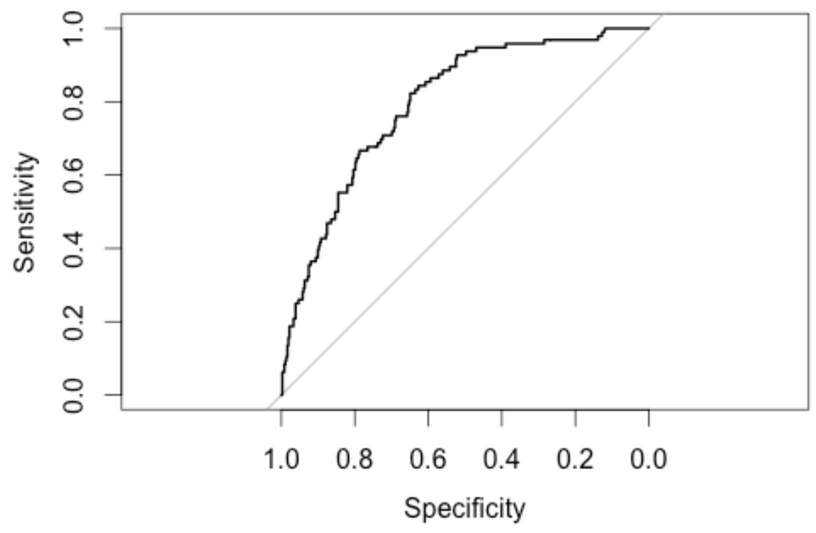


Results of Boosting Method

Having studied the relative influence of the different variables in the model, I proceeded to train the final model using the training set and 3875 iterations. For this, I used my test data in order to obtain the testing error of the model. We obtain the following confusion matrix:

	Forecast No Improvement	Forecast Improvement
No Improvement	348	14
Improvement	75	21

I calculated that the test misclassification error is of 19.43%, which is well below the 50% threshold. I also observed a specificity of 96% while we have sensitivity to improvement of 21%. We proceed to calculate that the area under the curve for this model is .7962. Finally, we show the plot of the AUC.



Comparison of Methods

Now I would like to explore the differences and similarities between the two methods developed. While the computational results of both methods are really similar, these models are almost virtually impossible to compare. The algorithms and methodology behind them are very different, so this makes a direct comparison almost impossible. However, I will discuss advantages and disadvantages of applying each model on this data.

We know that both of these models deal with very different tasks. They both do a pretty good job at classifying the data, and both of them have a similar test misclassification and area under the curve values. One of the advantages of the logistic regression is that we are able to visualize the effect of each mutation at a particular position and use the contrast matrix to compare different mutations against each other. On the other hand, the boosting method gives us a very graphic representation of the influence each variable has on the final prediction as well as marginal contributions of the variable to the class probability.

Ultimately, I have chosen the logistic regression model as the best choice since it is the simplest model out of the two and performs just as well on the test data. The option of being able to compare different mutations against one another in order to identify their effect is also really valuable since it allows us to identify good and bad mutations and the effect they have on patient health.

Validity of Results and Future Improvement

I needed to evaluate the validity of the results as well as to identify possible sources of error. I first started by discussing the external validity of the results. We know this dataset comes from a Kaggle competition, and it is a selection of HIV+ patients. We do not know how this sample of patient was selected and if it’s representative of the overall population of HIV patients. We also do not know what combination of drugs the patients are receiving, so this makes it difficult to generalize our results to the general population of patients. If this data collection were to be done again, it would be important to outline the process by which the patients were selected.

We also need to discuss the internal validity of this data set. We know that there is a big mistake when it comes to data since it does not provide a control group. We start this analysis from the assumption that the antiretroviral drugs should be working, we do not know if this is the case. Therefore, this hinders the internal validity of the results. If this study were to be repeated, it would be important to include a control group in order to evaluate the role of the antiretroviral drugs.

Another important source of error comes from the missing and ambiguous data. The method we developed to deal with this data is decent since it is able to use real probabilities of finding mutations in the virus. However, this is not entirely accurate. It would be better if we were able to determine with accuracy the genetic sequence of the virus.

Conclusion

From this analysis, we can conclude that the logistic regression predicts that patients with a higher viral load are more likely to improve in this dataset. We can also infer that the patients who have the wild type at PR10 and PR82 are more likely to improve compared to the ones who don't. The dataset has a lot of limitations, so it seems unreasonable to extend these results to patients outside this dataset. An important note about the results from this analysis is that Stanford's HIV Resistance Database indicates that mutations in PR82 are highly correlated to HIV resistance to antiretroviral drugs, which means that my analysis of the data may be generalized and my model is a good predictor for HIV progression.

Bibliography

"Codon Translation Table." *Programming in Bioinformatics*. N.p., n.d. Web. 21 Dec. 2014. <http://users-cs.au.dk/chili/PBI05/Exercises/codon_translation.html>.

"HIV/AIDS." *WHO*. World Health Organization, n.d. Web. 18 Dec. 2014. <http://www.who.int/mediacentre/factsheets/fs360/en/>.

"Mutation Prevalence According to Subtype and Treatment." *HIV Drug Resistance Database*. Stanford University, n.d. Web. 21 Dec. 2014. <http://hivdb.stanford.edu/cgi-bin/MutPrevBySubtypeRx.cgi>.

"Overview of HIV Treatments." *Overview of HIV Treatments*. AIDS.gov, n.d. Web. 19 Dec. 2014. <http://aids.gov/hiv-aids-basics/just-diagnosed-with-hiv-aids/treatment-options/overview-of-hiv-treatments/>.

"Patent EP2451977A1 - Genetic Markers Associated with Risk of Diabetes Mellitus." *Google Books*. N.p., n.d. Web. 21 Dec. 2014. <http://www.google.com/patents/EP2451977A1?cl=en>.

"Phenotypes of Common Patterns of Mutations at Position 82." *HIV Drug Resistance Database*. Stanford University, n.d. Web. 21 Dec. 2014. <http://hivdb.stanford.edu/cgi-bin/PositionPhenoSummary.cgi>.

"Predict HIV Progression." *Background*. Kaggle, n.d. Web. 21 Dec. 2014. <https://www.kaggle.com/c/hivprogression/details/Background>.

"Predict HIV Progression." *Description* -. Kaggle, n.d. Web. 21 Dec. 2014. <https://www.kaggle.com/c/hivprogression>.

Van Den Eede, P. "HIV-1 Genotyping of the Protease-reverse Transcriptase." *National Center for Biotechnology Information*. U.S. National Library of Medicine, n.d. Web. 21 Dec. 2014. <http://www.ncbi.nlm.nih.gov/pubmed/23821259>.

"Viral Load." *Viral Load*. AIDS.gov, n.d. Web. 21 Dec. 2014. <http://aids.gov/hiv-aids-basics/just-diagnosed-with-hiv-aids/understand-your-test-results/viral-load/>.