Multiple Linear Regression Analysis to predict the accuracy of computer-generated protein structure to a known structure of COVID-19 spike protein

Rayan Roy

Department of Statistics and Actuarial Science, University of Waterloo

December 11, 2020

Summary:

The objective of the report is to measure how close that computer-generated structure is to a known benchmark structure of the COVID-19 spike protein. This is indicated by the response variable *accuracy.*

Various predictors are analyzed out of the 686 predictors using VIF's, and different models have been created based on various criteria like stepwise forward selection with multiple penalties (including AIC, BIC) and ICM's. Finally, a 5-fold cross validation is conducted and RMSE scores are computed among various models to achieve the most optimal model for prediction.

The best fitted model has the search strategy of forward selection with search criteria of BIC. The model comprises of 116 predictors and has the lowest RMSE score among all six different models. We also found out that most predictors in the model comprised of atom type *aromatic* indicating that those predictors are influential in helping predict accuracy of the structure of the COVID-19 spike protein. Interestingly, we also saw that predictors containing type *scArgN* and *bbO* have the highest parameter estimates indicating that they are highly influential.

1 Exploratory Data Analysis

There are a total of 685 predictors provided in the training data set, with 1946 observations. Looking at the summary of the dataset, it is seen there are no missing values, and all the values of numerical type since they are counts of pair of atom types interacting at certain distance.

There is one predictor named *angles* that returns a score based on the configuration of angles in its structure and its values range from 113.8 to 139.2. The rest of the predictors are of *atom1_atom2* type and their values range from 0 to 100.

Each predictor has a unique name present in the dataset. The naming convention of the predictors is *atomtype1* and *atomtype2*, that indicate the names of different atoms types interacting at certain *distance* (like *vshort*, *medlong*). For instance, the variable name *carbonylC_bbCA_vshort* counts the number of pairs of atoms where one atom is *carbonylC* and other is *bbCA* are interacting at *vshort* distance.

From the summary statistics of protein data, it can be seen that the predictors containing the name *vshort* have really small count that range between 0 to 2. This indicates that there is very less count of pairs of atom types with very small atomic distance. We also see from the statistic, that as the *distance* increased gradually from *vshort* to *vlong*, higher number of atom types were seen to have interaction at those distance.

When we create a boxplot *(Fig. 1)* to see the distribution of scores based on configuration of angle (*angles),* we see that middle 50% of scores range from 122 to 127, with a few outliers present. Otherwise, the scores are normally distributed. Upon further exploration *(Fig. 2)* from histogram, we see the distribution of scores are actually normally distributed with no presence of outlier. Since the frequency of observation is so large, with wide variances in score, the histogram gives us a

more accurate depiction than the boxplot. We believe those few outliers wouldn't make a big difference in our overall prediction in the model.

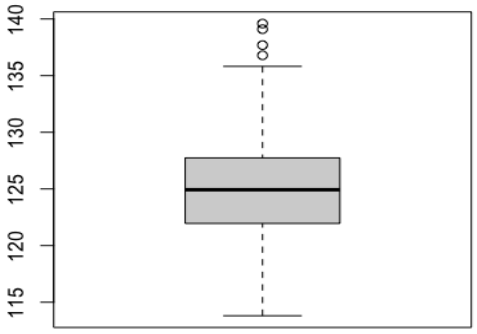

Figure 1: Boxplot shows that most of the scores lie between 122 to 127, with a few outliers present

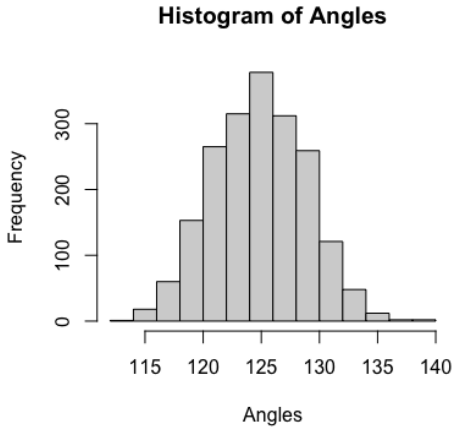

Figure 2: Histogram shows that scores based on configuration of the angles are normally distributed

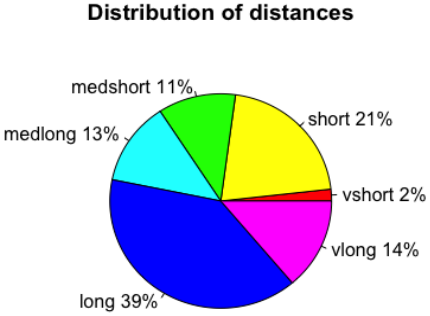

Figure 3: Pie chart shows the count of (in percentage) different distances types names in dataset

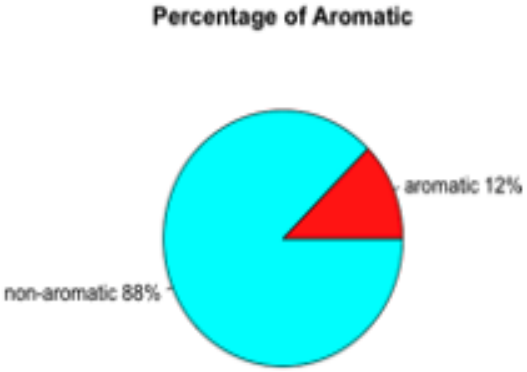

Figure 4: Comparison of number of predictors containing aromatic atom type and non-aromatic atom type

While exploring this data, we found out that majority of the distance between two atom types are long (*Fig.3).* This indicates that majority of protein folding might have some relation with higher distance which can be used for prediction. Another interesting fact was about *Aromatic* atom type. From our biochemistry knowledge, we know that carbonyl carbons and Nitrogen usually have the smallest bond length as they react very well and are closest in periodic table (called as aromatics) and act as protein folds drivers in human bodies (Krygowski & Szatylowics, 2015). We expect that majority of predictors in the final model would contain aromatic type as they are integral for protein folding, but it was only 12% (*Fig. 4)* of the total predictors that had aromatic atom type. But we believe that in the final model, there will be significant number of predictors that contain aromatic type as they usually are key for structural formation of proteins.

2 Methods

2.1 Removal of Predictors Using VIF and verifying key assumptions

Multiple Linear Regression (MLR) is used for predicting the variable *accuracy* since there are large number of predictors being used. The data set is loaded and set equal to *proteintrain.* Before conducting any model selection, two predictors: scArgN_bbC_medshort and scArgN_bbO_short are removed as there are perfectly multi-collinear. Multi-collinearity occurs when there is a strong linear relationship among predictors. In an MLR, a perfect multicollinear predictor can make the model inaccurate as variance of the predictors can become highly inflated.

Variance Inflation Factor (VIF) process is used on *proteintrain* data where any of the explanatory variables with VIF score of higher than 10 are removed. VIF assesses whether any of the predictors are correlated among each other. First the predictor with the highest VIF in the dataset is found, and then removed from the list of total predictors. Then the maximum VIF in the reduced model is found and then another predictor is removed based on that. This iterative process continues until a model is found with all the VIF's of the predictors in the model are less than 10. Through this process 115 predictors are removed from the dataset.

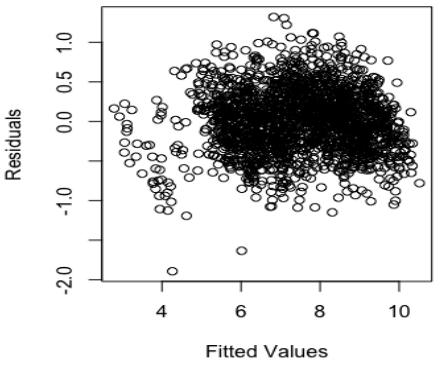

Figure 5: Scatterplot of Residual vs Fitted value shows points are randomly scattered

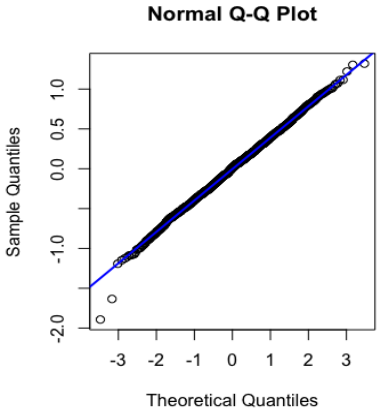

Figure 6: Normal QQ-plot after removal of predictors using VIF shows that Normality assumption is true with few heavy tail points in left

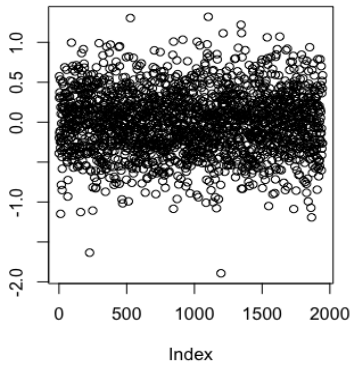

Figure 7: Residual vs Index Plot after removal of predictors using VIF shows the constant variance assumption is valid

Looking at the QQ-plot (*Fig. 6)*, we see no violation of the normality assumption as we would expect the observed distribution of the residual (sample quantiles) to roughly match the theoretical quantiles (i.e., the points would follow a 45-degree line). We do observe a slightly heavy tails on the left side, but it is quite close to the 45-degree line, thereby upholding the normality assumption.

From Residual vs Fitted Plot *(Fig. 5)*, we cannot see any patterns thereby holding the constant variance assumption. Similarly, from the Residual vs Index plot *(Fig. 7)* we see the points are scattered randomly indicating no violations of constant variance assumptions.

After removing all the predictors using VIF, the *proteintrain* dataset is split into 80/20 with 80% of data used for training (called *trainSet*) and 20% of data set used for validation (called *validSet*).

2.2 Various search strategies used

When we conduct regression analysis, the goal is to include all the predictors that are related to the dependent variable, *accuracy*, while also including as few predictors as possible because each irrelevant predictor decreases the precision of the estimated parameter estimate, thereby decrease the predictive power of the regression. The goal of variable selection is imperative in achieving a simplistic model while achieving a good fit. We use stepwise regression because there are many predictors, and we want to find the most useful set of predictors.

Although there are three types of stepwise regression namely forward selection, back elimination and forward/backward. The latter two aren't chosen as they are computationally expensive to work on large number of predictors. The forward selection is chosen which starts with an empty model and choses the best predictors and is the much faster than the other two.

We use BIC as a model selection criterion as compared to AIC because BIC accounts for the total length of the observations and the number of predictors as compared to AIC which only considers the number of predictors. Since we have a large dataset with lot of predictors, BIC would penalize more for having higher number of predictors. This is ideal as it helps us reduce the dimensionality (or the number of predictors) in our model. In other, words we can predict *accuracy* with a simpler model with less predictors.

The first search strategy to be implemented is forward selection with BIC as decision criteria (Appendix 1) onto the *trainSet* with the penalty of log(total number of rows of observation). The second search strategy is a forward selection with BIC as decision criteria, but the penalty is increased to twice that of first search strategy (2*log(no. of rows)) (Appendix 2). Similarly, for the third search strategy, first search strategy is repeated but instead with AIC as decision criteria and a penalty of 4 (which is less than log(1946= no. of observation) (Appendix 3). This was done to see if a lower penalty gives a lower RMSE on train and validation data than the BIC criterion.

We also use ICM as one of our search strategies which uses a stochastic method of determining the predictors. It usually gives a better BIC value with same penalty used in stepwise regression. In our fourth, fifth and sixth search strategy, we use ICM with BIC penalty (Appendix 4), ICM with twice the BIC penalty (Appendix 5) and ICM where penalty is 4 respectively (Appendix 6).

We use K-fold cross-validation to test if the model we got from ICM and stepwise regression can be generalized. This is a good technique because when we train any model on the training set, it tends to overfit most of the time, and in order to avoid this, we use techniques like cross-validation to see how it will be performing well on the actual test data. We do 5-fold cross-validation on the first three search strategies. We chose 5 because it would take less time to compute. We did the K-fold validation again but this including the three ICM models inside the K-folds

Lastly, the model with lowest RMSE score is chosen as the final model and used for prediction of the protein test dataset, which is the dataset who accuracy we are trying to predict.

3 Discussion and Results

3.1 Summary of Models and Key Findings

Below is the summary of results obtained by running various models:

Table 1: Summary statistics of various models (individual stepwise regression and ICM)

	Model 1- Forward (penalty BIC)	**ICM- Model 1 (penalty BIC)**	**Model 2- Forward (penalty 2*BIC)**	**ICM- Model 2 (penalty 2*BIC)**	**Model 3- Forward (penalty 4)**	**ICM- Model 3 (penalty 4)**
RSME of Validation	0.6172	0.6432	0.6772	0.6878	0.61707	0.6201
RSME of Train	0.4850	0.4985	0.5949	0.5820	0.4658	0.4439
AIC/BIC of Model	3039.501	2897.348	3505.675	3078.477	3032.292	2505.428
$\boldsymbol{R^2}$	0.89182	0.8856	0.8372	0.8440	0.9001	0.9093

Table 2: Summary statistics of stepwise regression and ICM done using K-fold Cross Validation

	Model 1- Forward (penalty BIC)	**ICM- Model 1 (penalty BIC)**	**Model 2- Forward (penalty 2*BIC)**	**ICM- Model 2 (penalty 2*BIC)**	**Model 3- Forward (penalty 4)**	**ICM- Model 3 (penalty 4)**
Mean RMSE	0.6129	0.6213	0.67178	0.7614	0.6095	0.6141

The RMSE measures the predictor error of a model. The RMSE on training is usually low because our model is already accustomed to the training data. A higher RMSE on validation than training indicates that we might want to increase the penalty to favour a simple model.

Looking at *Table 1*, we see the RSME on validation for Model 3 and Model 1 forward stepwise regression are the lowest out of all the 6 models used for comparison. Usually, ICM can typically find a better value of the criterion (e.g., AIC, BIC, L0 penalty) than stepwise as it searches more models. But it does not necessarily translate to better predictive RMSE, which the reason why we do cross validation to verify. This is what exactly happened in our case where we saw the ICM of each of the stepwise regression models higher than the RMSE of validation of the individual stepwise models.

We also observe the ICM has lower BIC value than individual stepwise models, and it makes sense since ICM uses a stochastic process to better fit the model. In both ICM and stepwise model, it is evident that models with less penalty have a lower AIC value as we see model 3 which has the lowest penalty has smallest AIC score as compared to model 2 with twice the penalty of BIC.

Although in model 3 with forward-stepwise regression we tried the penalty of 4, the RMSE of validation was almost identical to that of model 1 with penalty of BIC (log(number of rows)), the number of predictors in model 3 was way higher as compared to model 1. With such small difference in RMSE, it is not sensible to increase the number of predictors in the model. In fact, an ideal model should have simple model with less but accurate predictors which can be seen in model 1.

Furthermore, our observations are validated in the K-fold cross-validation (*Table 2)* where we see the mean RMSE of model 1 is almost identical to model 3 stepwise. As discussed earlier, having extra predictors for such tiny difference in RMSE score isn't ideal. Hence model 1 is the best fitted regression model.

Lastly, looking at the various R-squared values of model (*Table 1)*, we see that they increase as the penalty value decreases and are all relatively high. However, by just looking at R-square value, we cannot make inference about the RMSE scores or the predictive performance of the model, as there isn't any linear relationship among them.

3.2 Summary of Final Model and interesting observations

The final model we used chose was the BIC criterion with penalty = log(no. of rows). After fitting the model in the entirety of the train-protein dataset, we used 116 predictors in our final model to predict *accuracy*. The R-squared value of the final model was 0.898 and we think our model fitted the dataset well, although we can't be certain about RMSPE by just looking at R-squared value.

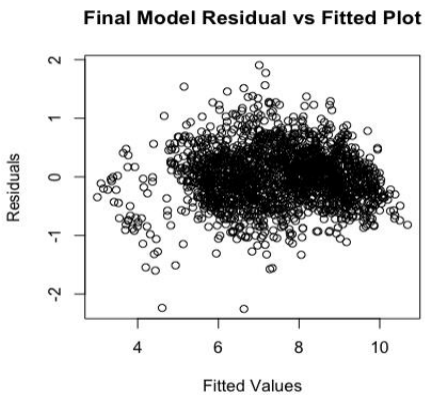

Figure 8: Residual vs Fitted value plot of points shows the constant variance assumption is true

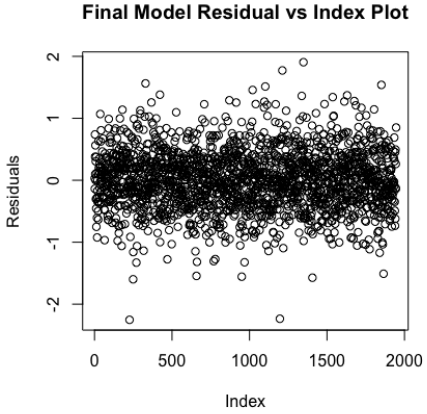

Figure 9: Residual vs Index Plot indicates shows the random scatter of points

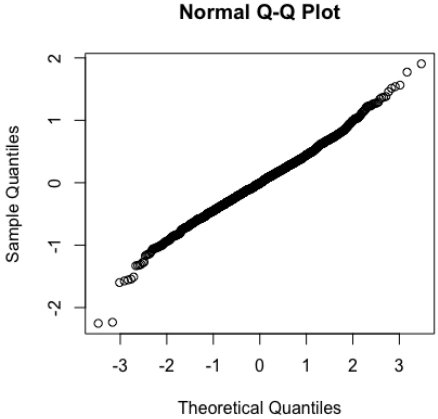

Figure 10: Normal QQ-plot shows normality assumption upheld

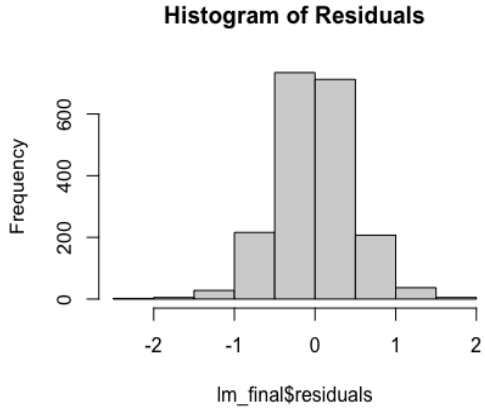

Figure 11: Histogram of Residuals illustrates that residuals are normal distributed with no outliers

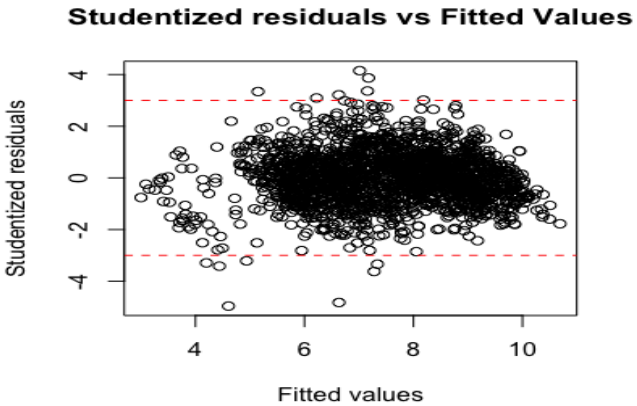

Figure 12: Studentized residual vs fitted value plot indicates that majority of observation are within [-3,3] with very few outliers.

We verify our assumptions of the final fitted model using various plots. Looking at *Fig. 8,* we see that residuals and fitted values are uncorrelated to each other as the data points are scattered randomly. We don't observe any obvious patterns in the Residual vs Index plot indicating no violation of constant variance assumption (*Fig. 9)*. The normality assumption also holds as we see the model is normally distributed (*Fig. 10),* although there are a few points that are outliers (that deviate from the standard 45-degree line). From *Fig. 11, a* bell-shaped pattern is observed when plotting the histogram of residuals indicating that residuals are normally distributed.

We then plot the studentized residual against the fitted value and see that that only 14 observations lie outside the [-3,3] range. Given there are 1946 observation, we can conclude over 99.7% of the studentized residual lie within the range and we believe that 14/1946~0.7% of observations that deviate won't make big impact. We can plausibly say that the constant variance assumption holds.

Although we had found some outlier earlier in our methods section, we felt it's fine to keep them, as they usually don't impact the predictors with such few in count. In fact, we believe it can lead to incorrect result if we remove them as those values can be important for the predictive performance of the model. We think there may legitimately be an interaction amongst the predictors if those outliers are kept and removing them may shift the predictive values. Overall, we believe like influential points are not usually a problem in the model, and their removal from dataset may either keep the parameter estimate unchanged.

From the 684 predictors that was initially given to us, we removed all the linear independent predictors that were insignificant and didn't contribute to explanatory power of the model. We used VIF to remove the predictors that had high multicollinearity among them. After implementing various search strategy, more predictors were removed that didn't contribute to the improving the predictive performance of the model.

Out of the 116 predictors that were used in the final model, not all the predictors are affecting the model significantly. We see that the p-value for hypothesis test for which the t value is the test statistic of predictor *scLysN_carboxylO_vlong, carbonylC_aromaticC_medshort, aliph1HC_ aliph3HC_ long* and *carboxylC_carboxylC_vlong* are 0.545, 0.307, 0.2 and 0.06 respectively and aren't significant at alpha = 0.05. This indicates that those four predictors aren't significantly adding anything to predictive performance of the model. Further investigation has to be done on those predictors to determine if they should be dropped or not. Those mentioned above are just a few examples of the predictors that are insignificant.

However, there are predictors that do affect the performance of the model significantly and are very important. These include like scArgN_bbO_medshort, scArgN_bbO_medlong with high predictor estimates.

Some of the interpretation of the important predictors are:
1. scArgN_bbO_medshort: For every additional pair of atom type *scArgN* and *bbO* interacting at *medshort* distance, the estimated change in expected accuracy is 0.681 while holding all other predictors constant (significant at alpha = 0.05, p-value = 0.0000317)
2. scArgN_bbO_medlong: The estimated change in expected accuracy is 0.9314 for every additional pair of atom type *scArgN* and atom type *bbO* interacting at *medlong* distance, while

holding all other predictors constant (significant at alpha = 0.05, p-value = 0.0000231).This predictor estimate is the most influential predictor to estimate *accuracy* since it has a highest coefficient.

3. aliph3HC_hydroxylO_medshort: For every increase in pair type *aliph3HC* and *hydroxylO* interacting at *medshort* distance, the estimated change in expected accuracy is -0.0672 holding all other predictors constant (significant at alpha = 0.05, p-value = 0.0000026).

4. carbonylC_aromaticC_short: For every addition in pair of atoms with type *carbonylC* and *aromaticC* interacting at short distance, the estimated change in accuracy is 0.1024 holding all other explanatory variables constant (significant at alpha = 0.05, p-value = 0.0002486).

We believe that our final model should fit the protein test set well. The final model had passed all the normality and independence assumptions. The RMSE value we achieved in our final model tested on proteintrain dataset was 0.6129 and when we did a 5-fold cross validation the final model had the lowest RMSE score of all the various models we had tested. Since our model hadn't seen the validation dataset before, we expect that the model would perform similarly on the protein-test dataset. If we do apply the fitted model on the new protein-test dataset, we expect the MSPE to be around 0.33 to 0.43 range. We got this based on the estimated relative error of 0.05 and absolute error of 0.3756 (0.6129^2) which is close to our validation MSE.

There were several interesting things we observed from final model. We had assumed earlier during our EDA that aromatic compounds should play a key role for protein folding, and it was indeed the case where we saw quite a few predictors containing the atom type *aromatic*. We expected that predictor *angles* would be included in our model as the score on the configuration of the structures of two different atom types should have an impact on the protein folded accuracy. Additionally, we had thought that ICM would be a much better search strategy since it uses a stochastic search for the different penalties like BIC/AIC, however it wasn't the case with ours. It was quite surprising to see highest parameter estimates of predictors containing atom type *scArgN* and *bbO*, indicating a higher influence on the prediction performance of the model but as based on prior biochemistry knowledge we thought that *aromatic* type would be most influential predictor.

In summary, the fitted model chosen was model with search strategy BIC criterion with penalty of log(number of row). For predicting the *accuracy*, we should definitely consider the predictors containing combinations of *scArgN, bb0, aromatic* to help generate the computer structure as we believe they are most important factors for determining the overall protein fold structure. The other predictors are important as well that can contribute to enhancing the predictive accuracy.

4 Reference

Krygowski, T M, and H Szatylowicz. “Aromaticity: what does it mean?.” Chemtexts vol. 1,3 (2015): 12. doi:10.1007/s40828-015-0012-2

5 Appendix

#Rcode for Protein Train data

```
options(max.print = 20000000)
set.seed(12345678)
library(MASS)
library(car)
library(dplyr)
trainprotein <- read.csv("/Project/protein-train.csv")

#########################################################

# ************ Exploratory Data Analysis ************

# Box Plot and histogram of Angles
boxplot(trainprotein$angles,)
hist(trainprotein$angles, xlab="Angles", main = "Histogram of Angles")

# Finding the distribution of aromatic and non aromatic atom types
aromatic_count <- ncol(select(trainprotein,contains("aromatic")))
non_count <-684 - aromatic_count
slices <- c(aromatic_count, non_count)
lbls <- c("aromatic", "non-aromatic")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, col=rainbow(length(lbls)),
   main="Percentage of Aromatic")

# Distribution of various types of distances types in predictors
vshort <- ncol(select(trainprotein,contains("vshort")))
short <- ncol(select(trainprotein,contains("short")))
medshort <- ncol(select(trainprotein,contains("medshort")))
medlong <- ncol(select(trainprotein,contains("medlong")))
long <- ncol(select(trainprotein,contains("long")))
vlong <- ncol(select(trainprotein,contains("vlong")))

# Pie Chart
slices <- c(vshort, short, medshort, medlong, long,vlong)
lbls <- c("vshort", "short", "medshort", "medlong", "long", "vlong")
```

```
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, col=rainbow(length(lbls)),
   main="Distribution of distances")
#########################################################

# Methods
(summary(trainprotein)) #Indicates no singularity present (no perfect multicollinearity)
min(trainprotein$angles)
table(sum(is.na(trainprotein))) #No empty values present
full <- lm(accuracy ~., data = trainprotein)
summary(full)
alias(full)
trainprotein$scArgN_bbC_medshort <- NULL
trainprotein$scArgN_bbO_short <- NULL
#########################################################

# Methods
# VIF Calculations
# Segment dataset into half
s1 <- lm(trainprotein[,1] ~ ., trainprotein[, 2:301])
s2 <- lm(trainprotein[,1] ~ ., trainprotein[, 302:684])

which.max(vif(s1)) #carbonylC_aliph3HC_medshort

s2 <- lm(trainprotein[,1] ~ . -carbonylC_aliph3HC_medshort, trainprotein[, 2:301])
vif(s2)
which.max(vif(s2)) #aliph2HC_bbC_medlong

s3 <- lm(trainprotein[,1] ~ . -carbonylC_aliph3HC_medshort-aliph2HC_bbC_medlong,
trainprotein[, 2:301])
which.max(vif(s3)) #carboxylC_aliph2HC_medshort

# Continue this process for s1 and s2 and then combine for both s1 and s2 (call it s3)
# and repeat the VIF process until all predictors have VIF that are less than 10

# Final Predictors with VIF < 10 present in the model. A total of 115 predictors removed
which.max(vif(vif_model)) #aliph2HC_scLysN_long
max(vif(vif_model)) #9.880524

# Residual vs Fitted Plot
plot(vif_model$fitted.values, vif_model$residuals, xlab="Fitted Values", ylab="Residuals")

# Residual vs Index Plot
plot(1:nrow(trainprotein), vif_model$residuals, xlab = "Index", ylab = "Residuals")
```

```
#QQ Plot of Residuals
qqnorm(vif_model$residuals)
qqline(vif_model$residuals, col="blue", lwd=2)

# Hence we will remove the 115 explanatory variables
new_dataset <- trainprotein

# We will split the new_dataset in 80/20 with 80% allocated to training and rest to validation
N <- nrow(new_dataset)
trainInd <- sample(1:N , round(N *0.8), replace = F)
trainSet <- new_dataset[trainInd ,]
validSet <- new_dataset[ - trainInd ,]

#########################################################

# ******* APPENDIX 1 **********************************

# Model number 1 : We will run stepwise forward with BIC with the penalty log(N)

full <- lm(accuracy ~., data = trainSet)
empty <- lm(accuracy ~ 1, data = trainSet)

m_aic <- stepAIC(object= empty, scope = list(upper = full, lower = empty), direction =
"forward", k = log(nrow(trainSet)))
summary(m_aic)

AIC(model_1, k = log(nrow(trainSet)))  #3039.501
BIC(model_1) #3039.501 Both models matched
summary(model_1)$r.squared #0.8918248
summary(model_1)$adj.r.squared #0.8843952
pred9 <- predict(model_1, newdata = validSet)
sqrt(mean((validSet$accuracy - pred9)^2)) # RMSE on validation we get is 0.6171842
sqrt(mean(model_1$residuals^2)) # RMSE on train we get is 0.4849922

#########################################################################

#******************** APPENDIX 2 *********************************
# Model number 2: We will run stepwise forward with BIC with the penalty 2log(N)

stepAIC(object = empty, scope = list(upper = full, lower = empty), direction = "forward", k =
2*log(nrow(trainSet)))

AIC(model_2, k = 2*log(nrow(trainSet)))  #3505.675
BIC(model_2) #3505.675 Both models matched
summary(model_2)$r.squared #0.8372157
```

```
summary(model_2)$adj.r.squared #0.8325893
# Calculating the RMSE scores for model 2
pred2 <- predict(model_2, newdata = validSet)
sqrt(mean((validSet$accuracy - pred2)^2)) #RMSE on validation we get is 0.6772338
sqrt(mean(model_2$residuals^2)) #RMSE on train we get is 0.5949453

#########################################################################

#### ******************** APPENDIX 3 *********************************

# Model number 3: We will run stepwise forward with AIC criterion with the penalty 4
stepAIC(object= empty, scope = list(upper = full, lower = empty), direction = "forward", k = 4)
AIC(model_3, k = 4) #3032.292
summary(model_3)$r.squared #0.9000784
# Calculating the RMSE scores for model 3
pred_3 <- predict(model_3, newdata = validSet)
sqrt(mean((validSet$accuracy - pred_3)^2)) #RMSE on validation we get is 0.6170718
sqrt(mean(model_3$residuals^2)) #RMSE on train we get is 0.4658136

#########################################################################
# Running ICM model:

ICM <- function(trainSet,penalty) {
  pen <- penalty
  varlist = c()
  varnames = names(trainSet)
  n = nrow(trainSet)
  varorder <- sample(1:ncol(trainSet)) # random order of variables
  minCrit = Inf
  noChange = F
  while (!noChange) {
   noChange = T
    for (i in varorder) {
     if (i == 1)  #because col 1 is accuracy
       next
     if (i %in% varlist & length(varlist) > 1) {
       index = c(1, varlist[varlist != i])
       trainVars = trainSet[, index]

       fit = lm(accuracy ~ ., data = trainVars)

       if (AIC(fit, k = pen) < minCrit) {
         minCrit = AIC(fit, k = pen)
         varlist = varlist[varlist != i]
         print(paste0("Criterion: ", round(minCrit, 1), ", variables: ", paste0(varnames[varlist],
collapse = " ")))
```

```
        best.model = fit
        noChange = F
       }
      } else if (!i %in% varlist) {
       index = c(1, varlist, i)
       trainVars = trainSet[, index]

       fit = lm(accuracy ~ ., data = trainVars)

       if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = c(varlist, i)
        print(paste0("Criterion: ", round(minCrit, 1), ", variables: ", paste0(varnames[varlist],
collapse = " ")))
        best.model = fit
        noChange = F
       }
     }
   }
  }
 return (best.model)
 }

#### ******************** APPENDIX 4 **********************************
#ICM for model 1
model1 <- ICM(trainSet,log(nrow(trainSet)))
summary(model1)
AIC(model1, k = log(nrow(trainSet))) #3132.292
BIC(model1) #3132.292 Both models matched
summary(model1)$r.squared #0.9000784
pred_1_ICM <- predict(model1, newdata = validSet)
sqrt(mean((validSet$accuracy - pred_1_ICM)^2)) # RMSE on validation 0.6431905
sqrt(mean(model1$residuals^2)) # RMSE on train 0.498459

#### ******************** APPENDIX 5 **********************************
#ICM for model 2
model2_icm <- ICM(trainSet,2*log(nrow(trainSet)))
summary(model2_icm)
AIC(model2_icm, k = 2*log(nrow(trainSet))) #3132.292
BIC(model2_icm) #3132.292 Both models matched
summary(model2_icm)$r.squared #0.9000784
pred_2_ICM <- predict(model2_icm, newdata = validSet)
sqrt(mean((validSet$accuracy - pred_2_ICM)^2)) # RMSE on validation 0.6878186
sqrt(mean(model2_icm$residuals^2)) # RMSE on train 0.5819979

#### ******************** APPENDIX 6 **********************************
```

```
#ICM for model 3
model3_icm <- ICM(trainSet,4)
summary(model3_icm)
AIC(model3_icm, k = 4) #3132.292
summary(model3_icm)$r.squared #0.9000784
pred_3_ICM <- predict(model3_icm, newdata = validSet)
sqrt(mean((validSet$accuracy - pred_3_ICM)^2)) # RMSE on validation 0.6200904
sqrt(mean(model3_icm$residuals^2)) # RMSE on train 0.4438868


#############################################################
# K-fold cross validation

N <- nrow(trainprotein)
K <- 5
validSetSplits <- sample((1:N)%%K + 1)
RMSE1 <- c() #Model_1 stepwise
RMSE2 <- c() #Model_2 stepwise
RMSE3 <- c() #Model_3 stepwise
RMSE4 <- c() #Model_1 ICM
RMSE3 <- c() #Model_2 ICM
RMSE3 <- c() #Model_3 ICM

for (k in 1:K) {
  validSet <- new_dataset[validSetSplits==k,] #Applying the model on the data with removed
predictors of VIF
  trainSet <- new_dataset[validSetSplits!=k,]

  full <- lm(accuracy ~ ., data = trainSet)
  empty <- lm(accuracy ~ 1, data = trainSet)

  m1 <- stepAIC(object = empty, scope = list(upper = full, lower = empty),
                direction = "forward", k = log(nrow(trainSet)))
  predk1 <- predict(m1, newdata = validSet)
  RMSE1[k] <- sqrt(mean((validSet$accuracy - predk1)^2))

  m2 <- stepAIC(object = empty, scope = list(upper = full, lower = empty),
                direction = "forward", k = 2*log(nrow(trainSet)))
  predk2 <- predict(m2, newdata = validSet)
  RMSE2[k] <- sqrt(mean((validSet$accuracy - predk2)^2))

  m3 <- stepAIC(object = empty, scope = list(upper = full, lower = empty),
                direction = "forward", k = 4)
  predk3 <- predict(m3, newdata = validSet)
  RMSE3[k] <- sqrt(mean((validSet$accuracy - predk3)^2))
```

```
  m4 <- ICM(trainSet,log(nrow(trainSet)))
  predk4 <- predict(m4, newdata = validSet)
  RMSE4[k] <- sqrt(mean((validSet$accuracy - predk4)^2))

  m5 <- ICM(trainSet,2*log(nrow(trainSet)))
  predk5 <- predict(m5, newdata = validSet)
  RMSE5[k] <- sqrt(mean((validSet$accuracy - predk5)^2))

  m6 <- ICM(trainSet,4)
  predk6 <- predict(m6, newdata = validSet)
  RMSE5[k] <- sqrt(mean((validSet$accuracy - predk6)^2))

}

RMSE1  # 0.5883980 0.5804227 0.6432637 0.6167076 0.6356070
mean(RMSE1)  #0.6128798
RMSE2 # 0.6757525 0.6384283 0.6923043 0.6767486 0.6753434
mean(RMSE2)  #0.6717154
RMSE3 # 0.5839171 0.6130055 0.6234234 0.5907642 0.5867903
mean(RMSE3) #0.60952342
RMSE4 # 0.654191 0.6145002 0.645601 0.589045 0.613904
mean(RMSE4) #0.6213211
RMSE5 # 0.780323 0.7459023 0.760291 0.77398 0.784210
mean(RMSE5) #0.761401
RMSE6 # 0.60792 0.624500 0.68703 0.57345 0.612147
mean(RMSE6) #0.6141432

# It turns out that the model_1 with forward stepwise regression is the best procedure among all
# the models based on CV prediction error.

#####################################################################

newfull <- lm(accuracy ~., data = trainprotein)
newempty <- lm(accuracy ~ 1, data = trainprotein)

final_model <- stepAIC(object= newempty, scope = list(upper = newfull, lower = newempty),
direction = "forward", k = log(nrow(trainSet)))
summary(final_model)

lm_final <- lm(formula = accuracy ~ [all predictors given my final model])

summary(lm_final)
length(coef(lm_final))-1

#QQ-Plot of Residual
qqnorm(lm_final$residuals)
```

```
qqline(lm_final$residuals, col =" blue ", lwd = 2)

# Residual vs Fitted Plot
plot(lm_final$fitted.values, lm_final$residuals, main= "Final Model Residual vs Fitted Plot",
xlab="Fitted Values", ylab="Residuals")

# Residual vs Index Plot
plot(1:nrow(trainprotein), lm_final$residuals, main= "Final Model Residual vs Index Plot",xlab
= "Index", ylab = "Residuals")

# Histogram of Residuals
hist(lm_final$residuals, main = "Histogram of Residuals")

# Studentized Residual
plot(fitted(lm_final), rstudent(lm_final), main = "Studentized residuals vs Fitted Values", xlab ="
Fitted values ", ylab ="Studentized residuals" )
abline( h = c (3 , -3) , col = "red ", lty =2)
which(abs(studres(lm_final)) > 3)

testprotein <- read.csv("/Project/protein-test.csv")

prediction <- predict(lm_final, newdata = testprotein)
writeLines(as.character(prediction), "mypreds.txt")

library(broom)
library(knitr)
values_produce <- tidy(lm_final)
kable(values_produce)
```

The list below contains all the predictors and their estimate in the final model lm_final

Predictor	Estimate	Std.error	statistic	p-value
(Intercept)	1.8057783	0.5098072	3.5420806	0.0004069
aliph1HC_aliph2HC_long	0.0376595	0.0070814	5.3180674	0.0000001
scLysN_bbC_vlong	-0.1022802	0.0160802	-6.3606245	0
scArgN_bbN_medlong	0.2475479	0.060549	4.0883883	0.0000453
aliph2HC_bbN_medshort	-0.0355636	0.0058891	-6.0388726	0
aliph1HC_aromaticC_medshort	0.0539186	0.0119249	4.5215185	0.0000065
sulfur_bbC_vlong	-0.0316254	0.0062438	-5.065063	0.0000004
bbC_bbC_medshort	-0.0359049	0.0087413	-4.1074862	0.0000418
aliph1HC_aromaticC_vlong	0.0723108	0.0074274	9.7356695	0
aliph3HC_scArgN_medlong	0.3598207	0.0504446	7.1329843	0
aromaticC_hydroxylO_medlong	-0.0335766	0.0067787	-4.9532425	0.0000008
sulfur_bbC_medlong	-0.0629096	0.0091741	-6.8572724	0

aliph2HC_aromaticC_vlong	0.0189571	0.0024178	7.8407254	0
carboxylC_bbN_vlong	0.0414063	0.0120323	3.4412628	0.000592
aliph1HC_aromaticC_medlong	0.0820176	0.0083005	9.8810644	0
carbonylC_aromaticC_medshort	-0.0106272	0.0104078	-1.0210738	0.3073545
carboxylC_aromaticC_long	0.0417484	0.0138464	3.015115	0.0026042
aliph1HC_aromaticC_long	0.0721939	0.0075324	9.5844192	0
bbN_bbCA_medlong	-0.0522906	0.0042415	-12.3284	0
aliph1HC_aliph1HC_vlong	0.0460553	0.0240631	1.9139354	0.0557843
scAGN_bbN_long	0.0378714	0.0057445	6.5926479	0
aromaticC_hydroxylO_long	-0.0193506	0.0053523	-3.6153777	0.000308
aliph1HC_bbO_long	-0.0709068	0.0082821	-8.561476	0
carboxylC_bbN_long	0.0631773	0.0153621	4.1125482	0.0000409
aliph3HC_aliph3HC_short	0.0356267	0.0186103	1.9143532	0.0557309
aliph3HC_bbN_short	0.1163167	0.0159601	7.2879847	0
aliph1HC_bbO_medlong	-0.0751657	0.0097868	-7.6803041	0
scLysN_carboxylO_long	0.1157428	0.0277728	4.1674854	0.0000322
aliph1HC_bbProN_medlong	0.3903635	0.0508018	7.684054	0
aromaticC_sulfur_long	0.0478665	0.0088597	5.4026997	0.0000001
scLysN_carboxylO_vlong	0.0174417	0.0288357	0.6048638	0.5453445
scAGN_bbN_medlong	0.0244998	0.0071273	3.4374693	0.0006003
carbonylC_bbC_medlong	-0.0427097	0.0079154	-5.3957895	0.0000001
aliph2HC_bbN_medlong	0.0257703	0.0037051	6.955278	0
aliph3HC_bbC_vlong	0.0159526	0.0037346	4.2715258	0.0000204
bbCA_bbO_vshort	-0.4567218	0.0612896	-7.4518672	0
bbN_bbC_vlong	-0.0158633	0.0026025	-6.0953599	0
aliph3HC_bbN_medlong	0.0159481	0.0055936	2.8511415	0.0044051
aliph2HC_hydroxylO_long	0.0148944	0.0056165	2.6519166	0.0080728
aromaticC_bbO_vlong	0.0118238	0.002417	4.8918843	0.0000011
scAGN_carbonylO_medlong	0.059571	0.0145189	4.1029811	0.0000426
aliph1HC_aliph3HC_long	0.0163683	0.0126168	1.2973451	0.1946761
aliph1HC_sulfur_short	0.251407	0.0422542	5.9498662	0
aliph2HC_aliph3HC_vlong	0.0102579	0.0042049	2.4395046	0.0148019
carbonylC_aromaticC_short	0.1024477	0.0279076	3.6709632	0.0002486
carbonylC_bbProN_medlong	-0.0922733	0.0252407	-3.6557309	0.0002637
aliph1HC_hydroxylO_vlong	0.0728563	0.014408	5.0566675	0.0000005
aliph2HC_aliph2HC_short	0.0520215	0.0163083	3.1898849	0.0014474
aliph1HC_aromaticC_short	0.0920435	0.0226066	4.0715223	0.0000487
scArgN_bbO_vshort	-0.353703	0.0705751	-5.0117243	0.0000006
carbonylC_sulfur_medshort	0.0666023	0.0196958	3.3815491	0.000736

carbonylC_bbN_vlong	-0.0202136	0.0052286	-3.8659771	0.0001145
aliph2HC_bbN_vlong	0.011788	0.0023067	5.1103395	0.0000004
bbO_bbO_long	-0.0166188	0.0044174	-3.7621385	0.0001738
aromaticC_hydroxylO_medshort	-0.0345974	0.0090823	-3.8093366	0.0001439
aliph2HC_aromaticC_medshort	0.0187337	0.0044814	4.1803109	0.0000305
bbN_bbO_vlong	0.0106814	0.0027512	3.8823917	0.0001071
aliph2HC_scArgN_vlong	0.0968871	0.0216024	4.4850148	0.0000077
carbonylC_hydroxylO_long	0.0315705	0.0124104	2.5438746	0.0110447
aliph2HC_scArgN_medlong	-0.2325776	0.0576806	-4.0321647	0.0000575
scLysN_bbN_medlong	-0.1009929	0.0236363	-4.2727836	0.0000203
carbonylC_carbonylO_vlong	0.067147	0.0146219	4.5922106	0.0000047
carboxylO_bbC_medshort	0.0550611	0.0188783	2.9166375	0.0035814
carboxylC_hydroxylO_short	-0.1290748	0.0480311	-2.6873157	0.0072681
carboxylO_carboxylO_vlong	-0.0779525	0.0200365	-3.8905248	0.0001036
scArgN_bbO_medlong	0.9314272	0.1504298	6.1917741	0
bbN_bbCA_medshort	-0.041218	0.0066121	-6.2336969	0
aliph3HC_sulfur_long	0.0387226	0.0117234	3.3030145	0.000975
aliph3HC_scArgN_long	0.2976905	0.0449362	6.6247303	0
aliph1HC_scAGN_medshort	0.0325676	0.016544	1.9685463	0.049156
carbonylC_bbCA_medlong	-0.0233301	0.007583	-3.0766441	0.0021245
aliph3HC_aromaticC_short	0.0354387	0.0082773	4.2814272	0.0000195
bbN_bbCA_long	-0.0180282	0.0034691	-5.196841	0.0000002
aromaticC_bbN_vlong	0.0090047	0.0025661	3.5091272	0.0004605
hydroxylO_bbC_medlong	0.0257001	0.0069728	3.6857764	0.0002346
bbProN_bbCA_medshort	0.0841772	0.0202425	4.1584442	0.0000335
aliph1HC_bbProN_long	0.180274	0.0355435	5.0719304	0.0000004
carboxylC_carboxylC_vlong	-0.044419	0.0346823	-1.2807399	0.2004474
aliph3HC_hydroxylO_short	-0.1251423	0.0190014	-6.5859649	0
aliph1HC_bbProN_vlong	0.1153372	0.0263471	4.3776019	0.0000127
sulfur_sulfur_vlong	0.1183655	0.0349855	3.3832755	0.0007314
scAGN_bbN_medshort	0.0280667	0.0104085	2.6965239	0.007071
scLysN_bbC_medlong	-0.13168	0.0256982	-5.1240844	0.0000003
aliph3HC_hydroxylO_medshort	-0.067175	0.0142533	-4.7129431	0.0000026
aromaticC_aromaticC_vlong	-0.0069138	0.0023666	-2.9213922	0.0035274
scAGN_scAGN_medshort	0.0609927	0.022183	2.749527	0.0060269
scArgN_bbO_medshort	0.6815002	0.1633877	4.1710621	0.0000317
aromaticC_scLysN_vlong	-0.0999354	0.0194391	-5.1409415	0.0000003
aliph1HC_hydroxylO_medlong	0.1605013	0.0259347	6.1886769	0
scArgN_bbCA_medshort	0.6258291	0.1281267	4.8844559	0.0000011

aromaticC_bbO_vshort	-0.2341505	0.042101	-5.5616445	0
aromaticC_bbC_short	0.1102342	0.0214534	5.1383113	0.0000003
carbonylC_hydroxylO_vlong	-0.0427506	0.0102939	-4.1529804	0.0000343
bbO_bbO_short	-0.0298282	0.0107511	-2.774439	0.0055859
aliph1HC_aliph3HC_medlong	-0.0573579	0.0140946	-4.0694944	0.0000491
aliph1HC_aliph1HC_long	0.0971338	0.0330833	2.9360397	0.0033658
aliph1HC_hydroxylO_medshort	0.1472844	0.0294359	5.0035678	0.0000006
carbonylC_aliph3HC_medshort	0.0504539	0.0113603	4.4412493	0.0000095
scLysN_bbN_vlong	-0.0711511	0.0154807	-4.5961059	0.0000046
aliph2HC_bbN_long	0.0125013	0.0028105	4.4480339	0.0000092
bbCA_bbCA_vlong	-0.0175189	0.0042777	-4.0954554	0.000044
carbonylC_aliph3HC_medlong	0.0472785	0.0104717	4.5148987	0.0000067
aliph2HC_bbC_medlong	-0.0133497	0.0037063	-3.6019185	0.0003243
aliph1HC_bbCA_medlong	-0.0346393	0.0091189	-3.7986353	0.0001502
aliph3HC_bbCA_medlong	0.0207428	0.005639	3.6784155	0.0002415
aliph2HC_aliph3HC_long	-0.019083	0.0044865	-4.2534466	0.0000221
aliph1HC_aliph3HC_short	0.1083398	0.0312127	3.4710204	0.0005305
scArgN_bbC_short	-0.6925344	0.1555985	-4.4507772	0.0000091
aliph1HC_hydroxylO_long	0.0685381	0.0181692	3.7722038	0.000167
scArgN_bbCA_long	0.1840745	0.0526398	3.496868	0.000482
bbProN_carbonylO_long	-0.0775705	0.0217494	-3.5665509	0.000371
scLysN_hydroxylO_long	0.1168269	0.0352238	3.3167044	0.0009287
carboxylO_bbO_vlong	0.0203439	0.0075803	2.6837696	0.0073454
hydroxylO_sulfur_short	0.0966386	0.0322641	2.9952311	0.0027792
aromaticC_scArgN_vlong	-0.0609078	0.0154428	-3.9440908	0.0000831
carboxylC_bbC_medlong	0.0505615	0.0180268	2.8047913	0.0050881
aromaticC_scArgN_medshort	0.1497952	0.0539175	2.7782318	0.0055214

Residual standard error: 0.485 on 1829 degrees of freedom
Multiple R-squared: 0.8982, Adjusted R-squared: 0.8918
F-statistic: 139.1 on 116 and 1829 DF, p-value: < 2.2e-16