Validation and Novel Predictive Modeling Techniques for Classification of Chemical Biodegradation Based on QSAR

Aaron Carr, Sreeja Kurapaty, Ivan Chavez

Shiley-Marcos School of Engineering, University of San Diego

Abstract

The delicacy of an ecosystem is important to maintain, but chemicals accumulate in them can have long lasting effects if they are not biodegradable. It is known that chemicals that are not ready biodegradable can linger and if the effects of the chemical are not entirely known, adverse effects may present themselves at some point. These adverse effects can last for an unspecified amount of time and impact surrounding wildlife, as well as human health. Due to the importance of classifying chemicals, the authors of "*Quantitative Structure−Activity Relationship Models for Ready Biodegradability of Chemicals*", created a dataset and conducted research identifying a set of predictive models that yielded the best performance (Mansouri et al., 2013a). To validate the study and improve on its performance, the team conceptualized various modeling segments: Baseline Models, Validation Models, and Novel Models. Each of the segments contains various algorithms to validate the study's performance metrics and create entirely new models to compare against the validation models. All models will be compared using the following metrics: Sensitivity; specificity; classification error rate (ER), which is the complement of the average of sensitivity and specificity; and area under the receiver operating characteristic curve (AUROC). The reason for using the stated metrics is due to the nature of the problem–correctly classifying chemicals as not ready biodegradable is equally important as correctly classifying for biodegradable. The current study found that the Validation Models were unsuccessfully replicated, and the best performing novel algorithm was Random Forests.

Keywords: QSAR, chemical, ready biodegradable, machine learning, predictive modeling, baseline, validation, novel

Table of Contents

List of Tables

List of Figures

Introduction

Background and Practical Implications

Chemical biodegradability experiments use indigenous microbiomes that are critical components in the environmental evaluation of chemicals. Biodegradability experiments are typically confined by a single focus, either on the chemical and probable transformation products or on the specific microbial species degrading the substance (Özel Duygan, 2021). Many impacts of potentially hazardous chemicals on microbial populations can be difficult to determine. A few substances in the ecosystem may be altered or entirely destroyed by a few or numerous strains, whilst others may have cytotoxic effects. A methodology was investigated to determine whether a specific chemical is biodegradable, which is an invaluable tool in the analysis of its overall effect on the environment, as it is a property that, when combined with other chemical properties such as toxicity, can theoretically have long-term effects on natural habitats (Boethling et al., 2007).

Purpose, Objectives, and Hypothesis of Present Study

Having a way to predict ready biodegradability based on other molecular features of a compound, especially during the development of new substances, could help prevent the release of chemicals into the environment that will stay there for great lengths of time. Mansouri et al. (2013a) published results on the application of several predictive modeling techniques to dichotomously classify chemicals into ready biodegradable or not. This current study uses the same final data set (N = 1,055) and has two main objectives: 1) Perform modeling that serves to reproduce the methods and results of Mansouri et al., specifically related to the *K*NN, PLSDA, and SVM models—note, the consensus models' results are not assessed in this particular paper; and 2) Implement modeling techniques novel to this specific data set, in order to assess both

whether predictive performance can be increased, as well as whether a less complex model (i.e., logistic regression [LR]) would at least match previous performance, but also allow increased interpretability. As such, the general hypotheses are: 1) Through model tuning and application of novel approaches, higher rates for both specificity (aka true negative rate) and sensitivity (aka true positive rate) can be achieved relative to Mansouri et al.; 2) While examining an additional evaluation measure, the receiver operating characteristic (ROC) curve, an area under the curve (AUC) of greater than 90% can be achieved; and 3) At least one novel model will provide more useful insight into how it works, through having a higher interpretability than the validation models.

Methods

Data Collection, Preprocessing, and Cleaning

This study is a retrospective analysis using a data set downloaded from the UCI Machine Learning Repository (Mansouri et al., 2013b). Following initial download, the data was loaded into R, where all processing and analyses were performed using the RStudio IDE. The downloaded file did not come with attribute names, but name mapping information was provided in the repository. Therefore the first task was to label the predictors and response variable. Before performing a split of the data into a training sample and a test sample, a bar graph was created to review the target attribute class makeup and distribution–see Sample Characteristics section for further details. There were no missing values in the data set, so no imputation was necessary. For the majority of models, feature values were transformed in the following ways: Corrected for skewness, using Box-Cox transformation; centered; and scaled. Additionally, for logistic regression, including the penalized versions, features that contained outliers were transformed using spatial sign, which effectively moves the outlier values closer to the majority

values to mitigate the effect of anomalous data points on models sensitive to them (Kuhn & Johnson, 2016). To avoid data leakage, the transformed values were fit solely to the training data subset, which was then applied to the test data accordingly (without refitting).

Sample Characteristics

The original data set contained 1,055 instances, and was composed of 41 predictor variables and one categorical response variable with two classes, RB (referring to a ready biodegradable compound) and NRB (referring to a not ready biodegradable compound). For the purposes of model performance evaluations, RB was considered the positive outcome and NRB was considered the negative one. The values for the predictor variables are all continuous or numerically discrete, with three of them being binary: B01.C.Br., B03.C.Cl., and B04.C.Br. The data was then split into two separate data sets using a stratified random split to ensure the distribution of class values remained the same. One extreme outlier was dropped from the training set, which resulted in a training set that contained 80.1% of the original data ($n = 844$) and a testing set that contained 19.9% ($n = 210$).

For modeling purposes, several different data subsets were created, with differing numbers of predictors. For instance, of the 41 original predictors (P), nine were eliminated due to near zero variances, which is needed for algorithms that are sensitive to them, such as k-nearest neighbor (KNN) and logistic regression ($P = 32$); or, to taking a different tack, six were filtered out using univariate analysis ($P = 35$)–see below for an explanation of this process. Relative to the subset without near-zero variances, three additional features were eliminated due to severely high correlation values (using the default threshold of .9), such that for a pair-wise feature combinations that met the criteria, the one of the two that had the highest average correlation with other features was removed–this is recommended for algorithms that are heavily influenced

by multicollinearity, such as LR ($P = 29$); or five features were filtered out using univariate analysis ($P = 27$). Univariate analysis was accomplished via the R caret package selection by filtering (sbf) function which recursively tests each individual predictor variable's performance to select the optimal variables through cross validation results. For a complete list of features and whether or not they were removed, see Table 1 in Appendix A. See the Descriptive Statistics section for details on measures of centrality and dispersion for some key features, as well as a summary of those that have significant outliers. In the modeling section, it is indicated for each model which subset is used, i.e., the number of predictors.

Descriptive Statistics Associated with Key Characteristics

Given the quantity of predictors, a visual representation of the individual frequency distributions was generated for each predictor value–see Figure 1. A quick overview of the various distributions reveals some potential issues that will require scaling methods due to the presence of right skewed values such as nHM and nCP. One particular distribution that is critical to visualize is the class variable, as shown in Figure 2. In order to identify if over/under sampling methods are needed, the distribution of the class variable should be visualized and examined for extreme inequalities. Fortunately, the distributions for both classes (RB and NRB) are well represented in the dataset, thus over/under sampling techniques are not needed.

Prior to modeling, it is important to perform exploratory data analysis (EDA) to get a sense of the data's structure and discover any initial patterns that can be used to guide the modeling process. Boxplots were generated for all of the predictors, as seen in Figure 3, however because the scales of features are inconsistent, several subplots were created to compare features within similar value ranges. Figure 4 shows the boxplots for six variables, which have values ranging from 0 to 7.723. It should be noted that comparing these boxplots should be done with

extreme caution, since even though they have similar ranges, they each have unique distributions based on the fact that their scales are completely independent and are based on the measurement values of each specific variable; having them combined on one plot is for the most part meant to be generally illustrative and for space saving reasons. Many of the features have somewhat similar variances as indicated by the size of the box and length of the whiskers. One difference is that SpMax_A has a much smaller box and shorter whiskers, and therefore most of its values are generally closer to the median. In terms of skew, LOC does seem to have some right skew because its upper whisker is longer and a greater number of outliers are present in the positive direction, which is confirmed by its skewness value (1.224); J_Dz.e. (1.202) and Psi_i_A = (1.476) are similarly skewed; SpMax_L (-0.716) and SpMax_A = (-1.006) are negatively (left) skewed; and HyWi_B.m. has relatively little skew (0.353).

Detailed descriptive statistics of the features in Figure 4, plus four more, are shown in Table 2. These include the measures of central tendency (such as mean and median) and variability (such as standard deviation, range, and the 25th and 75th percentiles). Almost all of the 41 predictors have outliers (B03.C.Cl. is the only one that does not), which for summary purposes were those with more than three standard deviations from the mean after standardization into *z*-scores. For those features in Table 2, the amount of outliers range from 0.1% of the sample (n = 1) for nCIR to 4.6% (n = 39) for SpMax_B.m. For nCIR, the one outlier value was 27.7 standard deviations from the mean, which is either due to a data entry error or the chemical has a makeup that is vastly different than all of the others; the latter is most likely the case given that the observations for that instance were also far from the mean for features F04.C.N. and F03.C.N. To avoid one molecule skewing the overall results, it was removed from the training set prior to modeling–see Figure 5. As noted in the Data Collection section, all other

outliers remain in the data set and are only transformed using spatial sign for models that are sensitive to them.

Modeling Techniques

The modeling process involved separating the modeling algorithms into three segments: Baseline, Validation, and Novel Modeling. The Baseline models, as implied by the name, serve to gauge the predictive performance of sophisticated models. Validation Modeling seeks to replicate the model performance of the algorithms mentioned in the journal "*Quantitative Structure−Activity Relationship Models for Ready Biodegradability of Chemicals*"; an important part of research is reproducibility and this step is critical to evaluate the findings mentioned in the journal–all models were trained using 5-fold cross-validation (Mansouri et al., 2013a). Finally, Novel Modeling is aimed at implementing new algorithms not mentioned in the journal, to determine if a modeling technique with a stronger predictive performance can be built–all models were trained using 10-fold cross-validation. All of the models were tuned based on applicable parameters for each algorithm, and then run on the test set. Several measures are summarized in Table 3, including the classification error rate ($ER = 1 - (TPR + TNR)/2$), sensitivity, specificity, and area under the ROC (AUROC) curve.

Baseline Modeling

Models 1.1 and 1.2 (M1.1 and M1.2): Single Decision Trees.

The single decision tree model type was selected as the baseline model due to it being a simple and relatively low computationally intensive algorithm, while also being interpretable. Also, decision trees generally have low bias, meaning that they tend to capture the trends and shape of the underlying data, but due to generally much higher variances than ensemble methods, they are more useful as part of EDA in order to review the nature of the predictor relationships,

than as long-term, high-performance predictive models. Figure 6 is the output of $M_{1.1}$, which is a classification and regression tree (CART) algorithm, using the full predictor set ($P = 41$), without any transformations–CART is robust to both noise and highly correlated features. As a comparison, a C5.0 decision tree model ($M_{1.2}$) was run using the same training set. Both CART and C5.0 focus on maximizing purity gain, though they have some differences, including the purity measure: CART uses the Gini index and C5.0 uses entropy information gain.

Validation Modeling

$M_{2.1}$ and $M_{2.2}$: *K*-Nearest Neighbor (*K*NN).

*K*NN is a simple algorithm that classifies training data based on distances between predictors. There are different distance measures that can be used for *K*NN, such as Euclidean, Manhattan, or Cosine Similarity. For the purpose of replication, the distance metric to be used is the Euclidean distance with 5-fold cross validation to identify an optimal *k* value for model testing. $M_{2.1}$ used the predictor set with $P = 32$ and $M_{2.2}$ used the set with $P = 35$, and both included transformation of features based on centering and scaling.

$M_{3.1}$: Support Vector Machine (SVM).

Support vector machines (SVMs) are a group of supervised learning techniques for classifying data, performing regression analysis, and identifying outliers. They process high-dimensional spaces effectively and if the dimensions are more than the samples, the method is still effective. Better memory efficiency is provided by using a portion of the training data for the decision function commonly known as support vectors which can be provided with a variety of kernel functions. $M_{3.1}$ ($P = 41$) included transformation of features based on centering and scaling.

$M_{4.1}$: Partial Least Squares Discriminant Analysis (PLSDA).

Discriminant analysis essentially finds the predictor combinations that best separates them into groups. PLSDA implements partial least squares to better identify predictor relationships and utilizes latent variables to reduce the dimension and optimize correlations for the classification response variable. Tuning requirements for PLSDA involve 5-fold cross-validation to identify the optimal latent variable that yields the strongest set of predictors. $M_{4.1}$ ($P = 41$) included transformation of features based on centering and scaling.

Novel Modeling

$M_{5.1}$, $M_{5.2}$, and $M_{5.3}$: Logistic Regression (LR).

Given a binary response variable, LR is a powerful probabilistic method for determining the likelihood of an observation to belong to one of the classes. Using log odds and a nonlinear (logit) function, LR creates a linear boundary between the two classes. By default, a probability >=0.5 assigns/predicts an instance to the positive class, whereas anything less is predicted as the negative class. Three different variations of LR were applied: $M_{5.1}$ used the predictor set without near zero variance values and with three additional highly correlated features removed ($P = 29$), and included transformation of features based on skewness, centering, scaling, and spatial sign; $M_{5.2}$ using the same feature set as $M_{5.1}$, but this time included principal component analysis (PCA) as part of the preprocessing step; and $M_{5.3}$ used the predictor set derived through univariate analysis ($P = 27$), and included transformation of features based on skewness, centering, scaling, and spatial sign.

$M_{6.1}$ and $M_{6.2}$: Penalized LR (PLR).

Penalization in this case is a way of regularizing the data to reduce complexity, with the goal of significantly decreasing variance while only slightly increasing bias (Kuhn & Johnson).

For PLR, a penalty is added to the error measure, which becomes more pronounced the higher parameter estimates become. The R caret package train() function allows for two parameters: Alpha (α) determines the mix between a purely ridge regression model ($\alpha = 0$), which is based on the sum of the squared parameters, and a purely lasso model ($\alpha = 1$), which is based on the sum of the absolute values of the parameters; and lambda (λ) is the penalization amount hyperparameter. Two variations of PLR were applied. In order to find the best tuning hyperparameters, both were tuned over a grid of a range of seven α values (between 0 and 1) and 20 λ values (between .01 and .25). Also, both models included transformation of features based on skewness, centering, scaling, and spatial sign. $M_{6.1}$ used the predictor set with $P = 29$ and $M_{6.2}$ used the set with $P = 27$.

$M_{7.1}$: Linear Discriminant Analysis (LDA).

Data visualization, dimension reduction, and classification are all done using LDA. It is both a reliable classification method and a dimension reduction tool. LDA features can be utilized with or without the assumption of data normality, which accounts for its robustness. LDA frequently yields reliable, respectable, and understandable classification results. LDA is frequently used as a benchmarking technique before using other, more intricate and flexible techniques to solve classification problems in the real world. $M_{7.1}$ ($P = 41$) included transformation of features based on centering and scaling.

$M_{8.1}$: Nearest Shrunken Centroid (NSC).

The NSC algorithm was selected as it offers a new approach to discriminating for class predictor variables. The centroid is found for each individual predictor and an overall centroid is calculated from those values. Additionally, those inputs that contribute the least to model discrimination are dropped. Since the model does its own feature selection and works well with

datasets that contain large amounts of predictors, it was selected as a novel model. $M_{8.1}$ ($P = 41$) included transformation of features based on centering and scaling.

***$M_{9.1}$*: Random Forests.**

Random trees construct a forest like decision trees and combine them together to ensure a reliable prediction. Multiple trees are built from training data in a manner that correlation between the trees is simply reduced. Random forests help to maintain or increase the speed of the decision trees and to achieve good accuracy. $M_{9.1}$ ($P = 41$) included transformation of features based on centering and scaling.

Results

For a full summary of ER, specificity, sensitivity, and AUROC values for each of the models, see Table 3 in Appendix A.

Baseline Models

The no information rate (NIR), which is the equivalent to predicting all instances as the majority class (negative in this instance) is .662. Therefore, for a model to be considered as the candidate for overall best performer, it must at a minimum perform with higher accuracy than 66.2%.

Based on review of the resampling results of the 10-fold cross-validated decision tree models, neither CART nor C5.0 performed very well given the full predictor set ($P = 41$), with no transformations. The sensitivity for both was higher than the published result fors *K*NN and SVM (but not PLSDA)–both on fit and 5-fold cross-validation–but the specificity was lower than all of the three published models; as a result, the ER for the decision trees was much higher; the lowest rate on the three published models was .14, whereas C5.0 was .199. As surmised, the performance relative to the training and test sets declined for the sensitivity, but surprisingly the

specificity increased and the ER decreased for both models. This means that both models got worse at predicting an actual positive instance as positive, but got significantly better at predicting an actual negative instance as negative. This is discussed further in the Discussion section. Although both models' ER was lower in the test set, neither were less than the *K*NN, PLSDA, and SVM published results. Figure 6 is the unpruned decision tree plot that was generated from the CART model; it is rather complex, with 24 inner nodes and 25 leaf nodes. For the latter, purity is inconsistent, with several being purely one class, whereas due to CART's greedy nature, for some of them there was not enough to split further, so they were very impure (e.g., LOC >= 0.56 resulted in a root node that was 43% NRB and 57% RB).

Validation Models

The train and test datasets were analyzed and validated using *K*NN ($M_{2.1}$), *K*NN with univariate predictors ($M_{2.2}$), PLSDA ($M_{3.1}$) and SVM ($M_{4.1}$). The models were validated using 5-fold cross validation. From Table 3, for the training dataset, it is observed that $M_{4.1}$ has the highest sensitivity (0.921) and the lowest ER (0.161), whereas $M_{2.2}$ has the high specificity (0.825). Compared to all the models $M_{4.1}$ has a higher AUROC than the other three for the validation done on the training dataset.

$M_{2.1}$ and $M_{2.2}$, based on *K*NN, both have an apparent average accuracy (86.7% and 83.9%, respectively) that exceeds the NIR. Sensitivity of $M_{2.1}$ and $M_{2.2}$ are almost similar for the training set. Considering the test set, from Table 3, $M_{4.1}$ has a high sensitivity (0.932), but the ER obtained for the *K*NN with univariate predictors ($M_{4.2}$) is lower than the other validation models.

Novel Models

All of the LR models that did not include any penalization parameters ($M_{5.1}$, $M_{5.2}$, and $M_{5.3}$) seemed to have very similar results. Of the three, $M_{5.1}$ had both the highest sensitivity (.884)

and specificity (.757), and therefore the lowest ER (.180). $M_{5.1}$ also had the highest AUROC of the three regular LR models (.905), but there were other novel models with higher values. Also, in relation to the published models, even though the sensitivity was the same as PLSDA, the specificity was lower than all of the fit and 5-fold cross-validated models, meaning that $M_{5.1}$ was just not very good at predicting actual negative instances as negative. Of the two PLR models ($M_{6.1}$ and $M_{6.2}$), $M_{6.1}$ ($\alpha = 0$ [for a fully ridge regression-type model]; $\lambda = 0.479$) had higher sensitivity (.902), the same specificity as $M_{6.2}$ (.726), a slightly lower ER (.186), and a slightly higher AUROC (.912). When compared to LR model $M_{5.1}$, $M_{6.1}$ had slightly higher sensitivity and AUROC, but lower specificity.

The other three novel models were based on LDA ($M_{7.1}$), NSC ($M_{8.1}$), and Random Forests ($M_{9.1}$) algorithms. While none of them had a better sensitivity than $M_{6.1}$, Random Forests did have the highest specificity (.909) and AUROC (0.933) of any novel model, as well as the lowest ER (.109). In fact, the ER was lower than any of the three published models we examined, on both fit and 5-fold cross-validation, making it a potentially a high performing predictive model when applied to new data. Interestingly enough, the Random Forests model did not have a higher sensitivity than the highest published model examined (PLSDA was .88) or a higher specificity than the published SVM model (0.92), but the combination of the two high measures together is what made the difference in the ER.

When the resampled novel models were run on the test data, the ER for all of them went down, which is an unusual result. However, the Random Forests model remained the best performer, with a sensitivity of .873 (which tied LR as the highest) and a specificity that went up significantly relative to the training sample (.964). As a result, the ER rate dropped to .082.

Discussion

While the current study was not able to fully replicate the results of the three published models, specifically in terms of achieving exact ERs, the Random Forests model was able to achieve results that are indicative of a strongly performing model that can sufficiently predict whether or not a chemical is ready biodegradable. Random Forests' power comes from the fact that it is an ensemble algorithm that runs parallel decision trees and averages the error measure across all of the trees. In terms of hypothesis 1, the objective was technically not achieved. The final model did not have higher specificity or sensitivity than the published models with the highest values, however, because both measures were high, an overall lower ER was achieved–this occurred in both the training and test sets. In terms of hypothesis 2, several models were able to exceed the AUROC threshold of 90%, both in the Validation Modeling segment and the Novel Modeling segment. Again, the Random Forests model AUROC well exceeds any others, especially from the validation models. Regarding hypothesis 3, LR was performed and a linear equation that is interpretable at the individual coefficient level was generated, however, the number of predictors included were too many for any interpretation of the model as a whole to produce profound insights. Also, given that the best (non-penalized) LR model ($M_{5.1}$) was not as high performing as Random Forests, which is not an interpretable model, the results are ambiguous. Evidence for the conclusion that the desired result was achieved is that $M_{5.1}$ did almost as well as Random Forests on the test set.

For future research, firstly an analysis should be done to investigate why test results were across-the-board better than the training sample results. An initial theory is that even though the initial train/test split was based on stratified random sampling, the underlying structure of the test set was somehow very different. Additionally, there may have been some additional variance

introduced during model training based on the *k*-fold cross-validation being based on simple random sampling. To resolve the discrepancy, the following steps could be done: 1) Resplit the data and rerun the analyses to see if the issue was based on split randomness; and 2) More critically, search out completely novel data on which to test the models' performances. Secondly, even though the results are promising for Random Forests, such that the model performed well on both the training and test data sets, further research will need to be done to confirm that the algorithm has not overfit the data, making a model that too closely resembles the underlying data structure. Lastly, further tuning could be done on the data to more closely replicate the specificity, sensitivity, and ER of the published models.

References

Boethling, R. S., Sommer, E., & DiFiore, D. (2007). Designing small molecules for biodegradability. *Chemical Reviews*, *107*(6), 2207–2227. https://doi.org/10.1021/cr050952t

Kuhn, M., & Johnson, K. (2016). *Applied predictive modeling*. Springer.

Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., & Consonni, V. (2013a). Quantitative structure-activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information and Modeling*, *53*, 867–878. https://doi.org/10.1021/ci4000213

Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., & Consonni, V. (2013b). *QSAR biodegradation* [Data set]. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation#

Özel Duygan, B. D., Rey, S., Leocata, S., Baroux, L., Seyfried, M., & van der Meer, J. R. (2021). Assessing Biodegradability of Chemical Compounds from Microbial Community Growth Using Flow Cytometry. *MSystems*, *6*(1). https://doi.org/10.1128/msystems.01143-20

Appendix A

Table 1

Feature Map from UCI Machine Learning Repository

Feature #	Feature	Feature Description	Removed from Model	Removal Reason (if applicable)
1	SpMax_L	Leading eigenvalue from Laplace matrix		
2	J_Dz.e.	Balaban-like index from Barysz matrix weighted by Sanderson electronegativity	M2.2, M5.3, M6.2	Random Forest Filter
3	nHM	Number of heavy atoms		
4	F01.N.N.	Frequency of N-N at topological distance 1	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
5	F04.C.N.	Frequency of C-N at topological distance 4		
6	NssssC	Number of atoms of type ssssC		
7	nCb.	Number of substituted benzene C(sp2)		
8	C.	Percentage of C atoms		
9	nCp	Number of terminal primary C(sp3)		
10	nO	Number of oxygen atoms		
11	F03.C.N.	Frequency of C-N at topological distance 3		
12	SdssC	Sum of dssC E-states		

Feature #	Feature	Feature Description	Removed from Model	Removal Reason (if applicable)
13	HyWi_B.m.	Hyper-Wiener-like index (log function) from Burden matrix weighted by mass		
14	LOC	Lopping centric index		
15	SM6_L	Spectral moment of order 6 from Laplace matrix	M5.1, M5.2	High Corr
16	F03.C.O.	Frequency of C - O at topological distance 3	M2.2, M5.3, M6.2	Random Forest Filter
17	Me	Mean atomic Sanderson electronegativity (scaled on Carbon atom)		
18	Mi	Mean first ionization potential (scaled on Carbon atom)		
19	nN.N	Number of N hydrazines	M2.1, M2.2, M5.1, M5.2, M5.3, M6.1, M6.2	NZV, Random Forest Filter
20	nArNO2	Number of nitro groups (aromatic)	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
21	nCRX3	Number of CRX3	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
22	SpPosA_B.p.	Normalized spectral positive sum from Burden matrix weighted by polarizability		
23	nCIR	Number of circuits		

Feature #	Feature	Feature Description	Removed from Model	Removal Reason (if applicable)
24	B01.C.Br.	Presence/absence of C - Br at topological distance 1	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
25	B03.C.Cl.	Presence/absence of C - Cl at topological distance 3		
26	N.073	Ar2NH / Ar3N / Ar2N-Al / R..N..R	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
27	SpMax_A	Leading eigenvalue from adjacency matrix (Lovasz-Pelikan index)	M5.1, M5.2	High Corr
28	Psi_i_1d	Intrinsic state pseudo connectivity index - type 1d	M2.2, M5.3, M6.2	Random Forest Filter
29	B04.C.Br.	Presence/absence of C - Br at topological distance 4	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
30	SdO	Sum of dO E-states	M2.2, M5.3, M6.2	Random Forest Filter
31	TI2_L	Second Mohar index from Laplace matrix		
32	nCrt	Number of ring tertiary C(sp3)	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
33	C.026	R--CX--R		
34	F02.C.N.	Frequency of C - N at topological distance 2		

Feature #	**Feature**	**Feature Description**	**Removed from Model**	**Removal Reason (if applicable)**
35	nHDon	Number of donor atoms for H-bonds (N and O)	M2.2, M5.3, M6.2	Random Forest Filter
36	SpMax_B.m.	Leading eigenvalue from Burden matrix weighted by mass		
37	Psi_i_A	Intrinsic state pseudo connectivity index - type S average		
38	nN	Number of Nitrogen atoms		
39	SM6_B.m.	Spectral moment of order 6 from Burden matrix weighted by mass	M5.1, M5.2	High Corr
40	nArCOOR	Number of esters (aromatic)	M2.1, M5.1, M5.2, M5.3, M6.1, M6.2	NZV
41	nX	Number of halogen atoms		
42	experimental class	ready biodegradable (RB); not ready biodegradable (NRB)		

Table 2

Descriptive Statistics for Selected Quantitative Features

	SpMax_L	J_Dz.e.	F04.C.N.	F03.C.N.	HyWi_B. m.	LOC	nCIR	SpMax_A	SpMax_B. m.	Psi_i_A
Total N:										
Count	845	845	845	845	845	845	845	845	845	845
NA Count	0	0	0	0	0	0	0	0	0	0
Mean	4.793	3.073	1.030	1.484	3.476	1.364	1.408	2.218	3.899	2.561
Median	4.838	3.058	0.000	0.000	3.444	1.187	1.000	2.248	3.720	2.452
Standard Deviation	0.539	0.799	2.458	3.261	0.586	0.788	5.257	0.222	0.943	0.640
Variance	0.290	0.639	6.043	10.632	0.343	0.620	27.640	0.049	0.890	0.409
Range	4.496	6.919	36.000	44.000	4.157	4.491	147.000	1.859	8.425	4.283
Min	2.000	0.804	0.000	0.000	1.544	0.000	0.000	1.000	2.267	1.467
Max	6.496	7.723	36.000	44.000	5.701	4.491	147.000	2.859	10.692	5.750
25th Percentile	4.499	2.524	0.000	0.000	3.106	0.875	0.000	2.105	3.495	2.107
75th Percentile	5.126	3.448	1.000	2.000	3.809	1.705	1.000	2.358	3.972	2.881
Subset w/o Outliers:										
Count	835	836	827	828	837	830	844	836	806	834
%	98.80%	98.90%	97.90%	98%	99.10%	98.20%	99.90%	98.90%	95.40%	98.70%
Outlier %	1.20%	1.10%	2.10%	2%	0.90%	1.80%	0.10%	1.10%	4.60%	1.30%
NA Count	0	0	0	0	0	0	0	0	0	0
Mean	4.812	3.034	0.771	1.167	3.466	1.316	1.236	2.228	3.722	2.525
Median	4.843	3.049	0.000	0.000	3.442	1.185	1.000	2.253	3.705	2.436
Standard Deviation	0.494	0.705	1.504	2.126	0.555	0.706	1.581	0.202	0.415	0.559
Variance	0.244	0.498	2.261	4.521	0.308	0.499	2.498	0.041	0.172	0.312
Range	2.902	4.645	8.000	11.000	3.336	3.711	15.000	1.241	3.541	2.950
Min	3.414	0.804	0.000	0.000	1.883	0.000	0.000	1.618	2.267	1.467
Max	6.316	5.449	8.000	11.000	5.219	3.711	15.000	2.859	5.808	4.417

Table 3

Summary of Evaluation Metrics for Each Model

BASELINE MODELS								
		10-fold Cross-validation				**Test Set**		
Model	**Algo.**	**ER**	**Sen**	**Spe**	**ROC**	**ER**	**Sen**	**Spe**
$M_{1.1}$	CART	0.239	0.859	0.664	0.833	0.160	0.789	0.892
$M_{1.2}$	C5.0	0.199	0.875	0.727	0.851	0.152	0.775	0.921
VALIDATION MODELS								
		5-fold Cross-validation				**Test Set**		
Model	**Algo.**	**ER**	**Sen**	**Spe**	**ROC**	**ER**	**Sen**	**Spe**
$M_{2.1}$	*K*NN	0.177	0.864	0.782	0.890	0.149	0.775	0.928
$M_{2.2}$	*K*NN w/ UP	0.165	0.846	0.825	0.900	0.110	0.859	0.921
$M_{3.1}$	PLSDA	0.178	0.898	0.747	0.907	0.121	0.817	0.942
$M_{4.1}$	SVM	0.161	0.921	0.758	0.924	0.134	0.932	0.801
NOVEL MODELS								
		10-fold Cross-validation				**Test Set**		
Model	**Algo.**	**ER**	**Sen**	**Spe**	**ROC**	**ER**	**Sen**	**Spe**
$M_{5.1}$	LR	0.180	0.884	0.757	0.905	0.096	0.873	0.935
$M_{5.2}$	LR w/ PCA	0.191	0.871	0.747	0.896	0.114	0.859	0.914
$M_{5.3}$	LR w/ UP	0.197	0.877	0.730	0.903	0.121	0.845	0.914
$M_{6.1}$	PLR	0.186	0.902	0.726	0.912	0.107	0.859	0.928
$M_{6.2}$	PLR w/ UP	0.189	0.896	0.726	0.909	0.124	0.831	0.921
$M_{7.1}$	LDA	0.171	0.893	0.765	0.923	0.089	0.859	0.964
$M_{8.1}$	NSC	0.250	0.844	0.656	0.862	0.220	0.676	0.885
$M_{9.1}$	Random Forests	0.109	0.873	0.909	0.933	0.082	0.873	0.964

Note: PCA = Principal Component Analysis; UP = Univariate Predictors

Table 4

Hyperparameters for Each Model

BASELINE MODELS						
Model	**Algo.**	**CP**				
$M_{1.1}$	CART	0				
$M_{1.2}$	C5.0	—				
VALIDATION MODELS						
Model	**Algo.**	***k***	**LV**	**C**	**Sigma**	
$M_{2.1}$	*K*NN	6	—	—	—	
$M_{2.2}$	*K*NN w/ Univ. Preds.	18	—	—	—	
$M_{3.1}$	PLSDA	—	16	—	—	
$M_{4.1}$	SVM	—	—	1	0.034	
NOVEL MODELS						
Model	**Algo.**	**Alpha**	**Lambda**	**Shrinkage**	**Mtry**	**LV**
$M_{5.1}$	LR	—	—	—	—	—
$M_{5.2}$	LR w/ PCA	—	—	—	—	16
$M_{5.3}$	LR w/ Univ. Preds.	—	—	—	—	—
$M_{6.1}$	PLR	0	0.479	—	—	—
$M_{6.2}$	PLR w/ Univ. Preds.	0	0.023	—	—	—
$M_{7.1}$	LDA	—	—	—	—	—
$M_{8.1}$	NSC	—	—	0	—	—
$M_{9.1}$	Random Forests	—	—	—	6.403	—

Figure 1

Predictor Frequency Distribution

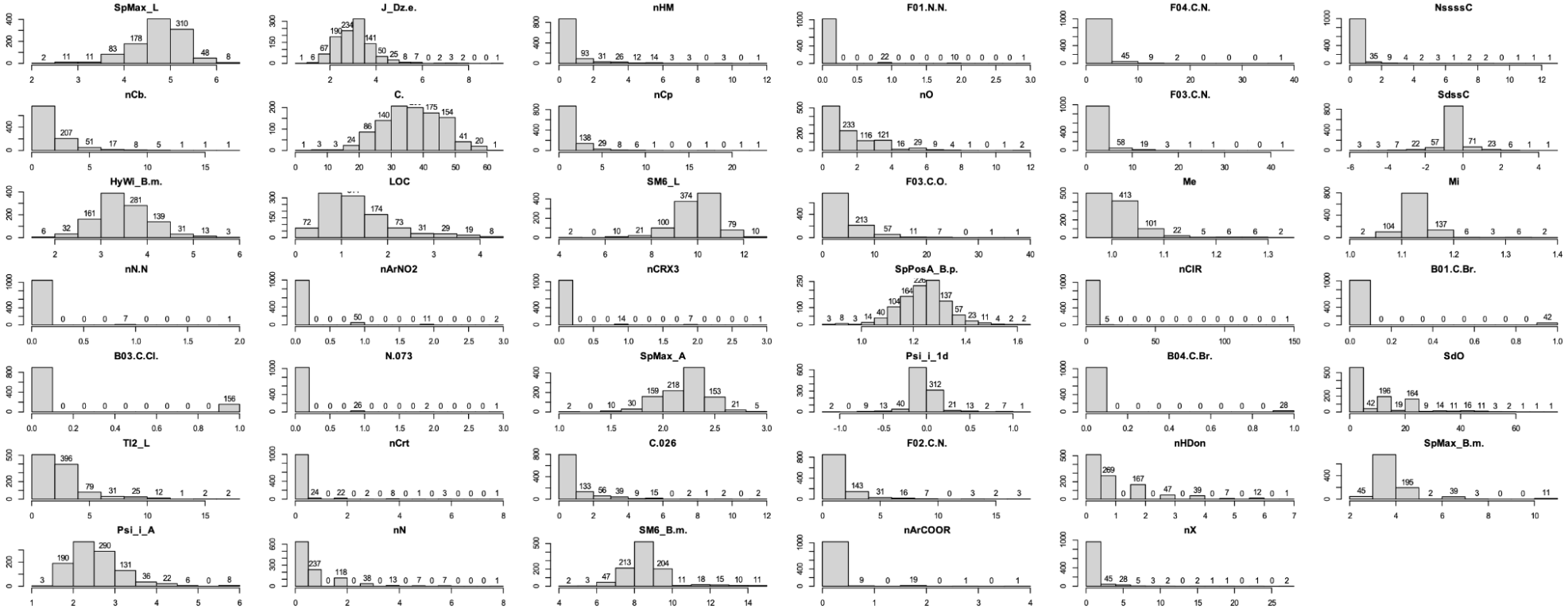


Figure 2

Response Frequency Distribution

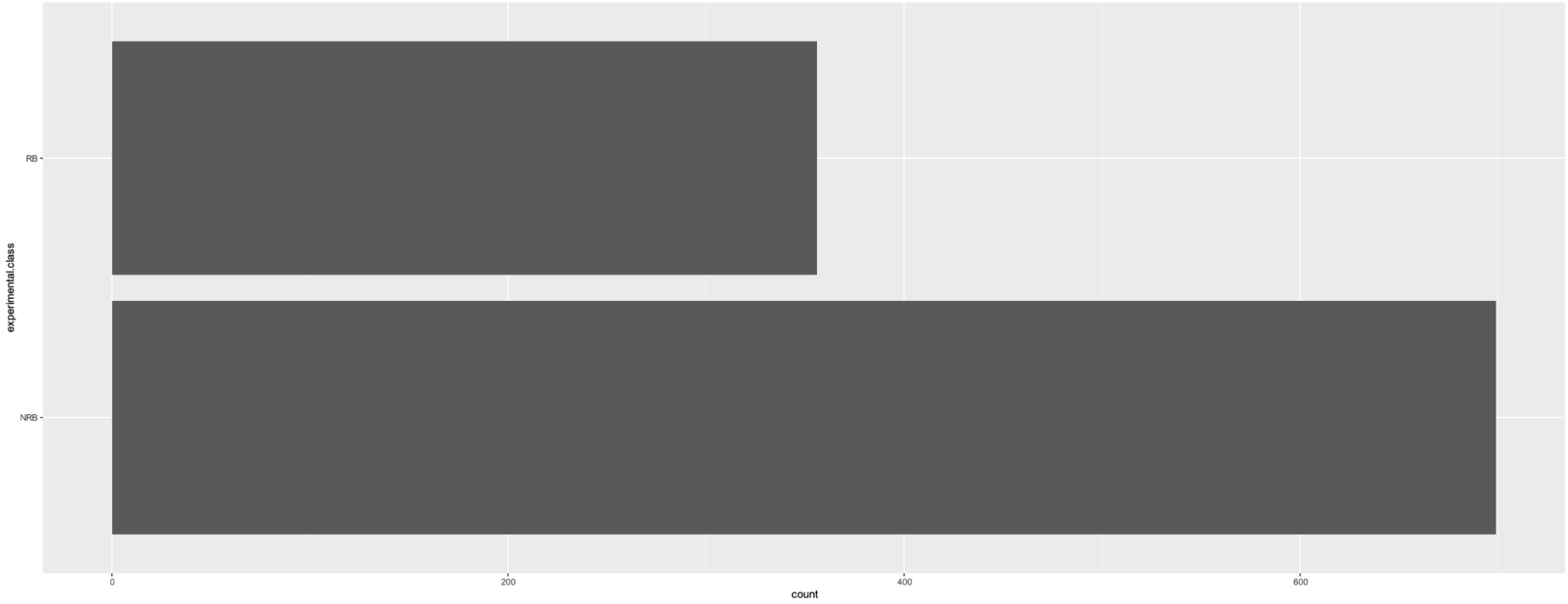


Figure 3
Boxplot of All Predictors

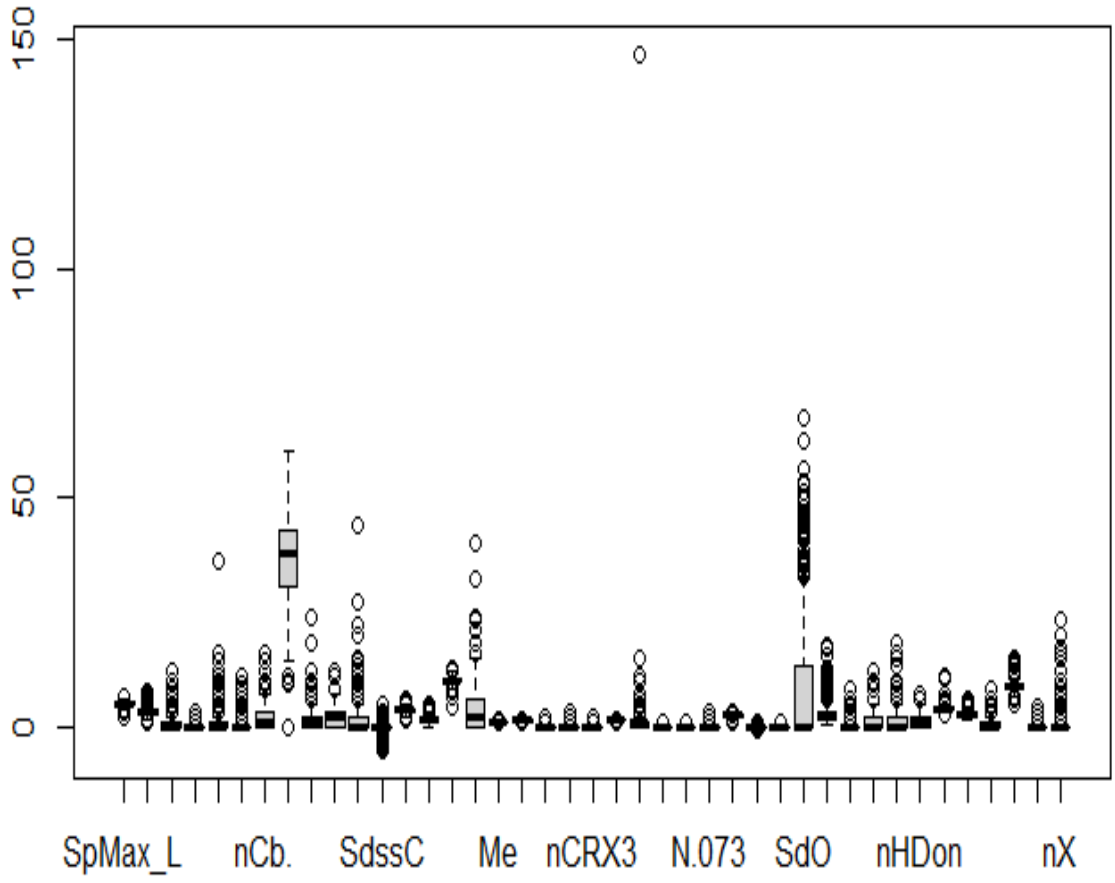


Figure 4

Boxplots of Features with Similar Ranges

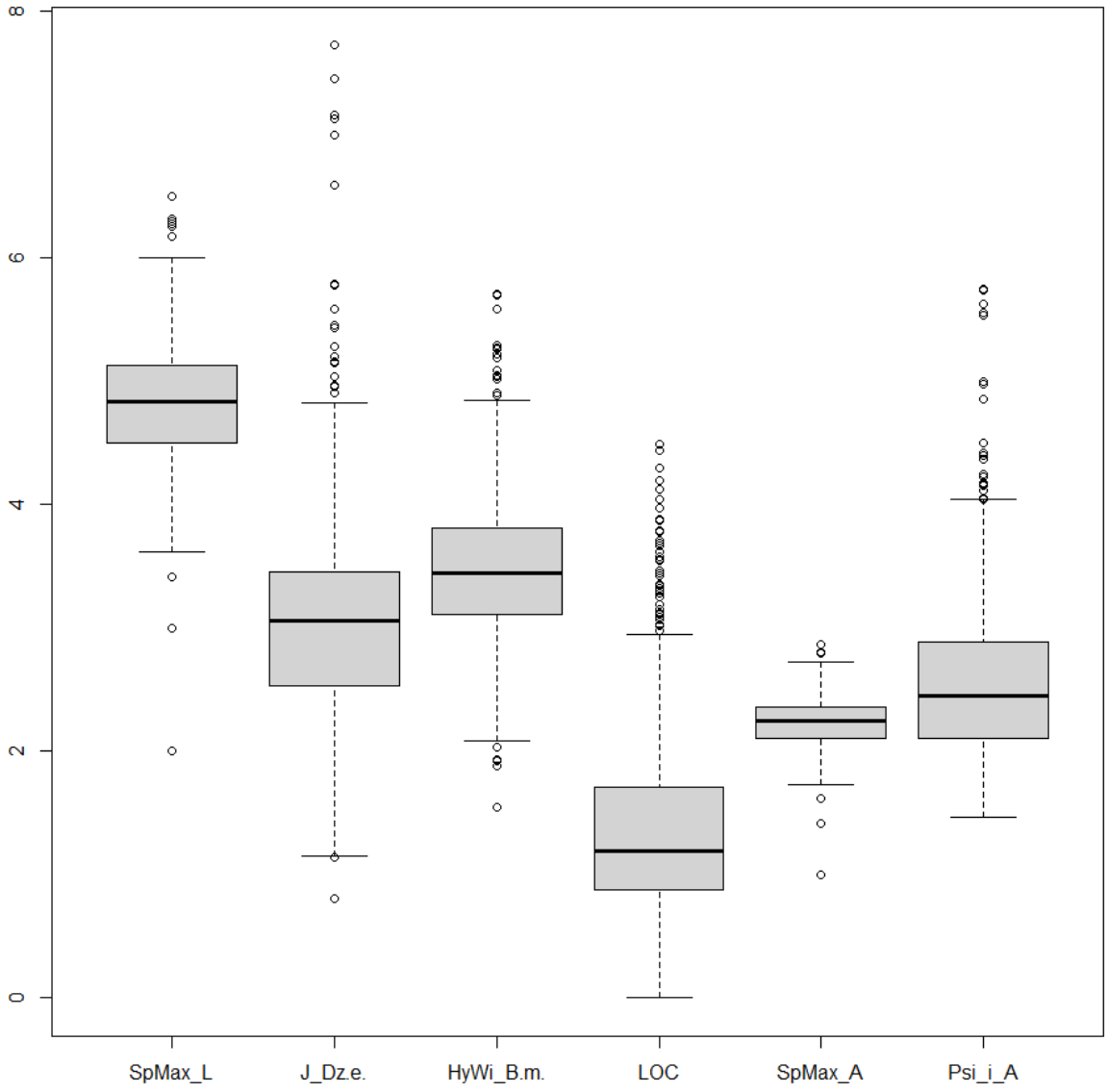


Figure 5

Boxplots of nCIR Feature Before (Left) and After (Right) Removal of Extreme Outlier Instance

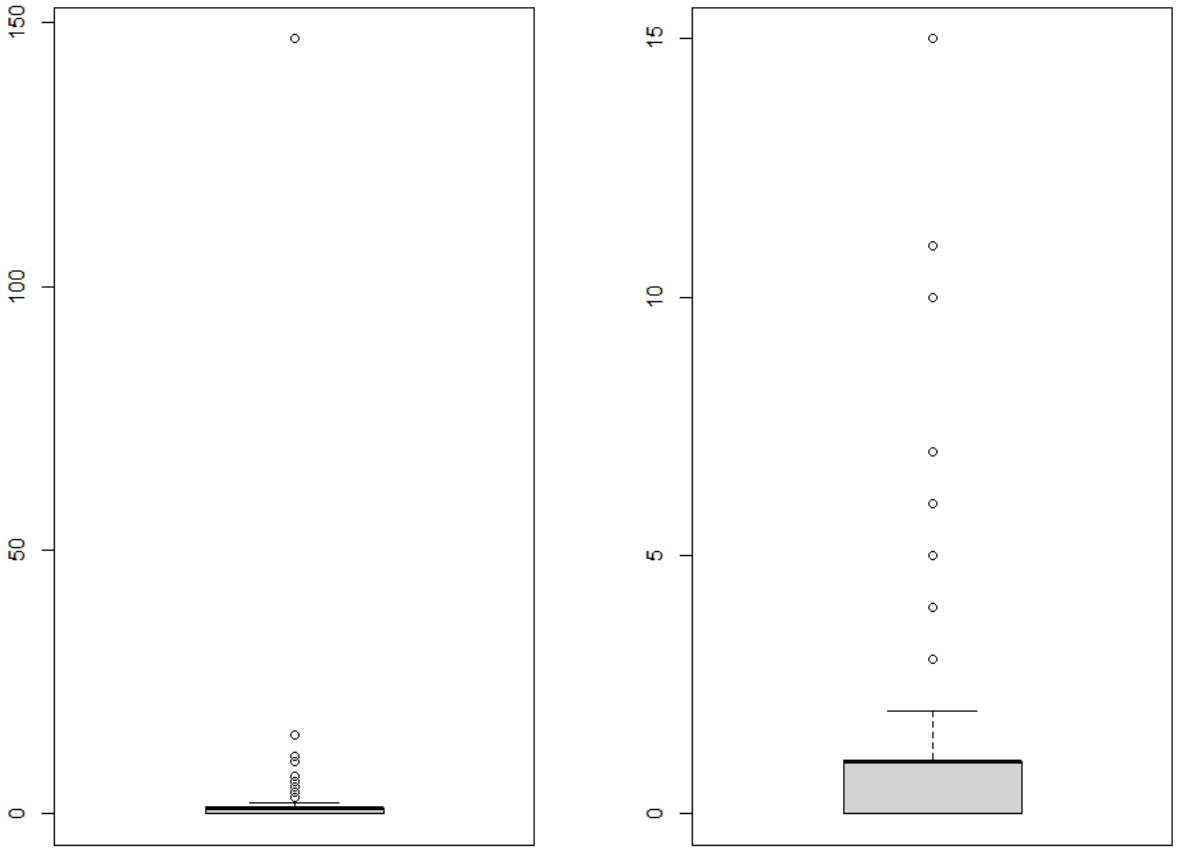
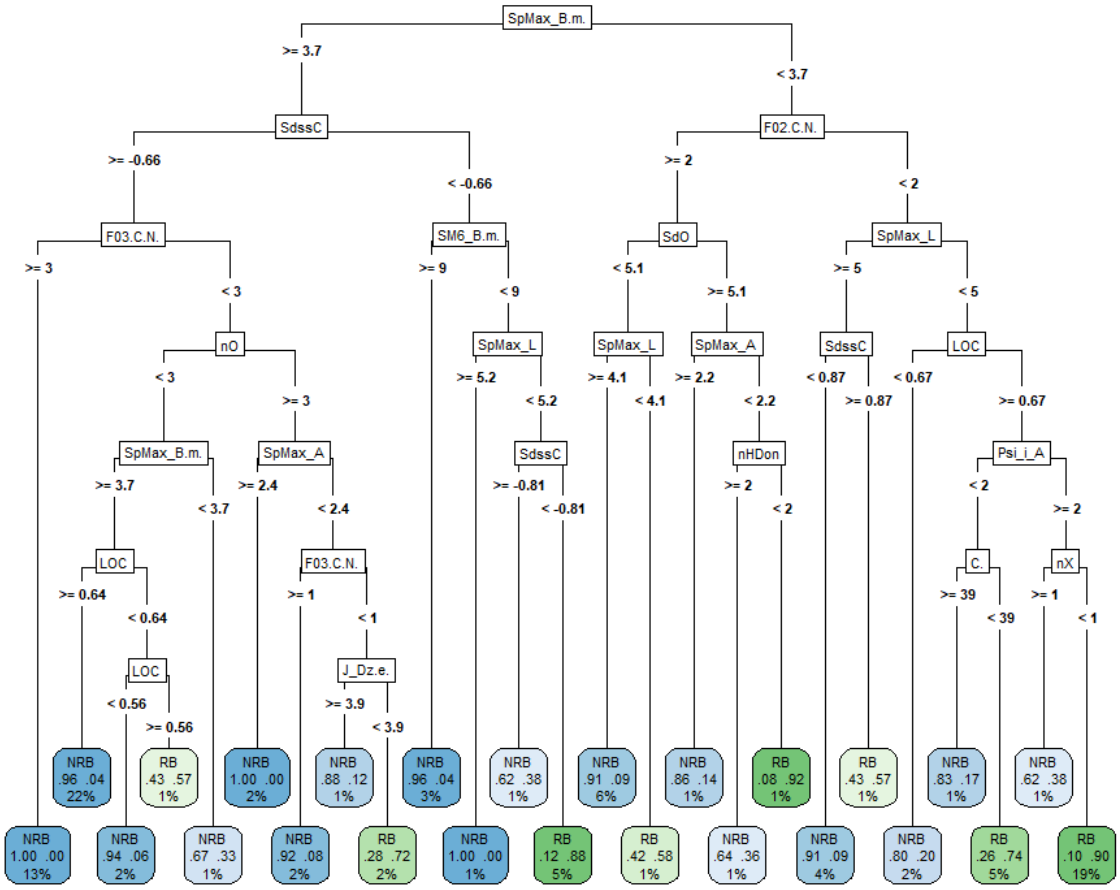


Figure 6

Classification and Regression Tree (CART) Model Output



Appendix B

Data Splitting

Ivan A Chavez

```
# Loading packages and setting seed
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✔ tibble  3.1.7     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
## ✔ purrr   0.3.4
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

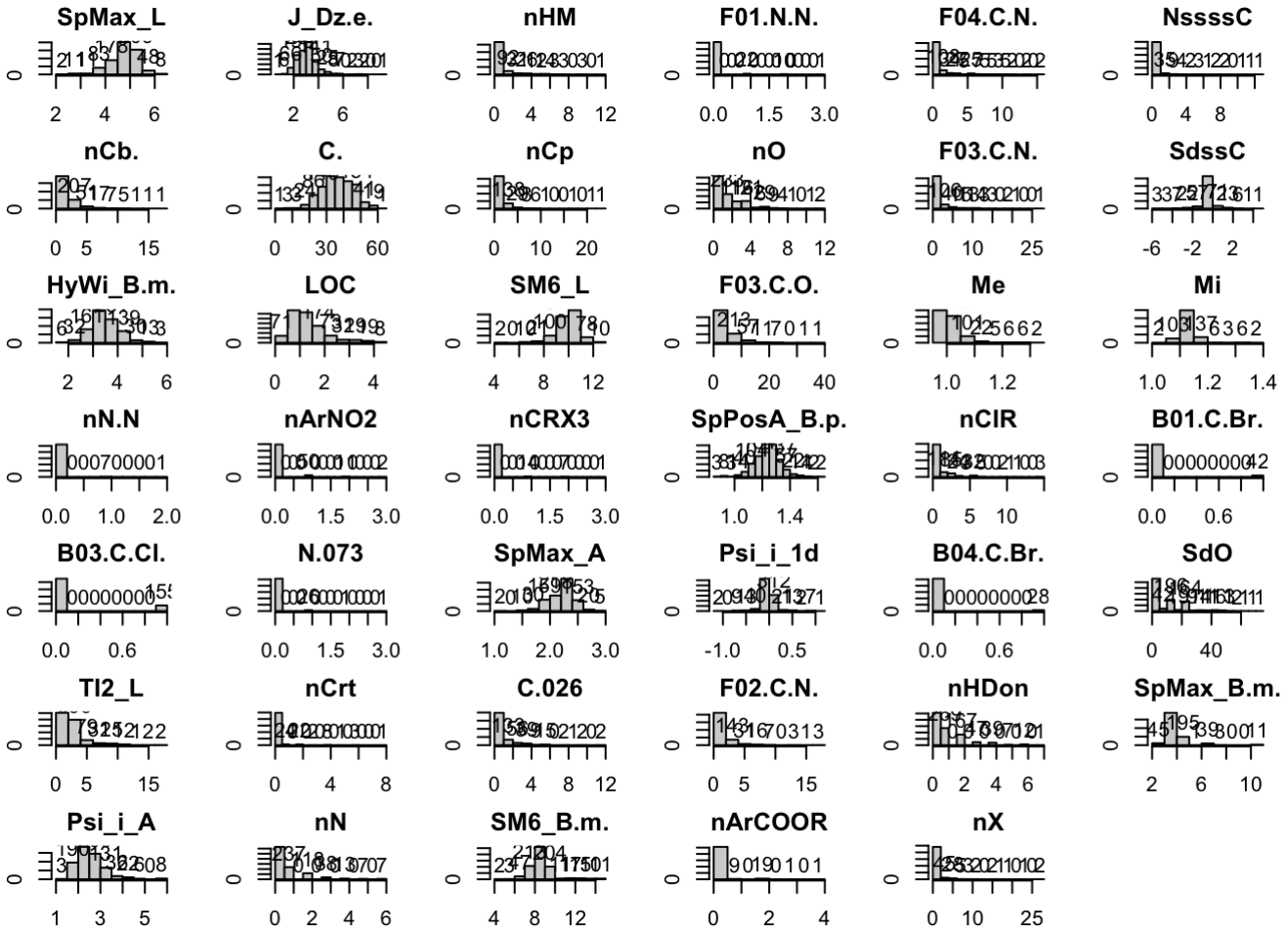
```
set.seed(100)
```

Importing and assigning column names

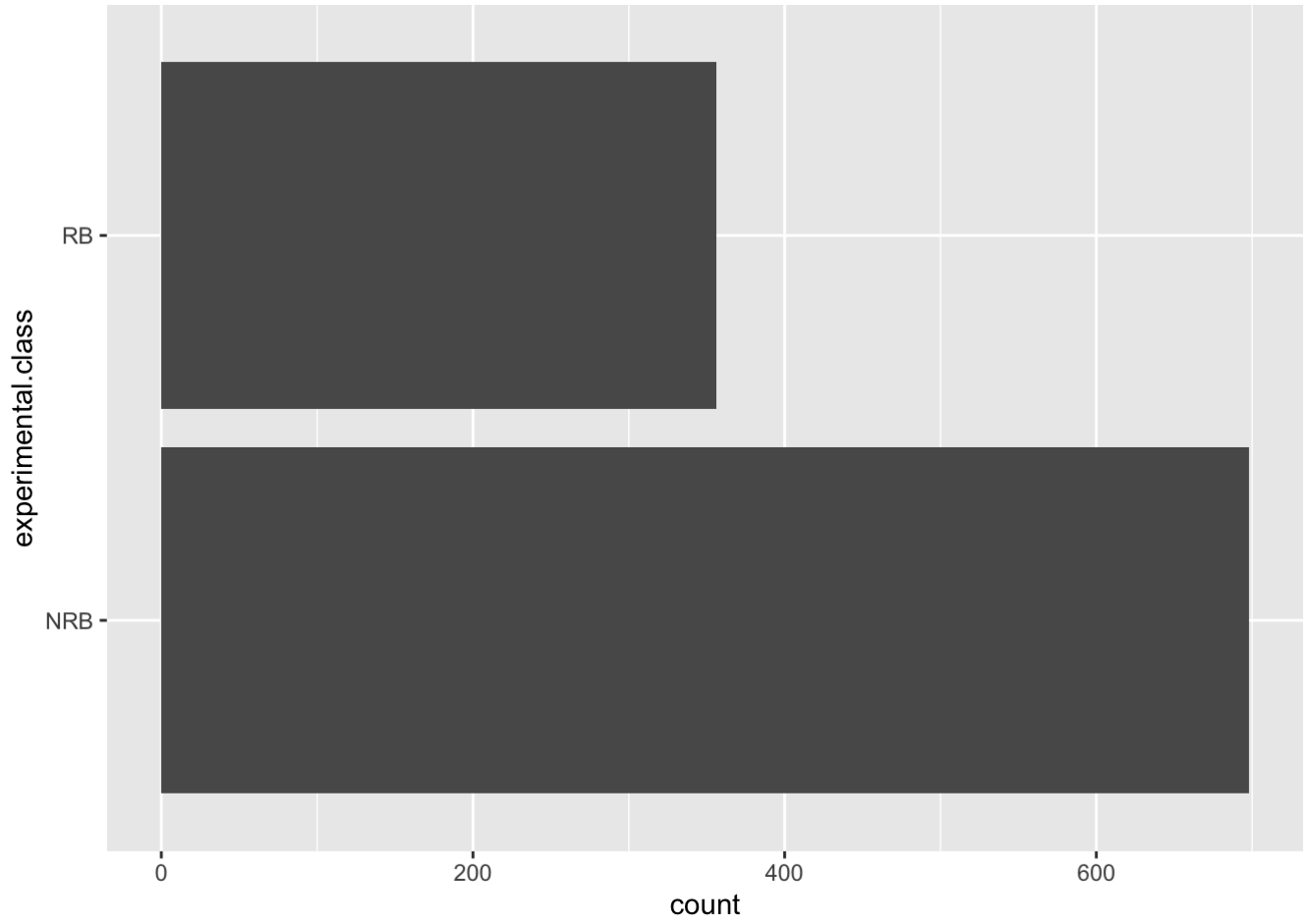
```
names <- c("SpMax_L", "J_Dz(e)", "nHM", "F01[N-N]", "F04[C-N]", "NssssC", "nCb-", "C%",
"nCp", "nO", "F03[C-N]", "SdssC", "HyWi_B(m)", "LOC", "SM6_L", "F03[C-O]", "Me", "Mi",
"nN-N", "nArNO2", "nCRX3", "SpPosA_B(p)", "nCIR", "B01[C-Br]", "B03[C-Cl]", "N-073", "Sp
Max_A", "Psi_i_1d", "B04[C-Br]", "SdO", "TI2_L", "nCrt", "C-026", "F02[C-N]", "nHDon",
"SpMax_B(m)", "Psi_i_A", "nN", "SM6_B(m)", "nArCOOR", "nX", "experimental class")

biodeg <- read.csv("../data/biodeg.csv", header = FALSE, sep = ";", col.names = names)

#dropping extereme nCIR outlier row 1054
biodeg <- biodeg[-1054,]
```

Plotting original frequency distributions of predictor variables

```
biodeg_hist <- biodeg[,-42]
par(mfrow = c(7,6), mar=c(2,2,2,2))

for(i in names(biodeg_hist)){
  hist(biodeg_hist[[i]] ,main=i,xlab="x",label=TRUE,plot = TRUE)
}
```




```
ggplot(biodeg, aes(experimental.class)) + geom_bar() + coord_flip()
```

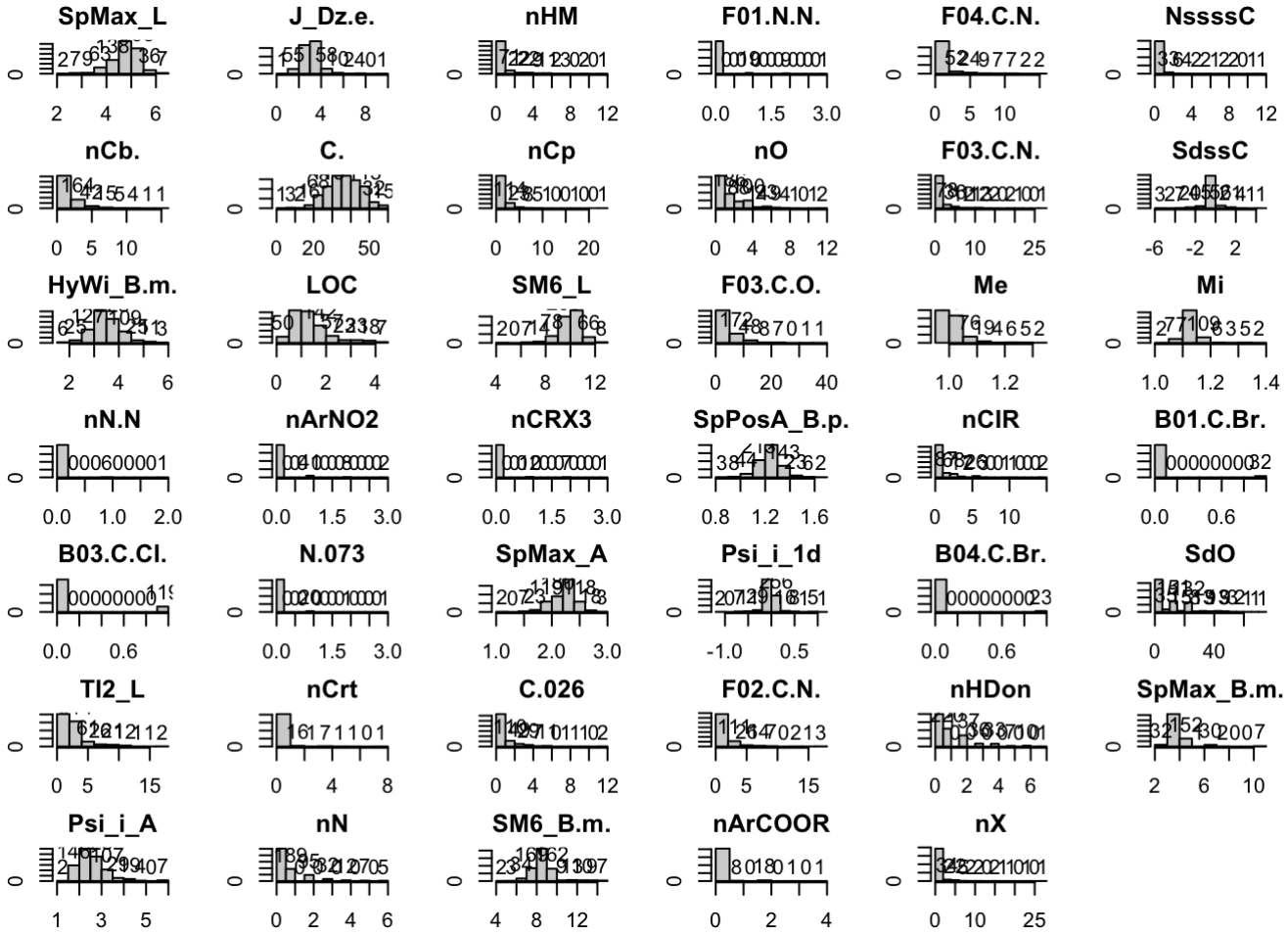


Implementing stratified sampling to maintain frequency distribution

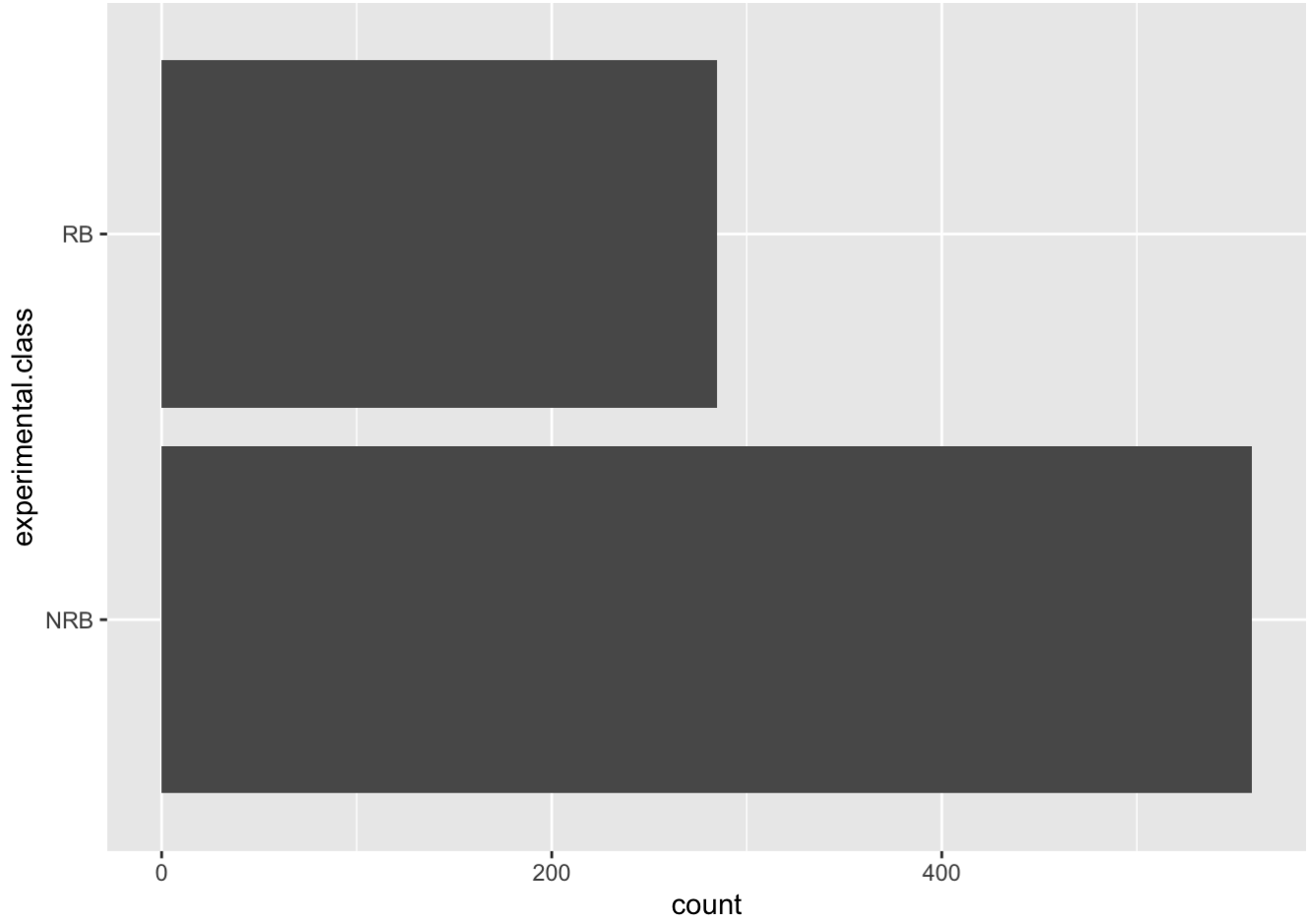
```
index_biodeg <- createDataPartition(biodeg$experimental.class, p=.80, list = FALSE)
biodeg_trainset <- biodeg[index_biodeg,]
biodeg_testset <- biodeg[-index_biodeg,]

response_train <- biodeg_trainset[,42]
response_test <- biodeg_testset[,42]

biodeg_train <- biodeg_trainset[,-42]
biodeg_test <- biodeg_testset[,-42]
```

Verifying training set distributions

```
par(mfrow = c(7,6), mar=c(2,2,2,2))

for(i in names(biodeg_train)){
  hist(biodeg_train[[i]] ,main=i,xlab="x",label=TRUE, plot = TRUE)
}
```

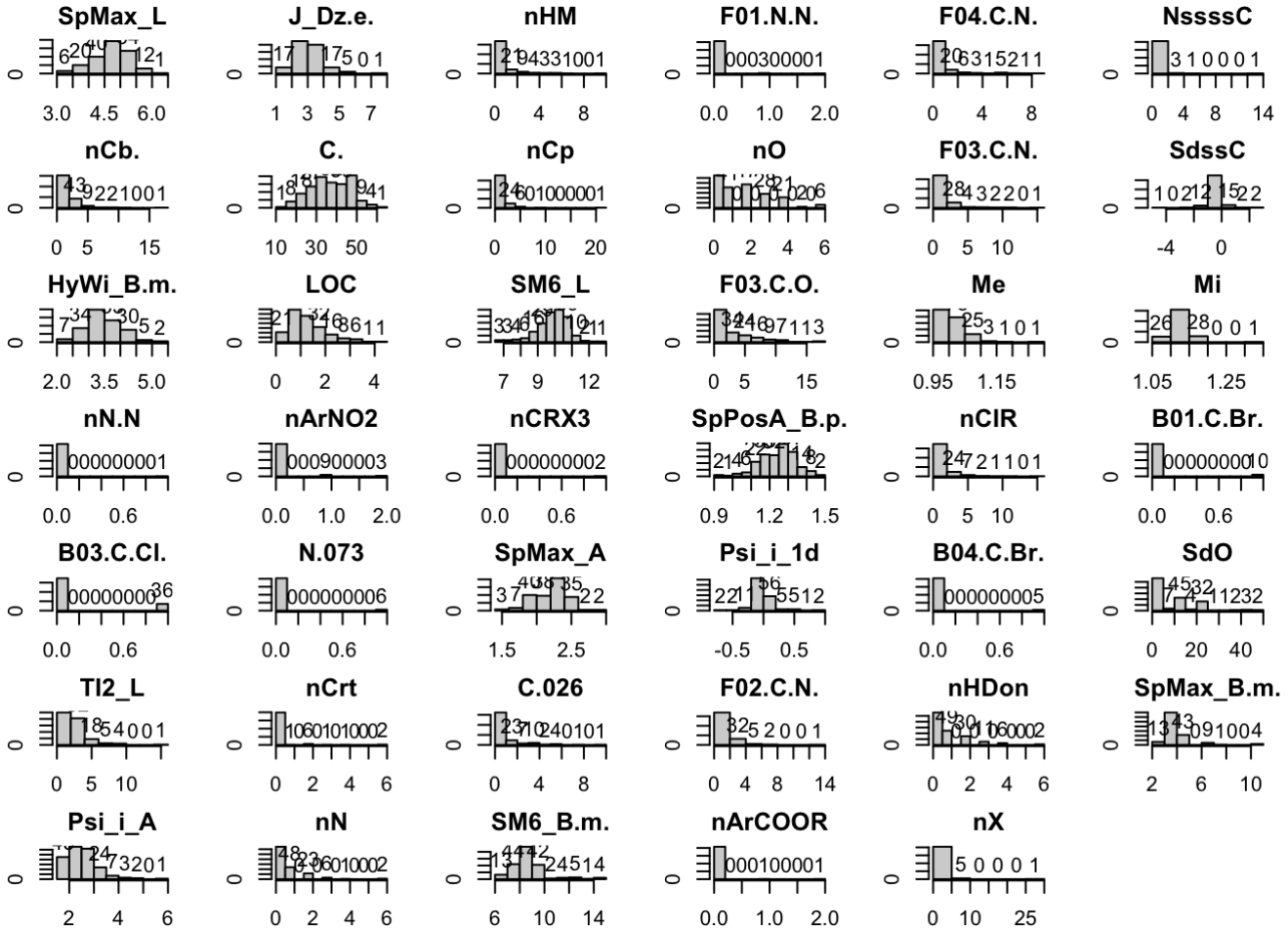



```
ggplot(biodeg_trainset, aes(experimental.class)) + geom_bar() + coord_flip()
```

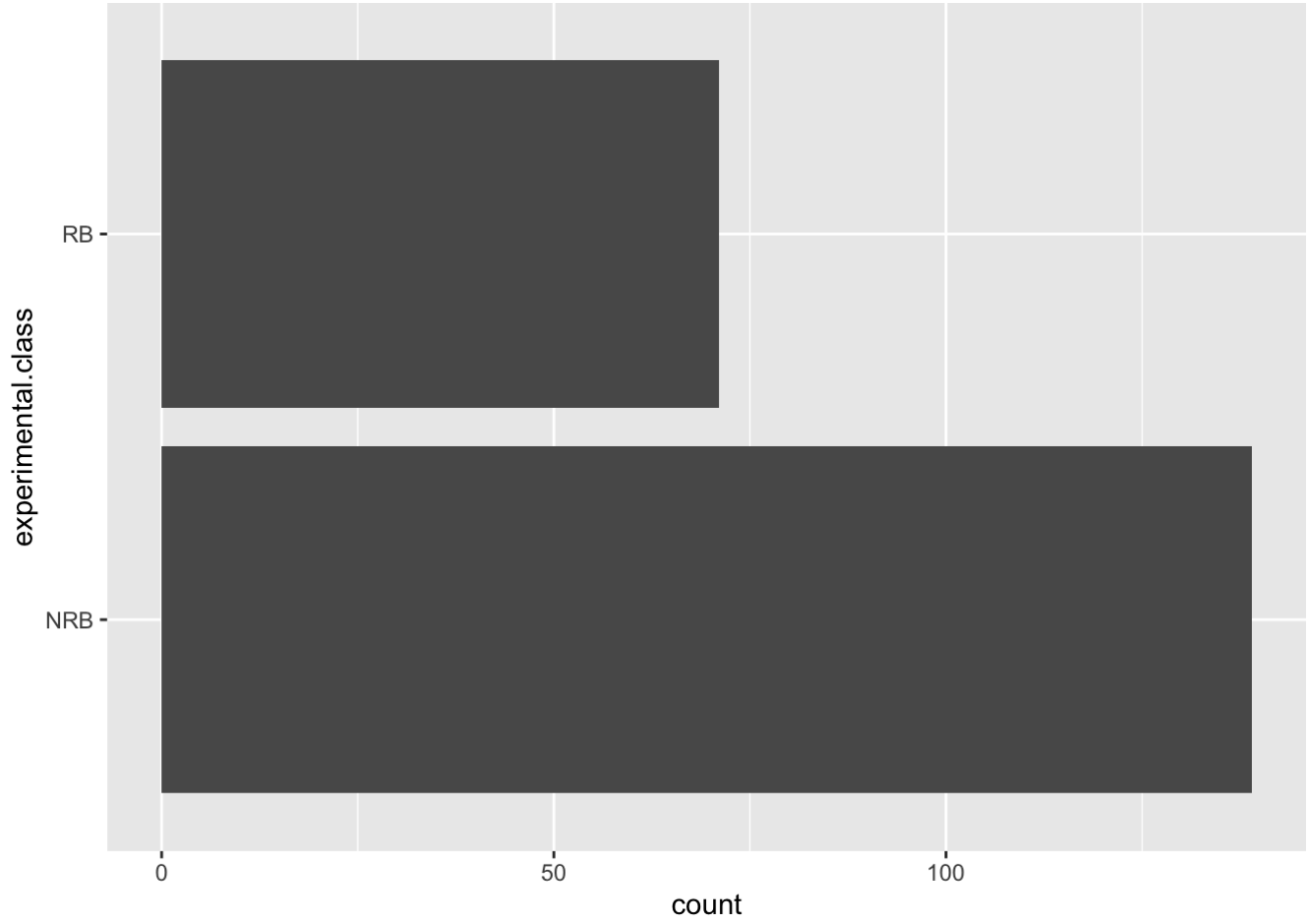


Verifying test set distributions

```
par(mfrow = c(7,6), mar=c(2,2,2,2))

for(i in names(biodeg_test)){
  hist(biodeg_test[[i]] ,main=i,xlab="x",label=TRUE, plot = TRUE)
}
```




```
ggplot(biodeg_testset, aes(experimental.class)) + geom_bar() + coord_flip()
```



Exporting train/test datasets for preprocessing

```
write.csv(biodeg_train,"../data/biodeg_train.csv", row.names = FALSE)
write.csv(biodeg_test,"../data/biodeg_test.csv", row.names = FALSE)
write.csv(response_train,"../data/response_train.csv", row.names = FALSE)
write.csv(response_test,"../data/response_test.csv", row.names = FALSE)
```

ADS503-02-SP22 - Final Project: Team 3

Carr_Aaron

06/27/2022

RMarkdown global setup

```
knitr::opts_chunk$set(
  fig.align = 'center'
  )
```

```
library(caret)
library(e1071)
```

Create function to generate boxplots for continuous variables

```
# Define function to produce formatted boxplots
box_comp <- function(xcol = c(), df = NA, rtn_met = TRUE) {
  sig <- 3
  metrics_df01 <- data.frame(metric = c("",
                                        "Total N:",
                                        "Count",
                                        "NA Count",
                                        "Mean",
                                        "Median",
                                        "Standard Deviation",
                                        "Variance",
                                        "Range",
                                        "Min",
                                        "Max",
                                        "25th Percentile",
                                        "75th Percentile",
                                        "Subset w/o Outliers:",
                                        "Count",
                                        "%",
                                        "Outlier %",
                                        "NA Count",
                                        "Mean",
                                        "Median",
                                        "Standard Deviation",
                                        "Variance",
                                        "Range",
                                        "Min",
                                        "Max"
                                        ))
  for (var in xcol) {
    df_s1 <- df[, var]
    df_s1s1 <-data.frame(df_s1)
    df_s1_fit <- preProcess(df_s1s1,
                            method = c("center", "scale"))
    df_s1_trans <- predict(df_s1_fit, df_s1s1)


    # Calculate quartiles
    var_iqr_lim <- IQR(df_s1) * 1.5
    var_q1 <- quantile(df_s1, probs = c(.25))
    var_otlow <- var_q1 - var_iqr_lim
    var_q3 <- quantile(df_s1, probs = c(.75))
    var_othigh <- var_q3 + var_iqr_lim

    # Subset non-outlier data
    var_non_otlr_df01 <- subset(df, (abs(df_s1_trans) <= 3))
    #var_non_otlr_df01 <- subset(df, (df_s1 > var_otlow & df_s1 < var_othigh))
    df_s2 <- var_non_otlr_df01[, var]

    # Begin calculating measures of centrality & dispersion
    var_mean <- mean(df_s1)
    var_non_otlr_df01_trunc_mean <- mean(df_s2)
```

```
    var_med <- median(df_s1)
    var_non_otlr_df01_trunc_med <- median(df_s2)
    var_mode <- mode(df_s1)
    var_non_otlr_df01_trunc_mode <- mode(df_s2)
    var_stde <- sd(df_s1)
    var_non_otlr_df01_trunc_stde <- sd(df_s2)
    var_vari <- var(df_s1)
    var_non_otlr_df01_trunc_vari <- var(df_s2)
    var01_min <- min(df[, var])
    var01_max <- max(df[, var])
    var01_range <- var01_max - var01_min
    var02_min <- min(var_non_otlr_df01[, var])
    var02_max <- max(var_non_otlr_df01[, var])
    var02_range <- var02_max - var02_min

    # Configure y-axis min & max to sync graphs
    plot_min <- min(var01_min, var02_min)
    plot_max <- max(var01_max, var02_max)
    nonoutlier_perc <- round((as.numeric(dim(var_non_otlr_df01)[1] / as.numeric(dim(df)[
1]))) * 100, 1)
    measure_val01 <- c(paste0("Variable: ", var),
                       "",
                       as.character(dim(df)[1]),
                       sum(is.na(df_s1)),
                       round(var_mean, sig),
                       round(var_med, sig),
                       round(var_stde, sig),
                       round(var_vari, sig),
                       round(var01_range, sig),
                       round(var01_min, sig),
                       round(var01_max, sig),
                       round(var_q1, sig),
                       round(var_q3, sig),
                       "",
                       as.character(dim(var_non_otlr_df01)[1]),
                       paste0(nonoutlier_perc, "%"),
                       paste0(round(100 - nonoutlier_perc, 1), "%"),
                       sum(is.na(df_s2)),
                       round(var_non_otlr_df01_trunc_mean, sig),
                       round(var_non_otlr_df01_trunc_med, sig),
                       round(var_non_otlr_df01_trunc_stde, sig),
                       round(var_non_otlr_df01_trunc_vari, sig),
                       round(var02_range, sig),
                       round(var02_min, sig),
                       round(var02_max, sig)
                       )

    var_name <- paste0("Variable: ", var)
    metrics_df01[, ncol(metrics_df01) + 1] <- measure_val01
}
  boxplot(df)
  if(rtn_met == TRUE) {
```

```
    return(metrics_df01)
  }
}
```

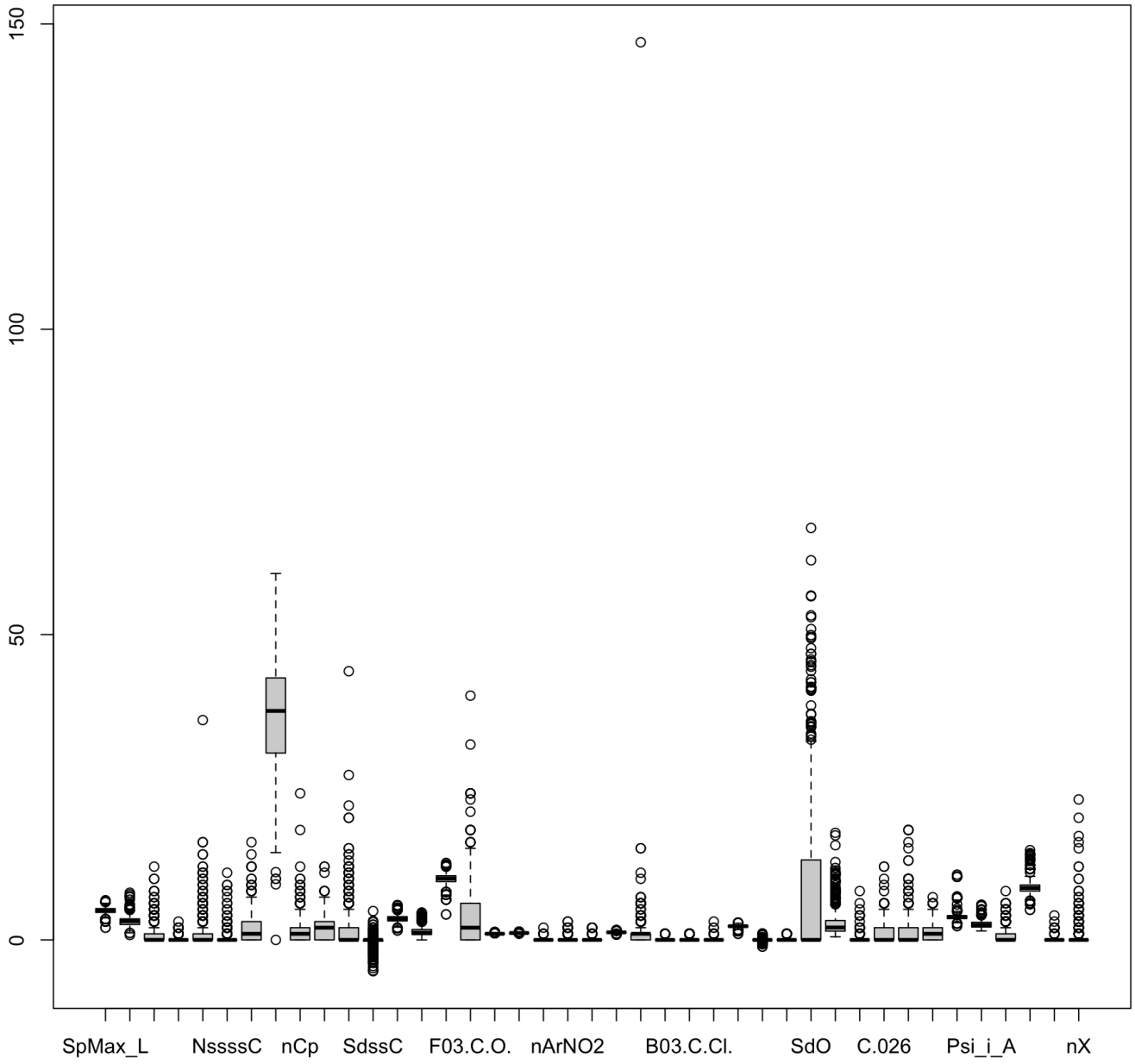
Importing Train/Test Datasets

```
train_x01_df01 <- read.csv("../data/outlier-included/biodeg_train.csv", header = TRUE, s
ep = ",")
test_x01_df01 <- read.csv("../data/outlier-included/biodeg_test.csv", header = TRUE, sep
= ",")

train_y01_df01 <- read.csv("../data/outlier-included/response_train.csv", header = TRUE,
sep = ",")
test_y01_df01 <- read.csv("../data/outlier-included/response_test.csv", header = TRUE, s
ep = ",")

train_y01_vc01 <- train_y01_df01[["x"]]
test_y01_vc01 <- test_y01_df01[["x"]]
```

Run function to create comparative boxplots

```
x01_lst01 <- c("SpMax_L",
               "J_Dz.e.",
               "nHM",
               "F01.N.N.",
               "F04.C.N.",
               "NssssC",
               "nCb.",
               "C.",
               "nCp",
               "nO",
               "F03.C.N.",
               "SdssC",
               "HyWi_B.m.",
               "LOC",
               "SM6_L",
               "F03.C.O.",
               "Me",
               "Mi",
               "nN.N",
               "nArNO2",
               "nCRX3",
               "SpPosA_B.p.",
               "nCIR",
               "B01.C.Br.",
               "B03.C.Cl.",
               "N.073",
               "SpMax_A",
               "Psi_i_1d",
               "B04.C.Br.",
               "SdO",
               "TI2_L",
               "nCrt",
               "C.026",
               "F02.C.N.",
               "nHDon",
               "SpMax_B.m.",
               "Psi_i_A",
               "nN",
               "SM6_B.m.",
               "nArCOOR",
               "nX")

x01_lst02 <- c("SpMax_L",
               "J_Dz.e.",
               "HyWi_B.m.",
               "LOC",
               "SpMax_A",
               "Psi_i_A")

x01_lst03 <- c("C.")

x01_lst04 <- c("SdO")
```

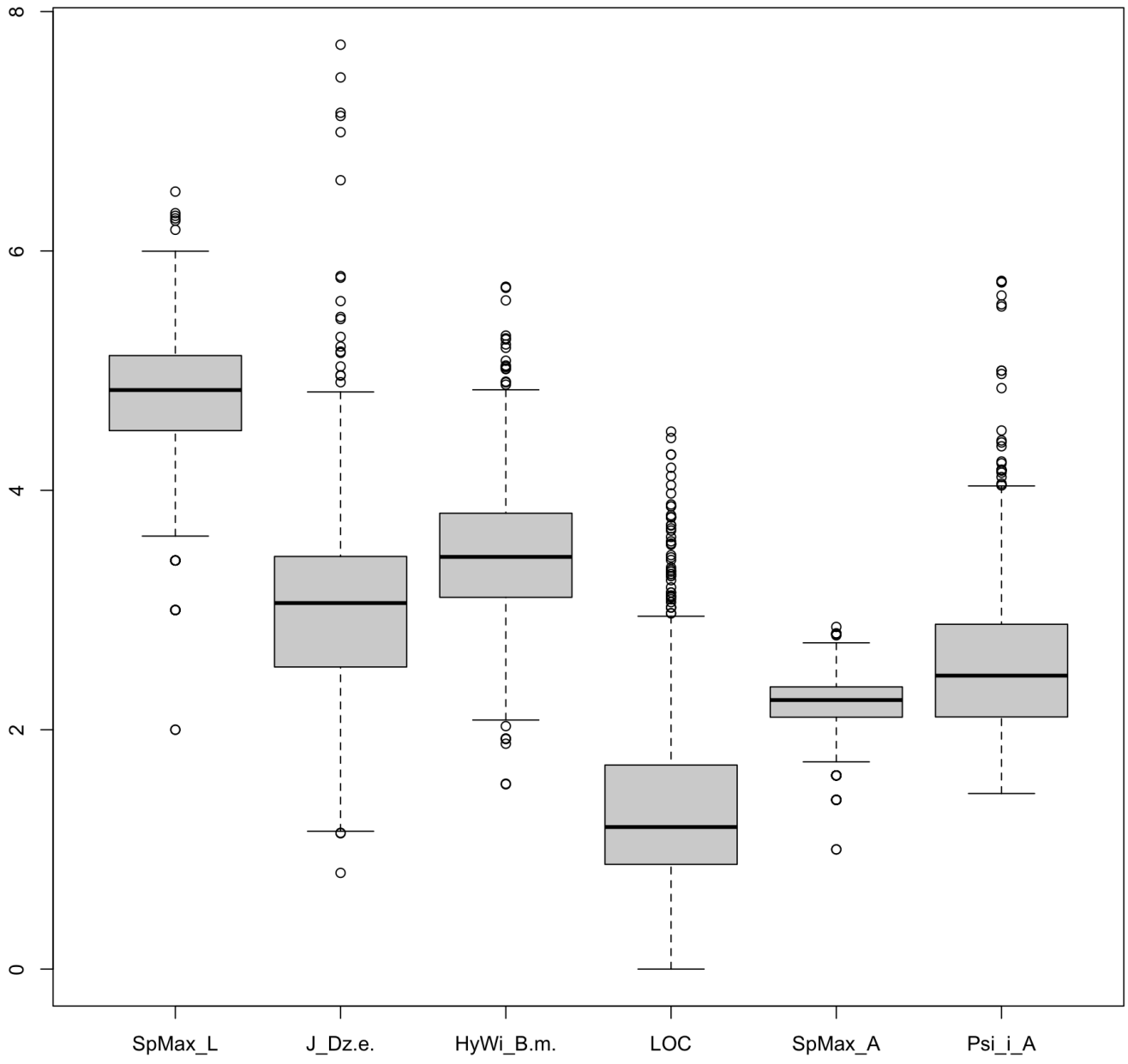
```
x01_lst05 <- c("nCIR")

x01_lst06 <- c("F01.N.N.",
               "nN.N",
               "nArNO2",
               "nCRX3",
               "B01.C.Br.",
               "B03.C.Cl.",
               "N.073",
               "B04.C.Br.",
               "nArCOOR")

x01_lst07 <- c("F04.C.N.",
               "F03.C.N.",
               "F03.C.O.")

x01_lst08 <- c("SdssC")

x01_lst09 <- c("Me",
               "Mi",
               "SpPosA_B.p.")

x01_lst10 <- c("SM6_L",
               "SM6_B.m.")

x01_lst11 <- c("nCb.",
               "nCp",
               "nO",
               "C.026",
               "F02.C.N.",
               "nHDon")

x01_lst12 <- c("TI2_L")

x01_lst13 <- c("nHM",
               "NssssC",
               "nCrt")

x01_lst14 <- c("nX")

x01_lst15 <- c("Psi_i_1d")

x01_lst16 <- c("SpMax_B.m.",
               "nN")

train_x01_df01_cols01 <- colnames(train_x01_df01)
train_x01_df01_metrics <- box_comp(xcol = train_x01_df01_cols01, df = train_x01_df01)
```



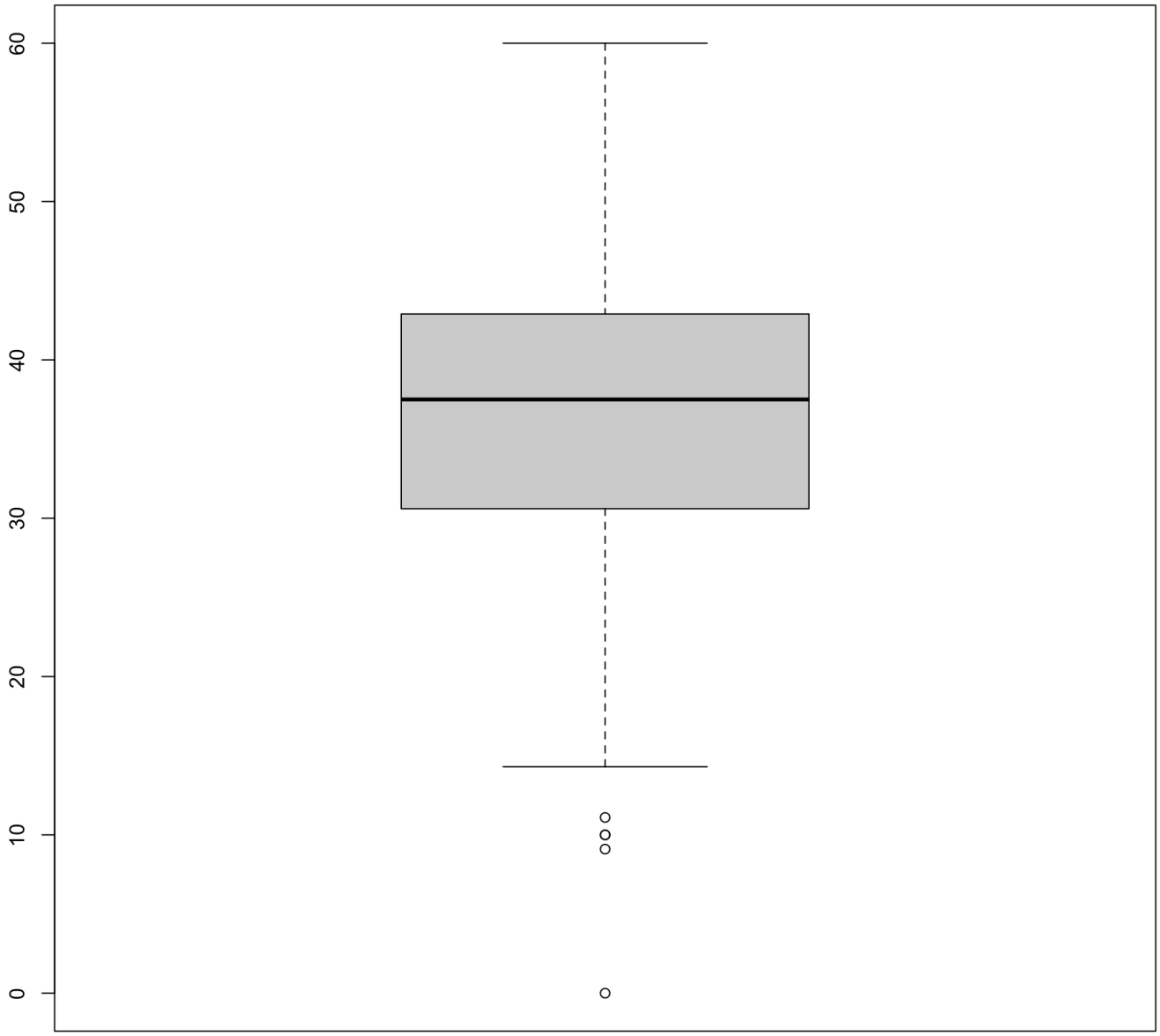

```
train_x01_df01_metrics
```

```
##                  metric                 V2                 V3             V4
## 1                       Variable: SpMax_L Variable: J_Dz.e. Variable: nHM
## 2             Total N:
## 3                 Count                845                845            845
## 4             NA Count                  0                  0              0
## 5                  Mean              4.793              3.073          0.701
## 6                Median              4.838              3.058              0
## 7    Standard Deviation              0.539              0.799          1.457
## 8              Variance               0.29              0.639          2.122
## 9                 Range              4.496              6.919             12
## 10                  Min                  2              0.804              0
## 11                  Max              6.496              7.723             12
## 12      25th Percentile              4.499              2.524              0
## 13      75th Percentile              5.126              3.448              1
## 14 Subset w/o Outliers:
## 15                Count                835                836            826
## 16                    %              98.8%              98.9%          97.8%
## 17            Outlier %               1.2%               1.1%           2.2%
## 18             NA Count                  0                  0              0
## 19                 Mean              4.812              3.034          0.552
## 20               Median              4.843              3.049              0
## 21   Standard Deviation              0.494              0.705          1.058
## 22             Variance              0.244              0.498           1.12
## 23                Range              2.902              4.645              5
## 24                  Min              3.414              0.804              0
## 25                  Max              6.316              5.449              5
##                    V5                 V6                 V7             V8
## 1  Variable: F01.N.N. Variable: F04.C.N. Variable: NssssC Variable: nCb.
## 2
## 3                 845                845                845            845
## 4                   0                  0                  0              0
## 5               0.047              1.03              0.293          1.647
## 6                   0                  0                  0              1
## 7               0.271              2.458              0.995          2.186
## 8               0.074              6.043               0.99          4.781
## 9                   3                 36                 11             16
## 10                  0                  0                  0              0
## 11                  3                 36                 11             16
## 12                  0                  0                  0              0
## 13                  0                  1                  0              3
## 14
## 15                816                827                831            833
## 16              96.6%              97.9%              98.3%          98.6%
## 17               3.4%               2.1%               1.7%           1.4%
## 18                  0                  0                  0              0
## 19                  0              0.771               0.19           1.51
## 20                  0                  0                  0              1
## 21                  0              1.504              0.529          1.859
## 22                  0              2.261              0.279          3.457
## 23                  0                  8                  3              8
## 24                  0                  0                  0              0
## 25                  0                  8                  3              8
```

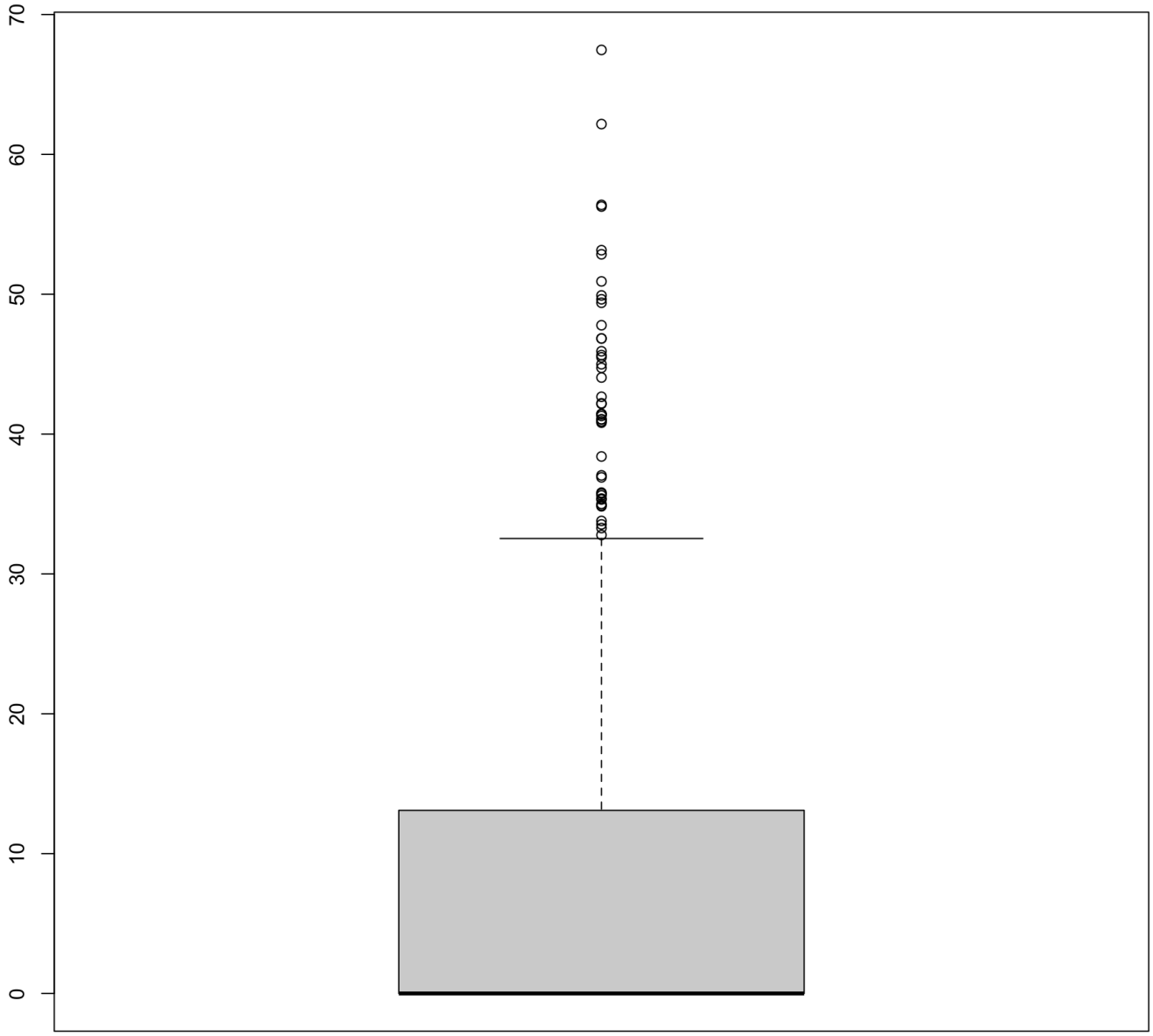
```
write.csv(train_x01_df01_metrics,"../outputs/demos.csv", row.names = FALSE)

box_comp(xcol = x01_lst02, df = subset(x = train_x01_df01, select = x01_lst02), rtn_met
 = FALSE)
```



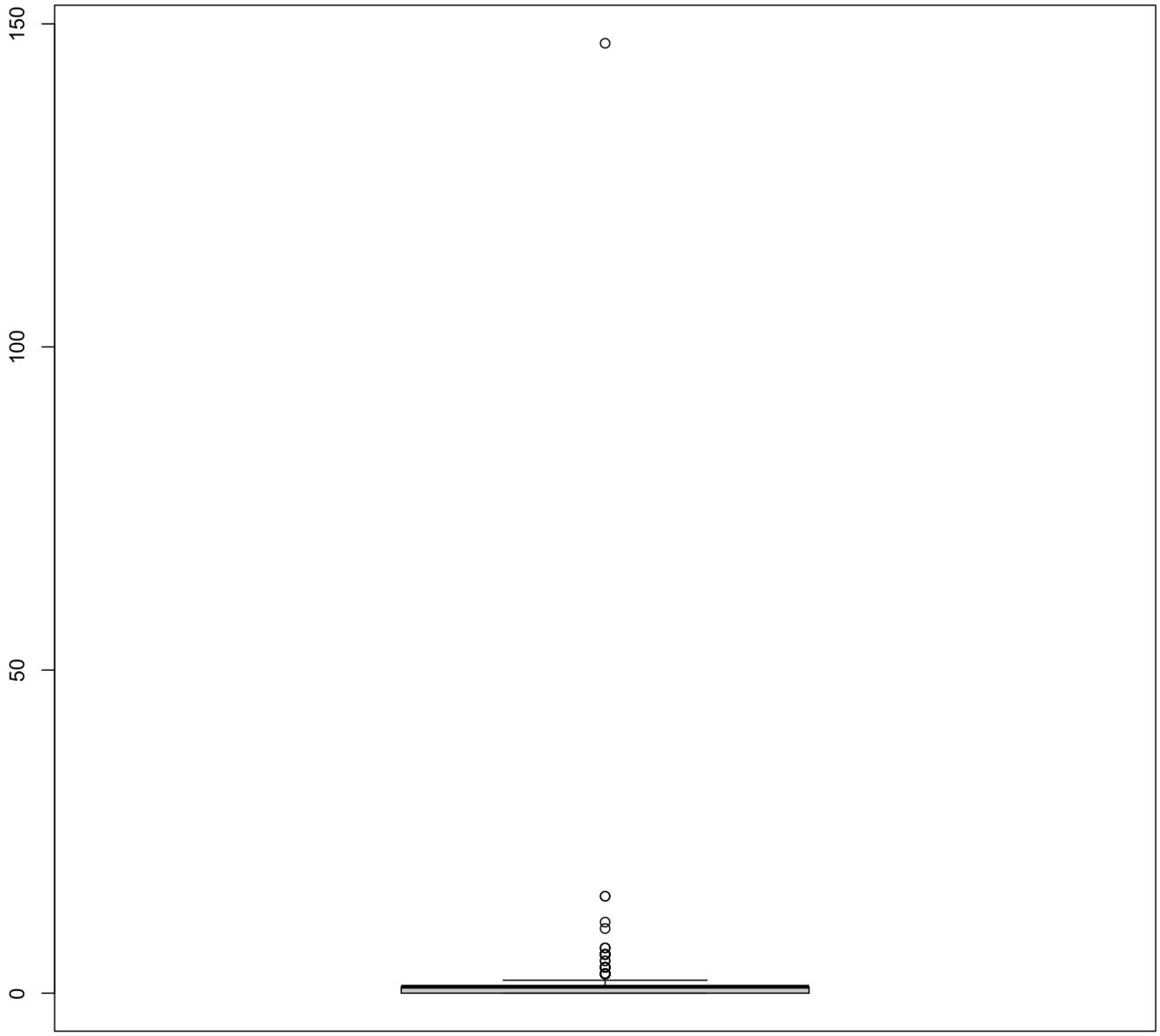

```
box_comp(xcol = x01_lst03, df = subset(x = train_x01_df01, select = x01_lst03), rtn_met
= FALSE)
```



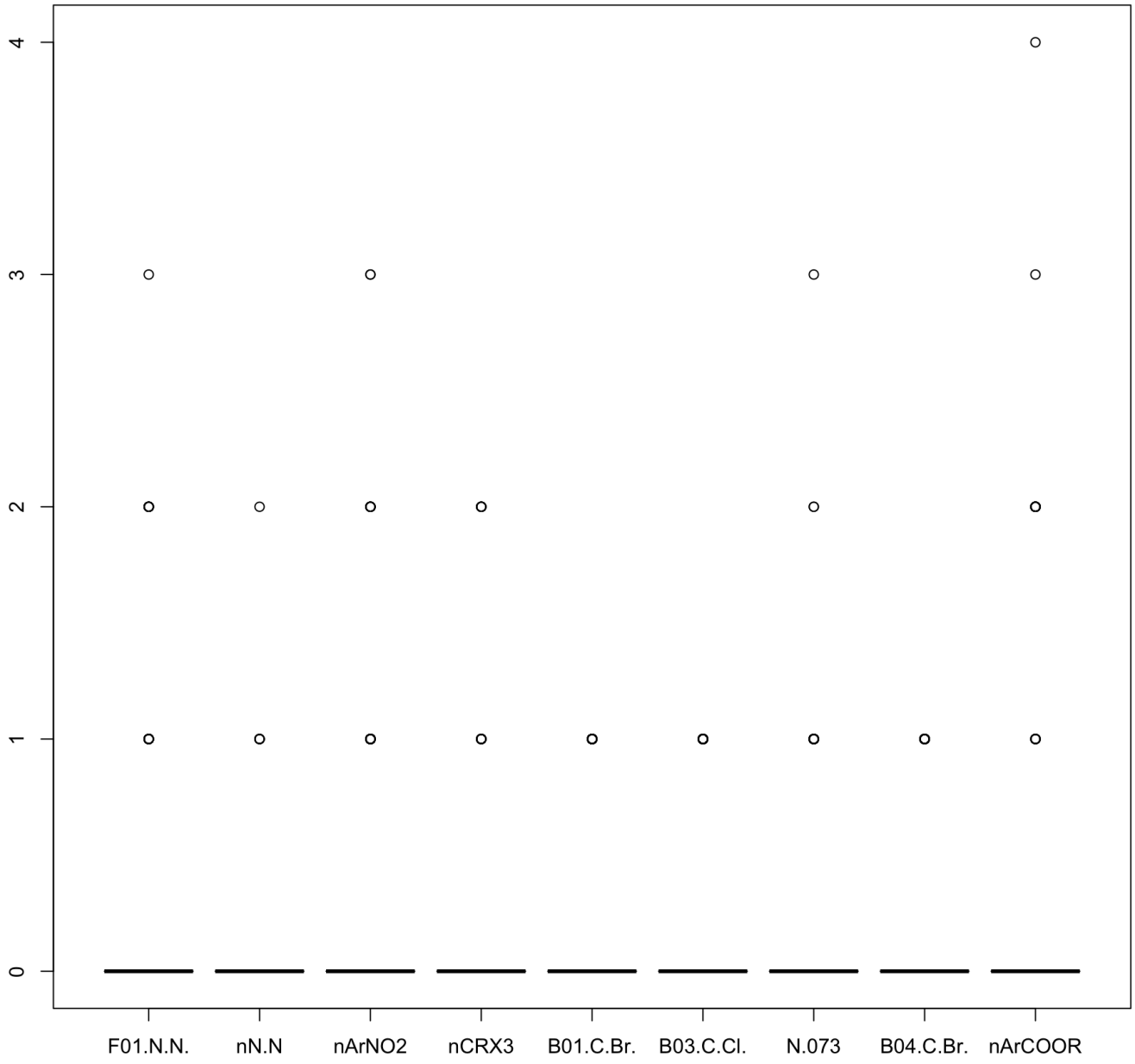

```
box_comp(xcol = x01_lst04, df = subset(x = train_x01_df01, select = x01_lst04), rtn_met
= FALSE)
```



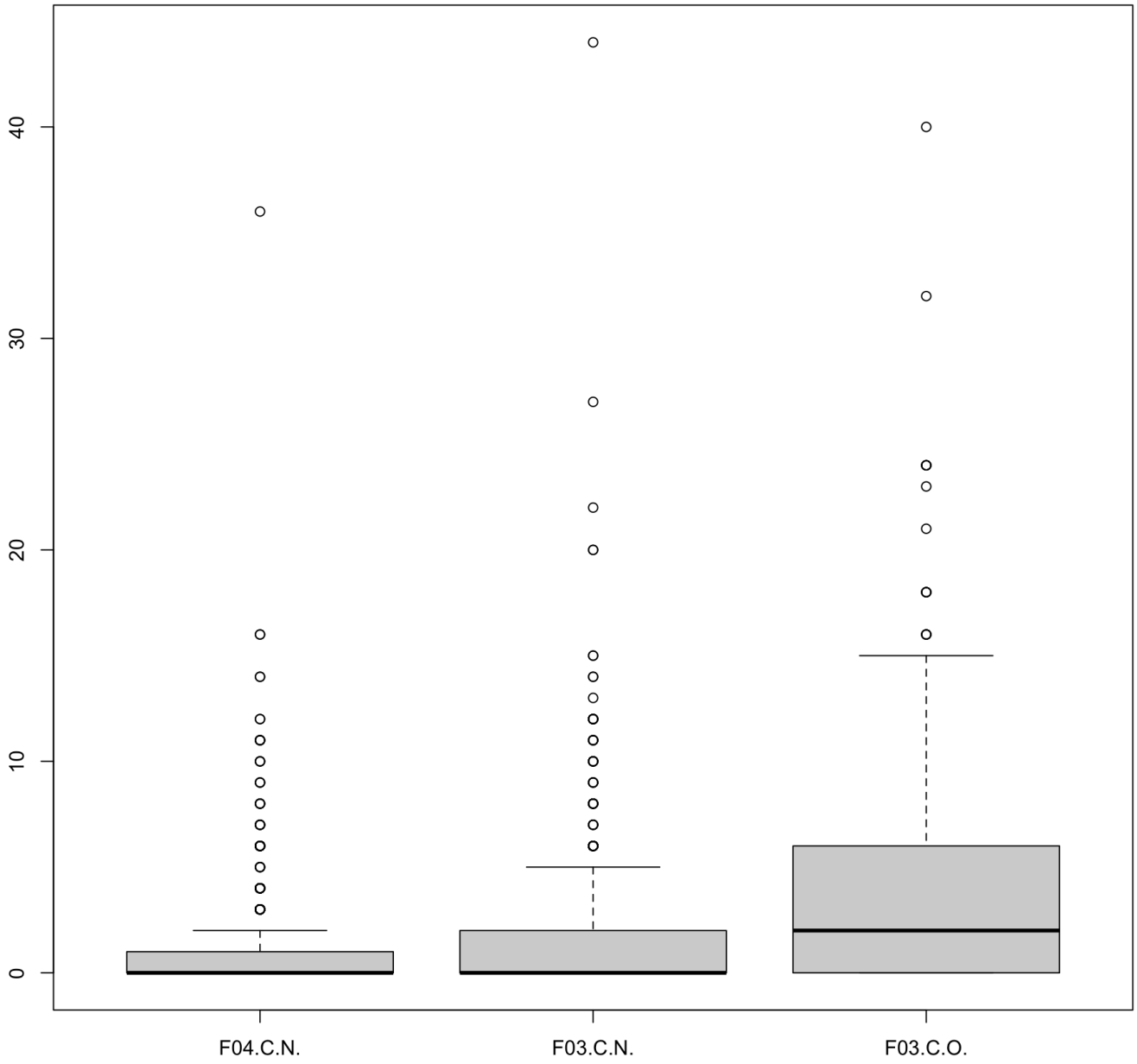

```
box_comp(xcol = x01_lst05, df = subset(x = train_x01_df01, select = x01_lst05), rtn_met
= FALSE)
```



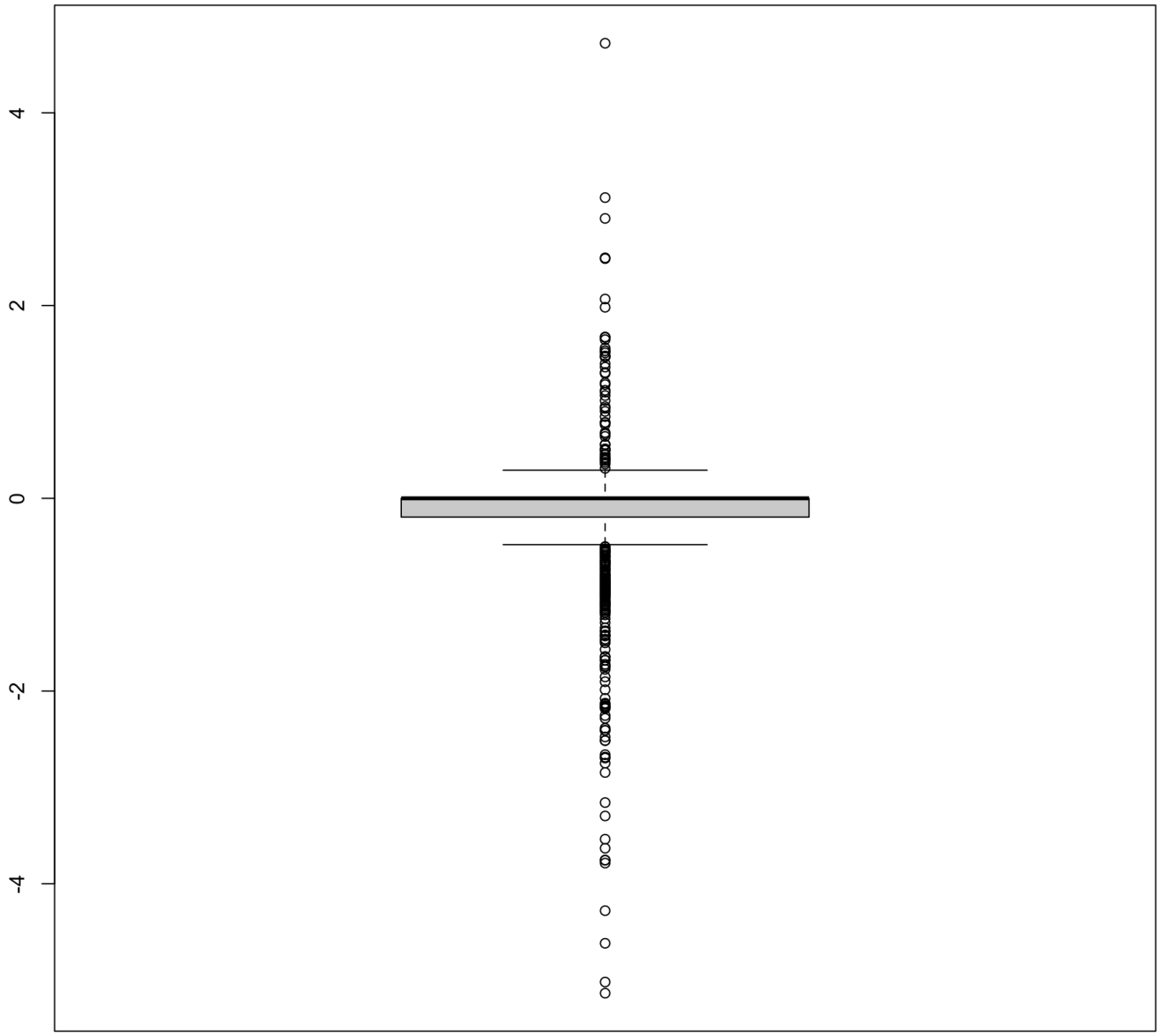

```
box_comp(xcol = x01_lst06, df = subset(x = train_x01_df01, select = x01_lst06), rtn_met
= FALSE)
```



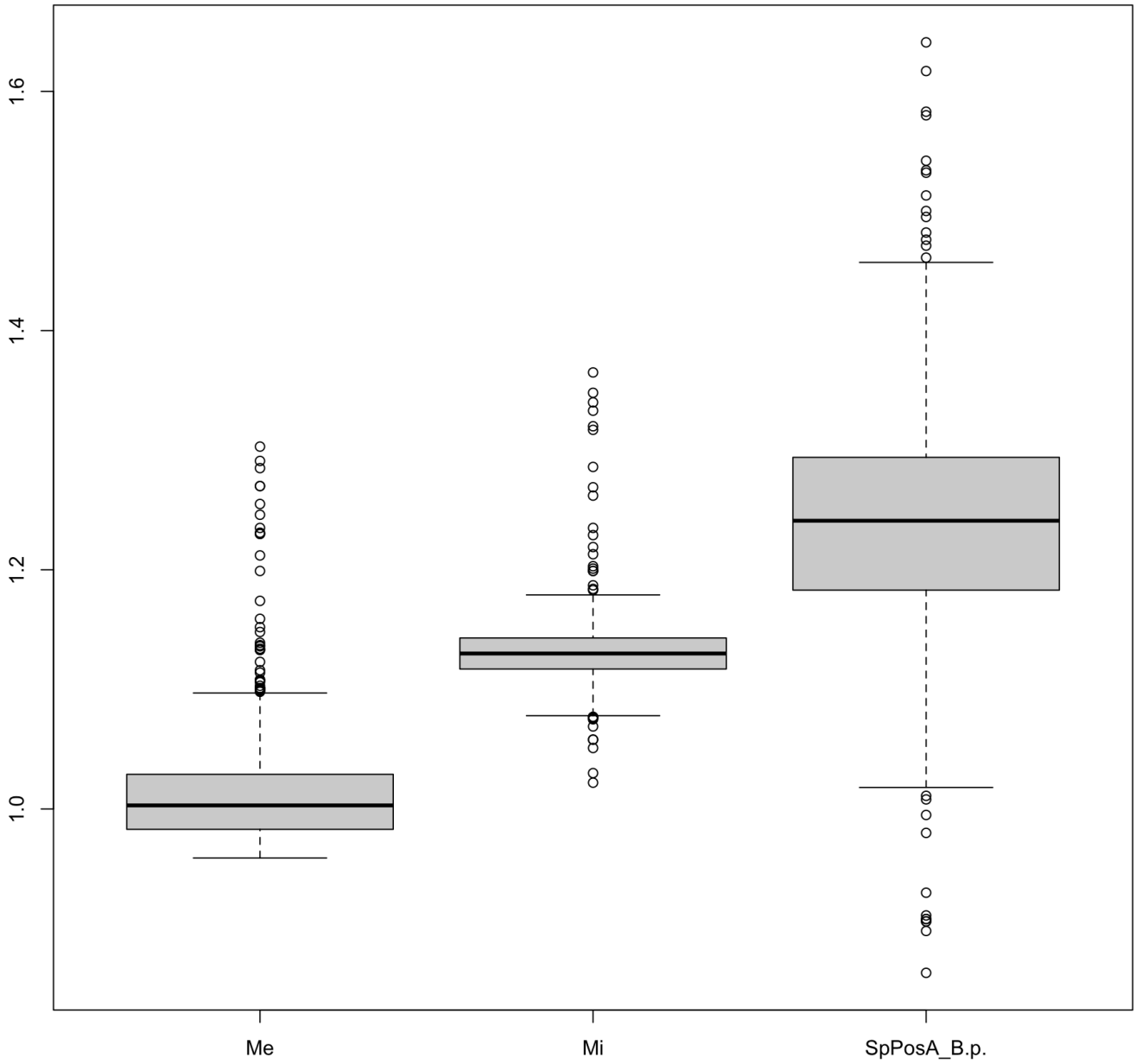

```
box_comp(xcol = x01_lst07, df = subset(x = train_x01_df01, select = x01_lst07), rtn_met
= FALSE)
```



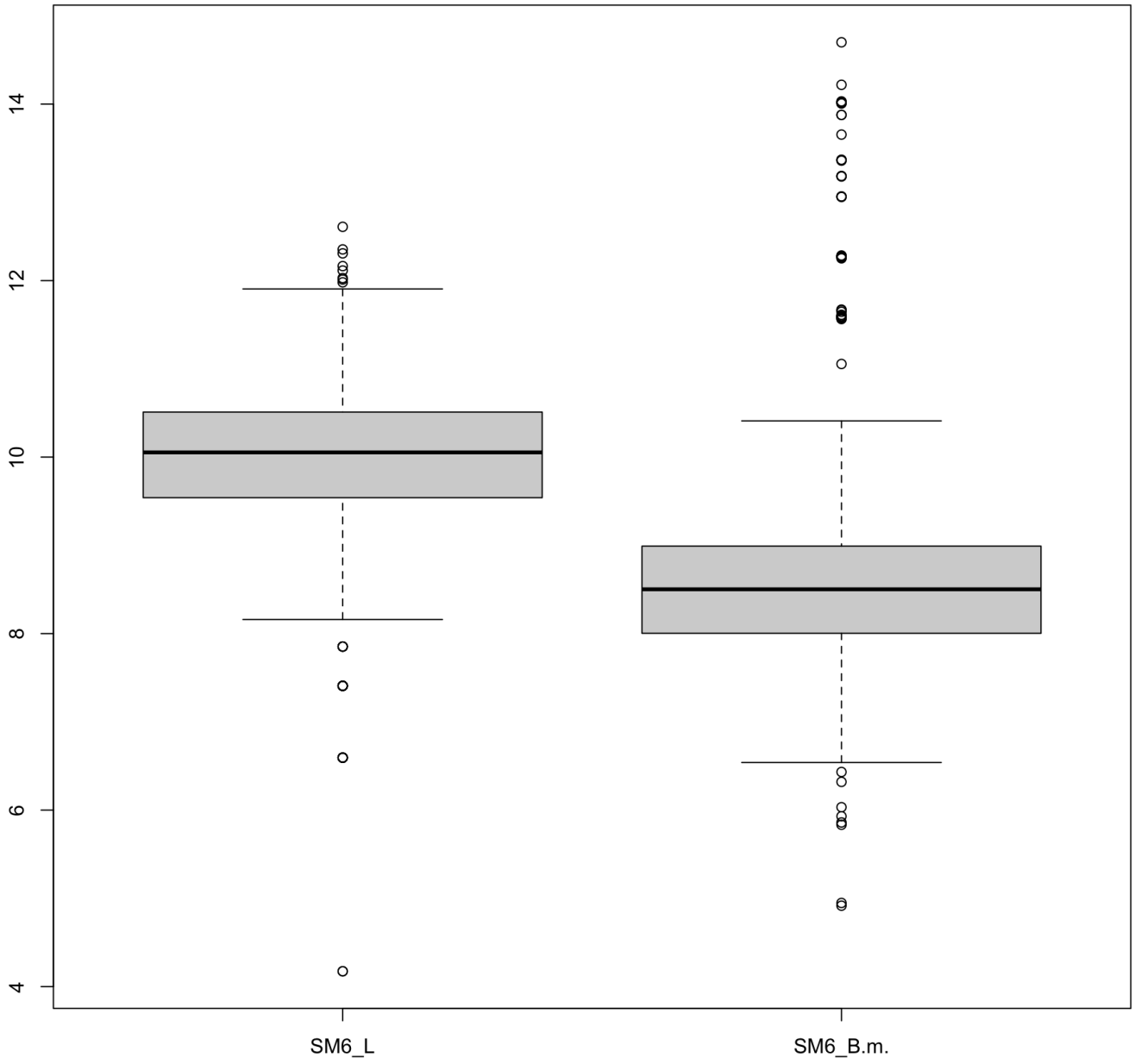

```
box_comp(xcol = x01_lst08, df = subset(x = train_x01_df01, select = x01_lst08), rtn_met
= FALSE)
```



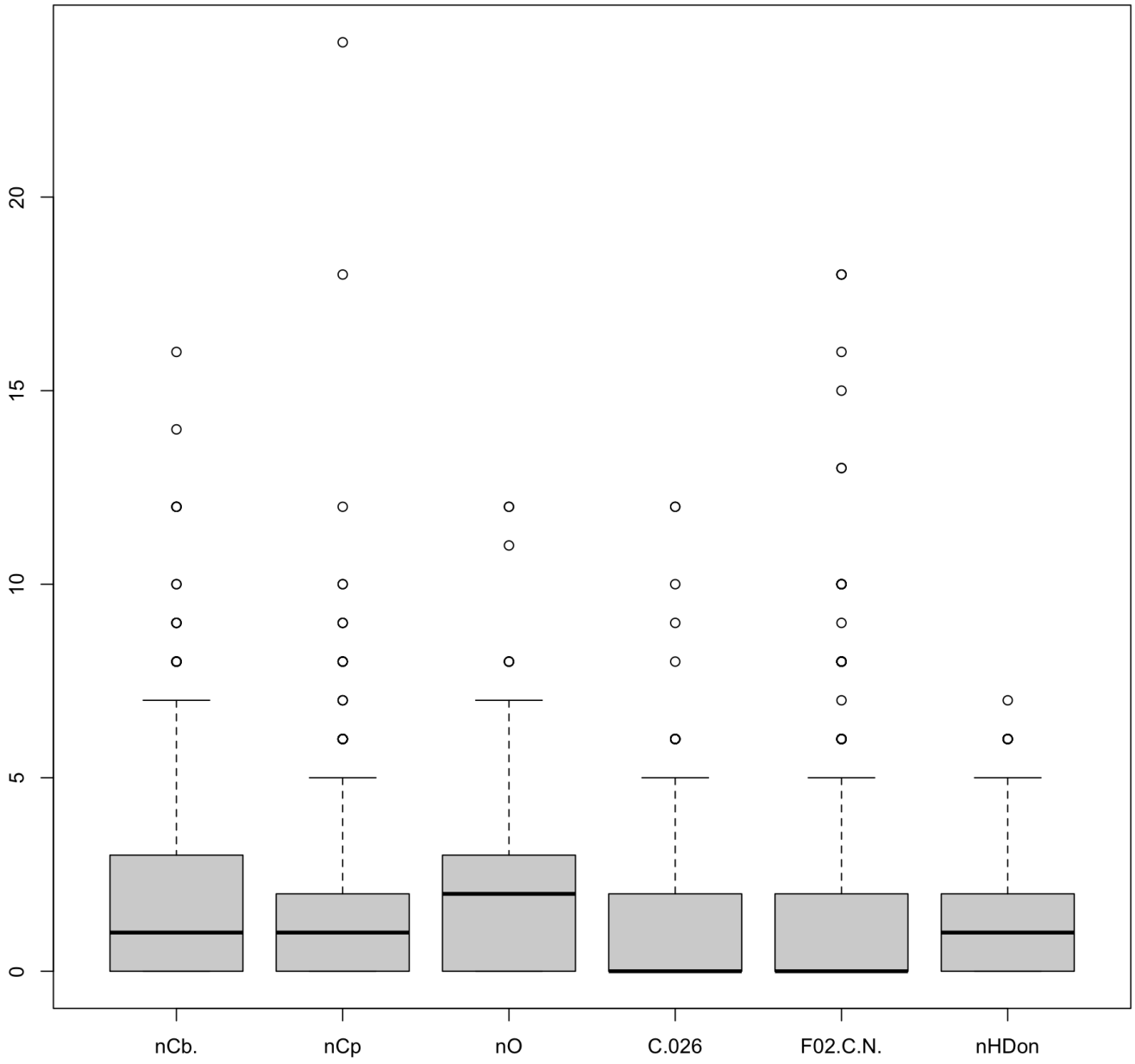

```
box_comp(xcol = x01_lst09, df = subset(x = train_x01_df01, select = x01_lst09), rtn_met
= FALSE)
```



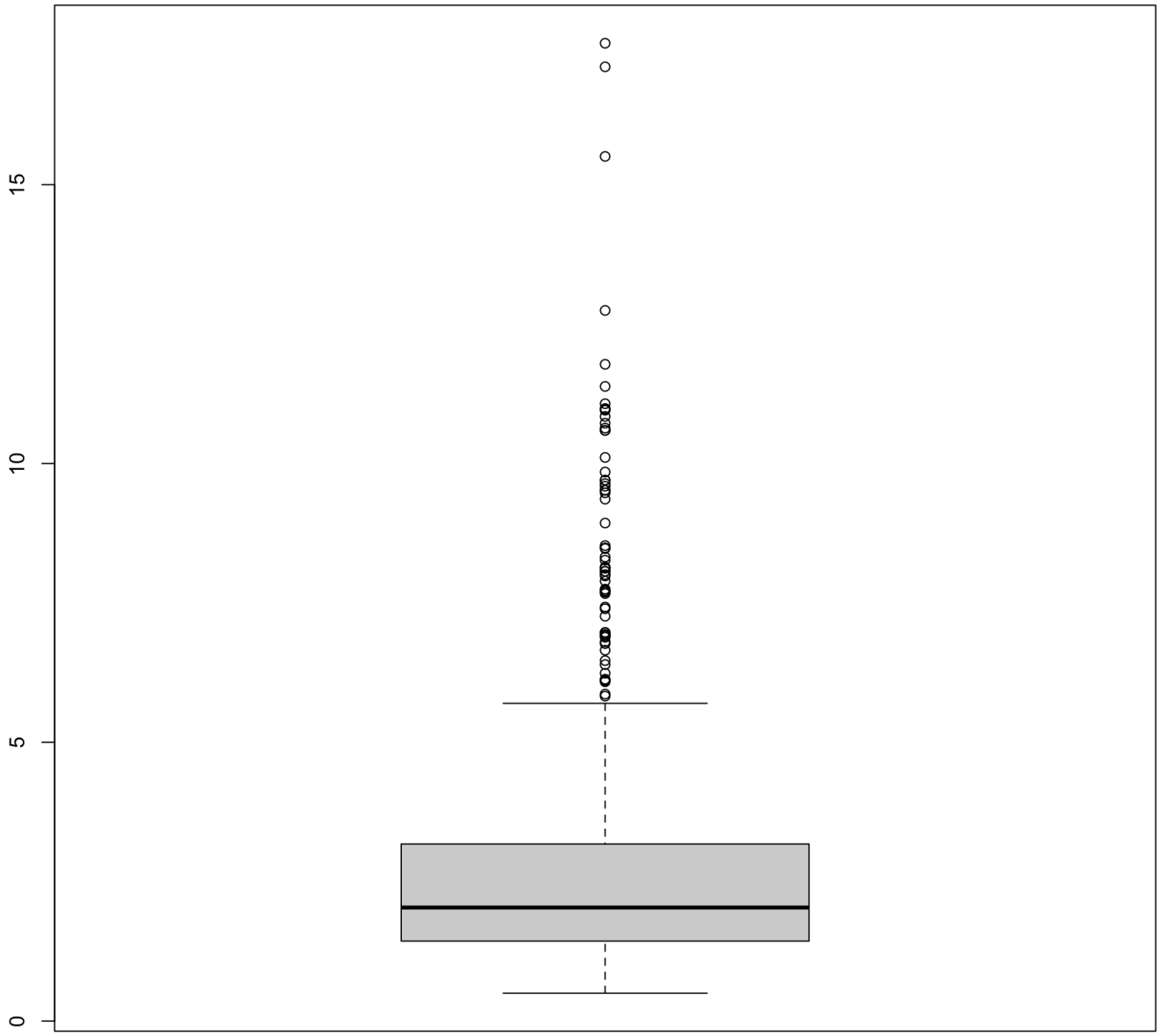

```
box_comp(xcol = x01_lst10, df = subset(x = train_x01_df01, select = x01_lst10), rtn_met
= FALSE)
```



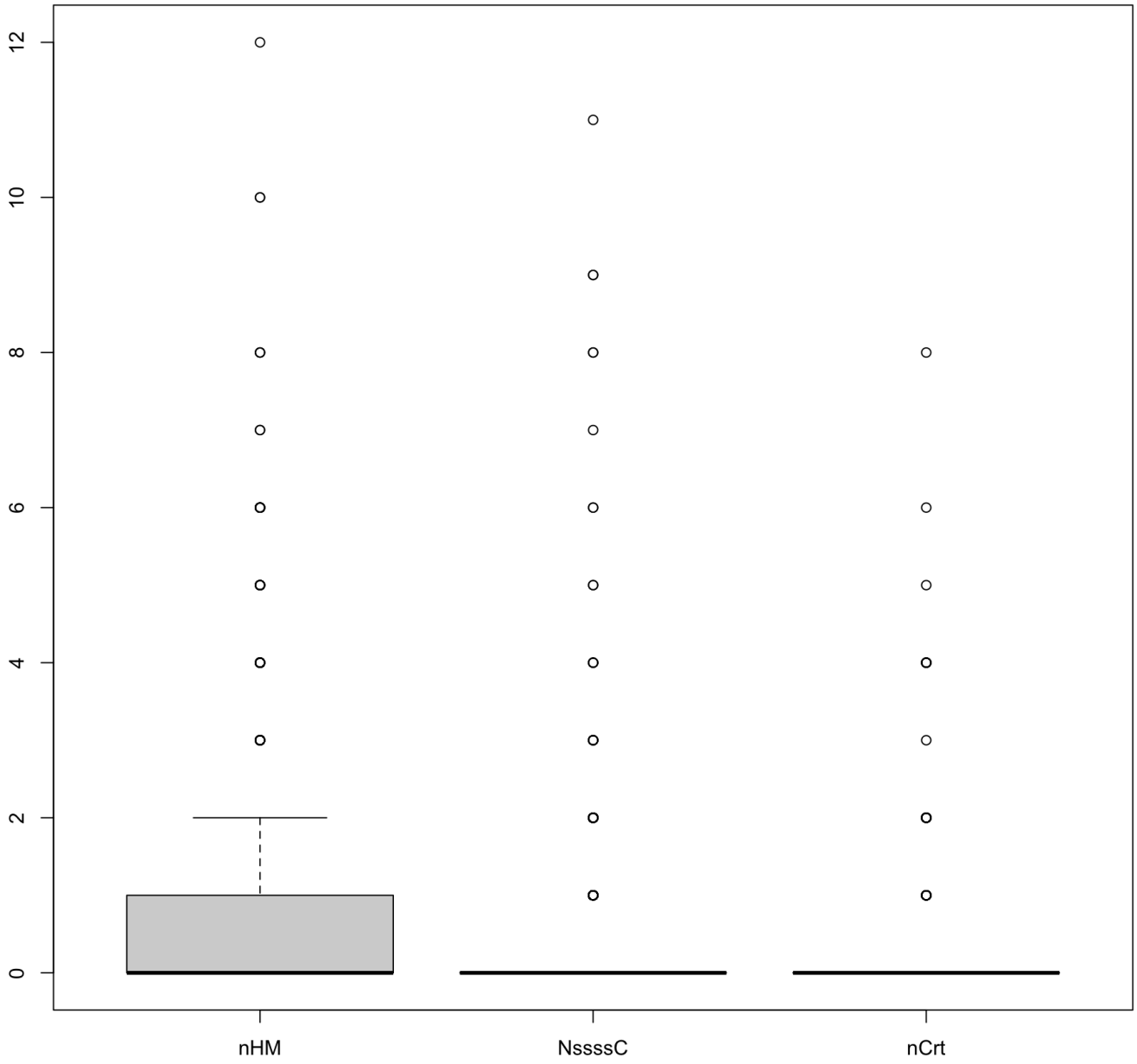

```
box_comp(xcol = x01_lst11, df = subset(x = train_x01_df01, select = x01_lst11), rtn_met
= FALSE)
```



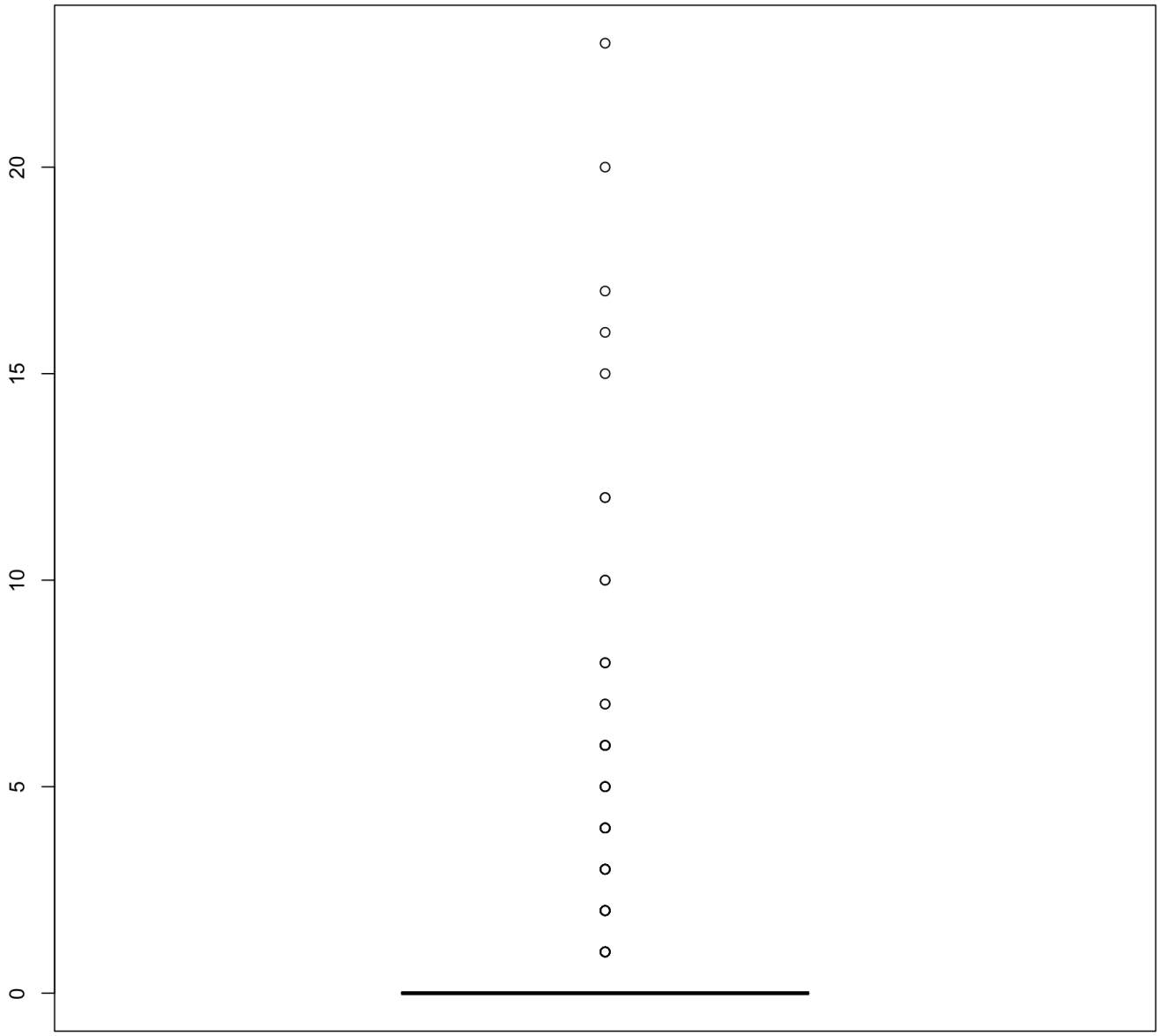

```
box_comp(xcol = x01_lst12, df = subset(x = train_x01_df01, select = x01_lst12), rtn_met
= FALSE)
```




```
box_comp(xcol = x01_lst13, df = subset(x = train_x01_df01, select = x01_lst13), rtn_met
= FALSE)
```

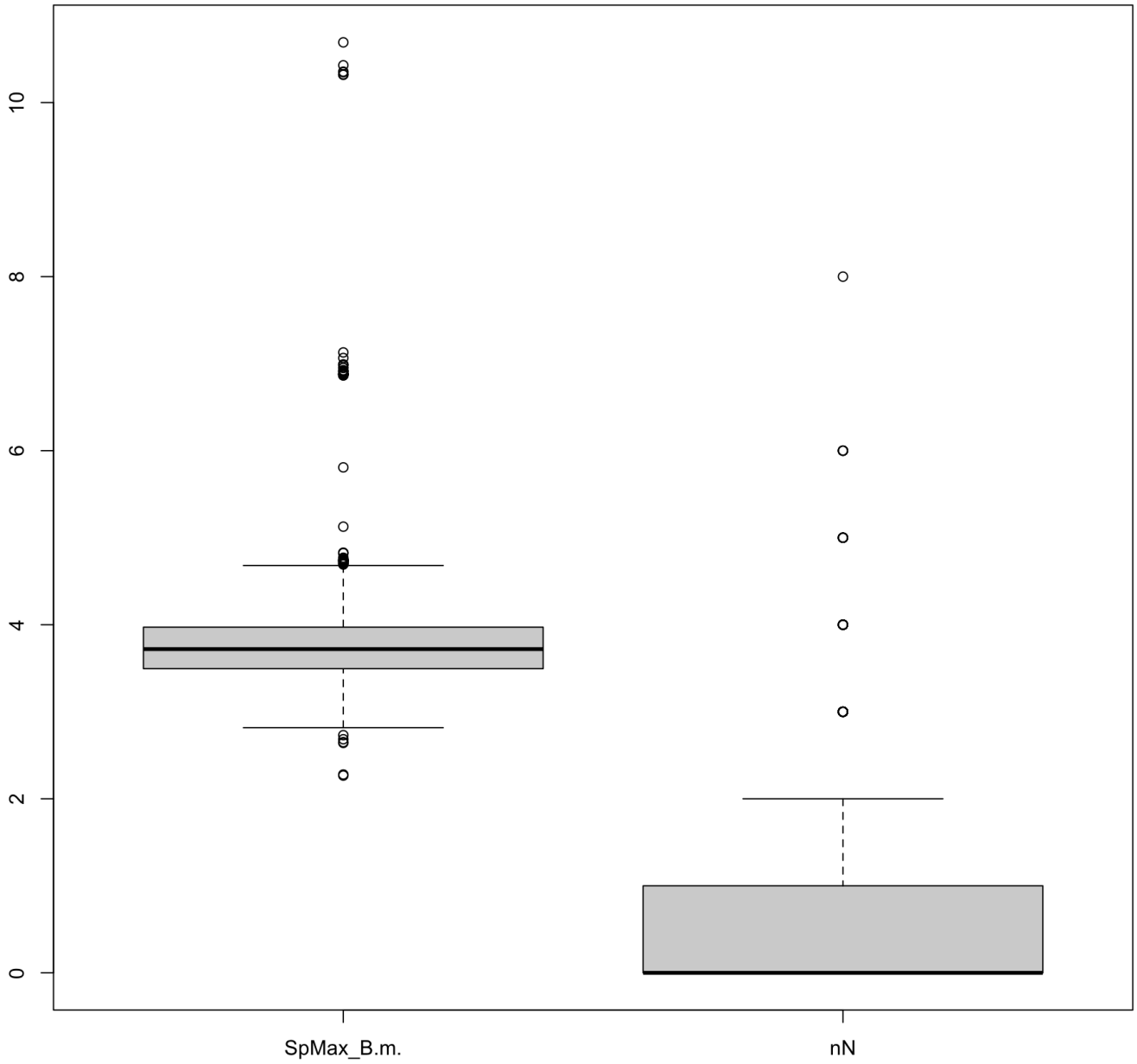



```
box_comp(xcol = x01_lst14, df = subset(x = train_x01_df01, select = x01_lst14), rtn_met
= FALSE)
```




```
box_comp(xcol = x01_lst15, df = subset(x = train_x01_df01, select = x01_lst15), rtn_met
= FALSE)
```


```
box_comp(xcol = x01_lst16, df = subset(x = train_x01_df01, select = x01_lst16), rtn_met
= FALSE)
```




```
train_x01_df01[train_x01_df01$nCIR > 68,]
```

```
##     SpMax_L J_Dz.e. nHM F01.N.N. F04.C.N. NssssC nCb.   C. nCp nO F03.C.N.
## 844   5.158  1.6914   2        0       36      0    9 56.1   0  0       44
##     SdssC HyWi_B.m.   LOC  SM6_L F03.C.O.    Me    Mi nN.N nArNO2 nCRX3
## 844     0     4.902 0.257 11.817        0 1.007 1.093    0      0     0
##     SpPosA_B.p. nCIR B01.C.Br. B03.C.Cl. N.073 SpMax_A Psi_i_1d B04.C.Br. SdO
## 844        1.41  147         0         1     2   2.622        0         0   0
##     TI2_L nCrt C.026 F02.C.N. nHDon SpMax_B.m. Psi_i_A nN SM6_B.m. nArCOOR nX
## 844 1.535    0     1       16     0      5.808   2.055  8   11.055       0  1
```

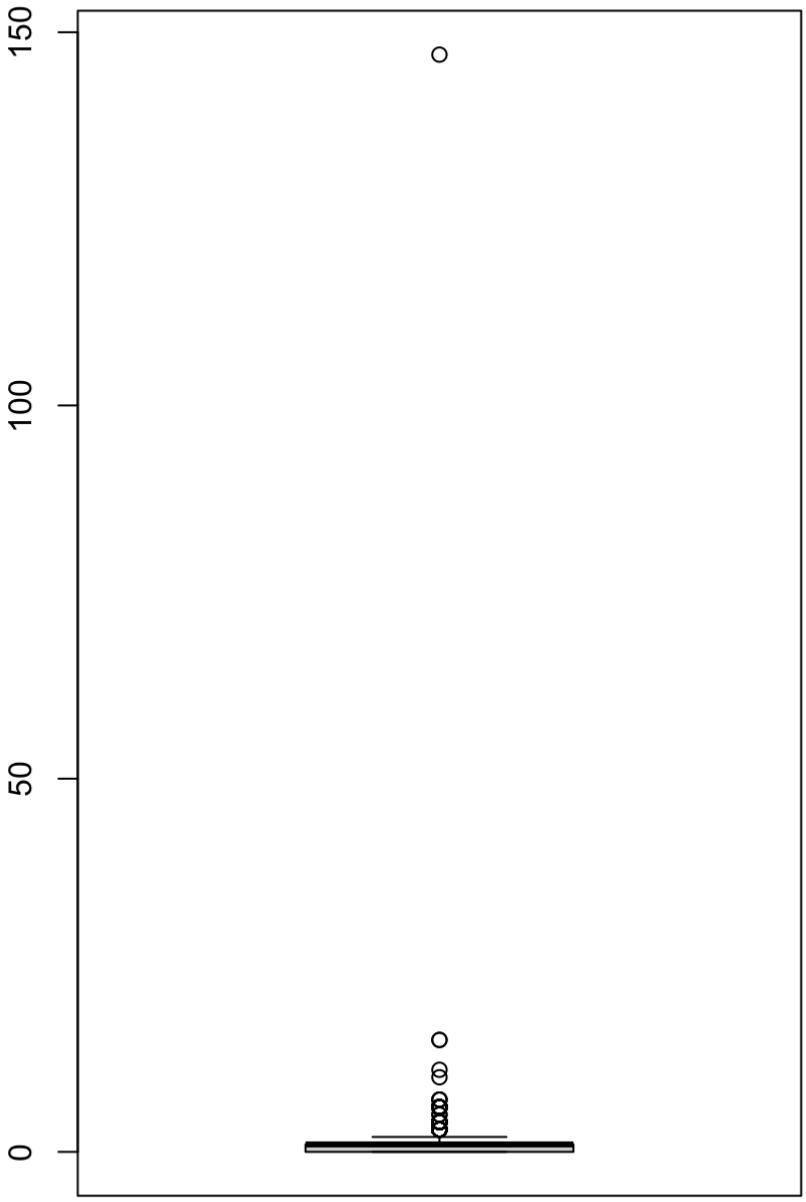
```
train_x01_df01[train_x01_df01$F04.C.N. > 33,]
```

```
##     SpMax_L J_Dz.e. nHM F01.N.N. F04.C.N. NssssC nCb.   C. nCp nO F03.C.N.
## 844   5.158  1.6914   2        0       36      0    9 56.1   0  0       44
##     SdssC HyWi_B.m.   LOC  SM6_L F03.C.O.    Me    Mi nN.N nArNO2 nCRX3
## 844     0     4.902 0.257 11.817        0 1.007 1.093    0      0     0
##     SpPosA_B.p. nCIR B01.C.Br. B03.C.Cl. N.073 SpMax_A Psi_i_1d B04.C.Br. SdO
## 844        1.41  147         0         1     2   2.622        0         0   0
##     TI2_L nCrt C.026 F02.C.N. nHDon SpMax_B.m. Psi_i_A nN SM6_B.m. nArCOOR nX
## 844 1.535    0     1       16     0      5.808   2.055  8   11.055       0  1
```

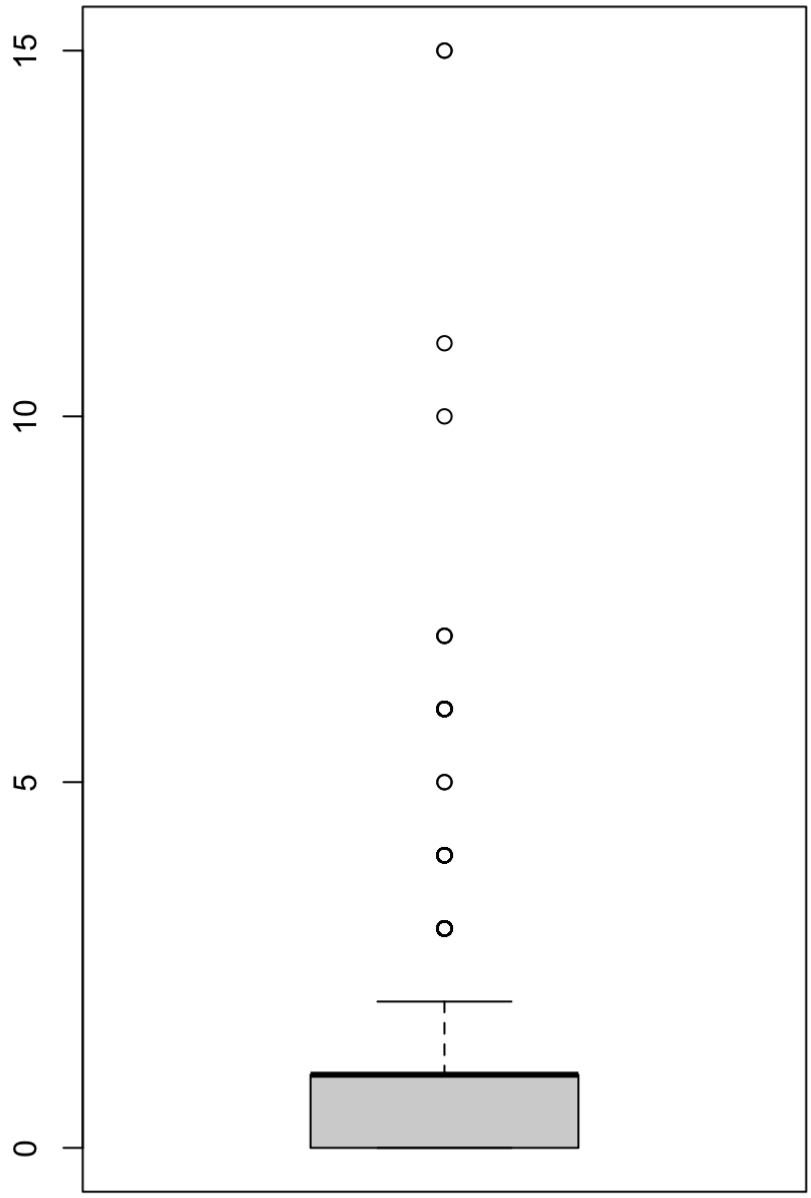
```
train_x01_df01[train_x01_df01$F03.C.N. > 44,]
```

```
##  [1] SpMax_L     J_Dz.e.     nHM         F01.N.N.    F04.C.N.    NssssC
##  [7] nCb.        C.          nCp         nO          F03.C.N.    SdssC
## [13] HyWi_B.m.   LOC         SM6_L       F03.C.O.    Me          Mi
## [19] nN.N        nArNO2      nCRX3       SpPosA_B.p. nCIR        B01.C.Br.
## [25] B03.C.Cl.   N.073       SpMax_A     Psi_i_1d    B04.C.Br.   SdO
## [31] TI2_L       nCrt        C.026       F02.C.N.    nHDon       SpMax_B.m.
## [37] Psi_i_A     nN          SM6_B.m.    nArCOOR     nX
## <0 rows> (or 0-length row.names)
```

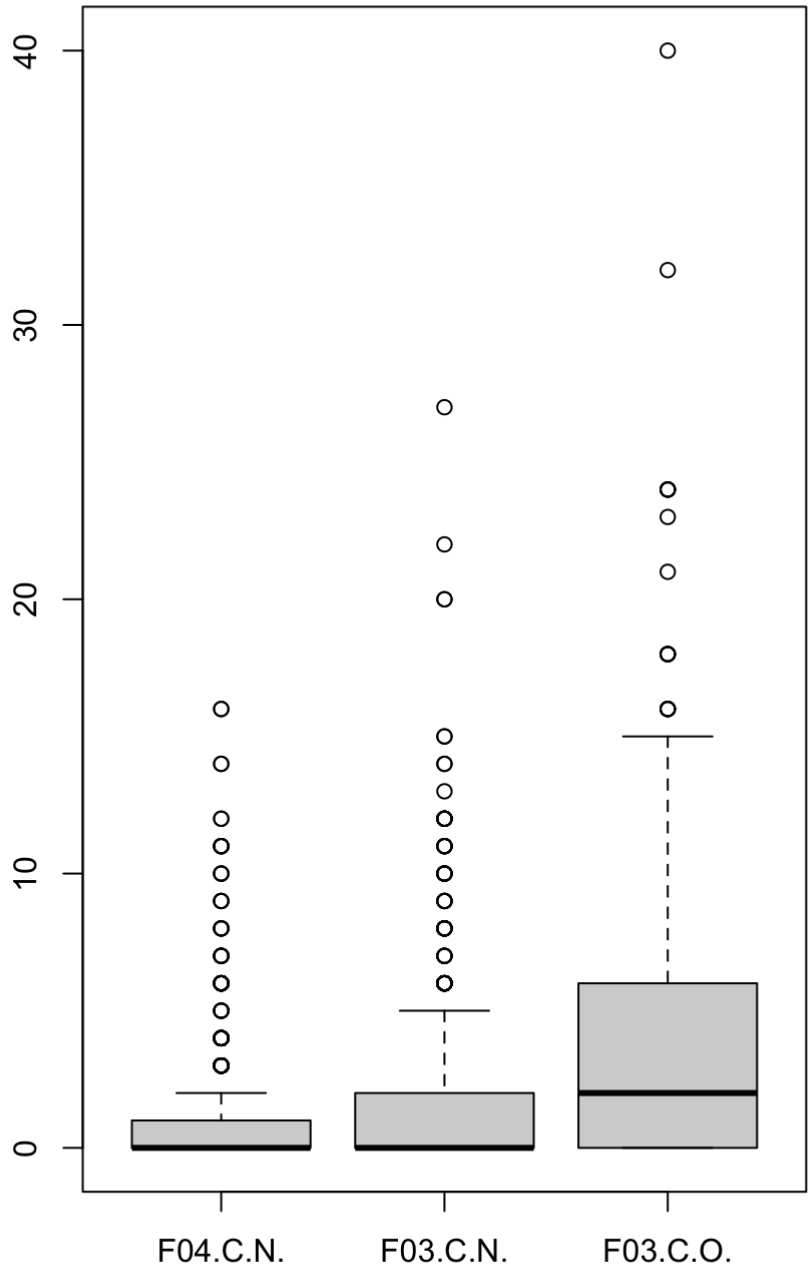
```
train_x01_df02 <- train_x01_df01[train_x01_df01$nCIR < 68,]
train_x01_df02 <- train_x01_df02[train_x01_df02$F04.C.N. < 33,]
train_x01_df02 <- train_x01_df02[train_x01_df02$F03.C.N. < 44,]

box_comp(xcol = x01_lst05, df = subset(x = train_x01_df01, select = x01_lst05), rtn_met
 = FALSE)
```




```
box_comp(xcol = x01_lst05, df = subset(x = train_x01_df02, select = x01_lst05), rtn_met
= FALSE)
```




```
box_comp(xcol = x01_lst07, df = subset(x = train_x01_df02, select = x01_lst07), rtn_met
= FALSE)
```




```
train_x01_df01_tbl01 <- table(train_x01_df01$B01.C.Br.)
train_x01_df01_tbl02 <- rbind(train_x01_df01_tbl01, round(prop.table(train_x01_df01_tbl0
1), 3))
rownames(train_x01_df01_tbl02) <- c("Count", "%")
train_x01_df01_tbl02
```

```
##             0      1
## Count 813.000 32.000
## %       0.962  0.038
```

```
train_x01_df01_tbl03 <- table(train_x01_df01$B03.C.Cl.)
train_x01_df01_tbl04 <- rbind(train_x01_df01_tbl03, round(prop.table(train_x01_df01_tbl0
3), 3))
rownames(train_x01_df01_tbl04) <- c("Count", "%")
train_x01_df01_tbl04
```

```
##             0       1
## Count 724.000 121.000
## %       0.857   0.143
```

```
train_x01_df01_tbl05 <- table(train_x01_df01$B04.C.Br.)
train_x01_df01_tbl06 <- rbind(train_x01_df01_tbl05, round(prop.table(train_x01_df01_tbl0
5), 3))
rownames(train_x01_df01_tbl06) <- c("Count", "%")
train_x01_df01_tbl06
```

```
##             0      1
## Count 822.000 23.000
## %       0.973  0.027
```

```
sapply(train_x01_df01, function(x) skewness(x))
```

```
##      SpMax_L      J_Dz.e.          nHM     F01.N.N.     F04.C.N.       NssssC
## -0.715734946  1.201674548  3.095209730  6.467526981  5.611477940  5.822338020
##         nCb.           C.          nCp           nO     F03.C.N.        SdssC
##  1.933944864 -0.057342873  3.780481570  1.340676964  5.045564445 -1.549970539
##    HyWi_B.m.          LOC        SM6_L     F03.C.O.           Me           Mi
##  0.352635490  1.223596820 -1.157870100  2.126576012  2.539690998  2.710728417
##         nN.N       nArNO2        nCRX3  SpPosA_B.p.         nCIR    B01.C.Br.
## 12.731567332  4.956649133  7.608922009  0.003010695 25.210180599  4.833473907
##    B03.C.Cl.        N.073      SpMax_A     Psi_i_1d    B04.C.Br.          SdO
##  2.033686241  7.948807551 -1.005599432 -0.099833884  5.800635272  1.514137530
##        TI2_L         nCrt        C.026     F02.C.N.        nHDon   SpMax_B.m.
##  2.679123868  6.795988929  2.613420017  3.042041970  1.630725564  3.878578920
##      Psi_i_A           nN     SM6_B.m.      nArCOOR           nX
##  1.476296015  2.182897210  1.793370995  6.327040774  5.491467120
```

```
x01_df01_train_fit01 <- preProcess(train_x01_df01,
                                   method = c("center", "scale"))

train_x01_trans_df01 <- predict(x01_df01_train_fit01, train_x01_df01)
sapply(train_x01_trans_df01, function(x) sum(abs(x) > 3))
```

```
##     SpMax_L     J_Dz.e.         nHM    F01.N.N.    F04.C.N.      NssssC
##          10           9          19          29          18          14
##       nCb.          C.         nCp          nO    F03.C.N.       SdssC
##          12           4          13           7          17          21
##   HyWi_B.m.         LOC       SM6_L    F03.C.O.          Me          Mi
##           8          15           9          12          15          13
##       nN.N      nArNO2       nCRX3 SpPosA_B.p.        nCIR   B01.C.Br.
##           7          11          19          15           1          32
##   B03.C.Cl.       N.073     SpMax_A    Psi_i_1d   B04.C.Br.         SdO
##           0          23           9          27          23          16
##      TI2_L        nCrt       C.026    F02.C.N.       nHDon  SpMax_B.m.
##          25          28          16          14          18          39
##     Psi_i_A          nN    SM6_B.m.     nArCOOR          nX
##          11          13          28          20          14
```

1.0-sk-lda

Sreeja K

2022-06-25

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✔ tibble  3.1.7     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
## ✔ purrr   0.3.4
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(mlbench)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(100)
```

```
biodeg_train <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
biodeg_test <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")

response_train <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
response_test <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

response_train <- as.factor(as.matrix(response_train))
response_test <- as.factor(as.matrix(response_test))

y_pred_test <- read.csv("../data/test-results/y_pred.csv", header = TRUE, sep = ",")
y_pred_test <- lapply(y_pred_test, function(x) (as.factor(x)))
```

#Support_Vector_Machines

```
ctrl <- trainControl(method = "cv", number = 5, summaryFunction = twoClassSummary, class
Probs = TRUE,   savePredictions = TRUE)

#svm fit
#SVMFIT <- train(x = biodeg_train, y = response_train, method = "svmLinear", preProcess
 = c("center","scale"),  trControl = ctrl)

SVMFIT <- train(x = biodeg_train, y = response_train, method = "svmRadial", preProcess =
c("center","scale"),  trControl = ctrl)
```

```
## Warning in train.default(x = biodeg_train, y = response_train, method =
## "svmRadial", : The metric "Accuracy" was not in the result set. ROC will be used
## instead.
```

```
svm_pred<- data.frame(obs = response_test, SVM = predict(SVMFIT, biodeg_test))
#svm_ROC = pROC::roc(response = response_train, predictor = svm_Pred[, 1])

#svm_AUC = svm_ROC$auc[1]
#svm = list(classifier = SVMFIT, predictions = svm_Pred, roc = svm_ROC, auc = svm_AUC)
SVMFIT
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 844 samples
##  41 predictor
##   2 classes: 'NRB', 'RB'
##
## Pre-processing: centered (41), scaled (41)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 675, 676, 675, 675, 675
## Resampling results across tuning parameters:
##
##   C     ROC        Sens       Spec
##   0.25  0.9106657  0.9087033  0.7263158
##   0.50  0.9160302  0.9141088  0.7263158
##   1.00  0.9239526  0.9212677  0.7578947
##
## Tuning parameter 'sigma' was held constant at a value of 0.02866623
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.02866623 and C = 1.
```

```
Test_svm <- data.frame(obs = response_test,svm = predict(SVMFIT, biodeg_test))
```

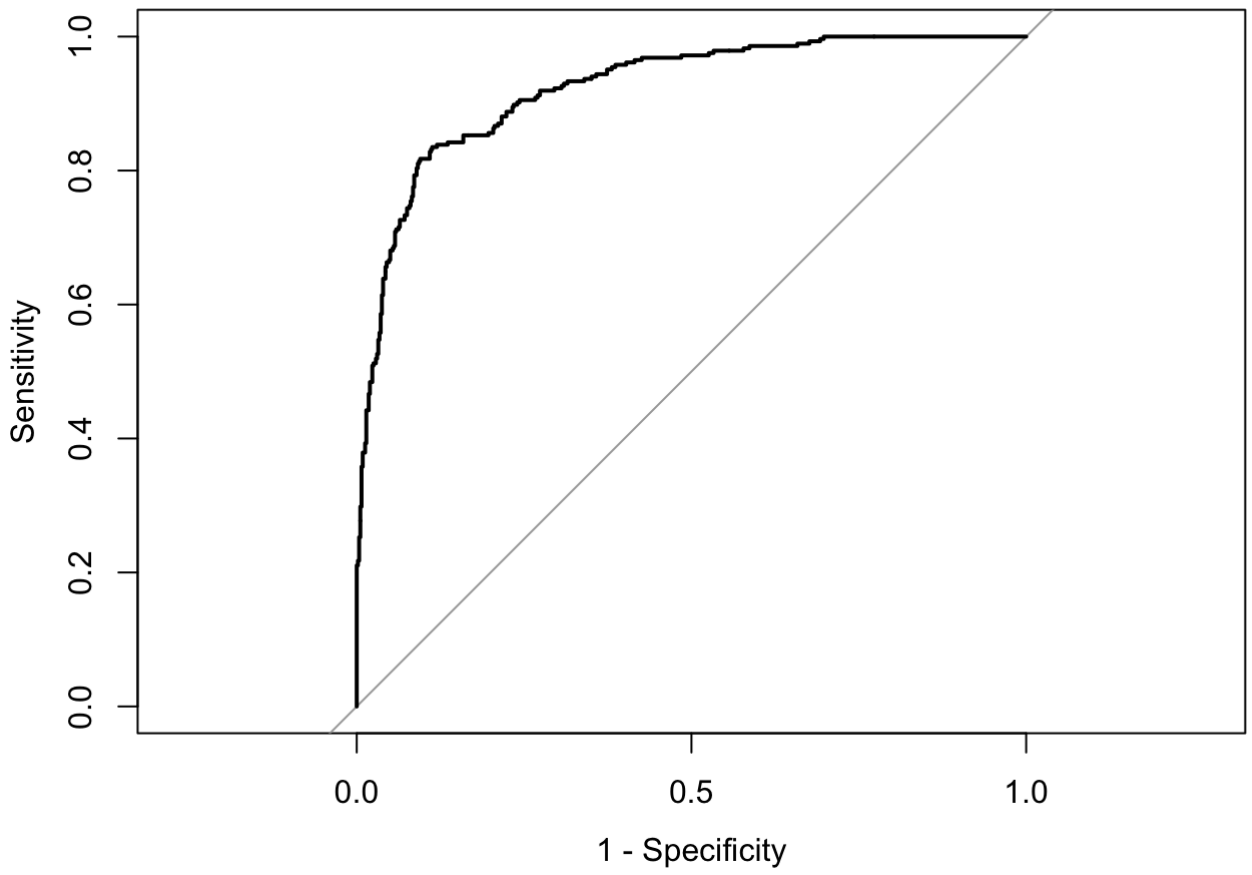
#Linear_Discriminant_Analysis

```
set.seed(100)
ctrl1 <- trainControl(method = "cv", number = 10, summaryFunction = twoClassSummary, cla
ssProbs = TRUE,   savePredictions = TRUE)
lda_fit = train(x=biodeg_train, y = response_train, method = "lda", preProc = c("center"
, "scale"), metric = "ROC", trControl = ctrl1)

lda_Pred  = predict(lda_fit,biodeg_train, type = "prob")
lda_ROC = pROC::roc(response = response_train, predictor = lda_Pred[, 1])
```

```
## Setting levels: control = NRB, case = RB
```

```
## Setting direction: controls > cases
```

```
lda_AUC = lda_ROC$auc[1]
lda = list(classifier = lda_fit, predictions = lda_Pred, roc = lda_ROC, auc = lda_AUC)
plot(lda_ROC, legacy.axes = TRUE)
```




```
Test_lda <- data.frame(obs = response_test,lda = predict(lda_fit, biodeg_test))
```

#RandomForest

```
set.seed(100)

rf_fit = randomForest(x=biodeg_train, y = response_train, preProc = c("center", "scale"
), metric = "ROC", trControl = ctrl1)

#rf_Pred  = predict(rf_fit,biodeg_test, type = "prob")
#rf_ROC = pROC::roc(response = response_train, predictor = rf_Pred[, 1])

#rf_AUC = rf_ROC$auc[1]
#rf = list(classifier = rf_fit, predictions = rf_Pred, roc = rf_ROC, auc = rf_AUC)
#plot(rf_ROC, legacy.axes = TRUE)
```

```
Test_rf <- data.frame(obs = response_test,rf = predict(rf_fit, biodeg_test))
```

```
#Models <- resamples(list(lda = lda_fit,
                         #rf = rf_fit,
                         #svm = SVMFIT))
#Summ_Mod <- summary(Models)
#Summ_Mod
```

```
confusionMatrix(Test_lda$lda, Test_lda$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 131  13
##        RB    8  58
##
##                Accuracy : 0.9
##                  95% CI : (0.8512, 0.937)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 8.985e-16
##
##                   Kappa : 0.7727
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.8169
##             Specificity : 0.9424
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 0.9097
##              Prevalence : 0.3381
##          Detection Rate : 0.2762
##    Detection Prevalence : 0.3143
##       Balanced Accuracy : 0.8797
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(Test_rf$rf, Test_rf$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 133   9
##        RB    6  62
##
##                Accuracy : 0.9286
##                  95% CI : (0.8849, 0.9595)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8387
##
##  Mcnemar's Test P-Value : 0.6056
##
##             Sensitivity : 0.8732
##             Specificity : 0.9568
##          Pos Pred Value : 0.9118
##          Neg Pred Value : 0.9366
##              Prevalence : 0.3381
##          Detection Rate : 0.2952
##    Detection Prevalence : 0.3238
##       Balanced Accuracy : 0.9150
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(Test_svm$svm, Test_svm$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 131  11
##        RB    8  60
##
##                Accuracy : 0.9095
##                  95% CI : (0.8623, 0.9446)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7957
##
##  Mcnemar's Test P-Value : 0.6464
##
##             Sensitivity : 0.8451
##             Specificity : 0.9424
##          Pos Pred Value : 0.8824
##          Neg Pred Value : 0.9225
##              Prevalence : 0.3381
##          Detection Rate : 0.2857
##    Detection Prevalence : 0.3238
##       Balanced Accuracy : 0.8938
##
##        'Positive' Class : RB
##
```

```
#writing the output
#write.csv(y_pred_test,"C:/Users/kurap/OneDrive/Documents/GitHub/predictive-modeling/dat
a/model-output/y_pred_sr.csv", row.names = FALSE)
```

Univariate Feature Selection

Ivan A Chavez

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(100)
```

```
# Loading transformed data to perform modeling
biodeg_train <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
biodeg_test <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")
response_train <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
response_test <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

response_train <- as.factor(as.matrix(response_train))
response_test <- as.factor(as.matrix(response_test))
```

Implementing Univariate Analysis to identify features of interest. Helperfunction used in control is random forest Selection by filtering(rfSBF).

```
Ctrl <- sbfControl(functions = rfSBF, method = "cv")


biodeg_features <- sbf(biodeg_train, response_train, sbfControl = Ctrl)

#biodeg_features

biodeg_features$optVariables
```

```
##  [1] "SpMax_L"     "nHM"         "F01.N.N."    "F04.C.N."    "NssssC"
##  [6] "nCb."        "C."          "nCp"         "nO"          "F03.C.N."
## [11] "SdssC"       "HyWi_B.m."   "LOC"         "SM6_L"       "Me"
## [16] "Mi"          "nArNO2"      "nCRX3"       "SpPosA_B.p." "nCIR"
## [21] "B01.C.Br."   "B03.C.Cl."   "N.073"       "SpMax_A"     "B04.C.Br."
## [26] "TI2_L"       "nCrt"        "C.026"       "F02.C.N."    "SpMax_B.m."
## [31] "Psi_i_A"     "nN"          "SM6_B.m."    "nArCOOR"     "nX"
```

Subsetting based on univariate selected features

```
biodeg_train_features <- biodeg_train[,biodeg_features$optVariables]
biodeg_test_features <- biodeg_test[,biodeg_features$optVariables]
```

Saving subsetted dataset

```
write.csv(biodeg_train_features,"../data/univariate-selection-data/biodeg_train_univaria
te.csv", row.names = FALSE)
write.csv(biodeg_test_features,"../data/univariate-selection-data/biodeg_test_univariat
e.csv", row.names = FALSE)
```

ADS503-02-SP22 - Final Project: Team 3

Carr_Aaron

06/27/2022

```
library(caret)
library(class)
library(corrplot)
library(datasets)
library(dplyr)
library(e1071)
library(Hmisc)
library(mlbench)
library(ggplot2)
library(gridExtra)
library(psych)
library(randomForest)
library(RANN)
library(rpart)
library(rpart.plot)
library(scales)
library(stats)
```

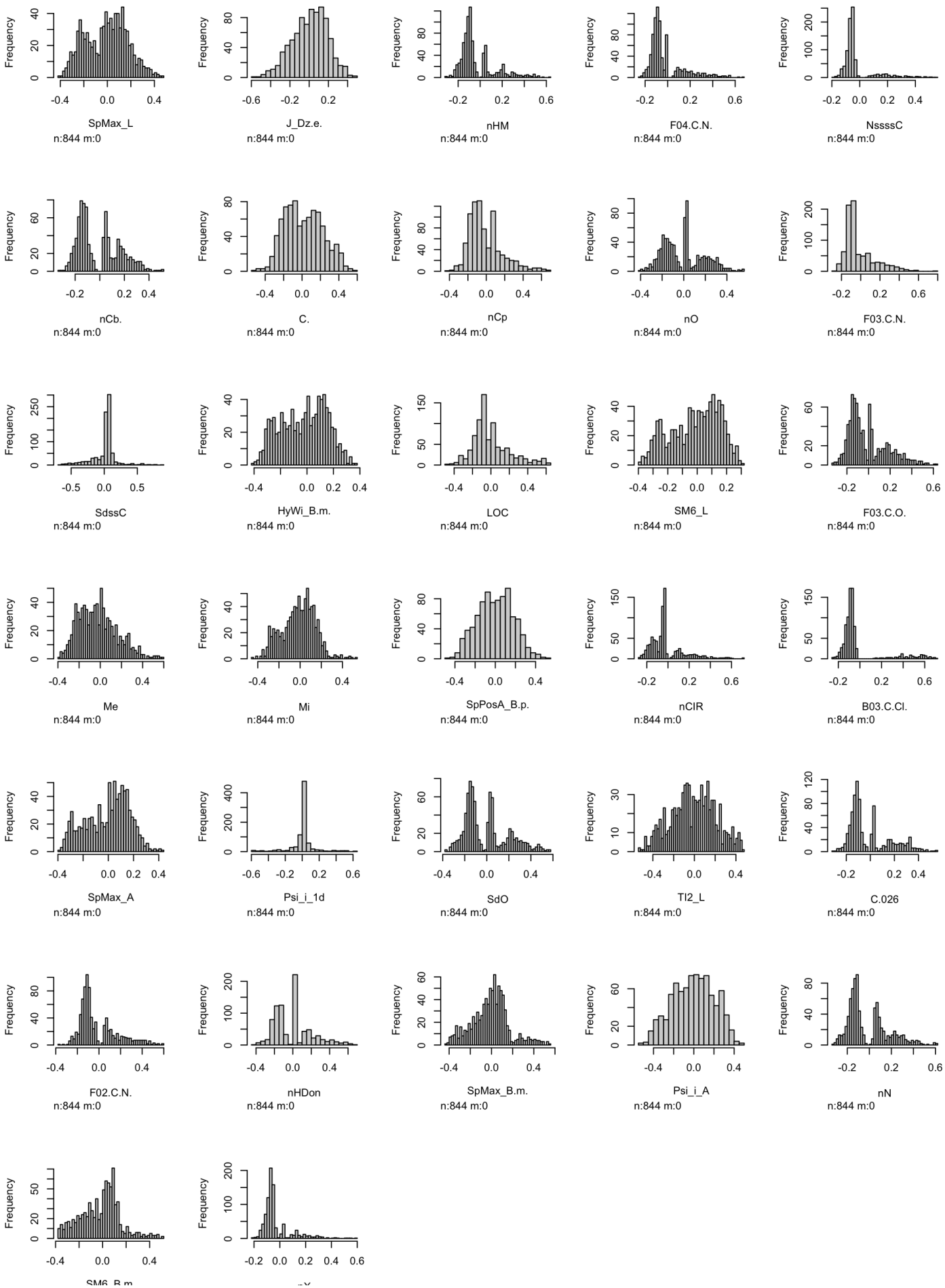
Importing Train/Test Datasets

```
train_x01_df01 <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
test_x01_df01 <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")

train_y01_df01 <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
test_y01_df01 <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

train_y01_vc01 <- train_y01_df01[["x"]]
test_y01_vc01 <- test_y01_df01[["x"]]
```

```
# Transform predictor set based on skew, centering, scaling, NZV, & outliers
set.seed(100)
x01_df01_train_fit01 <- preProcess(train_x01_df01,
                                   method = c("BoxCox",
                                              "center",
                                              "scale",
                                              "nzv",
                                              "spatialSign"))
x01_df01_train_fit01
```

```
## Created from 844 samples and 41 variables
##
## Pre-processing:
##   - Box-Cox transformation (12)
##   - centered (32)
##   - ignored (0)
##   - removed (9)
##   - scaled (32)
##   - spatial sign transformation (32)
##
## Lambda estimates for Box-Cox transformation:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.00000 -1.25000 -0.10000 -0.01667  1.40000  2.00000
```

```
# Transform training set
train_x01_trans_df01 <- predict(x01_df01_train_fit01, train_x01_df01)
hist.data.frame(train_x01_trans_df01)
```


Frequency
SpMax_L
n:844 m:0
J_Dz.e.
n:844 m:0
nHM
n:844 m:0
F04.C.N.
n:844 m:0
NssssC
n:844 m:0
nCb.
n:844 m:0
C.
n:844 m:0
nCp
n:844 m:0
nO
n:844 m:0
F03.C.N.
n:844 m:0
SdssC
n:844 m:0
HyWi_B.m.
n:844 m:0
LOC
n:844 m:0
SM6_L
n:844 m:0
F03.C.O.
n:844 m:0
Me
n:844 m:0
Mi
n:844 m:0
SpPosA_B.p.
n:844 m:0
nCIR
n:844 m:0
B03.C.Cl.
n:844 m:0
SpMax_A
n:844 m:0
Psi_i_1d
n:844 m:0
SdO
n:844 m:0
Tl2_L
n:844 m:0
C.026
n:844 m:0
F02.C.N.
n:844 m:0
nHDon
n:844 m:0
SpMax_B.m.
n:844 m:0
Psi_i_A
n:844 m:0
nN
n:844 m:0


```
# Transform test set using training set-based fit
test_x01_trans_df01 <- predict(x01_df01_train_fit01, test_x01_df01)
#_____________________________________________________________________________
# Transform predictor set based on skew, centering, scaling, & NZV
set.seed(100)
x01_df01_train_fit02 <- preProcess(train_x01_df01,
                                   method = c("BoxCox",
                                              "center",
                                              "scale",
                                              "nzv"))
x01_df01_train_fit02
```

```
## Created from 844 samples and 41 variables
##
## Pre-processing:
##   - Box-Cox transformation (12)
##   - centered (32)
##   - ignored (0)
##   - removed (9)
##   - scaled (32)
##
## Lambda estimates for Box-Cox transformation:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.00000 -1.25000 -0.10000 -0.01667  1.40000  2.00000
```

```
# Transform training set
train_x01_trans_df02 <- predict(x01_df01_train_fit02, train_x01_df01)

# Transform test set using training set-based fit
test_x01_trans_df02 <- predict(x01_df01_train_fit02, test_x01_df01)

# Perform principal components analysis (PCA) on transformed training set
set.seed(100)
# Citation:
# Larose, C. D., & Larose, D. T. (2019). Data science using Python and R. John Wiley & S
ons.
train_x01_trans_df02_cor_mtx01 <- cor(train_x01_trans_df02)
train_x01_trans_df02_cor_mtx01
```

```
##                SpMax_L     J_Dz.e.          nHM      F04.C.N.       NssssC
## SpMax_L      1.00000000  0.19304025  0.271557749  0.1911033455  0.518790798
## J_Dz.e.      0.19304025  1.00000000  0.066816208 -0.2192235269  0.236939730
## nHM          0.27155775  0.06681621  1.000000000 -0.0078140155  0.070893636
## F04.C.N.     0.19110335 -0.21922353 -0.007814016  1.0000000000  0.012145267
## NssssC       0.51879080  0.23693973  0.070893636  0.0121452674  1.000000000
## nCb.         0.40889071 -0.26767767  0.250924598  0.4030233007  0.062891968
## C.           0.32820592 -0.26528883  0.094216197  0.1403389248 -0.078520178
## nCp          0.24691778  0.11169462 -0.131723512  0.0374482783  0.477499815
## nO           0.25170684  0.10905494 -0.100468758  0.1090233996  0.045938283
## F03.C.N.     0.18287680 -0.16811486 -0.030566002  0.8392350400 -0.013801965
## SdssC       -0.11486789 -0.15031190  0.041989098 -0.1165732609 -0.124309636
## HyWi_B.m.    0.64325534 -0.01651684  0.470560648  0.2886092352  0.253300108
## LOC         -0.10381973  0.28606960 -0.097029574 -0.0904492799 -0.009027324
## SM6_L        0.89269125  0.03060851  0.234875709  0.3247553338  0.456890908
## F03.C.O.     0.40802553  0.04023202 -0.049215931  0.1072540368  0.148832402
## Me           0.30299612  0.42387003  0.386595271  0.0103617935  0.319693217
## Mi          -0.09070728  0.32513945 -0.360240309 -0.0231298820  0.434681015
## SpPosA_B.p.  0.18440101 -0.17816184  0.520374410  0.0778356919 -0.226044273
## nCIR         0.42464862 -0.53808420  0.169409767  0.3162449551  0.097596376
## B03.C.Cl.    0.14556650  0.01058745  0.537316165  0.1013025984 -0.013460403
## SpMax_A      0.90628911 -0.03738947  0.232620565  0.2801933123  0.377255345
## Psi_i_1d     0.04072547  0.02676464 -0.005815232  0.0077761292  0.009482272
## SdO          0.24503322  0.05830033 -0.063206206  0.3275445709 -0.029400880
## TI2_L        0.04301575 -0.09345301 -0.070600338  0.1866087880  0.070591641
## C.026        0.29518795 -0.12169896  0.424836827  0.3986759798 -0.037702319
## F02.C.N.     0.12171008 -0.21342791 -0.044130997  0.8075210726 -0.040851422
## nHDon        0.03557271  0.01136330 -0.141098436  0.2751227923  0.013370073
## SpMax_B.m.   0.57560353  0.15682725  0.535195414  0.1303441083  0.075251239
## Psi_i_A      0.12646537  0.43483596  0.079631075 -0.0004837395  0.172900042
## nN           0.04410029 -0.10195825 -0.047259775  0.7567209289 -0.079521460
## SM6_B.m.     0.59937319  0.10383932  0.547380073  0.1959017902  0.138663287
## nX           0.39850319  0.27929462  0.542075913 -0.0248277827  0.666265384
##                   nCb.          C.          nCp          nO     F03.C.N.
## SpMax_L      0.40889071  0.32820592  0.246917782  0.25170684  0.182876795
## J_Dz.e.     -0.26767767 -0.26528883  0.111694617  0.10905494 -0.168114856
## nHM          0.25092460  0.09421620 -0.131723512 -0.10046876 -0.030566002
## F04.C.N.     0.40302330  0.14033892  0.037448278  0.10902340  0.839235040
## NssssC       0.06289197 -0.07852018  0.477499815  0.04593828 -0.013801965
## nCb.         1.00000000  0.60352124  0.127914896  0.13522978  0.360298072
## C.           0.60352124  1.00000000 -0.219413445 -0.13798026  0.107171374
## nCp          0.12791490 -0.21941345  1.000000000  0.19887791  0.009698958
## nO           0.13522978 -0.13798026  0.198877907  1.00000000  0.116724751
## F03.C.N.     0.36029807  0.10717137  0.009698958  0.11672475  1.000000000
## SdssC       -0.12334670  0.02370898  0.019946877 -0.52769869 -0.100309651
## HyWi_B.m.    0.51126030  0.35904317  0.175965361  0.34201171  0.261220550
## LOC         -0.30592960 -0.36914245  0.245189205  0.32115129 -0.113116251
## SM6_L        0.52438665  0.39433549  0.301671217  0.38621321  0.308627171
## F03.C.O.     0.29169773  0.05443174  0.348658682  0.84280429  0.108507705
## Me           0.07614984 -0.07598553 -0.254294213  0.25218043  0.029204864
## Mi          -0.48779479 -0.72147167  0.190377255  0.09109251  0.015443857
## SpPosA_B.p.  0.43696598  0.61650380 -0.129954041 -0.35478573  0.061714717
```

```
## nCIR          0.52242025  0.58305281 -0.096547198  0.04298961  0.281990142
## B03.C.Cl.     0.29208072  0.25412415 -0.152898613 -0.10653839  0.097784814
## SpMax_A       0.52408594  0.54492326  0.116008690  0.23297848  0.275350466
## Psi_i_1d      0.01803029  0.05996245  0.002039826 -0.00830005  0.012127807
## SdO           0.23501451  0.03519397  0.011610116  0.74781246  0.313407515
## TI2_L         0.05993928 -0.12954972  0.266800546  0.41264541  0.134050137
## C.026         0.81463622  0.41829226 -0.070215070  0.10922009  0.378199983
## F02.C.N.      0.35435363  0.10094833 -0.019342979  0.07083507  0.808098558
## nHDon         0.09534061 -0.24784977  0.019685954  0.19258529  0.257867864
## SpMax_B.m.    0.40449615  0.37535655 -0.061143548  0.06842164  0.133804098
## Psi_i_A      -0.02360347 -0.19372312 -0.197170906  0.39100154  0.023018786
## nN            0.20102799 -0.09389869 -0.100040207  0.03490619  0.844066720
## SM6_B.m.      0.45583122  0.38147995  0.024113153  0.18202189  0.183882183
## nX            0.12046395  0.04517202 -0.047116317 -0.13118458 -0.042125507
##                     SdssC   HyWi_B.m.          LOC       SM6_L      F03.C.O.
## SpMax_L      -0.1148678859  0.64325534 -0.103819726  0.89269125  0.4080255308
## J_Dz.e.      -0.1503118959 -0.01651684  0.286069604  0.03060851  0.0402320163
## nHM           0.0419890984  0.47056065 -0.097029574  0.23487571 -0.0492159306
## F04.C.N.     -0.1165732609  0.28860924 -0.090449280  0.32475533  0.1072540368
## NssssC       -0.1243096365  0.25330011 -0.009027324  0.45689091  0.1488324017
## nCb.         -0.1233467013  0.51126030 -0.305929598  0.52438665  0.2916977271
## C.            0.0237089794  0.35904317 -0.369142452  0.39433549  0.0544317369
## nCp           0.0199468775  0.17596536  0.245189205  0.30167122  0.3486586821
## nO           -0.5276986911  0.34201171  0.321151291  0.38621321  0.8428042853
## F03.C.N.     -0.1003096510  0.26122055 -0.113116251  0.30862717  0.1085077046
## SdssC         1.0000000000 -0.18425919 -0.121738565 -0.17965157 -0.3970174883
## HyWi_B.m.    -0.1842591858  1.00000000  0.224558742  0.83664078  0.4597878756
## LOC          -0.1217385655  0.22455874  1.000000000  0.07354996  0.2145282955
## SM6_L        -0.1796515655  0.83664078  0.073549960  1.00000000  0.5290067061
## F03.C.O.     -0.3970174883  0.45978788  0.214528296  0.52900671  1.0000000000
## Me           -0.3476181834  0.16774265 -0.174951842  0.16466629  0.1161700036
## Mi           -0.1121252736 -0.30015351  0.273635139 -0.15477249 -0.0540152355
## SpPosA_B.p.   0.2490832009  0.41345978 -0.304248837  0.20242206 -0.1362974445
## nCIR          0.0120766257  0.38214501 -0.424363320  0.49656164  0.1920139892
## B03.C.Cl.    -0.0005386118  0.20662366 -0.131962978  0.12589940 -0.0779445216
## SpMax_A      -0.1009993988  0.69074466 -0.170267723  0.90762685  0.3976331114
## Psi_i_1d      0.0139042294  0.02166812  0.021862614  0.05034712  0.0119319732
## SdO          -0.5039136461  0.33412714  0.168111532  0.36517274  0.6844343227
## TI2_L        -0.1835125107  0.47479239  0.702889883  0.34014571  0.3476991727
## C.026        -0.0411164480  0.44583401 -0.219970948  0.37735712  0.1757290544
## F02.C.N.     -0.0660370656  0.20325429 -0.137878912  0.25500764  0.0264977420
## nHDon        -0.2161619355 -0.04652300 -0.003426084  0.04860271  0.0899887250
## SpMax_B.m.   -0.0275117351  0.69196839 -0.151672714  0.53893444  0.1855792370
## Psi_i_A      -0.3928445441 -0.12486437 -0.126912400 -0.04328680  0.1987878420
## nN           -0.0707106042  0.11036265 -0.094329308  0.14722246 -0.0002323556
## SM6_B.m.     -0.1004053716  0.87403618  0.014306232  0.66928021  0.2947954352
## nX           -0.1066105888  0.37407302 -0.073516753  0.32162158 -0.1062883833
##                        Me           Mi SpPosA_B.p.        nCIR     B03.C.Cl.
## SpMax_L       0.302996118 -0.090707283  0.18440101  0.42464862  0.1455665044
## J_Dz.e.       0.423870026  0.325139451 -0.17816184 -0.53808420  0.0105874486
## nHM           0.386595271 -0.360240309  0.52037441  0.16940977  0.5373161650
## F04.C.N.      0.010361793 -0.023129882  0.07783569  0.31624496  0.1013025984
```

```
## NssssC       0.319693217  0.434681015 -0.22604427  0.09759638 -0.0134604034
## nCb.         0.076149840 -0.487794788  0.43696598  0.52242025  0.2920807177
## C.          -0.075985531 -0.721471672  0.61650380  0.58305281  0.2541241462
## nCp         -0.254294213  0.190377255 -0.12995404 -0.09654720 -0.1528986129
## nO           0.252180431  0.091092512 -0.35478573  0.04298961 -0.1065383883
## F03.C.N.     0.029204864  0.015443857  0.06171472  0.28199014  0.0977848143
## SdssC       -0.347618183 -0.112125274  0.24908320  0.01207663 -0.0005386118
## HyWi_B.m.    0.167742646 -0.300153508  0.41345978  0.38214501  0.2066236550
## LOC         -0.174951842  0.273635139 -0.30424884 -0.42436332 -0.1319629783
## SM6_L        0.164666287 -0.154772486  0.20242206  0.49656164  0.1258994044
## F03.C.O.     0.116170004 -0.054015236 -0.13629744  0.19201399 -0.0779445216
## Me           1.000000000  0.249849943 -0.14740822 -0.03241080  0.2707813341
## Mi           0.249849943  1.000000000 -0.81289103 -0.43420295 -0.2634459671
## SpPosA_B.p. -0.147408222 -0.812891032  1.00000000  0.35639110  0.3087083375
## nCIR        -0.032410795 -0.434202947  0.35639110  1.00000000  0.1334984416
## B03.C.Cl.    0.270781334 -0.263445967  0.30870834  0.13349844  1.0000000000
## SpMax_A      0.198656724 -0.268476798  0.29381647  0.64297525  0.1804394963
## Psi_i_1d    -0.016775073 -0.011416758  0.02684666  0.01931312  0.0146560658
## SdO          0.301626857  0.004938824 -0.15085707  0.14583260 -0.0107683438
## TI2_L       -0.181019227  0.134277247 -0.21008181 -0.08330935 -0.1362485848
## C.026        0.226774384 -0.394160542  0.38212402  0.28295224  0.4419865357
## F02.C.N.     0.007290958  0.021139913  0.04974745  0.29007800  0.0921735053
## nHDon        0.071650667  0.238758716 -0.29951550 -0.02532685 -0.0743190609
## SpMax_B.m.   0.304788283 -0.416132463  0.63902780  0.30113857  0.2469003298
## Psi_i_A      0.821726553  0.315895134 -0.35972729 -0.17192030  0.1237275212
## nN           0.057244337  0.165624810 -0.03003336  0.15572919  0.0504828856
## SM6_B.m.     0.270600646 -0.400170350  0.60274619  0.32368455  0.2483667890
## nX           0.620582562  0.245458495  0.03753180  0.05520241  0.3306002588
##                 SpMax_A     Psi_i_1d          SdO       TI2_L       C.026
## SpMax_L      0.90628911  0.040725466  0.245033223  0.04301575  0.29518795
## J_Dz.e.     -0.03738947  0.026764640  0.058300334 -0.09345301 -0.12169896
## nHM          0.23262056 -0.005815232 -0.063206206 -0.07060034  0.42483683
## F04.C.N.     0.28019331  0.007776129  0.327544571  0.18660879  0.39867598
## NssssC       0.37725535  0.009482272 -0.029400880  0.07059164 -0.03770232
## nCb.         0.52408594  0.018030288  0.235014507  0.05993928  0.81463622
## C.           0.54492326  0.059962452  0.035193972 -0.12954972  0.41829226
## nCp          0.11600869  0.002039826  0.011610116  0.26680055 -0.07021507
## nO           0.23297848 -0.008300050  0.747812463  0.41264541  0.10922009
## F03.C.N.     0.27535047  0.012127807  0.313407515  0.13405014  0.37819998
## SdssC       -0.10099940  0.013904229 -0.503913646 -0.18351251 -0.04111645
## HyWi_B.m.    0.69074466  0.021668116  0.334127145  0.47479239  0.44583401
## LOC         -0.17026772  0.021862614  0.168111532  0.70288988 -0.21997095
## SM6_L        0.90762685  0.050347117  0.365172742  0.34014571  0.37735712
## F03.C.O.     0.39763311  0.011931973  0.684434323  0.34769917  0.17572905
## Me           0.19865672 -0.016775073  0.301626857 -0.18101923  0.22677438
## Mi          -0.26847680 -0.011416758  0.004938824  0.13427725 -0.39416054
## SpPosA_B.p.  0.29381647  0.026846660 -0.150857068 -0.21008181  0.38212402
## nCIR         0.64297525  0.019313119  0.145832602 -0.08330935  0.28295224
## B03.C.Cl.    0.18043950  0.014656066 -0.010768344 -0.13624858  0.44198654
## SpMax_A      1.00000000  0.048957083  0.272011622  0.03312862  0.37147552
## Psi_i_1d     0.04895708  1.000000000 -0.001798859  0.01694891  0.01455682
## SdO          0.27201162 -0.001798859  1.000000000  0.26974267  0.20544801
```

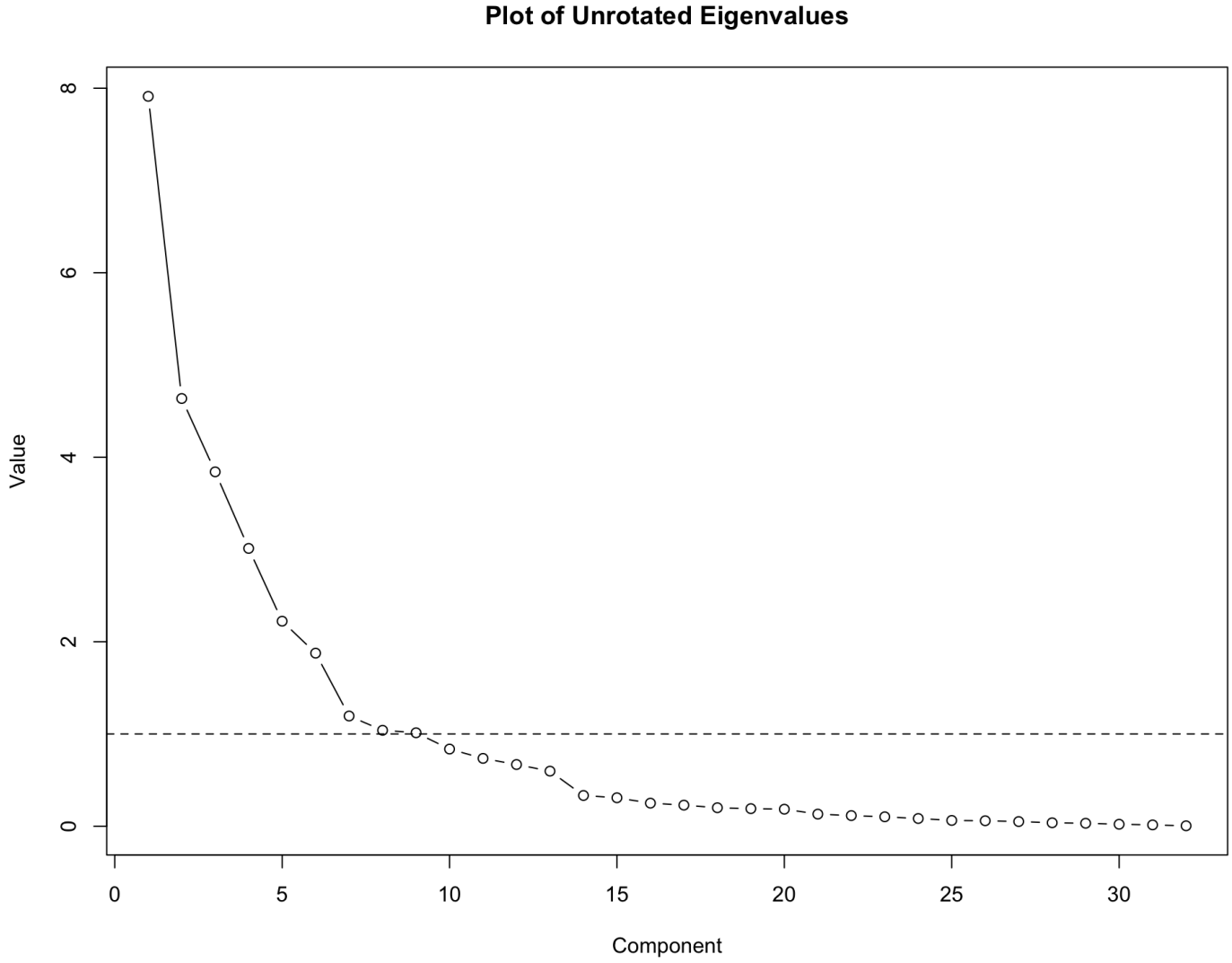
```
## TI2_L        0.03312862  0.016948913  0.269742668  1.00000000  0.02433811
## C.026        0.37147552  0.014556817  0.205448015  0.02433811  1.00000000
## F02.C.N.     0.21839545  0.018343558  0.298547894  0.12441452  0.41099679
## nHDon        0.01289193 -0.023153743  0.084228483  0.04687291  0.13299553
## SpMax_B.m.   0.56823429  0.007319998  0.164015834 -0.05889255  0.39232102
## Psi_i_A     -0.02215021 -0.043523015  0.408836414 -0.16959339  0.09669372
## nN           0.12230862  0.017306793  0.268668028  0.07571599  0.26861622
## SM6_B.m.     0.61581598  0.013443957  0.248573356  0.17990146  0.43225005
## nX           0.31034294  0.005701087 -0.095455973 -0.03402839  0.23875479
##                 F02.C.N.        nHDon   SpMax_B.m.       Psi_i_A            nN
## SpMax_L      0.121710082  0.035572706  0.575603530  0.1264653724  0.0441002912
## J_Dz.e.     -0.213427915  0.011363297  0.156827253  0.4348359592 -0.1019582530
## nHM         -0.044130997 -0.141098436  0.535195414  0.0796310748 -0.0472597755
## F04.C.N.     0.807521073  0.275122792  0.130344108 -0.0004837395  0.7567209289
## NssssC      -0.040851422  0.013370073  0.075251239  0.1729000420 -0.0795214599
## nCb.         0.354353630  0.095340612  0.404496150 -0.0236034715  0.2010279885
## C.           0.100948330 -0.247849775  0.375356547 -0.1937231163 -0.0938986933
## nCp         -0.019342979  0.019685954 -0.061143548 -0.1971709059 -0.1000402072
## nO           0.070835071  0.192585288  0.068421640  0.3910015400  0.0349061895
## F03.C.N.     0.808098558  0.257867864  0.133804098  0.0230187855  0.8440667202
## SdssC       -0.066037066 -0.216161935 -0.027511735 -0.3928445441 -0.0707106042
## HyWi_B.m.    0.203254292 -0.046523000  0.691968394 -0.1248643702  0.1103626530
## LOC         -0.137878912 -0.003426084 -0.151672714 -0.1269123999 -0.0943293080
## SM6_L        0.255007639  0.048602708  0.538934443 -0.0432867987  0.1472224569
## F03.C.O.     0.026497742  0.089988725  0.185579237  0.1987878420 -0.0002323556
## Me           0.007290958  0.071650667  0.304788283  0.8217265526  0.0572443367
## Mi           0.021139913  0.238758716 -0.416132463  0.3158951341  0.1656248103
## SpPosA_B.p.  0.049747453 -0.299515501  0.639027801 -0.3597272934 -0.0300333567
## nCIR         0.290078001 -0.025326847  0.301138575 -0.1719203005  0.1557291943
## B03.C.Cl.    0.092173505 -0.074319061  0.246900330  0.1237275212  0.0504828856
## SpMax_A      0.218395453  0.012891933  0.568234292 -0.0221502064  0.1223086183
## Psi_i_1d     0.018343558 -0.023153743  0.007319998 -0.0435230147  0.0173067926
## SdO          0.298547894  0.084228483  0.164015834  0.4088364140  0.2686680285
## TI2_L        0.124414521  0.046872908 -0.058892554 -0.1695933925  0.0757159948
## C.026        0.410996786  0.132995529  0.392321020  0.0966937248  0.2686162168
## F02.C.N.     1.000000000  0.288365514  0.074421162  0.0040636076  0.7945650464
## nHDon        0.288365514  1.000000000 -0.106630652  0.2554699378  0.4536056733
## SpMax_B.m.   0.074421162 -0.106630652  1.000000000  0.0687595763  0.0424247973
## Psi_i_A      0.004063608  0.255469938  0.068759576  1.0000000000  0.0798558057
## nN           0.794565046  0.453605673  0.042424797  0.0798558057  1.0000000000
## SM6_B.m.     0.125122209 -0.095141179  0.943321747 -0.0055056648  0.0671381860
## nX          -0.052342602 -0.112837445  0.308013431  0.2943553375 -0.0878078447
##                 SM6_B.m.           nX
## SpMax_L      0.599373188  0.398503189
## J_Dz.e.      0.103839316  0.279294620
## nHM          0.547380073  0.542075913
## F04.C.N.     0.195901790 -0.024827783
## NssssC       0.138663287  0.666265384
## nCb.         0.455831222  0.120463954
## C.           0.381479953  0.045172025
## nCp          0.024113153 -0.047116317
## nO           0.182021894 -0.131184583
```

```
## F03.C.N.     0.183882183 -0.042125507
## SdssC       -0.100405372 -0.106610589
## HyWi_B.m.    0.874036182  0.374073025
## LOC          0.014306232 -0.073516753
## SM6_L        0.669280211  0.321621577
## F03.C.O.     0.294795435 -0.106288383
## Me           0.270600646  0.620582562
## Mi          -0.400170350  0.245458495
## SpPosA_B.p.  0.602746187  0.037531804
## nCIR         0.323684547  0.055202408
## B03.C.Cl.    0.248366789  0.330600259
## SpMax_A      0.615815980  0.310342937
## Psi_i_1d     0.013443957  0.005701087
## SdO          0.248573356 -0.095455973
## TI2_L        0.179901463 -0.034028390
## C.026        0.432250050  0.238754787
## F02.C.N.     0.125122209 -0.052342602
## nHDon       -0.095141179 -0.112837445
## SpMax_B.m.   0.943321747  0.308013431
## Psi_i_A     -0.005505665  0.294355338
## nN           0.067138186 -0.087807845
## SM6_B.m.     1.000000000  0.355347271
## nX           0.355347271  1.000000000
```

```
train_x01_trans_df02_pca01 <- principal(r = train_x01_trans_df02_cor_mtx01, rotate = "no
ne", nfactors = 32)
train_x01_trans_df02_pca01
```

```
## Mean item complexity =  4.1
## Test of the hypothesis that 32 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0
##
## Fit based upon off diagonal values = 1
```

```
plot(train_x01_trans_df02_pca01$values, type = "b", main = "Plot of Unrotated Eigenvalue
s", ylab = "Value", xlab = "Component")
abline(h = 1, lty = 2)
```




```
train_x01_trans_df02_pca01$loadings
```

```
##
## Loadings:
##             PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9
## SpMax_L      0.758  0.183  0.297         0.269 -0.267 -0.189
## J_Dz.e.             0.388  0.517  0.238         0.223 -0.354  0.291  0.193
## nHM          0.470 -0.281  0.410  0.275         0.417  0.188
## F04.C.N.     0.479  0.174 -0.685  0.213  0.189  0.129
## NssssC       0.241  0.373  0.414         0.623 -0.296  0.179
## nCb.         0.748 -0.176 -0.209               -0.130  0.348  0.263  0.105
## C.           0.564 -0.534        -0.126 -0.169 -0.291        -0.129
## nCp                 0.319        -0.456  0.447         0.173  0.510  0.108
## nO           0.295  0.713        -0.278 -0.464
## F03.C.N.     0.457  0.188 -0.694  0.258  0.186  0.133 -0.113
## SdssC       -0.198 -0.533                0.394        -0.116  0.144
## HyWi_B.m.    0.850  0.108  0.166 -0.289         0.285        -0.111
## LOC         -0.166  0.471  0.108 -0.483         0.576        -0.123
## SM6_L        0.852  0.239  0.115 -0.265  0.243 -0.121
## F03.C.O.     0.442  0.547        -0.420 -0.339 -0.186         0.141
## Me           0.292  0.362  0.442  0.664 -0.202               -0.118
## Mi          -0.427  0.683         0.255  0.418               -0.107
## SpPosA_B.p.  0.496 -0.738                      0.219 -0.167  0.120
## nCIR         0.609 -0.291 -0.226               -0.460        -0.280 -0.131
## B03.C.Cl.    0.354 -0.248  0.150  0.362 -0.106  0.249  0.423         0.112
## SpMax_A      0.840         0.117 -0.120  0.189 -0.323 -0.146
## Psi_i_1d                                                    -0.264  0.931
## SdO          0.412  0.572 -0.176        -0.494               -0.106
## TI2_L        0.152  0.444 -0.128 -0.535         0.443  0.252 -0.253
## C.026        0.668 -0.147 -0.167  0.238 -0.148  0.151  0.383  0.268  0.130
## F02.C.N.     0.407  0.147 -0.721  0.281  0.183  0.121
## nHDon               0.375 -0.358  0.253                      0.341
## SpMax_B.m.   0.735 -0.193  0.313               0.215 -0.392  0.101
## Psi_i_A             0.514  0.251  0.611 -0.378 -0.151
## nN           0.281  0.232 -0.714  0.387  0.182  0.197 -0.192
## SM6_B.m.     0.816         0.280               0.305 -0.260
## nX           0.350         0.573  0.443  0.366         0.248 -0.241
##             PC10   PC11   PC12   PC13   PC14   PC15   PC16   PC17   PC18
## SpMax_L                    0.213                      -0.146
## J_Dz.e.      0.188 -0.285  0.144        -0.131  0.104                0.129
## nHM         -0.114  0.341               -0.248  0.152
## F04.C.N.     0.149                             0.119               -0.138
## NssssC                    -0.164                      0.162
## nCb.                -0.173 -0.145  0.155                             0.139
## C.           0.115 -0.400                0.100  0.112
## nCp          0.214  0.172 -0.163 -0.146  0.100
## nO                  0.164
## F03.C.N.     0.180                                          -0.226
## SdssC               0.281  0.331  0.493         0.101
## HyWi_B.m.
## LOC                 -0.160  0.188                              0.123
## SM6_L                      0.147
## F03.C.O.            0.213                            0.158 -0.184
## Me                               0.132
```

```
## Mi                                                  -0.218
## SpPosA_B.p.              -0.149 -0.101          0.110  0.105
## nCIR        -0.123  0.194                        0.139                 0.275
## B03.C.Cl.    0.106          0.440 -0.337  0.226 -0.107
## SpMax_A                     0.235
## Psi_i_1d    -0.140  0.141 -0.109
## SdO          0.232  0.109                              0.193  0.234
## TI2_L               -0.137         0.148  0.131  0.119 -0.108
## C.026               -0.123         0.283 -0.145 -0.143
## F02.C.N.     0.137                                             0.211
## nHDon       -0.703                                       0.118
## SpMax_B.m.               -0.135         0.180 -0.132
## Psi_i_A                         0.134  0.174  0.204
## nN                                                   0.118
## SM6_B.m.                 -0.147         0.160
## nX                       -0.124                       0.146
##             PC19   PC20   PC21   PC22   PC23   PC24   PC25   PC26   PC27
## SpMax_L
## J_Dz.e.                   0.137
## nHM
## F04.C.N.            0.292  0.108
## NssssC                          -0.134        -0.118
## nCb.        -0.110                                   0.132
## C.
## nCp
## nO           0.154                              -0.101
## F03.C.N.                 -0.151
## SdssC
## HyWi_B.m.                       0.112
## LOC                 0.123 -0.159         0.111
## SM6_L
## F03.C.O.                              0.146
## Me                              0.164
## Mi
## SpPosA_B.p.                                                  0.119  0.101
## nCIR
## B03.C.Cl.
## SpMax_A
## Psi_i_1d
## SdO         -0.169
## TI2_L               -0.134  0.129
## C.026                                                -0.113
## F02.C.N.     0.270 -0.121
## nHDon
## SpMax_B.m.
## Psi_i_A
## nN          -0.111 -0.143
## SM6_B.m.
## nX
##             PC28   PC29   PC30   PC31   PC32
## SpMax_L
## J_Dz.e.
```

```
## nHM
## F04.C.N.
## NssssC
## nCb.
## C.
## nCp
## nO
## F03.C.N.
## SdssC
## HyWi_B.m.
## LOC
## SM6_L
## F03.C.O.
## Me
## Mi                   0.107
## SpPosA_B.p.
## nCIR
## B03.C.Cl.
## SpMax_A
## Psi_i_1d
## SdO
## TI2_L
## C.026
## F02.C.N.
## nHDon
## SpMax_B.m.
## Psi_i_A
## nN
## SM6_B.m.
## nX
##
##                  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10
## SS loadings    7.912 4.637 3.841 3.012 2.223 1.876 1.195 1.040 1.013 0.837
## Proportion Var 0.247 0.145 0.120 0.094 0.069 0.059 0.037 0.032 0.032 0.026
## Cumulative Var 0.247 0.392 0.512 0.606 0.676 0.734 0.772 0.804 0.836 0.862
##                 PC11  PC12  PC13  PC14  PC15  PC16  PC17  PC18  PC19  PC20
## SS loadings    0.736 0.669 0.597 0.333 0.309 0.250 0.229 0.200 0.191 0.185
## Proportion Var 0.023 0.021 0.019 0.010 0.010 0.008 0.007 0.006 0.006 0.006
## Cumulative Var 0.885 0.906 0.925 0.935 0.945 0.952 0.960 0.966 0.972 0.978
##                 PC21  PC22  PC23  PC24  PC25  PC26  PC27  PC28  PC29  PC30
## SS loadings    0.131 0.115 0.102 0.084 0.063 0.059 0.051 0.038 0.032 0.021
## Proportion Var 0.004 0.004 0.003 0.003 0.002 0.002 0.002 0.001 0.001 0.001
## Cumulative Var 0.982 0.985 0.989 0.991 0.993 0.995 0.997 0.998 0.999 0.999
##                 PC31  PC32
## SS loadings    0.014 0.005
## Proportion Var 0.000 0.000
## Cumulative Var 1.000 1.000
```

```
train_x01_trans_df02_high_cor_cols <- findCorrelation(train_x01_trans_df02_cor_mtx01)
head(train_x01_trans_df02[,train_x01_trans_df02_high_cor_cols], 1)
```

```
##       SM6_L   SpMax_A  SM6_B.m.
## 1 -1.085451 -1.316236 -1.329006
```

```
write.csv(train_x01_trans_df01,
          "../data/biodeg_train_trans.csv", row.names = FALSE)
write.csv(test_x01_trans_df01,
          "../data/biodeg_test_trans.csv", row.names = FALSE)

write.csv(train_x01_trans_df02,
          "../data/biodeg_train_trans_wo_spasign.csv", row.names = FALSE)
write.csv(test_x01_trans_df02,
          "../data/biodeg_test_trans_wo_spasign.csv", row.names = FALSE)
```

K Nearest Neighbor Model

Ivan A Chavez

```
# Loading packages and setting seed
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(100)
```

```
# Loading transformed data to perform modeling
biodeg_train <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
biodeg_test <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")
response_train <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
response_test <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

biodeg_train_uni <- read.csv("../data/univariate-selection-data/biodeg_train_univariate.
csv", header = TRUE, sep = ",")
biodeg_test_uni <- read.csv("../data/univariate-selection-data/biodeg_test_univariate.cs
v", header = TRUE, sep = ",")

response_train <- as.factor(as.matrix(response_train))
response_test <- as.factor(as.matrix(response_test))
```

```
ctrl <- trainControl(method = "cv",
                     number = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

knnFit_pre <- train(x = biodeg_train, y = response_train,
                  method = "knn",
                  tuneGrid = data.frame(k = 1:20),
                  preProcess = c("nzv", "center", "scale"),
                  metric = "ROC",
                  trControl = ctrl)

knnFit_uni <- train(x = biodeg_train_uni, y = response_train,
                 method = "knn",
                 tuneGrid = data.frame(k = 1:20),
                 preProcess = c("center", "scale"),
                 metric = "ROC",
                 trControl = ctrl)

knnFit_uni_nzv <- train(x = biodeg_train_uni, y = response_train,
                    method = "knn",
                    tuneGrid = data.frame(k = 1:20),
                    preProcess = c("nzv","center", "scale"),
                    metric = "ROC",
                    trControl = ctrl)



knnFit_pre
```

```
## k-Nearest Neighbors
##
## 844 samples
##  41 predictor
##   2 classes: 'NRB', 'RB'
##
## Pre-processing: centered (32), scaled (32), remove (9)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 675, 676, 675, 675, 675
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.8019623  0.8390122  0.7649123
##    2  0.8535704  0.8121139  0.7684211
##    3  0.8755461  0.8461390  0.7894737
##    4  0.8868445  0.8443694  0.7789474
##    5  0.8881915  0.8585746  0.7789474
##    6  0.8902785  0.8639961  0.7824561
##    7  0.8873594  0.8675354  0.7789474
##    8  0.8870479  0.8496300  0.7754386
##    9  0.8843763  0.8549871  0.7824561
##   10  0.8855507  0.8443050  0.7964912
##   11  0.8850348  0.8424871  0.7789474
##   12  0.8856353  0.8460746  0.7894737
##   13  0.8863486  0.8371139  0.7894737
##   14  0.8847315  0.8389157  0.7824561
##   15  0.8855387  0.8334942  0.7894737
##   16  0.8855840  0.8263514  0.7894737
##   17  0.8852895  0.8227638  0.7859649
##   18  0.8837676  0.8156692  0.7754386
##   19  0.8834203  0.8156692  0.7894737
##   20  0.8838295  0.8156853  0.7964912
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 6.
```

```
knnFit_uni
```

```
## k-Nearest Neighbors
##
## 844 samples
##  35 predictor
##   2 classes: 'NRB', 'RB'
##
## Pre-processing: centered (35), scaled (35)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 675, 675, 676, 675, 675
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.8037158  0.8425193  0.7649123
##    2  0.8460231  0.8300193  0.7614035
##    3  0.8824638  0.8568533  0.8105263
##    4  0.8824893  0.8479086  0.8105263
##    5  0.8885841  0.8604408  0.8070175
##    6  0.8966899  0.8622265  0.8035088
##    7  0.8987579  0.8622748  0.8035088
##    8  0.8991392  0.8568855  0.8105263
##    9  0.8986478  0.8676319  0.8070175
##   10  0.8963477  0.8640444  0.7859649
##   11  0.8974624  0.8676480  0.8000000
##   12  0.8961964  0.8658623  0.8000000
##   13  0.8989247  0.8712194  0.7964912
##   14  0.8994498  0.8676641  0.8105263
##   15  0.8977239  0.8604569  0.8105263
##   16  0.8967982  0.8568855  0.8000000
##   17  0.8981800  0.8497426  0.8105263
##   18  0.8997154  0.8461390  0.8245614
##   19  0.8986015  0.8443533  0.8140351
##   20  0.8978852  0.8443211  0.8105263
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 18.
```

```
#knnFit_uni_nzv
```

```
test_preds <- data.frame(obs = response_test,
                         M2_KNN = predict(knnFit_pre, biodeg_test))

test_preds$M2_KNN_uni <- predict(knnFit_uni, biodeg_test)
test_preds$M2_KNN_uni_nvz <- predict(knnFit_uni_nzv, biodeg_test)



models_compare <- resamples(list(M2_KNN = knnFit_pre, M2_KNN_uni = knnFit_uni, M2_KNN_un
i_nvz = knnFit_uni_nzv))

models_compare_summ <- summary(models_compare)
models_compare_summ
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: M2_KNN, M2_KNN_uni, M2_KNN_uni_nvz
## Number of resamples: 5
##
## ROC
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M2_KNN         0.8366524 0.8463346 0.9136122 0.8902785 0.9190163 0.9357769    0
## M2_KNN_uni     0.8760182 0.8789944 0.9020073 0.8997154 0.9181548 0.9234023    0
## M2_KNN_uni_nvz 0.8810307 0.8878446 0.8947368 0.9026014 0.9189380 0.9304568    0
##
## Sens
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M2_KNN         0.8303571 0.8378378 0.8660714 0.8639961 0.8839286 0.9017857    0
## M2_KNN_uni     0.8125000 0.8214286 0.8378378 0.8461390 0.8482143 0.9107143    0
## M2_KNN_uni_nvz 0.8125000 0.8214286 0.8468468 0.8479408 0.8571429 0.9017857    0
##
## Spec
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M2_KNN         0.6140351 0.7368421 0.7894737 0.7824561 0.8596491 0.9122807    0
## M2_KNN_uni     0.7719298 0.8070175 0.8245614 0.8245614 0.8596491 0.8596491    0
## M2_KNN_uni_nvz 0.7543860 0.7719298 0.7894737 0.8140351 0.8771930 0.8771930    0
```

```
confusionMatrix(test_preds$M2_KNN, test_preds$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 129  16
##        RB   10  55
##
##                Accuracy : 0.8762
##                  95% CI : (0.8239, 0.9175)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 9.761e-13
##
##                   Kappa : 0.7175
##
##  Mcnemar's Test P-Value : 0.3268
##
##             Sensitivity : 0.7746
##             Specificity : 0.9281
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8897
##              Prevalence : 0.3381
##          Detection Rate : 0.2619
##    Detection Prevalence : 0.3095
##       Balanced Accuracy : 0.8514
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(test_preds$M2_KNN_uni, test_preds$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 128  10
##        RB   11  61
##
##                Accuracy : 0.9
##                  95% CI : (0.8512, 0.937)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 8.985e-16
##
##                   Kappa : 0.7773
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8592
##             Specificity : 0.9209
##          Pos Pred Value : 0.8472
##          Neg Pred Value : 0.9275
##              Prevalence : 0.3381
##          Detection Rate : 0.2905
##    Detection Prevalence : 0.3429
##       Balanced Accuracy : 0.8900
##
##        'Positive' Class : RB
##
```

```
write.csv(test_preds[,-1],"../data/model-output/y_pred_knn.csv", row.names = FALSE)
write.csv(data.frame(models_compare_summ$models, models_compare_summ$statistics),
          "../data/resampling-results/resamp_results_knn.csv", row.names = FALSE)
```

Nearest Shrunken Centroid Model

Ivan A Chavez

```
# Loading packages and setting seed
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(100)
```

```
# Loading transformed data to perform modeling
biodeg_train <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
biodeg_test <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")
response_train <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
response_test <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

response_train <- as.factor(as.matrix(response_train))
response_test <- as.factor(as.matrix(response_test))
```

```
ctrl <- trainControl(method = "cv",
                     number = 5,
                      summaryFunction = twoClassSummary,
                      classProbs = TRUE,
                      savePredictions = TRUE)

ctrl_10 <- trainControl(method = "cv",
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE,
                    savePredictions = TRUE)

nscFit_noPre <- train(x = biodeg_train,
                y = response_train,
                method = "pam",
                tuneGrid = data.frame(threshold = seq(0, 25, length = 30)),
                metric = "ROC",
                trControl = ctrl)
```

```
## 111111
```

```
nscFit_cs <- train(x = biodeg_train,
                y = response_train,
                method = "pam",
                tuneGrid = data.frame(threshold = seq(0, 25, length = 30)),
                preProcess = c('center','scale'),
                metric = "ROC",
                trControl = ctrl)
```

```
## 111111
```

```
nscFit_10_noPre <- train(x = biodeg_train,
                      y = response_train,
                      method = "pam",
                      tuneGrid = data.frame(threshold = seq(0, 25, length = 30)),
                      metric = "ROC",
                      trControl = ctrl_10)
```

```
## 11111111111
```

```
nscFit_10_cs <- train(x = biodeg_train,
                   y = response_train,
                   method = "pam",
                   tuneGrid = data.frame(threshold = seq(0, 25, length = 30)),
                   preProcess = c('center','scale'),
                   metric = "ROC",
                   trControl = ctrl_10)
```

```
## 11111111111
```

```
nscFit_10_cs
```

```
## Nearest Shrunken Centroids
##
## 844 samples
##  41 predictor
##   2 classes: 'NRB', 'RB'
##
## Pre-processing: centered (41), scaled (41)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 759, 760, 759, 759, 759, 761, ...
## Resampling results across tuning parameters:
##
##   threshold  ROC        Sens       Spec
##    0.000000  0.8620015  0.8443831  0.656034483
##    0.862069  0.8545862  0.8586688  0.589285714
##    1.724138  0.8449994  0.8873377  0.511699507
##    2.586207  0.8330481  0.9302597  0.314778325
##    3.448276  0.8158228  0.9910714  0.077093596
##    4.310345  0.7888923  0.9982143  0.003448276
##    5.172414  0.7531681  1.0000000  0.000000000
##    6.034483  0.5000000  1.0000000  0.000000000
##    6.896552  0.5000000  1.0000000  0.000000000
##    7.758621  0.5000000  1.0000000  0.000000000
##    8.620690  0.5000000  1.0000000  0.000000000
##    9.482759  0.5000000  1.0000000  0.000000000
##   10.344828  0.5000000  1.0000000  0.000000000
##   11.206897  0.5000000  1.0000000  0.000000000
##   12.068966  0.5000000  1.0000000  0.000000000
##   12.931034  0.5000000  1.0000000  0.000000000
##   13.793103  0.5000000  1.0000000  0.000000000
##   14.655172  0.5000000  1.0000000  0.000000000
##   15.517241  0.5000000  1.0000000  0.000000000
##   16.379310  0.5000000  1.0000000  0.000000000
##   17.241379  0.5000000  1.0000000  0.000000000
##   18.103448  0.5000000  1.0000000  0.000000000
##   18.965517  0.5000000  1.0000000  0.000000000
##   19.827586  0.5000000  1.0000000  0.000000000
##   20.689655  0.5000000  1.0000000  0.000000000
##   21.551724  0.5000000  1.0000000  0.000000000
##   22.413793  0.5000000  1.0000000  0.000000000
##   23.275862  0.5000000  1.0000000  0.000000000
##   24.137931  0.5000000  1.0000000  0.000000000
##   25.000000  0.5000000  1.0000000  0.000000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was threshold = 0.
```

```
#nscFit_5
```

```
test_preds <- data.frame(obs = response_test,
                         M8_NSC_no_pre = predict(nscFit_noPre, biodeg_test))

test_preds$M8_NSC_CS <- predict(nscFit_cs, biodeg_test)
test_preds$M8_NSC_10_no_pre <- predict(nscFit_10_noPre, biodeg_test)
test_preds$M8_10_NSC_CS <- predict(nscFit_10_cs, biodeg_test)



models_compare <- resamples(list(M8_NSC_no_pre = nscFit_noPre, M8_NSC_CS = nscFit_cs))

models_compare_summ <- summary(models_compare)
models_compare_summ
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: M8_NSC_no_pre, M8_NSC_CS
## Number of resamples: 5
##
## ROC
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M8_NSC_no_pre 0.7905801 0.8184524 0.8715539 0.8551398 0.8768797 0.9182331    0
## M8_NSC_CS     0.8320802 0.8425752 0.8562030 0.8596624 0.8828321 0.8846215    0
##
## Sens
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M8_NSC_no_pre 0.7946429 0.8018018 0.8035714 0.8157175 0.8303571 0.8482143    0
## M8_NSC_CS     0.8125000 0.8198198 0.8392857 0.8371782 0.8482143 0.8660714    0
##
## Spec
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M8_NSC_no_pre 0.5087719 0.7017544 0.7017544 0.6982456 0.7719298 0.8070175    0
## M8_NSC_CS     0.5964912 0.5964912 0.6315789 0.6561404 0.7017544 0.7543860    0
```

```
models_compare_10 <- resamples(list(M8_NSC_10_no_pre = nscFit_10_noPre, M8_10_NSC_CS = n
scFit_10_cs))

models_compare_summ_10 <- summary(models_compare_10)
models_compare_summ_10
```

```
##
## Call:
## summary.resamples(object = models_compare_10)
##
## Models: M8_NSC_10_no_pre, M8_10_NSC_CS
## Number of resamples: 10
##
## ROC
##                       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## M8_NSC_10_no_pre 0.815051 0.8528380 0.8674019 0.8595887 0.8739364 0.8799261
## M8_10_NSC_CS     0.812500 0.8309399 0.8628751 0.8620015 0.8945285 0.9056122
##                  NA's
## M8_NSC_10_no_pre    0
## M8_10_NSC_CS        0
##
## Sens
##                       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## M8_NSC_10_no_pre 0.7142857 0.7723214 0.8214286 0.8087338 0.8392857 0.8909091
## M8_10_NSC_CS     0.7142857 0.8258929 0.8558442 0.8443831 0.8750000 0.9107143
##                  NA's
## M8_NSC_10_no_pre    0
## M8_10_NSC_CS        0
##
## Spec
##                       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## M8_NSC_10_no_pre 0.5862069 0.6428571 0.6964286 0.6910099 0.7216749 0.8275862
## M8_10_NSC_CS     0.5172414 0.6105296 0.6785714 0.6560345 0.7155172 0.7586207
##                  NA's
## M8_NSC_10_no_pre    0
## M8_10_NSC_CS        0
```

```
confusionMatrix(test_preds$M8_10_NSC_CS, test_preds$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 123  23
##        RB   16  48
##
##                Accuracy : 0.8143
##                  95% CI : (0.755, 0.8645)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 7.191e-07
##
##                   Kappa : 0.5748
##
##  Mcnemar's Test P-Value : 0.3367
##
##             Sensitivity : 0.6761
##             Specificity : 0.8849
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.8425
##              Prevalence : 0.3381
##          Detection Rate : 0.2286
##    Detection Prevalence : 0.3048
##       Balanced Accuracy : 0.7805
##
##        'Positive' Class : RB
##
```

```
write.csv(data.frame(models_compare_summ$models, models_compare_summ$statistics),
          "../data/resampling-results/resamp_results_nsc_5.csv", row.names = FALSE)

write.csv(data.frame(models_compare_summ_10$models, models_compare_summ_10$statistics),
          "../data/resampling-results/resamp_results_nsc_10.csv", row.names = FALSE)

write.csv(test_preds[,-1],"../data/model-output/y_pred_nsc.csv", row.names = FALSE)
```

Partial Least Squares Discriminate Analysis

Ivan A Chavez

```
# Loading packages and setting seed
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(100)
```

```
# Loading transformed data to perform modeling
biodeg_train <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
biodeg_test <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")
response_train <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
response_test <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

response_train <- as.factor(as.matrix(response_train))
response_test <- as.factor(as.matrix(response_test))
```

```
ctrl <- trainControl(method = "cv",
                     number = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

#tune ncomp = 16
plsdaFit <- train(x = biodeg_train,
                  y = response_train,
                  method = "pls",
                  tuneGrid = expand.grid(ncomp = 1:41),
                  preProcess = c("center", "scale"),
                  metric = "ROC",
                  trControl = ctrl)


plsdaFit
```

```
## Partial Least Squares
##
## 844 samples
##  41 predictor
##   2 classes: 'NRB', 'RB'
##
## Pre-processing: centered (41), scaled (41)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 675, 676, 675, 675, 675
## Resampling results across tuning parameters:
##
##   ncomp  ROC        Sens       Spec
##    1     0.8586627  0.8819015  0.5684211
##    2     0.8832266  0.8818694  0.6561404
##    3     0.8949248  0.9015444  0.6701754
##    4     0.8929492  0.8997587  0.6771930
##    5     0.8991513  0.9105051  0.7122807
##    6     0.9021284  0.8997104  0.7228070
##    7     0.9004349  0.8997265  0.7228070
##    8     0.9016358  0.9086873  0.7228070
##    9     0.9019136  0.9086873  0.7333333
##   10     0.9032076  0.9051158  0.7298246
##   11     0.9035268  0.9104891  0.7368421
##   12     0.9050758  0.9033301  0.7403509
##   13     0.9054833  0.9051319  0.7473684
##   14     0.9054209  0.9051319  0.7578947
##   15     0.9060179  0.9015605  0.7473684
##   16     0.9077454  0.8979730  0.7473684
##   17     0.9071477  0.8979569  0.7438596
##   18     0.9069614  0.8979730  0.7403509
##   19     0.9067723  0.8997748  0.7473684
##   20     0.9069275  0.8997748  0.7508772
##   21     0.9067040  0.8997748  0.7543860
##   22     0.9068931  0.9015766  0.7543860
##   23     0.9062939  0.9015766  0.7578947
##   24     0.9062022  0.9015766  0.7578947
##   25     0.9064212  0.8997909  0.7578947
##   26     0.9063258  0.8979891  0.7543860
##   27     0.9067982  0.8997909  0.7508772
##   28     0.9068931  0.9015766  0.7578947
##   29     0.9071446  0.8997748  0.7543860
##   30     0.9070842  0.9015605  0.7508772
##   31     0.9066425  0.9051641  0.7543860
##   32     0.9060757  0.9033623  0.7543860
##   33     0.9061669  0.9033623  0.7543860
##   34     0.9061985  0.9015766  0.7543860
##   35     0.9060410  0.9033623  0.7543860
##   36     0.9061042  0.9033623  0.7543860
##   37     0.9061669  0.9033623  0.7543860
##   38     0.9061669  0.9033623  0.7543860
##   39     0.9061356  0.9033623  0.7543860
##   40     0.9061356  0.9033623  0.7543860
```

```
##   41     0.9061356  0.9033623  0.7543860
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 16.
```

```
test_preds <- data.frame(obs = response_test,
                         M3_PLSDA = predict(plsdaFit, biodeg_test))

confusionMatrix(test_preds$M3_PLSDA, test_preds$obs, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 131  13
##        RB    8  58
##
##                Accuracy : 0.9
##                  95% CI : (0.8512, 0.937)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 8.985e-16
##
##                   Kappa : 0.7727
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.8169
##             Specificity : 0.9424
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 0.9097
##              Prevalence : 0.3381
##          Detection Rate : 0.2762
##    Detection Prevalence : 0.3143
##       Balanced Accuracy : 0.8797
##
##        'Positive' Class : RB
##
```

```
write.csv(test_preds[,-1],"../data/model-output/y_pred_plsda.csv", row.names = FALSE)
```

ADS503-02-SP22 - Final Project: Team 3

Carr_Aaron

06/27/2022

RMarkdown global setup

```
knitr::opts_chunk$set(
  fig.align = 'center'
  )
```

```
library(caret)
library(pROC)
library(psych)
library(randomForest)
library(RANN)
library(ROCR)
library(rpart)
library(rpart.plot)
library(scales)
```

Importing Train/Test Datasets

```
train_x01_df01 <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
test_x01_df01 <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")

train_y01_df01 <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
test_y01_df01 <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

train_y01_vc01 <- as.factor(train_y01_df01[["x"]])
test_y01_vc01 <- as.factor(test_y01_df01[["x"]])
```

Setting up for models

```
set.seed(100)
train_y01_vc01_2cl_ctl <- trainControl(method = "cv",
                                       summaryFunction = twoClassSummary,
                                       classProbs = TRUE,
                                       savePredictions = TRUE)
```

Model 1.1 ($M_{1.1}$): CART DT Using `train()`

```
set.seed(100)
m1v1_cart <- train(x = train_x01_df01, y = train_y01_vc01,
                   method = "rpart",
                   tuneLength = 40,
                   metric = "ROC",
                   trControl = train_y01_vc01_2cl_ctl
                   )
test_pred_model_outcomes <- data.frame(obs = test_y01_vc01,
                                       M1.1.CART = predict(m1v1_cart, test_x01_df01))
test_pred_raw <- data.frame(M1.1.CART = predict(m1v1_cart, test_x01_df01))
```

Model 1.2 ($M_{1.2}$): C5.0 Using `train()`

```
set.seed(100)
m1v2_c50 <- train(x = train_x01_df01, y = train_y01_vc01,
                   method = "C5.0Tree",
                   metric = "ROC",
                   trControl = train_y01_vc01_2cl_ctl
                   )
test_pred_model_outcomes$M1.2.C5.0 = predict(m1v2_c50, test_x01_df01)
test_pred_raw$M1.2.C5.0 = predict(m1v2_c50, test_x01_df01)
```

```
sig <- 3
#m1v1_cart
#m1v1_cart$results
m1v1_cart_pred <- m1v1_cart$pred$obs
m1v1_cart_roc <- roc(response = m1v1_cart_pred,
                     predictor = m1v1_cart$pred$RB,
                     levels = levels(m1v1_cart_pred))
```

```
## Setting direction: controls < cases
```

```
confusionMatrix(m1v1_cart, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 480  96
##        RB   79 189
##
##  Accuracy (average) : 0.7927
```

```
m1v2_c50$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.8506733 0.8747727 0.7270936 0.03683413 0.04836271 0.06819443
```

```
m1v2_c50_pred <- m1v2_c50$pred$obs
m1v2_c50_roc <- roc(response = m1v2_c50_pred,
                    predictor = m1v2_c50$pred$RB,
                    levels = levels(m1v2_c50_pred))
```

```
## Setting direction: controls < cases
```

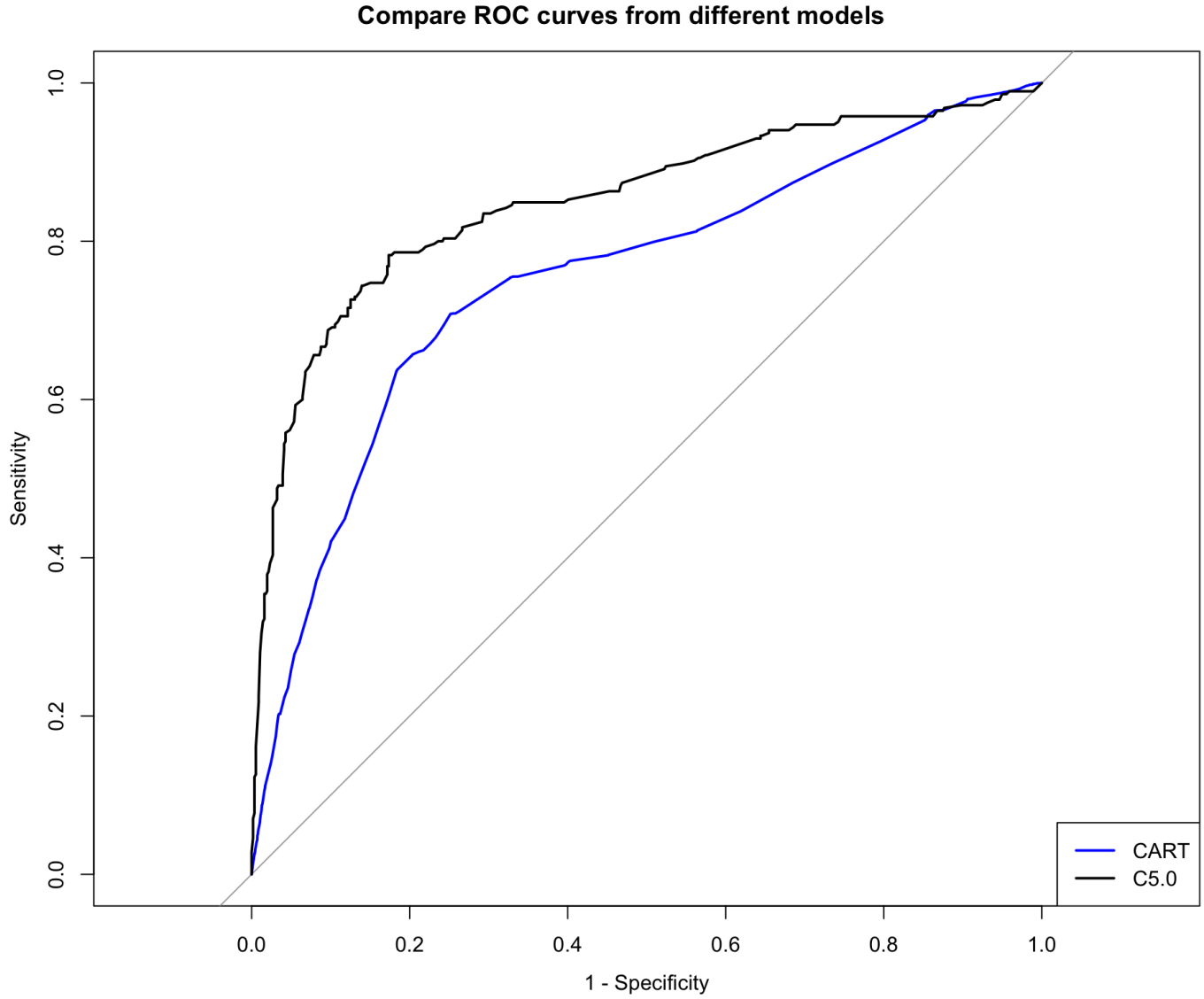
```
confusionMatrix(m1v2_c50, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 489  78
##        RB   70 207
##
##  Accuracy (average) : 0.8246
```

```
models_compare <- resamples(list(M1.1.CART = m1v1_cart,
                                 M1.2.C5.0 = m1v2_c50))
models_compare_summ <- summary(models_compare)
models_compare_summ
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: M1.1.CART, M1.2.C5.0
## Number of resamples: 10
##
## ROC
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M1.1.CART 0.7592474 0.8038793 0.8171182 0.8326233 0.8654638 0.9228316    0
## M1.2.C5.0 0.8069581 0.8246888 0.8384354 0.8506733 0.8707804 0.9180485    0
##
## Sens
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M1.1.CART 0.7857143 0.8392857 0.8558442 0.8586688 0.8750000 0.9285714    0
## M1.2.C5.0 0.8035714 0.8437500 0.8649351 0.8747727 0.9196429 0.9464286    0
##
## Spec
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M1.1.CART 0.5172414 0.5862069 0.6662562 0.6636700 0.7678571 0.7931034    0
## M1.2.C5.0 0.6206897 0.6896552 0.7321429 0.7270936 0.7564655 0.8571429    0
```

```
# Compare ROC curves
plot(m1v1_cart_roc, col = 'blue', legacy.axes = TRUE)
plot(m1v2_c50_roc, add = TRUE, col = 'black', legacy.axes = TRUE)
legend("bottomright", legend=c("CART", "C5.0"),
       col=c("blue", "black"), lwd = 2)
title(main = "Compare ROC curves from different models", outer = TRUE, line = -1)
```




```
print(paste0("Model 1.1: CART AUC = ", round(m1v1_cart_roc$auc, sig)))
```

```
## [1] "Model 1.1: CART AUC = 0.753"
```

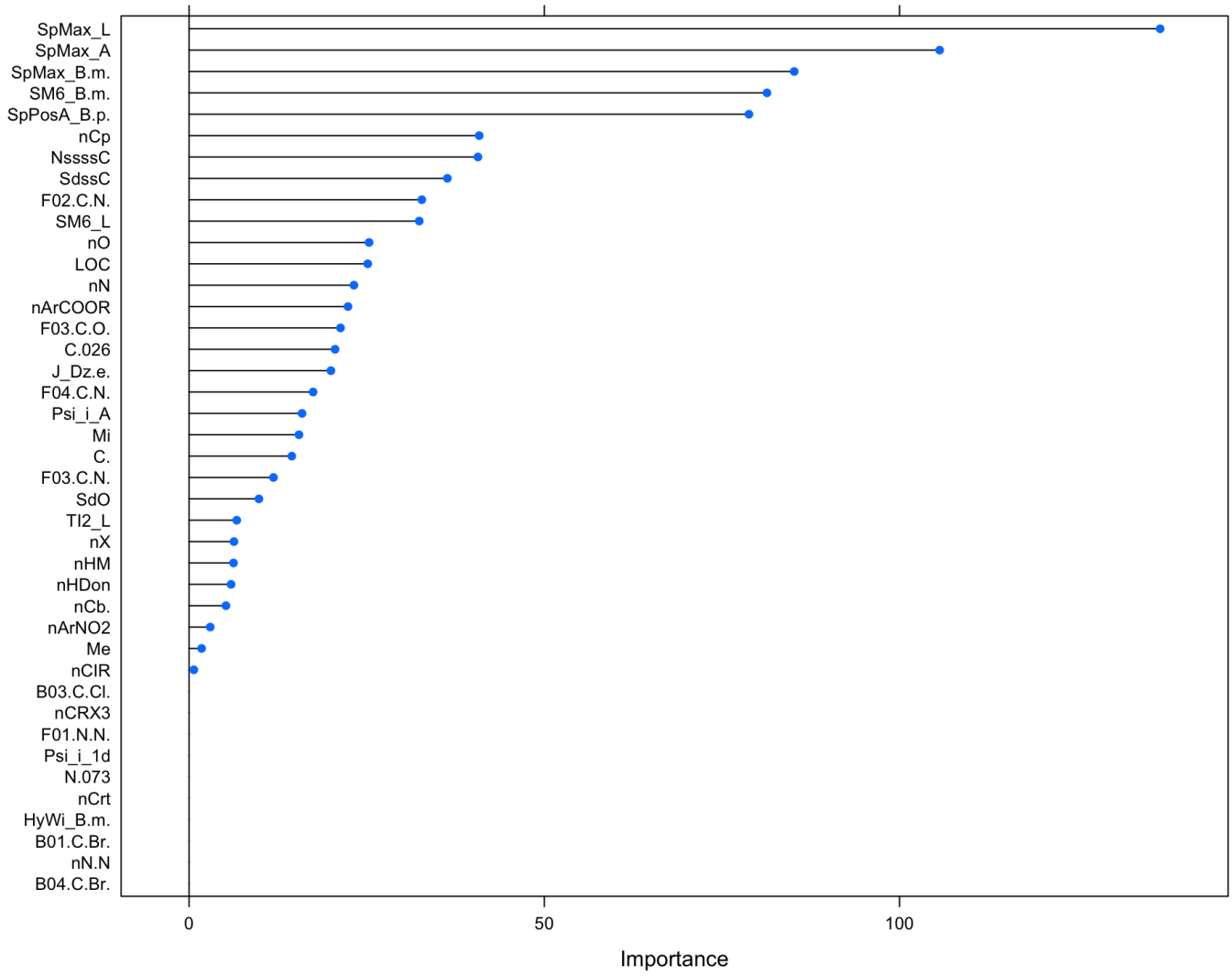
```
print(paste0("Model 1.2: C5.0 AUC = ", round(m1v2_c50_roc$auc, sig)))
```

```
## [1] "Model 1.2: C5.0 AUC = 0.85"
```

```
m1v1_cart_vip <- varImp(m1v1_cart, scale = FALSE)
m1v1_cart_vip
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 41)
##
##             Overall
## SpMax_L      136.68
## SpMax_A      105.64
## SpMax_B.m.    85.17
## SM6_B.m.      81.34
## SpPosA_B.p.   78.80
## nCp           40.83
## NssssC        40.66
## SdssC         36.36
## F02.C.N.      32.76
## SM6_L         32.40
## nO            25.34
## LOC           25.15
## nN            23.20
## nArCOOR       22.37
## F03.C.O.      21.32
## C.026         20.55
## J_Dz.e.       19.96
## F04.C.N.      17.46
## Psi_i_A       15.90
## Mi            15.47
```

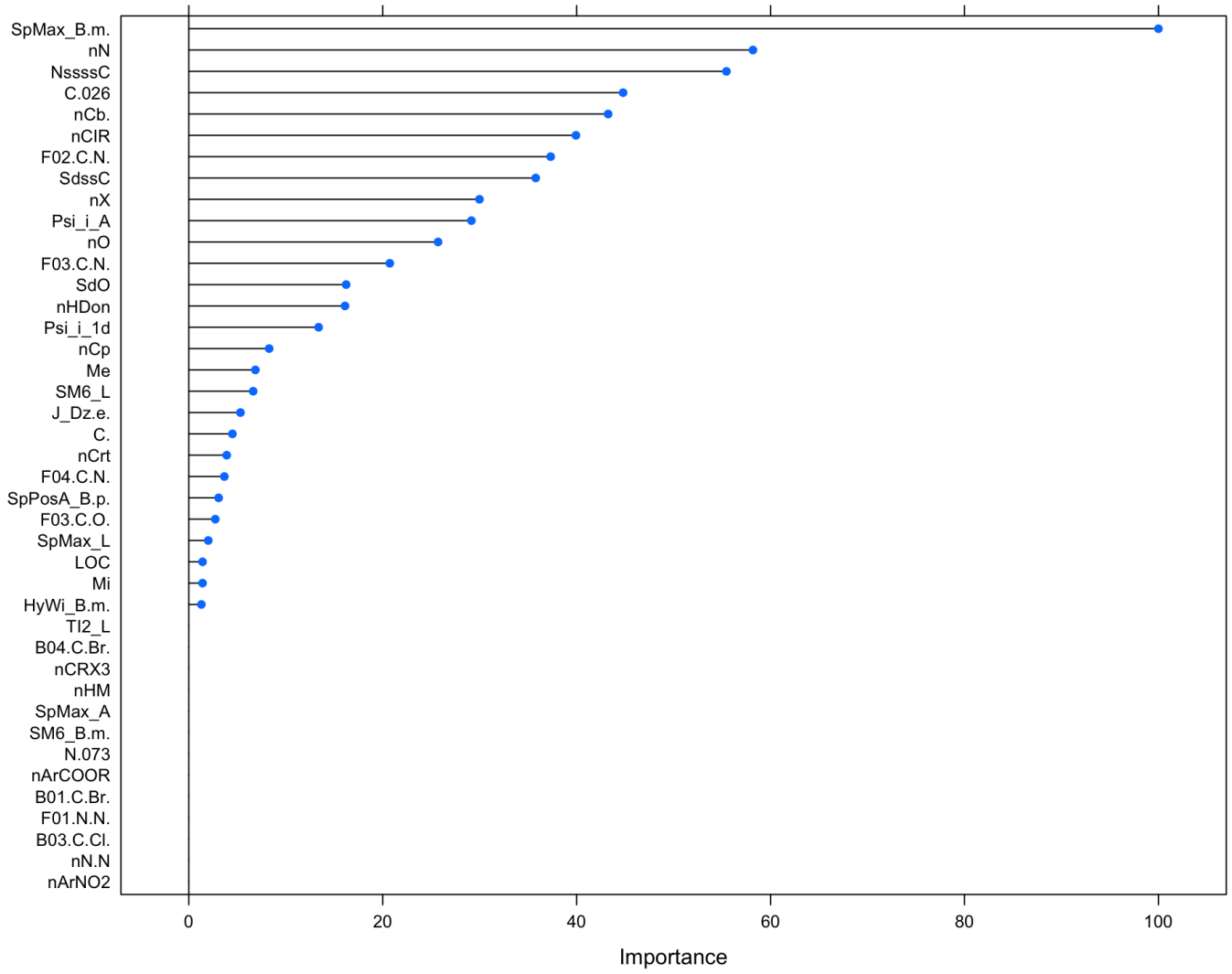
```
plot(m1v1_cart_vip)
```




```
m1v2_c50_vip <- varImp(m1v2_c50, scale = FALSE)
m1v2_c50_vip
```

```
## C5.0Tree variable importance
##
##   only 20 most important variables shown (out of 41)
##
##            Overall
## SpMax_B.m.  100.00
## nN           58.18
## NssssC       55.45
## C.026        44.79
## nCb.         43.25
## nCIR         39.93
## F02.C.N.     37.32
## SdssC        35.78
## nX           29.98
## Psi_i_A      29.15
## nO           25.71
## F03.C.N.     20.73
## SdO          16.23
## nHDon        16.11
## Psi_i_1d     13.39
## nCp           8.29
## Me            6.87
## SM6_L         6.64
## J_Dz.e.       5.33
## C.            4.50
```

```
plot(m1v2_c50_vip)
```



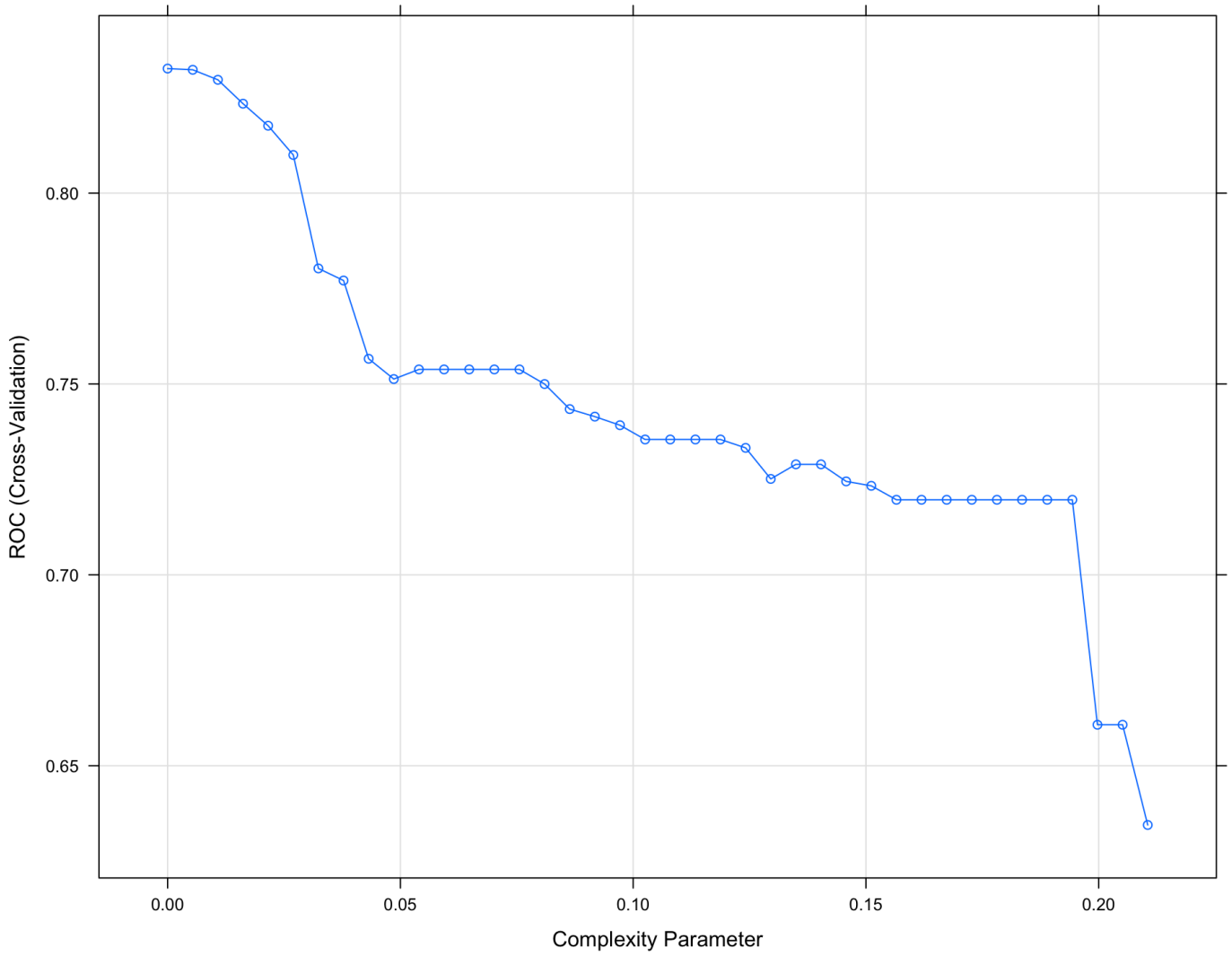

```
confusionMatrix(test_pred_raw$M1.1.CART, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 124  15
##        RB   15  56
##
##                Accuracy : 0.8571
##                  95% CI : (0.8024, 0.9015)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 1.2e-10
##
##                   Kappa : 0.6808
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7887
##             Specificity : 0.8921
##          Pos Pred Value : 0.7887
##          Neg Pred Value : 0.8921
##              Prevalence : 0.3381
##          Detection Rate : 0.2667
##    Detection Prevalence : 0.3381
##       Balanced Accuracy : 0.8404
##
##        'Positive' Class : RB
##
```

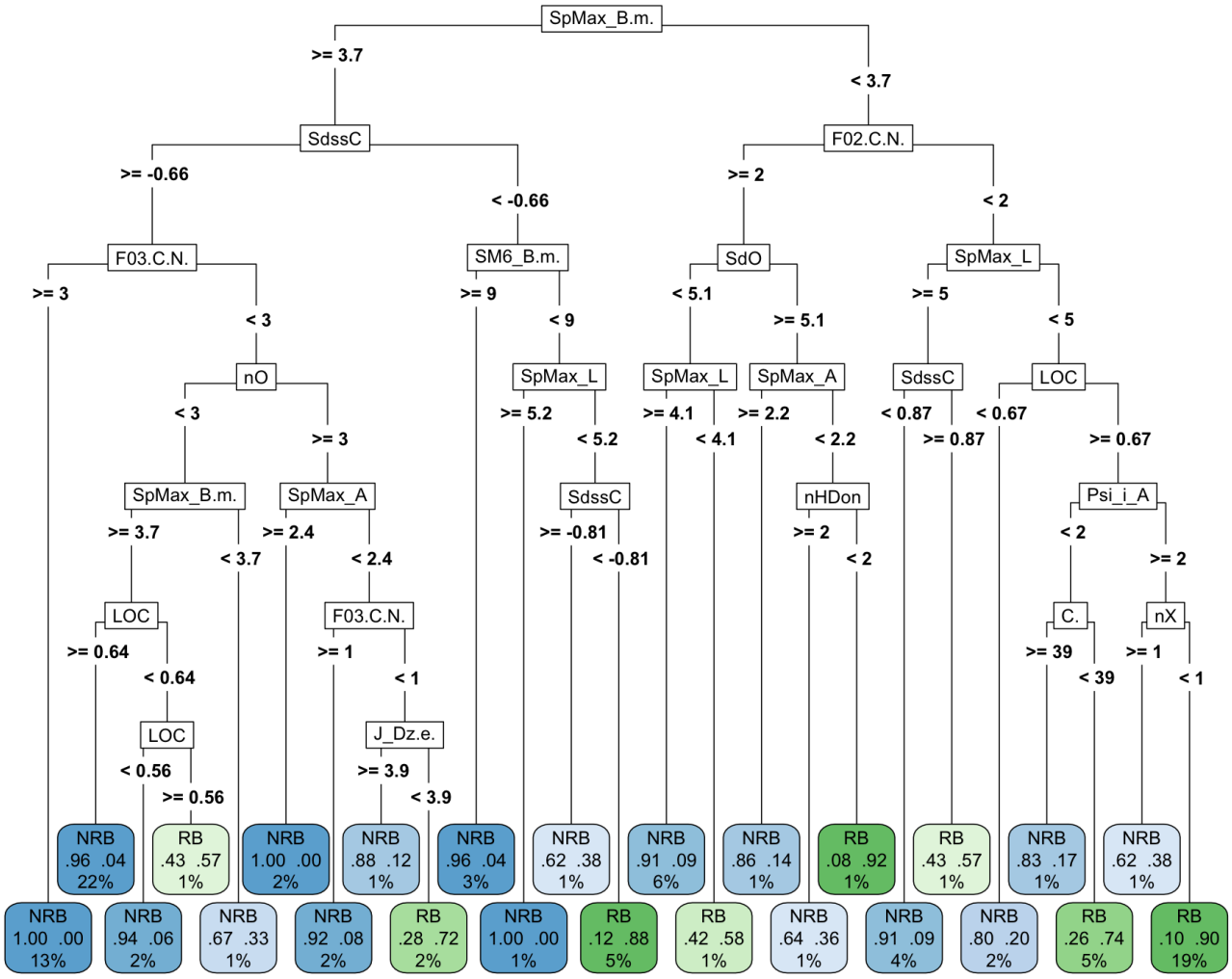
```
#m1v1_cart_y_pred_summ <- data.frame(obs = test_y01_vc01,
#                                    pred = predict(m1v1_cart, test_x01_df01))
#twoClassSummary(m1v1_cart_y_pred_summ, lev = c("RB", "NRB"))

confusionMatrix(test_pred_raw$M1.2.C5.0, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 128  16
##        RB   11  55
##
##                Accuracy : 0.8714
##                  95% CI : (0.8185, 0.9135)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 3.454e-12
##
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 0.4414
##
##             Sensitivity : 0.7746
##             Specificity : 0.9209
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8889
##              Prevalence : 0.3381
##          Detection Rate : 0.2619
##    Detection Prevalence : 0.3143
##       Balanced Accuracy : 0.8478
##
##        'Positive' Class : RB
##
```

```
#m1v2_c50_y_pred_summ <- data.frame(obs = test_y01_vc01,
#                                   pred = predict(m1v2_c50, test_x01_df01))
#twoClassSummary(m1v2_c50_y_pred_summ, lev = c("RB", "NRB"))

plot(m1v1_cart)
```




```
# Citation:
# https://stackoverflow.com/questions/39636186/plot-decision-tree-in-r-caret
rpart.plot(m1v1_cart$finalModel, type = 5, extra = 104)
```




```
#m1v2_c50_tree <- as.party(m1v2_c50$pred)
#plot.party(as.party(m1v2_c50[["pred"]]))
```

```
#write.csv(test_pred_model_outcomes, "../data/01_biodeg_test_outcomes.csv", row.names =
 FALSE)
write.csv(test_pred_raw,
          "../data/model-output/y_pred_dt.csv", row.names = FALSE)
write.csv(data.frame(models_compare_summ$models, models_compare_summ$statistics),
          "../data/resampling-results/resamp_results_dt.csv", row.names = FALSE)
```

ADS503-02-SP22 - Final Project: Team 3

Carr_Aaron

06/27/2022

RMarkdown global setup

```
knitr::opts_chunk$set(
  fig.align = 'center'
  )
```

```
library(caret)
library(pROC)
```

Importing Train/Test Datasets

```
train_x01_df01 <- read.csv("../data/biodeg_train.csv", header = TRUE, sep = ",")
test_x01_df01 <- read.csv("../data/biodeg_test.csv", header = TRUE, sep = ",")

train_y01_df01 <- read.csv("../data/response_train.csv", header = TRUE, sep = ",")
test_y01_df01 <- read.csv("../data/response_test.csv", header = TRUE, sep = ",")

train_y01_vc01 <- as.factor(train_y01_df01[["x"]])
test_y01_vc01 <- as.factor(test_y01_df01[["x"]])

train_x02_df01 <- read.csv("../data/univariate-selection-data/biodeg_train_univariate.cs
v", header = TRUE, sep = ",")
test_x02_df01 <- read.csv("../data/univariate-selection-data/biodeg_test_univariate.csv"
, header = TRUE, sep = ",")

test_pred_model_outcomes <- read.csv("../data/test-results/y_pred.csv", header = TRUE, s
ep = ",")
```

Setting up for models

```
set.seed(100)
train_y01_vc01_2cl_ctl <- trainControl(method = "cv",
                                       summaryFunction = twoClassSummary,
                                       classProbs = TRUE,
                                       savePredictions = TRUE)
```

Model 5.1 ($M_{5.1}$): Logistic Regression (LR)

```
# Train & test LR model
set.seed(100)
m5v1_glr <- train(x = train_x01_df01, y = train_y01_vc01,
                  method = "glm",
                  preProcess = c("nzv", "corr", "BoxCox", "center", "scale", "spatialSig
n"),
                  metric = "ROC",
                  trControl = train_y01_vc01_2cl_ctl)

test_pred_model_outcomes$M5.1.GLR <- predict(m5v1_glr, test_x01_df01)
test_pred_raw <- data.frame(M5.1.GLR = predict(m5v1_glr, test_x01_df01))
```

Model 5.2 ($M_{5.2}$): General Logistic Regression (GLR) w/ PCA

```
# Train & test LR model using principal component analysis (PCA)
set.seed(100)
m5v2_glr <- train(x = train_x01_df01, y = train_y01_vc01,
                  method = "glm",
                  preProcess = c("nzv", "corr", "BoxCox", "center", "scale", "pca", "spa
tialSign"),
                  metric = "ROC",
                  trControl = train_y01_vc01_2cl_ctl)

test_pred_model_outcomes$M5.2.GLR_PCA <- predict(m5v2_glr, test_x01_df01)
test_pred_raw$M5.2.GLR_PCA <- predict(m5v2_glr, test_x01_df01)
```

Model 5.3 ($M_{5.3}$): General Logistic Regression (GLR) w/ RF Univariate Analysis Feature Selection

```
# Train & test LR model using selected features
set.seed(100)
m5v3_glr <- train(x = train_x02_df01, y = train_y01_vc01,
                  method = "glm",
                  preProcess = c("nzv", "BoxCox", "center", "scale", "spatialSign"),
                  metric = "ROC",
                  trControl = train_y01_vc01_2cl_ctl)

test_pred_model_outcomes$M5.3.GLR_Univ <- predict(m5v3_glr, test_x02_df01)
test_pred_raw$M5.3.GLR_Univ <- predict(m5v3_glr, test_x02_df01)
```

Model 6.1 ($M_{6.1}$): Penalized Logistic Regression (PLR)

```
# Lambda controls regularization; alpha regulates region: 1 = lasso, 0 = ridge, 0<a<1 =
 combo
m6v1_plr_grd <- expand.grid(alpha = c(0, .1, .2, .4, .6, .8, 1),
                            lambda = seq(.01, .25, length = 20))
set.seed(100)
m6v1_plr <- train(x = train_x01_df01, y = train_y01_vc01,
                  method = "glmnet",
                  preProcess = c("nzv", "BoxCox", "center", "scale", "spatialSign"),
                  metric = "ROC",
                  trControl = train_y01_vc01_2cl_ctl,
                  tuneGrid = m6v1_plr_grd)
test_pred_model_outcomes$M6.1.PLR <- predict(m6v1_plr, test_x01_df01)
test_pred_raw$M6.1.PLR <- predict(m6v1_plr, test_x01_df01)
```

Model 6.2 ($M_{6.2}$): Penalized Logistic Regression (PLR) w/ RF Univariate Analysis Feature Selection

```
# Lambda controls regularization; alpha regulates region: 1 = lasso, 0 = ridge, 0<a<1 =
 combo
set.seed(100)
m6v2_plr <- train(x = train_x02_df01, y = train_y01_vc01,
                  method = "glmnet",
                  preProcess = c("nzv", "BoxCox", "center", "scale", "spatialSign"),
                  metric = "ROC",
                  trControl = train_y01_vc01_2cl_ctl,
                  tuneGrid = m6v1_plr_grd)
test_pred_model_outcomes$M6.2.PLR_Univ <- predict(m6v2_plr, test_x02_df01)
test_pred_raw$M6.2.PLR_Univ <- predict(m6v2_plr, test_x02_df01)
```

```
sig <- 3
m5v1_glr$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.9047424 0.8837662 0.7573892 0.02622391 0.03379501 0.07533931
```

```
m5v1_glr_pred <- m5v1_glr$pred$obs
m5v1_glr_roc <- roc(response = m5v1_glr_pred,
                    predictor = m5v1_glr$pred$RB,
                    levels = levels(m5v1_glr_pred))
```

```
## Setting direction: controls < cases
```

```
confusionMatrix(m5v1_glr, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 494  69
##        RB   65 216
##
##  Accuracy (average) : 0.8412
```

```
m5v2_glr
```

```
## Generalized Linear Model
##
## 844 samples
##  41 predictor
##   2 classes: 'NRB', 'RB'
##
## Pre-processing: Box-Cox transformation (9), centered (29), scaled
##  (29), principal component signal extraction (29), spatial sign
##  transformation (29), remove (12)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 759, 760, 759, 759, 759, 760, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8957497  0.8712013  0.7470443
```

```
m5v2_glr$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.8957497 0.8712013 0.7470443 0.03431199 0.02348263 0.06852231
```

```
m5v2_glr_pred <- m5v2_glr$pred$obs
m5v2_glr_roc <- roc(response = m5v2_glr_pred,
                    predictor = m5v2_glr$pred$RB,
                    levels = levels(m5v2_glr_pred))
```

```
## Setting direction: controls < cases
```

```
confusionMatrix(m5v2_glr, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 487  72
##        RB   72 213
##
##  Accuracy (average) : 0.8294
```

```
levels(m5v2_glr_pred)
```

```
## [1] "NRB" "RB"
```

```
rev(levels(m5v2_glr_pred))
```

```
## [1] "RB"  "NRB"
```

```
m5v3_glr$results
```

```
##   parameter     ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.90315 0.8767208 0.7295567 0.02629332 0.04782131 0.08236043
```

```
m5v3_glr_pred <- m5v3_glr$pred$obs
m5v3_glr_roc <- roc(response = m5v3_glr_pred,
                    predictor = m5v3_glr$pred$RB,
                    levels = levels(m5v3_glr_pred))
```

```
## Setting direction: controls < cases
```

```
confusionMatrix(m5v3_glr, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 490  77
##        RB   69 208
##
##  Accuracy (average) : 0.827
```

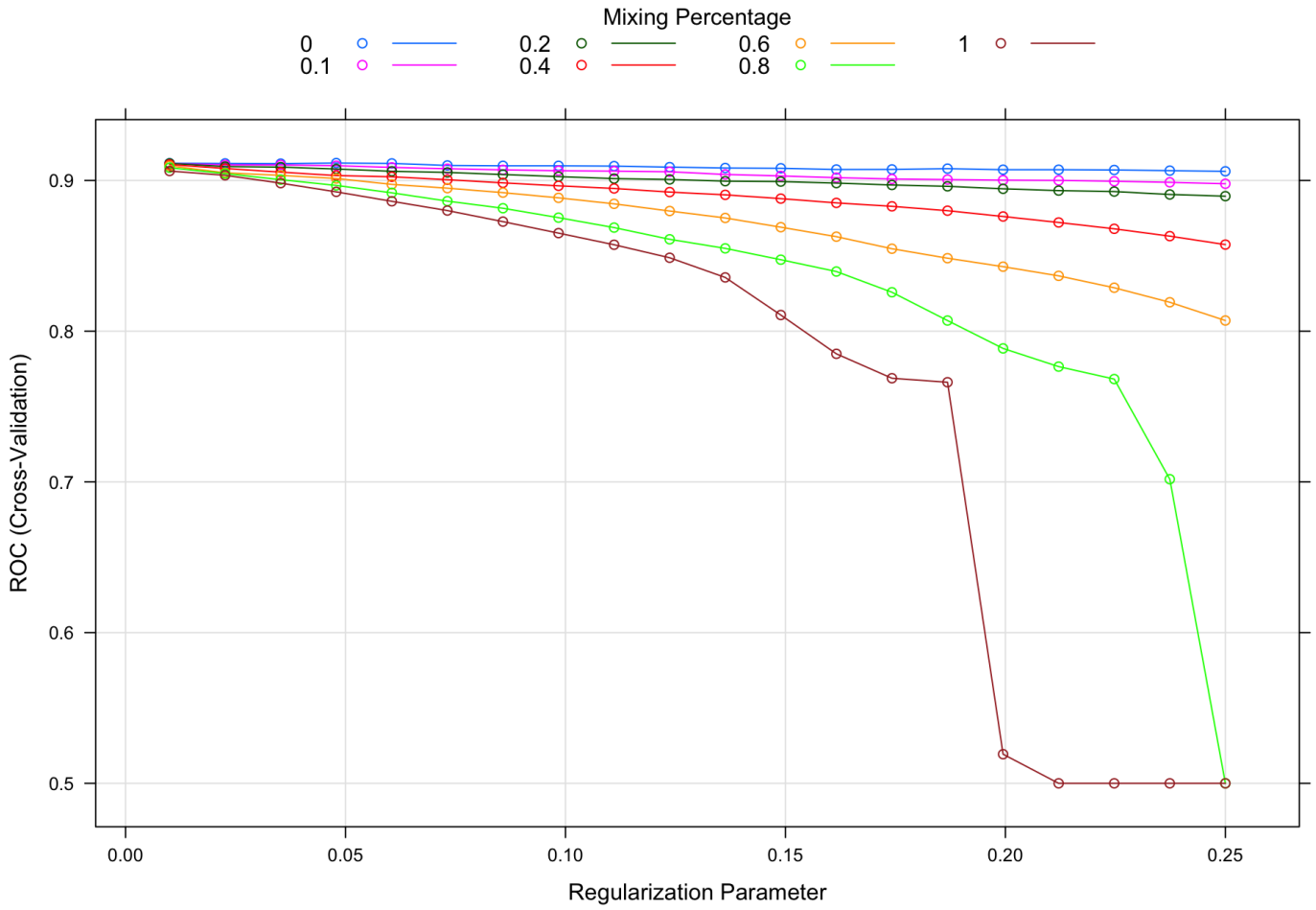
```
m6v1_plr$bestTune
```

```
##   alpha     lambda
## 4     0 0.04789474
```

```
#m6v1_plr
mean(m6v1_plr$results$ROC)
```

```
## [1] 0.8666543
```

```
plot(m6v1_plr)
```




```
m6v1_plr_pred <- m6v1_plr$pred$obs
m6v1_plr_roc <- roc(response = m6v1_plr_pred,
                    predictor = m6v1_plr$pred$RB,
                    levels = levels(m6v1_plr_pred))
```

```
## Setting direction: controls < cases
```

```
confusionMatrix(m6v1_plr, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 504  78
##        RB   55 207
##
##  Accuracy (average) : 0.8424
```

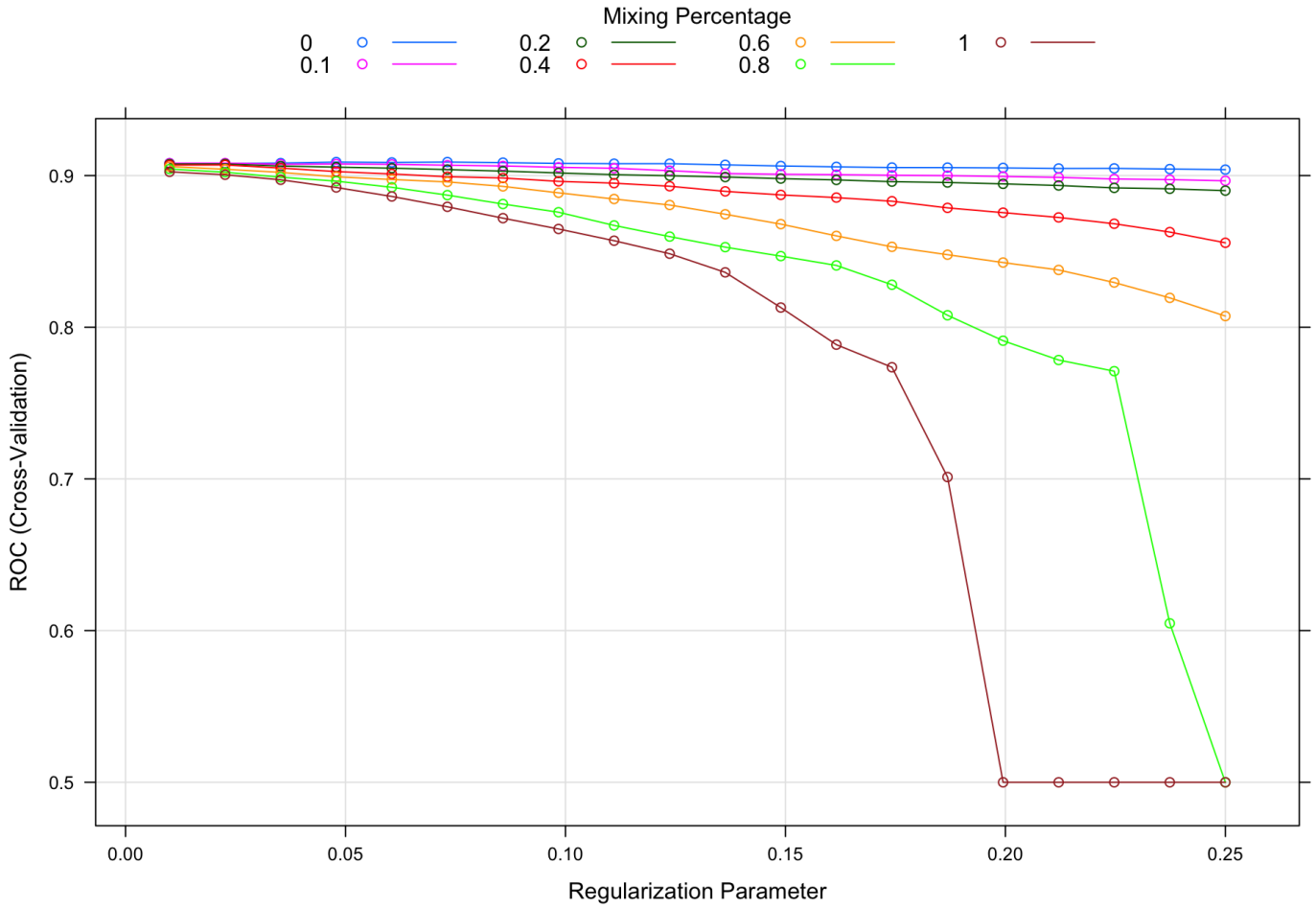
```
m6v2_plr$bestTune
```

```
##   alpha     lambda
## 6     0 0.07315789
```

```
#m6v2_plr
mean(m6v2_plr$results$ROC)
```

```
## [1] 0.8645381
```

```
plot(m6v2_plr)
```




```
m6v2_plr_pred <- m6v2_plr$pred$obs
m6v2_plr_roc <- roc(response = m6v2_plr_pred,
                    predictor = m6v2_plr$pred$RB,
                    levels = levels(m6v2_plr_pred))
```

```
## Setting direction: controls < cases
```

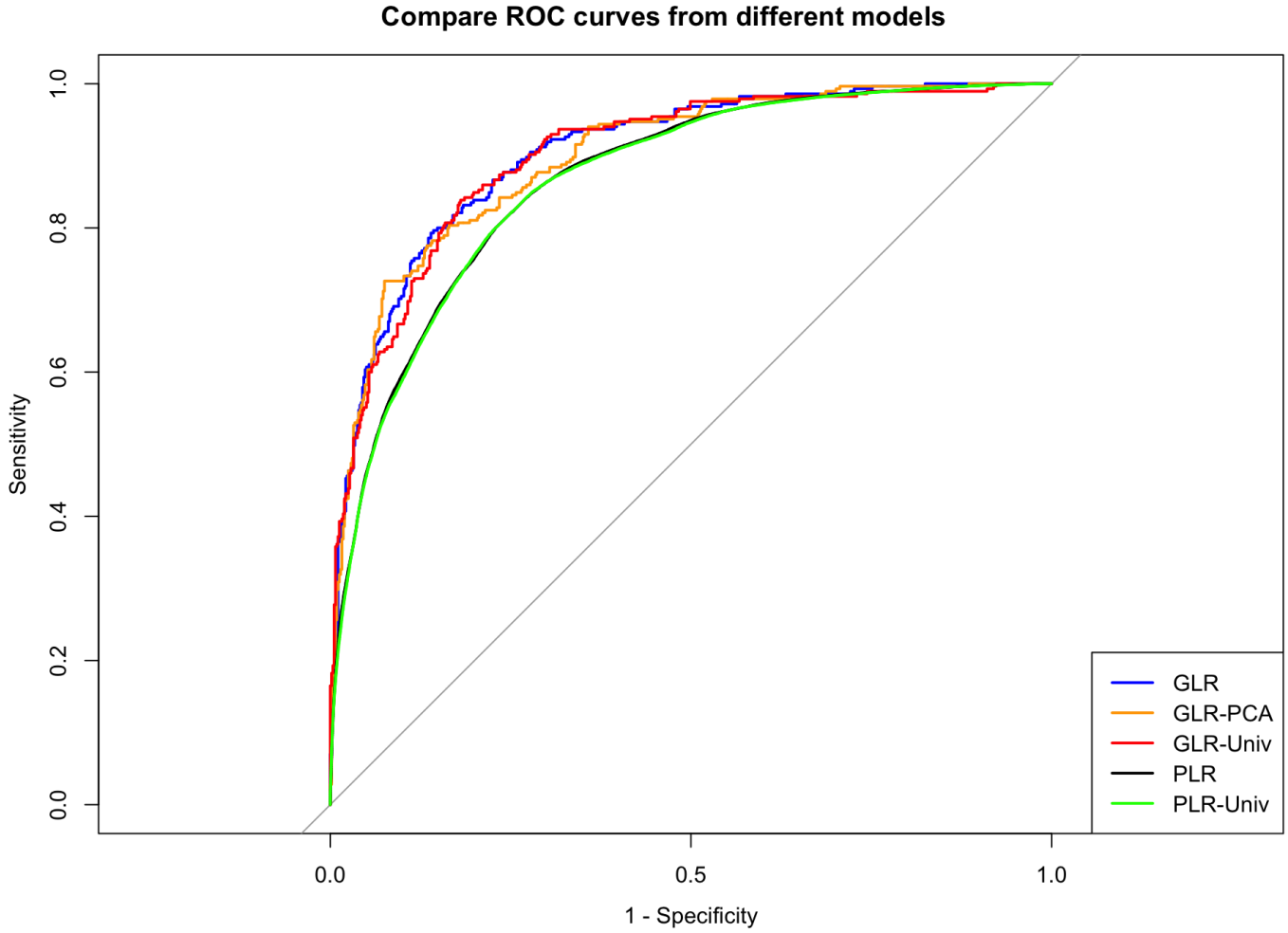
```
confusionMatrix(m6v2_plr, norm = "none")
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction NRB  RB
##        NRB 501  78
##        RB   58 207
##
##  Accuracy (average) : 0.8389
```

```
models_compare <- resamples(list(M5.1.GLR = m5v1_glr,
                                 M5.2.GLR_PCA = m5v2_glr,
                                 M5.3.GLR_Univ = m5v3_glr,
                                 M6.1.PLR = m6v1_plr,
                                 M6.2.PLR_Univ = m6v2_plr))
models_compare_summ <- summary(models_compare)
models_compare_summ
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: M5.1.GLR, M5.2.GLR_PCA, M5.3.GLR_Univ, M6.1.PLR, M6.2.PLR_Univ
## Number of resamples: 10
##
## ROC
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M5.1.GLR      0.8635204 0.8881125 0.9067338 0.9047424 0.9242061 0.9438776    0
## M5.2.GLR_PCA  0.8380102 0.8695131 0.9098566 0.8957497 0.9191992 0.9349490    0
## M5.3.GLR_Univ 0.8608374 0.8914222 0.9013239 0.9031500 0.9198080 0.9438776    0
## M6.1.PLR      0.8565271 0.9008236 0.9101205 0.9115487 0.9258957 0.9623724    0
## M6.2.PLR_Univ 0.8577586 0.8967771 0.9072396 0.9088909 0.9229288 0.9559949    0
##
## Sens
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M5.1.GLR      0.8392857 0.8616071 0.8750000 0.8837662 0.9005682 0.9464286    0
## M5.2.GLR_PCA  0.8392857 0.8571429 0.8738636 0.8712013 0.8883929 0.9107143    0
## M5.3.GLR_Univ 0.8214286 0.8392857 0.8660714 0.8767208 0.9151786 0.9636364    0
## M6.1.PLR      0.8214286 0.8750000 0.8928571 0.9016558 0.9416396 0.9642857    0
## M6.2.PLR_Univ 0.8571429 0.8616071 0.8839286 0.8962662 0.9237013 0.9642857    0
##
## Spec
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## M5.1.GLR      0.6785714 0.6875000 0.7413793 0.7573892 0.8189655 0.8928571    0
## M5.2.GLR_PCA  0.6551724 0.7142857 0.7192118 0.7470443 0.8057266 0.8620690    0
## M5.3.GLR_Univ 0.6071429 0.6896552 0.7192118 0.7295567 0.7564655 0.8620690    0
## M6.1.PLR      0.6551724 0.6875000 0.7192118 0.7262315 0.7500000 0.8214286    0
## M6.2.PLR_Univ 0.6428571 0.7142857 0.7192118 0.7262315 0.7500000 0.8214286    0
```

```
# Compare ROC curves
plot(m5v1_glr_roc, col = 'blue', legacy.axes = TRUE)
plot(m5v2_glr_roc, add = TRUE, col = 'orange', legacy.axes = TRUE)
plot(m5v3_glr_roc, add = TRUE, col = 'red', legacy.axes = TRUE)
plot(m6v1_plr_roc, add = TRUE, col = 'black', legacy.axes = TRUE)
plot(m6v2_plr_roc, add = TRUE, col = 'green', legacy.axes = TRUE)
legend("bottomright", legend=c("GLR", "GLR-PCA", "GLR-Univ", "PLR", "PLR-Univ"),
       col=c("blue", "orange", "red", "black", "green"), lwd = 2)
title(main = "Compare ROC curves from different models", outer = TRUE, line = -1)
```




```
print(paste0("Model 5.1: GLR AUC = ", round(m5v1_glr_roc$auc, sig)))
```

```
## [1] "Model 5.1: GLR AUC = 0.904"
```

```
print(paste0("Model 5.2: GLR-PCA AUC = ", round(m5v2_glr_roc$auc, sig)))
```

```
## [1] "Model 5.2: GLR-PCA AUC = 0.899"
```

```
print(paste0("Model 5.3: GLR-Univ AUC = ", round(m5v3_glr_roc$auc, sig)))
```

```
## [1] "Model 5.3: GLR-Univ AUC = 0.901"
```

```
print(paste0("Model 6.1: PLR AUC = ", round(m6v1_plr_roc$auc, sig)))
```

```
## [1] "Model 6.1: PLR AUC = 0.867"
```

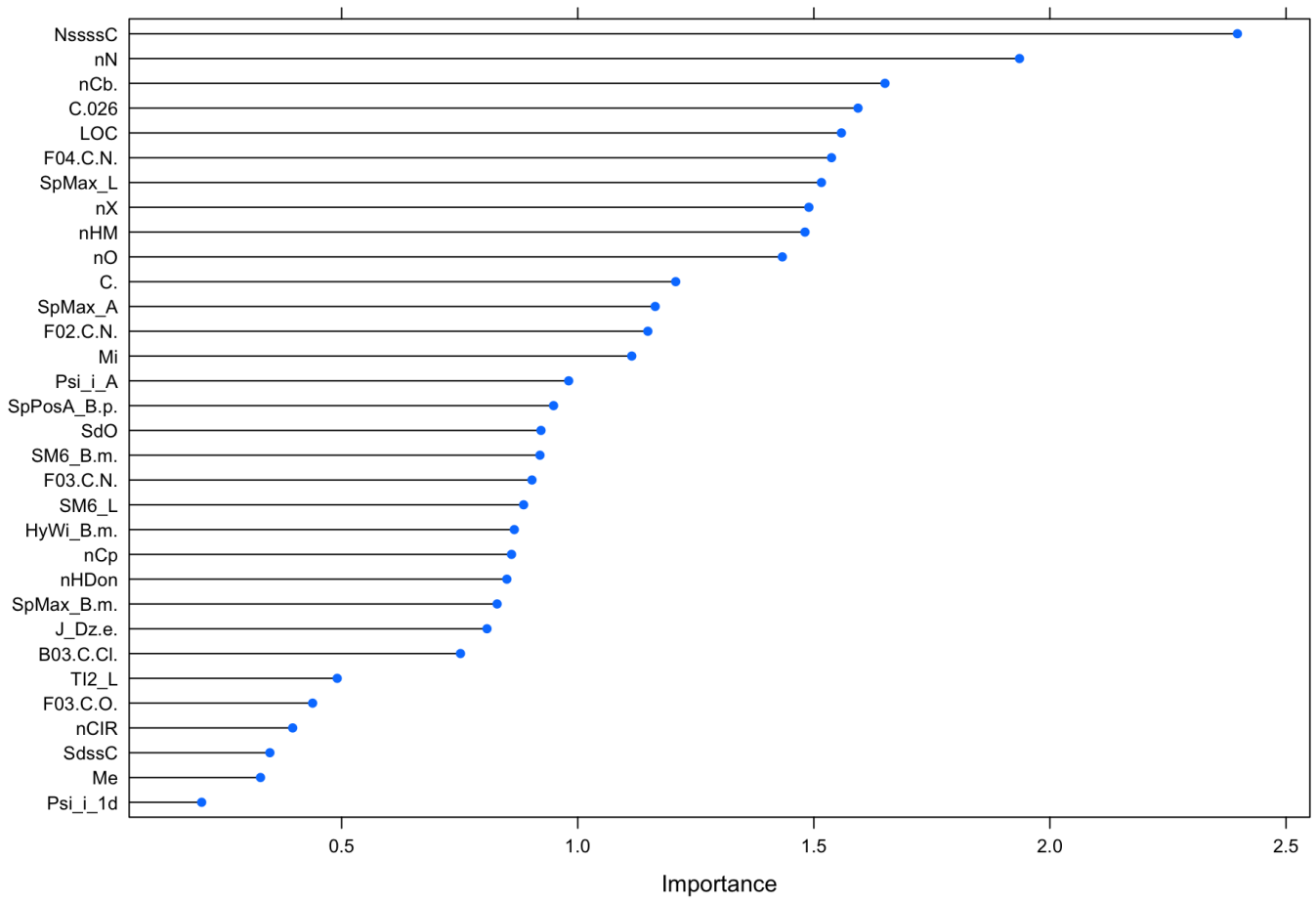
```
print(paste0("Model 6.2: PLR-Univ AUC = ", round(m6v2_plr_roc$auc, sig)))
```

```
## [1] "Model 6.2: PLR-Univ AUC = 0.865"
```

```
m6v1_plr_vip <- varImp(m6v1_plr, scale = FALSE)
m6v1_plr_vip
```

```
## glmnet variable importance
##
##   only 20 most important variables shown (out of 32)
##
##             Overall
## NssssC       2.3970
## nN           1.9356
## nCb.         1.6507
## C.026        1.5936
## LOC          1.5584
## F04.C.N.     1.5374
## SpMax_L      1.5161
## nX           1.4895
## nHM          1.4812
## nO           1.4333
## C.           1.2075
## SpMax_A      1.1640
## F02.C.N.     1.1484
## Mi           1.1144
## Psi_i_A      0.9808
## SpPosA_B.p.  0.9488
## SdO          0.9222
## SM6_B.m.     0.9200
## F03.C.N.     0.9031
## SM6_L        0.8854
```

```
plot(m6v1_plr_vip)
```




```
confusionMatrix(test_pred_raw$M5.1.GLR, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 130   9
##        RB    9  62
##
##                Accuracy : 0.9143
##                  95% CI : (0.8679, 0.9484)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8085
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8732
##             Specificity : 0.9353
##          Pos Pred Value : 0.8732
##          Neg Pred Value : 0.9353
##              Prevalence : 0.3381
##          Detection Rate : 0.2952
##    Detection Prevalence : 0.3381
##       Balanced Accuracy : 0.9042
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(test_pred_raw$M5.2.GLR_PCA, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 127  10
##        RB   12  61
##
##                Accuracy : 0.8952
##                  95% CI : (0.8457, 0.9332)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 4e-15
##
##                   Kappa : 0.7675
##
##  Mcnemar's Test P-Value : 0.8312
##
##             Sensitivity : 0.8592
##             Specificity : 0.9137
##          Pos Pred Value : 0.8356
##          Neg Pred Value : 0.9270
##              Prevalence : 0.3381
##          Detection Rate : 0.2905
##    Detection Prevalence : 0.3476
##       Balanced Accuracy : 0.8864
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(test_pred_raw$M5.3.GLR_Univ, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 127  11
##        RB   12  60
##
##                Accuracy : 0.8905
##                  95% CI : (0.8402, 0.9293)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 1.695e-14
##
##                   Kappa : 0.7561
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8451
##             Specificity : 0.9137
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.9203
##              Prevalence : 0.3381
##          Detection Rate : 0.2857
##    Detection Prevalence : 0.3429
##       Balanced Accuracy : 0.8794
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(test_pred_raw$M6.1.PLR, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 129  10
##        RB   10  61
##
##                Accuracy : 0.9048
##                  95% CI : (0.8567, 0.9409)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7872
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8592
##             Specificity : 0.9281
##          Pos Pred Value : 0.8592
##          Neg Pred Value : 0.9281
##              Prevalence : 0.3381
##          Detection Rate : 0.2905
##    Detection Prevalence : 0.3381
##       Balanced Accuracy : 0.8936
##
##        'Positive' Class : RB
##
```

```
confusionMatrix(test_pred_raw$M6.2.PLR_Univ, test_y01_vc01, positive = "RB")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 128  12
##        RB   11  59
##
##                Accuracy : 0.8905
##                  95% CI : (0.8402, 0.9293)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 1.695e-14
##
##                   Kappa : 0.7544
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8310
##             Specificity : 0.9209
##          Pos Pred Value : 0.8429
##          Neg Pred Value : 0.9143
##              Prevalence : 0.3381
##          Detection Rate : 0.2810
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.8759
##
##        'Positive' Class : RB
##
```

```
write.csv(data.frame(m5v1_glr$pred$obs, m5v1_glr$pred$RB),
          "../data/model-output/y_prob_m5v1.csv", row.names = FALSE)
write.csv(data.frame(m5v2_glr$pred$obs, m5v2_glr$pred$RB),
          "../data/model-output/y_prob_m5v2.csv", row.names = FALSE)
write.csv(data.frame(m5v3_glr$pred$obs, m5v3_glr$pred$RB),
          "../data/model-output/y_prob_m5v3.csv", row.names = FALSE)
write.csv(data.frame(m6v1_plr$pred$obs, m6v1_plr$pred$RB),
          "../data/model-output/y_prob_m6v1.csv", row.names = FALSE)
write.csv(data.frame(m6v2_plr$pred$obs, m6v2_plr$pred$RB),
          "../data/model-output/y_prob_m6v2.csv", row.names = FALSE)
```

```
#write.csv(test_pred_model_outcomes, "../data/test-results/y_pred_amc.csv", row.names =
 FALSE)
write.csv(test_pred_raw,
          "../data/model-output/y_pred_lr.csv", row.names = FALSE)
write.csv(data.frame(models_compare_summ$models, models_compare_summ$statistics),
          "../data/resampling-results/resamp_results_lr.csv", row.names = FALSE)
```

1.0-sk-scatterplots

Sreeja K

2022-06-25

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✔ tibble  3.1.7     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
## ✔ purrr   0.3.4
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

```
library(ggplot2)
library(GGally)
```

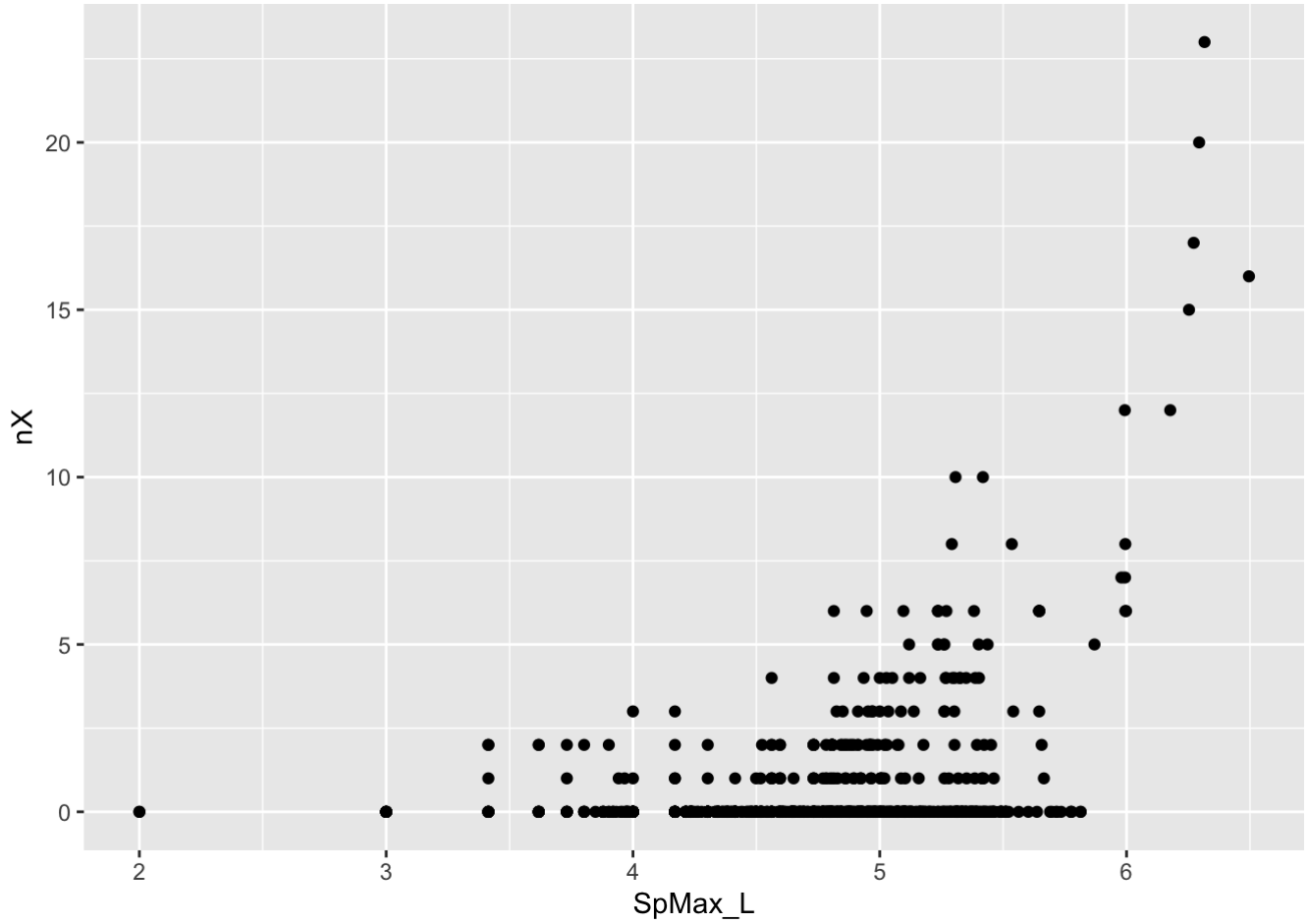
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
set.seed(100)
```

```
names <- c("SpMax_L", "J_Dz(e)", "nHM", "F01[N-N]", "F04[C-N]", "NssssC", "nCb-", "C%",
"nCp", "nO", "F03[C-N]", "SdssC", "HyWi_B(m)", "LOC", "SM6_L", "F03[C-O]", "Me", "Mi",
"nN-N", "nArNO2", "nCRX3", "SpPosA_B(p)", "nCIR", "B01[C-Br]", "B03[C-Cl]", "N-073", "Sp
Max_A", "Psi_i_1d", "B04[C-Br]", "SdO", "TI2_L", "nCrt", "C-026", "F02[C-N]", "nHDon",
"SpMax_B(m)", "Psi_i_A", "nN", "SM6_B(m)", "nArCOOR", "nX", "experimental class")

biodeg <- read.csv("../data/biodeg.csv", header = FALSE, sep = ";", col.names = names)
```

```
index_biodeg <- createDataPartition(biodeg$experimental.class, p=.80, list = FALSE)
biodeg_trainset <- biodeg[index_biodeg,]
biodeg_testset <- biodeg[-index_biodeg,]

response_train <- biodeg_trainset[,42]
response_test <- biodeg_testset[,42]

biodeg_train <- biodeg_trainset[,-42]
biodeg_test <- biodeg_testset[,-42]
```

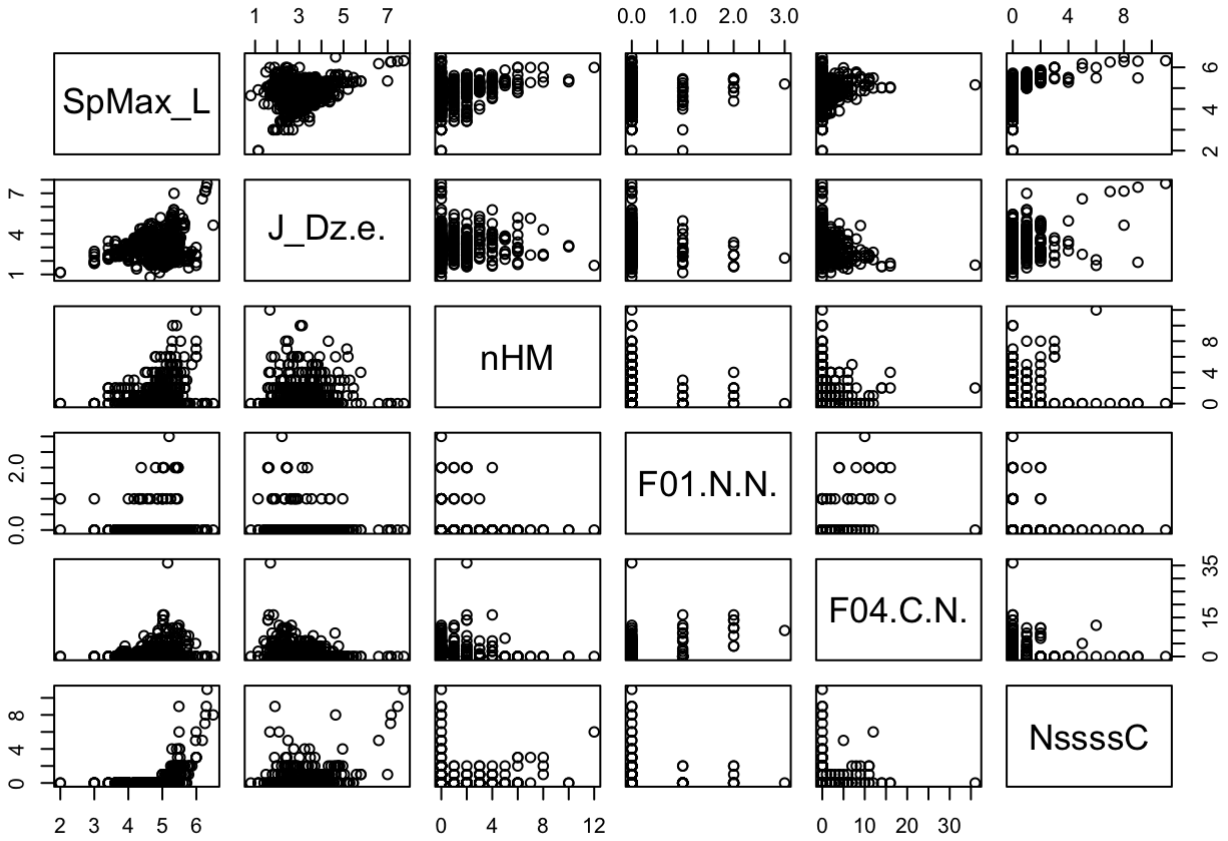
```
ggplot(biodeg_trainset, aes(x = SpMax_L, y = nX)) + geom_point()
```



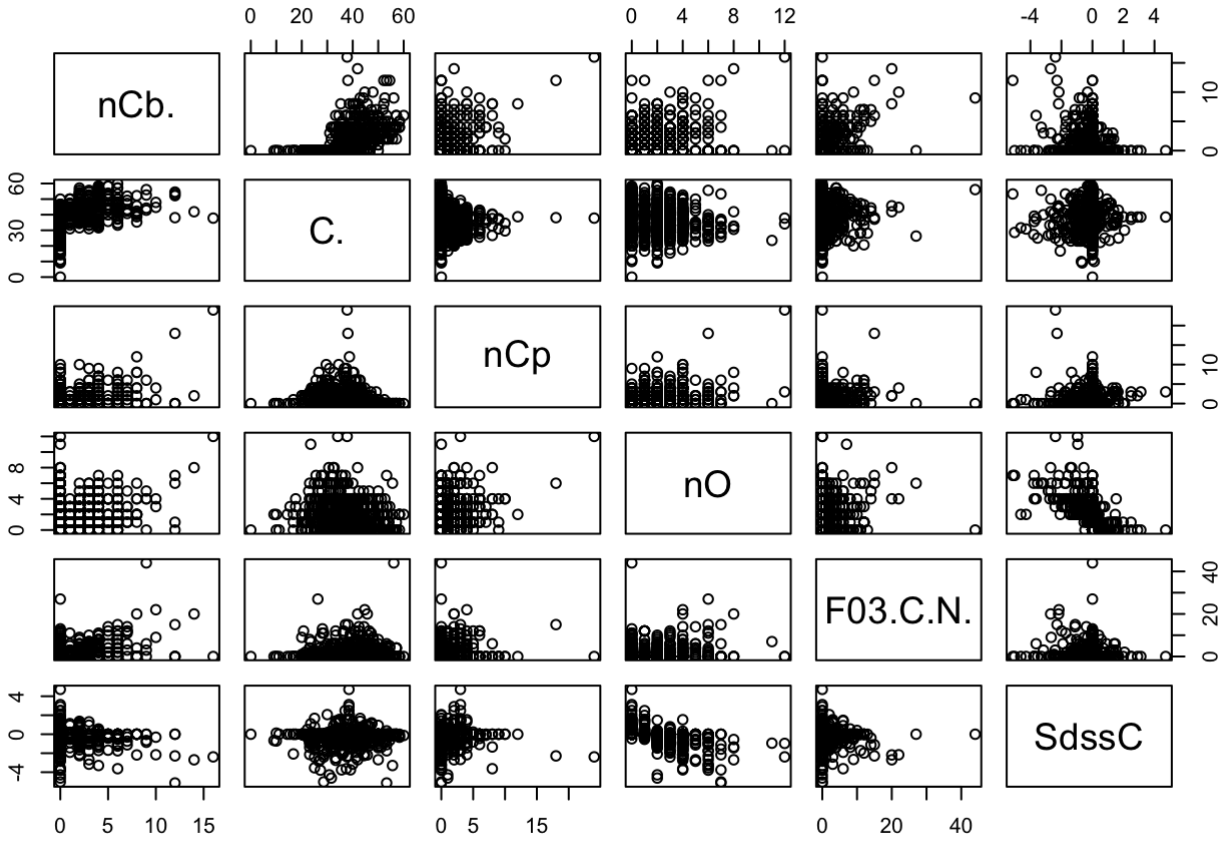

```
#Scatterplot matrix
'par("mar")
par(mar=c(1,1,1,1))'
```

```
## [1] "par(\"mar\")\npar(mar=c(1,1,1,1))"
```

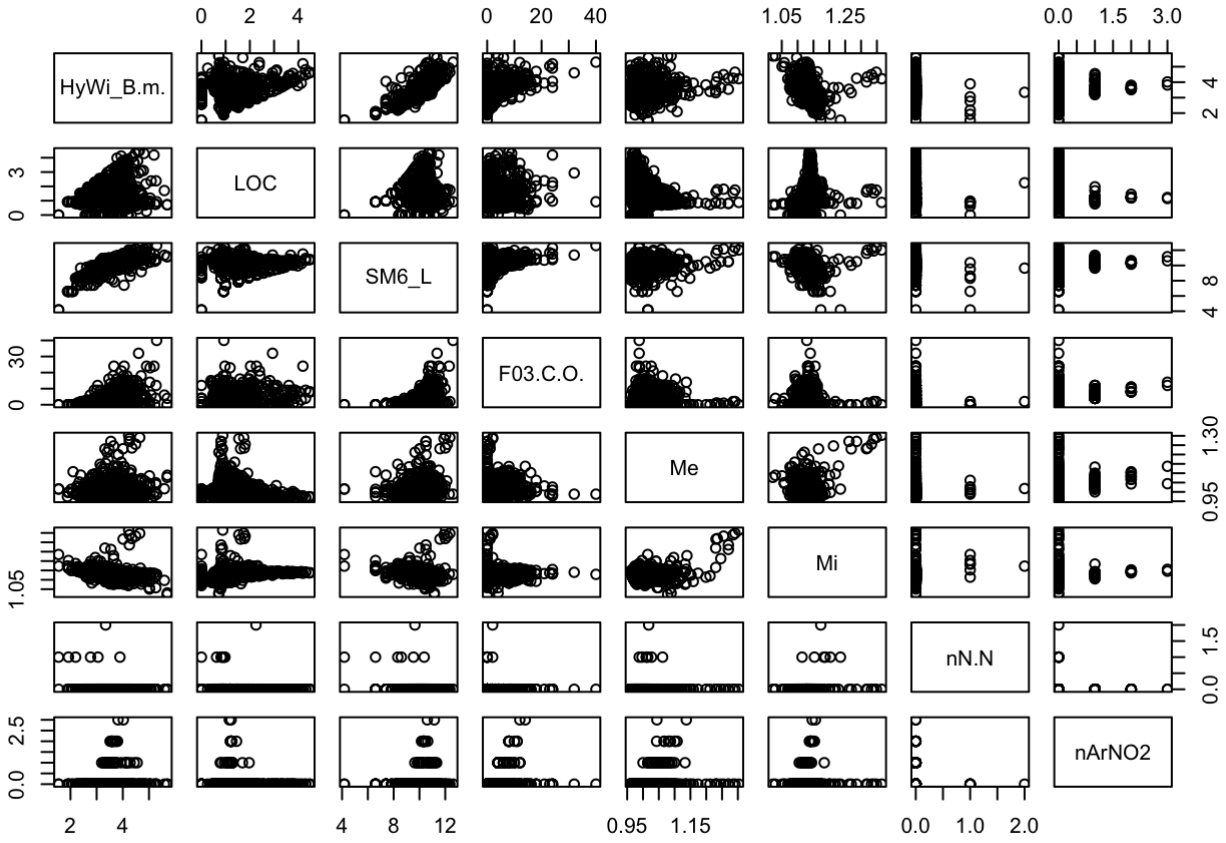
```
pairs(biodeg_trainset[1:6])
```



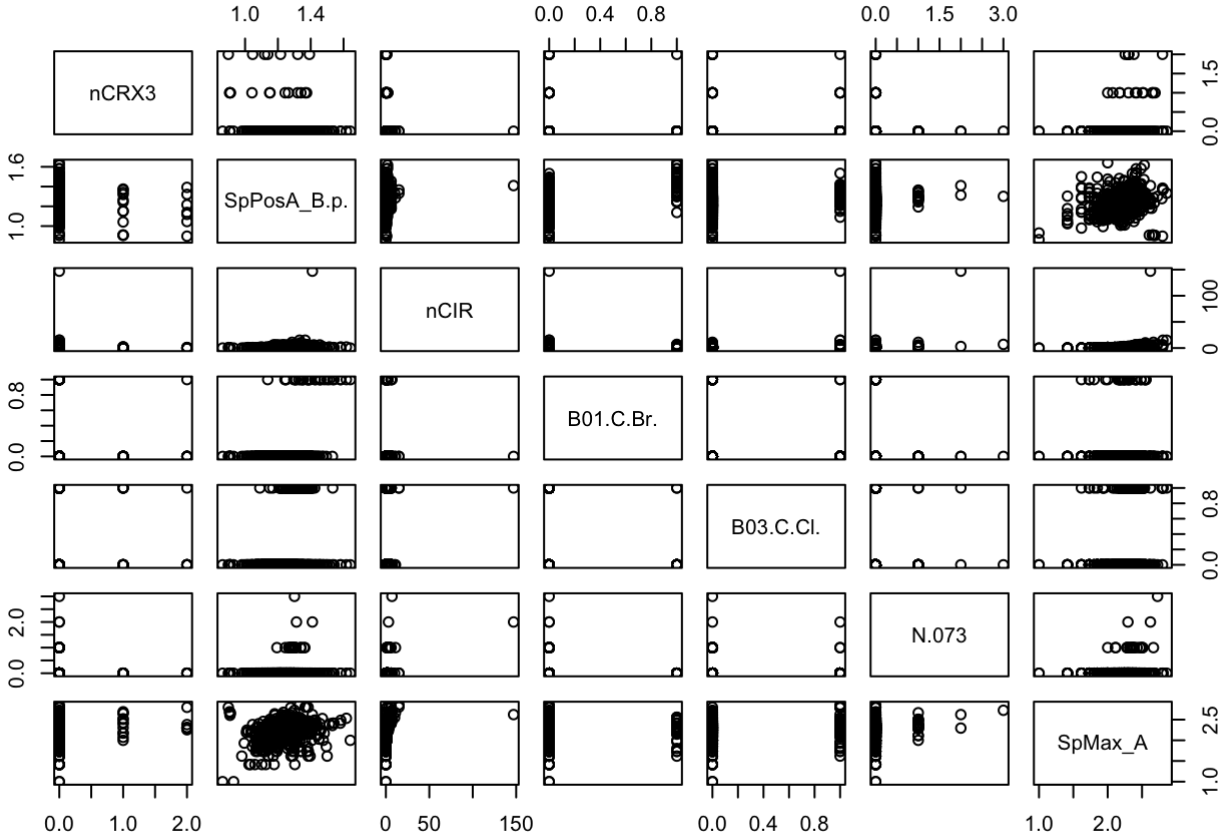

```
pairs(biodeg_trainset[7:12])
```



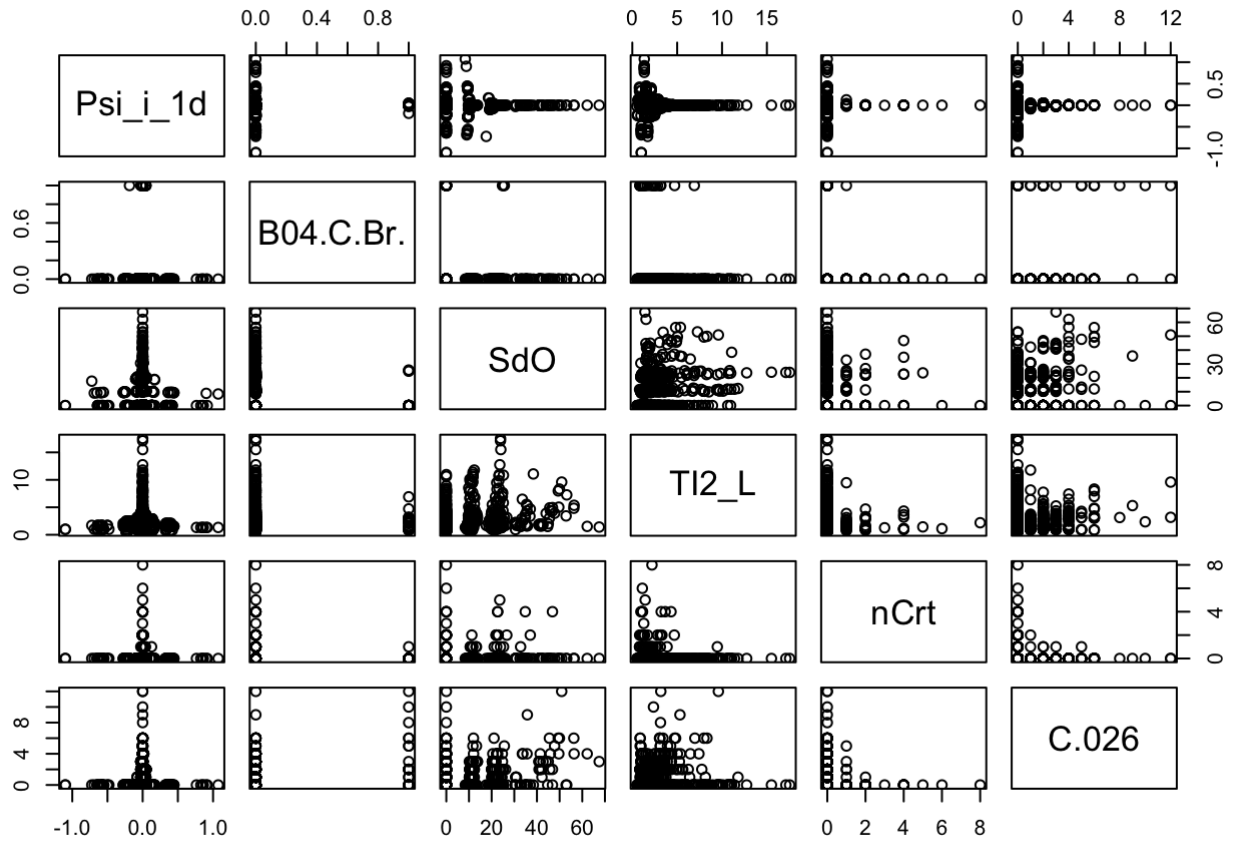

```
pairs(biodeg_trainset[13:20])
```



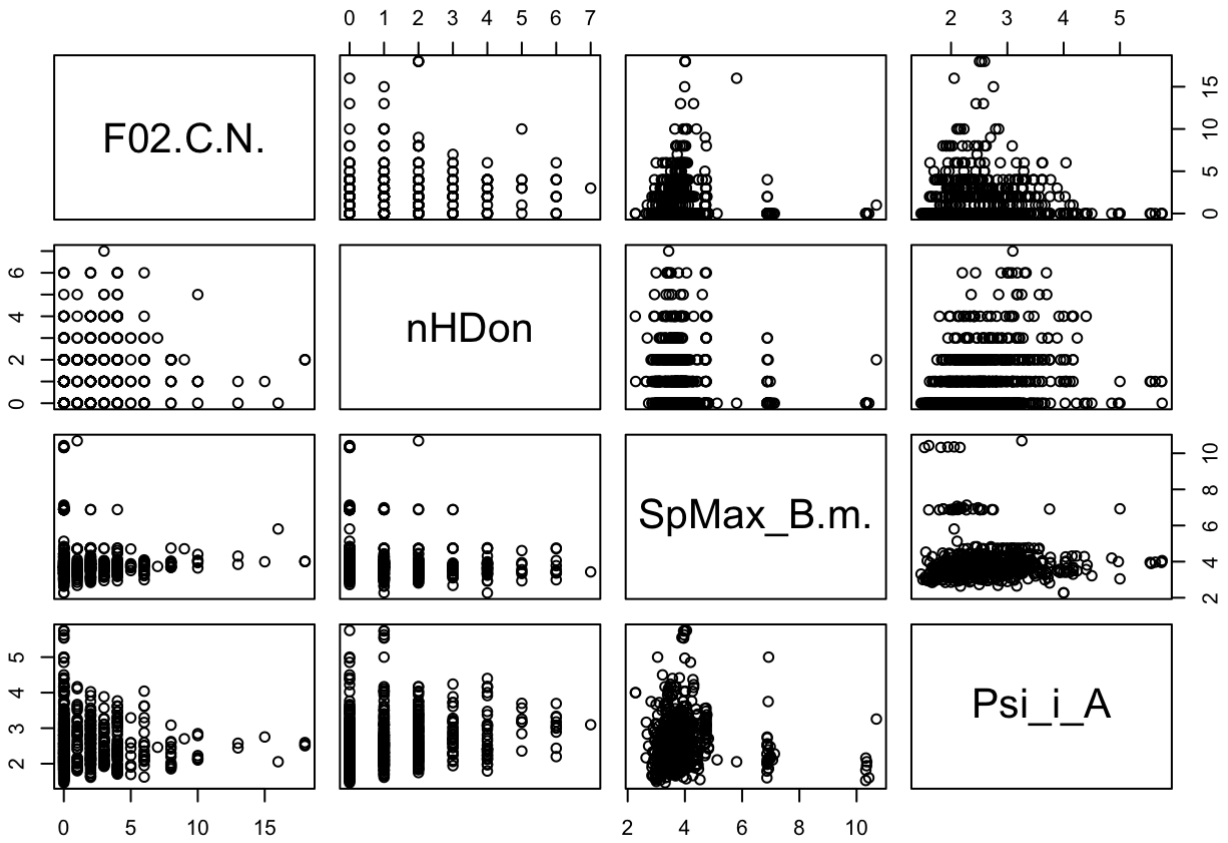

```
pairs(biodeg_trainset[21:27])
```



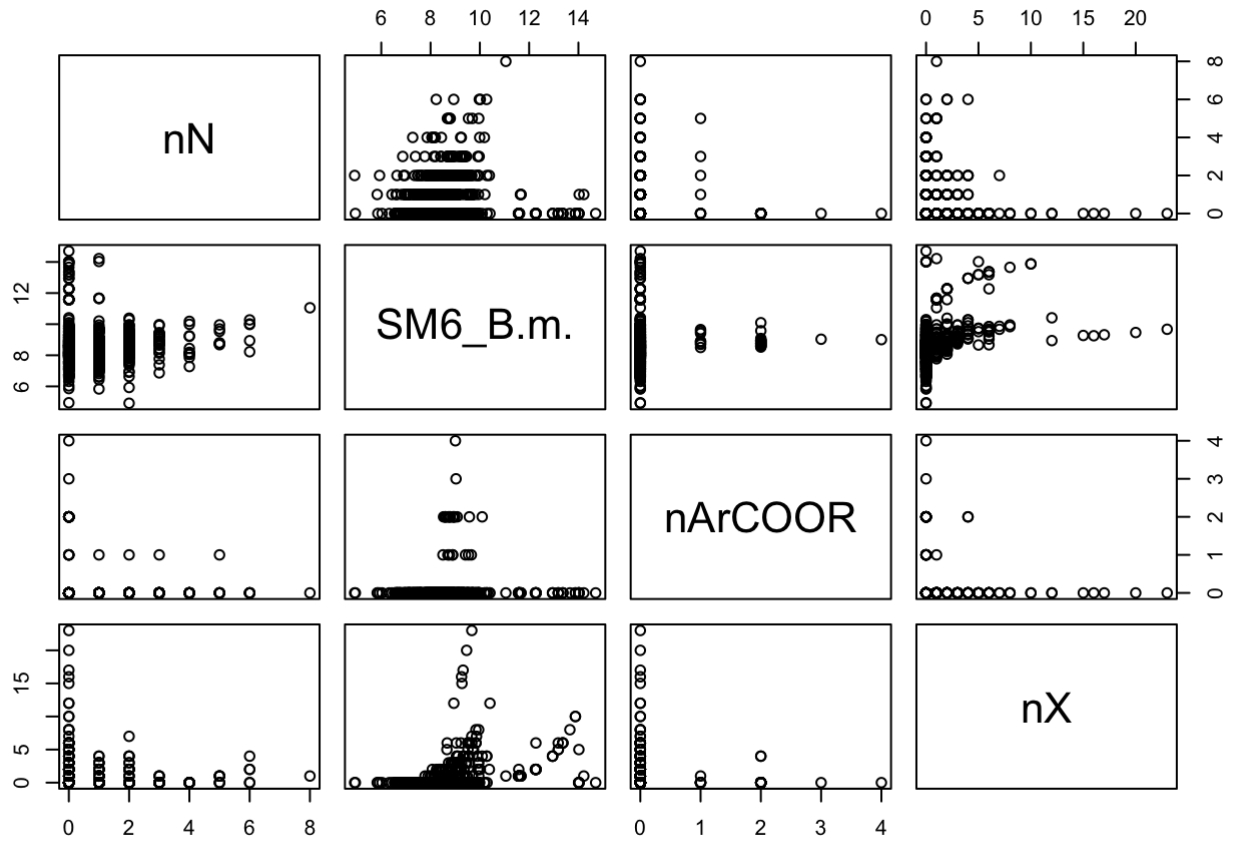

```
pairs(biodeg_trainset[28:33])
```




```
pairs(biodeg_trainset[34:37])
```




```
pairs(biodeg_trainset[38:41])
```


nN
SM6_B.m.
nArCOOR
nX
6 8 10 12 14
0 5 10 15 20
0 2 4 6 8
6 8 12
0 1 2 3 4
0 5 15
0 2 4 6 8
0 1 2 3 4