

Statistics 215B: Final Exam

UC Berkeley, Spring 2011

Hoxie Ackerman

4:00 PM, May 10, 2011

1 Partial Dependence Functions

1.1 Introduction and Definitions

Decision trees are a statistical learning technique that can be used for both regression and classification purposes. The most popular implementation of decision trees, CART ("classification and regression trees"), partitions the observation space into distinct regions via a greedy recursive partitioning algorithm and then fits a simple function to each region, often a constant function (regression) or a single class label (classification). CART has a number of advantages as a technique: growing trees and applying them to new data are computationally inexpensive, almost no tuning is required, many types of variables and missing observations are handled gracefully, and results are often robust to the inclusion of unimportant features and outliers.

Furthermore, a classification tree is highly interpretable: a binary trees can be examined visually and the variables responsible for splits, especially those towards the top of the tree, can be easily ascertained. Applying this idea numerically, we can use the average decrease in loss associated with the splits from each variable to compute relative importances of predictors. Once the important variables have been identified, understanding the relationship between these variables and the outcome can be important in many contexts. Partial dependence functions and plots can be used for this purpose.

Given p predictors (X_1, \dots, X_p) , we can evaluate the dependence of $f(X)$ on a small subset of all predictors X_S , with $S \subset \{1, \dots, p\}$. (Usually we take $|S| \in \{1, 2\}$ so that the partial dependence can be visualized in two or three dimensions, respectively.) Letting $C = \{1, \dots, p\} \setminus S$ be the remaining variables, classifier $f(X)$ can be written $f(X) = f(X_C, X_S)$. To understand how f varies with X_S , average out X_C :

$$f_S(X_S) = \mathbb{E}_{X_C}[f(X_S, X_C)] \quad (1)$$

Marginal average $f_S(X_S)$ can be approximated from training data via:

$$\hat{f}_S(X_S) = \sum_{i=1}^n f(X_S, x_{iC}), \quad (2)$$

where $\{x_{iC}\}_{i=1}^n$ are the training data values [1].

1.2 Partial Dependence for Trees

The computational expense of evaluating $\hat{f}_S(X_S)$ will depend generally on the dimension of S (more variables means more joint values at which to evaluate \hat{f}_S) and specifically on the form that f takes. For a single classification tree, the calculations are much easier than they are for other methods because they can be computed directly from the complete tree grown using all training data.

A naive algorithm for generating a partial dependence plot would look something like:

1. Build classifier f using all training data.
2. Partition all predictors (X_1, \dots, X_p) into X_S and X_C .
3. Holding the values of X_C constant at their observed training data values, evaluate $\hat{f}_S(X_S)$ across a grid of reasonable values for X_S . For each level of X_S , a `predict`-type function must be called.
4. Optionally smooth the predicted values using some density smoother and plot them against their respective X_S values.

This algorithm is relatively easy to perform for a classification tree f for two reasons. First, for observation x , evaluating $f(x)$ is computationally inexpensive: a series of binary decisions based on the levels of x puts x into one of the partitioned regions and that region's outcome value is returned. Note that once the tree has been built, the training data are technically no longer needed: the binary rules (and perhaps surrogate variables computed for each node in the tree) are enough to classify a new observation.

Second, and more importantly, using binary rules for decisions means that all x in certain subsets of the feature space will take the same path down a tree. After all, a classification tree partitions the feature space

into mutually exclusive recursively-defined regions. Rather than computing $\hat{f}_S(X_S)$ across an entire grid, a clever partial dependence algorithm should be able to figure out the boundaries between regions and thereby evaluate many grid points at once.

To evaluate the functionality of my naive algorithm above and to better understand dependence plots, I considered Fisher's Iris data, specifically predicting Sepal Length from the other three quantitative flower measurements: Sepal Width, Petal Length, and Petal Width. A random forest choosing just one variable for each split with 1000 trees explained about 82% of the variability in Sepal Length, and the variable that produced the greatest reductions in mean squared error and node impurity was Petal Length.

The reason that I started with random forests over single trees is that the `randomForest` R package includes a `partialPlot` function that creates partial dependence plots. To understand how Petal Length is related to Sepal Length, averaging across the other two predictors, I made the partial dependence plot seen in Figure 1. I then calculated partial dependences across a grid of Petal Length values and added those values to the plot. The alignment of points and line implies that in this case, at least, the naive algorithm succeeds.

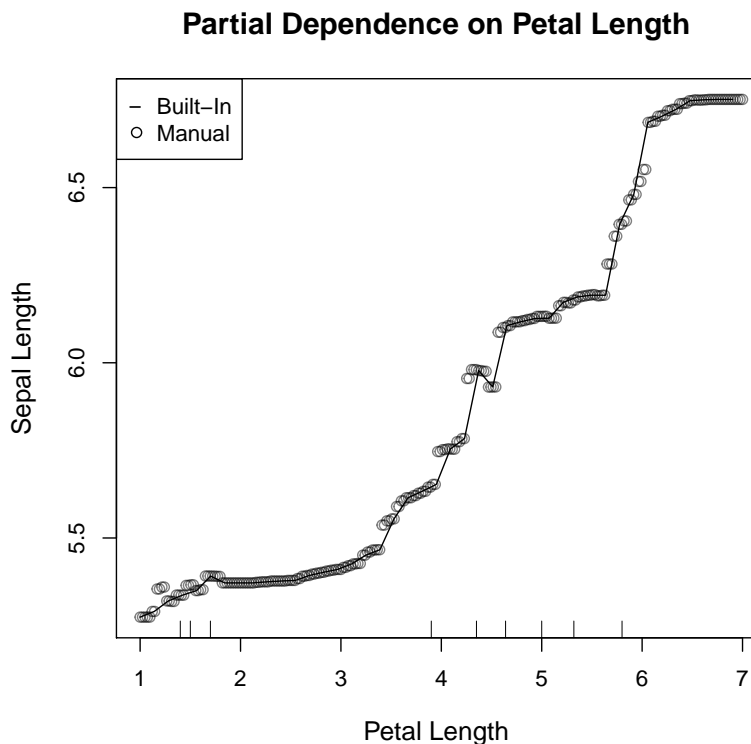
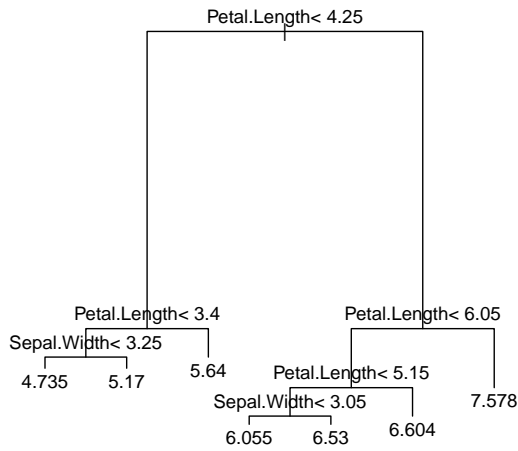


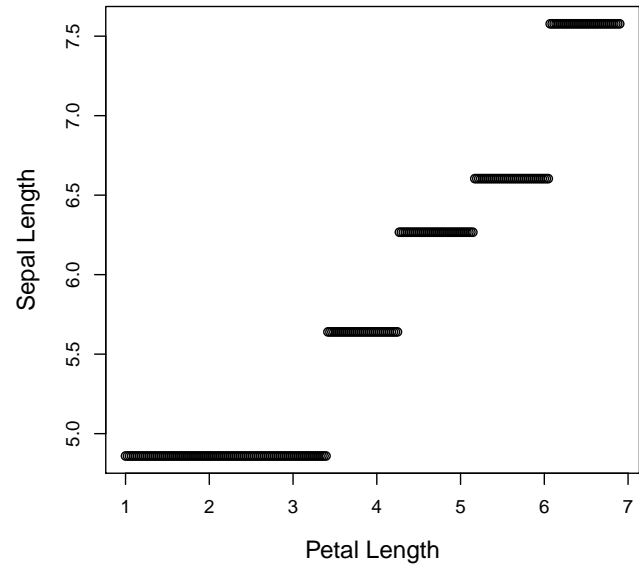
Figure 1: Partial dependence plot from the random forest regression of Sepal Length on Petal Length, the most informative predictor. As we'd expect, the two values are positively associated. Partial dependence values calculated manually are superimposed.

The question asked about classification trees, though, so I took my naive algorithm to a CART framework and grew the classification tree for this problem (Figure 2a). Given the small number of predictors, it's easy to see that indeed, Petal Length is the most important predictor. Note also that Sepal Width is never used to make a split at all. These facts are reflected in their partial dependence plots (Figures 2b-d). More importantly, note that in all cases, the functions are piecewise linear. Thus, by determining only the cut points, i.e. the values of predictor(s) X_S where the region changes, an efficient algorithm could compute partial dependencies very efficiently.

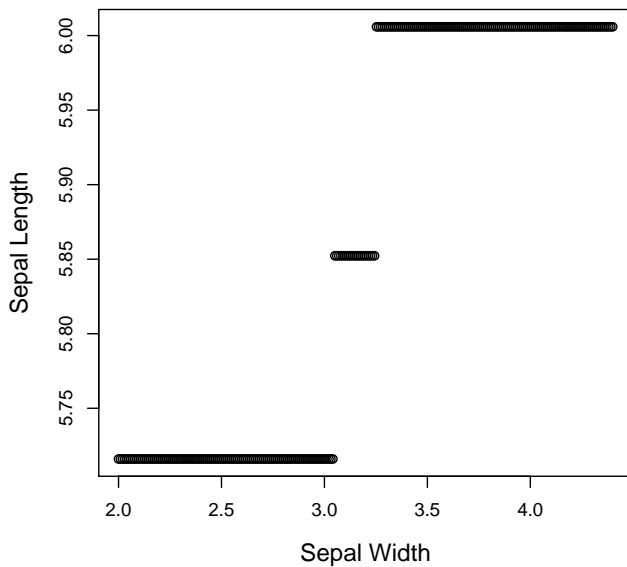
CART for Iris Data: Predict Sepal Length



Partial Dependence Plot: Petal Length



Partial Dependence Plot: Sepal Width



Partial Dependence Plot: Sepal Width

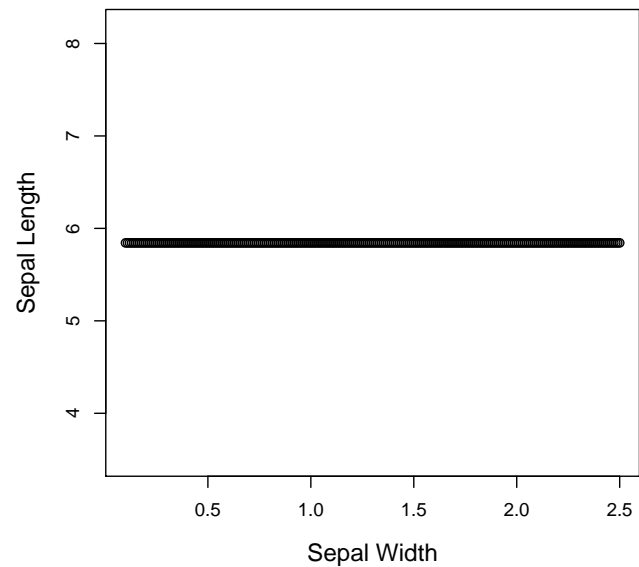


Figure 2: Exploring partial dependence plots in Fisher’s Iris data. (a) The tree with Sepal Length as response. (b-d) Partial dependence plots for the three predictors. In all cases, the plots are piecewise linear; by determining the cut points, an efficient partial dependence function algorithm could save much computation time.

2 Random Effects Modeling

2.1 Introduction

The data being analyzed in this problem come from an experiment conducted in rats, investigating the effect of a drug called carbachol on nucleotide activation in various regions of the brain. A saline solution was used as a control to obtain basal values. Though data were initially collected for seven brain regions, only three regions are considered here.

From an experimental design perspective, there are three factors in this experiment. The fixed effects are Treatment $\in \{\text{Basal}, \text{Carb}\}$ and Region $\in \{\text{BST}, \text{LS}, \text{VDB}\}$. Region is an unordered categorical variable. These are fixed effects because they're the true levels of the variables we're interested in learning more about. The third factor, Rat, is random. There are 5 levels of this variable, i.e. 5 rats were used in this experiment. This is a random effect because we don't care about these 5 rats specifically, but rather the population of all rats (and probably other species as well, though generalizing findings across species is a whole separate issue.) We have one measurement of nucleotide activation for each (Rat, Treatment, Region) combination, giving $5 \times 3 \times 2 = 30$ observations total. It's assumed that rats are independent of one another.

2.2 Exploratory Data Analysis

To get a quick feel for the data, I made dot plots, splitting on both fixed effects. Figure 3 preliminarily suggests that carbachol increases nucleotide activation in all three regions considered. We can also see that variability between rats increases when taking carbachol versus the control. Though this plot hints at the interactions between Treatment and Region, I think this effect is more apparent in Figure 4, where the differences between carbachol and control clearly depend on which region is being considered. Running an ANOVA with fixed effects only confirmed that the interaction effect between Region and Treatment is significant ($p = 0.008$, full table calculated but not given).

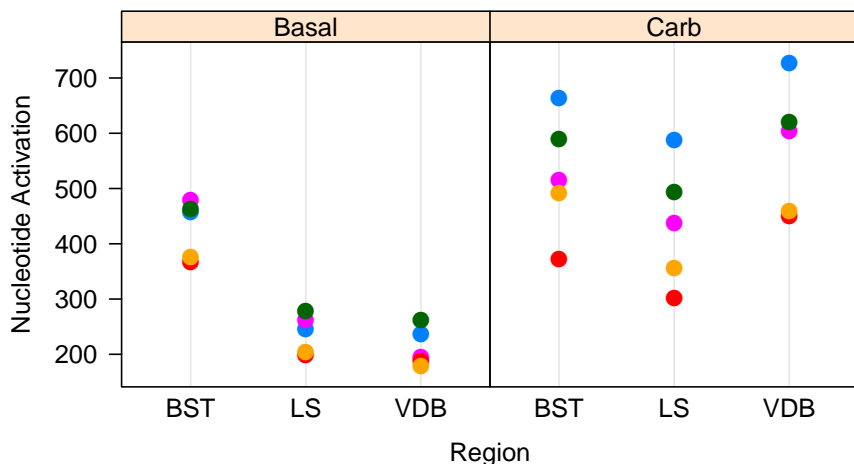


Figure 3: Split plot examining the effects of Treatment by Region. Carbachol appears to increase nucleotide activation in all three regions, though nucleotide activation values are noisier in the presence of carbachol.

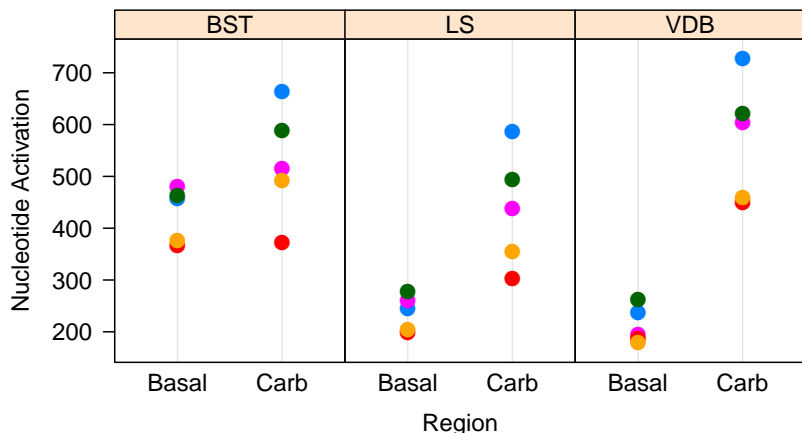


Figure 4: Split plot examining the effects of Region by Treatment. Interaction effects are clearly present here: to talk about the effect of carbachol without taking region into account is impossible.

2.3 Models

As noted earlier, each Activation value is indexed by a combination of (Rat, Treatment, Region). So let y_{ijk} be the Activation value from rat i , Treatment j , and Region k , where $i \in \{1, \dots, 5\}$, $j \in \{1, 2\}$, and $k \in \{1, 2, 3\}$.

Model 1 fits a first-order fixed effects interaction with a random intercept for each rat. Thus, its mathematical form is as follows:

$$\begin{aligned}
 y_{ijk} = & \mu + \beta_C \mathbb{I}(\text{Treatment} = \text{Carbachol}) + \beta_1 \mathbb{I}(\text{Region} = \text{LS}) + \beta_2 \mathbb{I}(\text{Region} = \text{VDB}) \\
 & + \beta_{1C} \mathbb{I}(\text{Region} = \text{LS}, \text{Treatment} = \text{Carbachol}) \\
 & + \beta_{2C} \mathbb{I}(\text{Region} = \text{VDB}, \text{Treatment} = \text{Carbachol}) + \gamma_0^i + \epsilon_{ijk},
 \end{aligned}$$

where $\mathbb{I}(\cdot)$ is an indicator variable, μ is the intercept common to all rats, $\beta = (\beta_C, \beta_1, \beta_2, \beta_{1C}, \beta_{2C})^T$ are the effects of their corresponding (Treatment, Region) levels common to all rats, γ_0^i is the random intercept for rat i , and ϵ_{ijk} is a random error term specific to each observation. Excluding ϵ terms, there are 5 random terms implied by this model: the five γ_0^i values.

Model 2 fits the same first-order fixed effects interaction as Model 1 did. In addition to fitting a random intercept for each rat, Model 2 also includes a random interaction term between rat and treatment. This model takes the following mathematical form:

$$\begin{aligned}
 y_{ijk} = & \mu + \beta_C \mathbb{I}(\text{Treatment} = \text{Carbachol}) + \beta_1 \mathbb{I}(\text{Region} = \text{LS}) + \beta_2 \mathbb{I}(\text{Region} = \text{VDB}) \\
 & + \beta_{1C} \mathbb{I}(\text{Region} = \text{LS}, \text{Treatment} = \text{Carbachol}) \\
 & + \beta_{2C} \mathbb{I}(\text{Region} = \text{VDB}, \text{Treatment} = \text{Carbachol}) + \gamma_0^i + \gamma_1^{ij} + \epsilon_{ijk},
 \end{aligned}$$

All repeated variables have the same meaning that they did in Model 1, and γ_1^{ij} represents the specific interaction between each rat i and each treatment j , for all i, j . Excluding ϵ terms, there are 15 random terms implied by this model: five γ_0^i values and ten γ_1^{ij} values.

Model 3 also keeps the same first-order fixed effects interaction as Model 1 did but fits a random intercept for each rat as well as a random carbachol effect for each rat. This model takes the following form:

$$\begin{aligned}
 y_{ijk} = & \mu + \beta_C \mathbb{I}(\text{Treatment} = \text{Carbachol}) + \beta_1 \mathbb{I}(\text{Region} = \text{LS}) + \beta_2 \mathbb{I}(\text{Region} = \text{VDB}) \\
 & + \beta_{1C} \mathbb{I}(\text{Region} = \text{LS}, \text{Treatment} = \text{Carbachol}) \\
 & + \beta_{2C} \mathbb{I}(\text{Region} = \text{VDB}, \text{Treatment} = \text{Carbachol}) + \gamma_0^i + \gamma_1^{i, \text{CARB}} + \epsilon_{ijk},
 \end{aligned}$$

where all terms defined in previous models remain the same and $\gamma_1^{i,CARB}$ is the random effect experienced by rat i of being treated with carbachol. Excluding ϵ terms, there are 10 random terms implied by this model: 5 γ_0^i values and 5 $\gamma_1^{i,CARB}$ values.

The fixed effects in all three models are the same and are the usual fixed effects when the factors are outcomes: coefficients corresponding to the effect of carbachol (versus basal), the effects of the two non-BST regions (versus BST), and two interaction effects. Based on Figures 3 and 4, these seem reasonable. After fitting a fixed-effect-only model, examining the residuals by subject should indicate what kinds of random effects will be useful or extraneous. These residual plots are given in Figure 5.

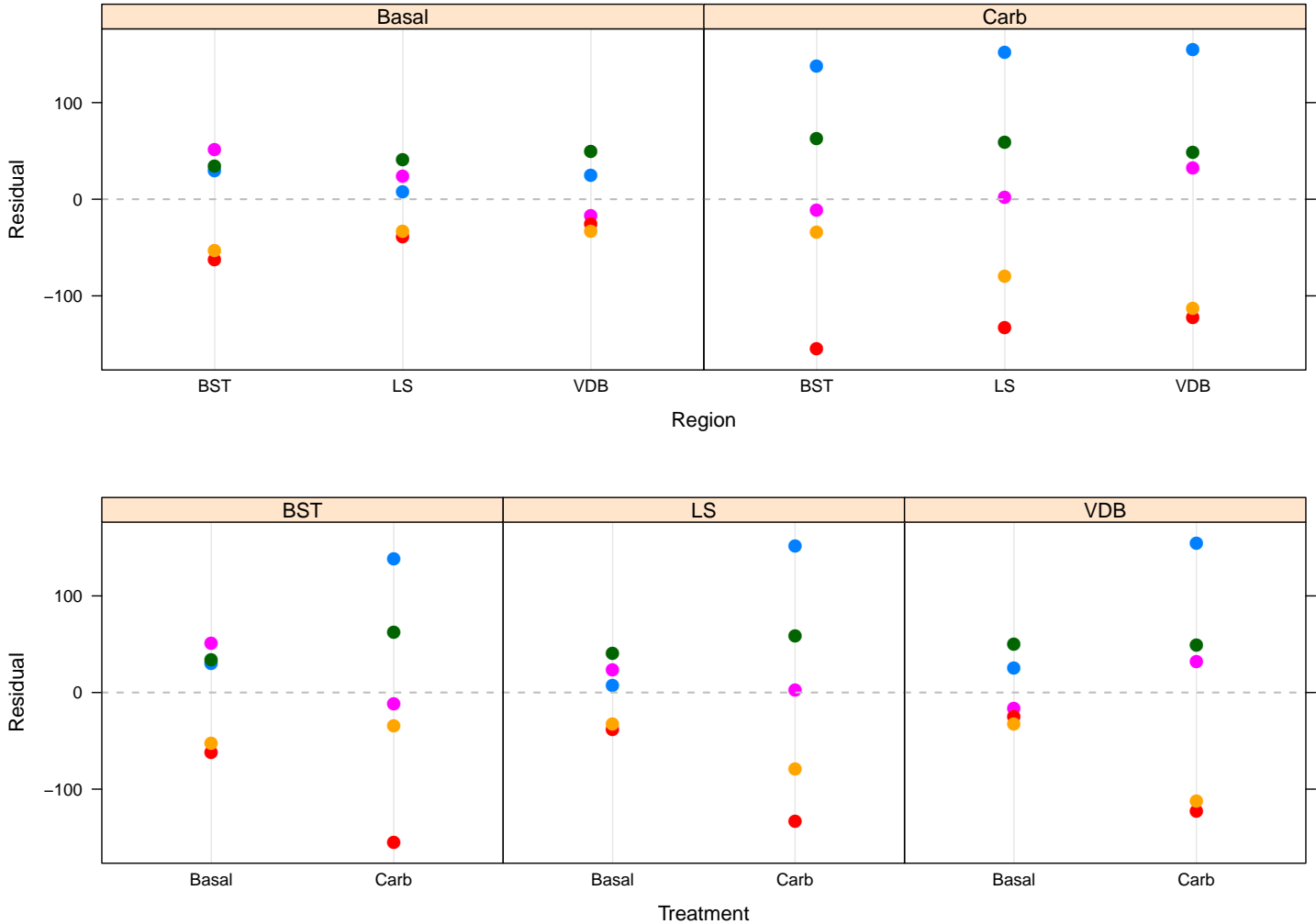


Figure 5: By-rat residual plots. (a) The fixed effects model appears to capture the effect of region adequately, as trends across regions are mostly unchanging for each rat. (b) Non-constant residuals here suggest that random effects to account for individual responses to carbachol will improve modeling.

Figure 5a, with its flat but shifted by-rat residuals, suggests that the fixed effects model adequately captured the effects of Region on Nucleotide Activation, though a random intercept term for each rat will account for the vertical shifts. The fixed effects do not appear to have captured all effects of carbachol, as indicated by the differences between Basal and Carb for each rat in Figure 5b. A random effect term for carbachol should improve the ability of our model to capture variability in Nucleotide Activation.

Comparing these thoughts to the three models given, Model 1 is a step in the right direction. It fits the random intercept for each rat, but will be unable to account for the specific reactions of each rat to taking Carbochol. Model 3, which includes both the random intercept and the $\gamma_1^{i,CARB}$ that is specific to each rat and applies only when the rat is given Carbochol, seems like a very reasonable model for these data. Model 2, which I believe is the model not explicitly given in the chapter, seems like overkill: it fits a random intercept and fits an additional effect term for each (Rat, Treatment) combination. Since Treatment only has two levels, however, this would seem to fit 5 unnecessary variables; the estimates of γ_1^{ij} when $j = \text{Control}$ should be able to be absorbed into the γ_0^i term. There is a slight difference between the fits of Model 2 and Model 3, though the residuals have a correlation of 0.95 (calculated and plotted but not shown), so these models are quite similar. (Model 1 and 3 residuals have a correlation of 0.52, for comparison.)

2.4 A Simpler Alternative?

Based on my discussion in the previous section, Model 3 does seem like a reasonable model for the problem at hand. After fitting the Model 3 random effects, the by-subject residuals look essentially like noise centered around 0, as seen in Figure 6. This suggests that most relevant information has been captured by this model.

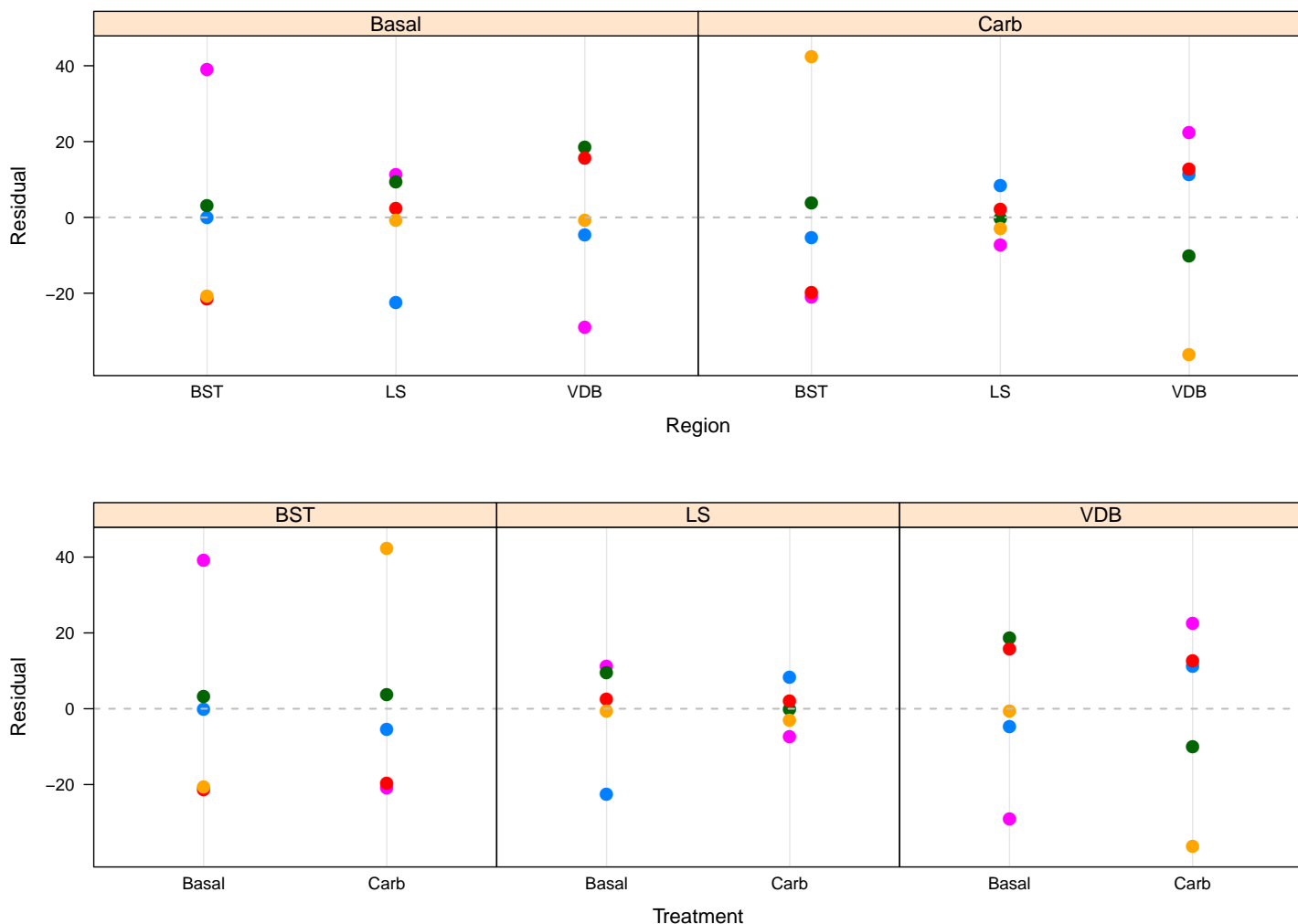


Figure 6: Residuals, color-coded by rat, after fitting Model 3. Though the number of x-values is small in both plots, the residuals appear to be approximately noise around 0.

Looking at Figure 5, it seems to me that a random slope and a coefficient for the effect of carbachol are good ideas. In terms of simplifying the random effects machinery, fitting only a random slope for each rat is insufficient, as indicated from the obvious trends in the residuals from Model 1 (calculated and plotted but not included). Having only a random intercept and individual effect for Carbachol is already pretty simple, appears to capture the trends in the data well, and makes sense intuitively. I see no reason to simplify it further.

3 Experimental Design & Optimization

3.1 Overview

In the first 215B lab report, we considered an industrial experiment run by two statisticians, Prat and Tort. They were hired by a rabbit food manufacturer to optimize the process of making rabbit pellets, and they conducted a $3/4 2^4$ experiment to obtain data about the process. The manufacturer's main goal was to make pellets that were less likely to crumble into powder before reaching the customer, but they were also interested in maintaining or improving process yield while minimizing both processing costs and the amount of product reduced to powder during the manufacturing process itself. Working with engineers, Prat and Tort considered four factors with two levels each and ran 12 experiments, measuring four outcome variables in each experiment [2]. The four factors and four outcome variables are summarized in Table 1. Though the Low and High levels had real meanings in the experiment conducted, they were treated as -1 and 1 values, respectively, when Prat and Tort conducted their analysis and fit the coefficients used in this problem. I therefore reparameterized to ± 1 in Table 1.

Factors				Responses	
Variable	Description	Low	High	Variable	Description
X1	Amount of Glue	-1	1	Y1	Powder at Plant
X2	Conditioning Temperature	-1	1	Y2	Powder at Customer
X3	Flow	-1	1	Y3	Yield
X4	Compression Zone in Die	-1	1	Y4	Cost

Table 1: Rabbit food experiment variables. The factor low and high levels had actual interpretations in terms of units of variables, but their ranges can be considered $[-1,1]$ for the purposes of this question.

3.2 Analytic Optimization

Analyzing their data, Prat and Tort produced the following models:

$$Y_1 = 1.109 - .038X_2 - .095X_4$$

$$Y_2 = 1.885 + .11X_1$$

$$Y_3 = 8.19 - 1.017X_4$$

$$Y_4 = 240 - 4.07X_2 + 10.87X_3 + 2.23X_2X_3$$

Assuming that all X_i values can be set to any arbitrary amount within the range tested in this experiment, which is $[-1,1]$ as noted previously, the following optimization problem is proposed:

$$\begin{aligned} \text{minimize: } & Y_4 = 240 - 4.07X_2 + 10.87X_3 + 2.23X_2X_3 \\ \text{subject to: } & Y_1 = 1.109 - .038X_2 - .095X_4 \leq 1.1 \\ & Y_3 = 8.19 - 1.017X_4 \geq 8.5 \\ & X_2, X_3, X_4 \in [-1, 1] \end{aligned}$$

The second condition implies that we need $X_4 \leq \frac{-0.31}{1.017}$. Thus, the problem becomes:

$$\begin{aligned} \text{minimize: } & Y_4 = 240 - 4.07X_2 + 10.87X_3 + 2.23X_2X_3 \\ \text{subject to: } & Y_1 = 1.109 - .038X_2 - .095X_4 \leq 1.1 \\ & X_2, X_3, \in [-1, 1], \quad X_4 \in \left[-1, \frac{-0.31}{1.017}\right] \end{aligned}$$

Looking at the expression to be minimized, specifically the signs, we can clearly see that this expression will be minimized when X_2 is as positive as possible and X_3 is as negative as possible, thus making X_2X_3 as negative as possible. Since there are no other constraints on X_3 , choose $\mathbf{X}_3 = -\mathbf{1}$.

Since there are other constraints on X_2 , we need to consider how positive X_2 can be. Working with the first constraint, we have

$$\begin{aligned} 1.109 - .038X_2 - .095X_4 &\leq 1.1 \\ \Leftrightarrow - .038X_2 - .095X_4 &\leq -0.009 \\ \Leftrightarrow X_2 &\geq \frac{.009}{.038} - \frac{.095}{.038}X_4 \end{aligned}$$

Choosing X_4 at the top of its adjusted range, i.e. $X_4 = \frac{-0.31}{1.017}$, we need $X_2 \geq 0.998887$. But we wanted X_2 as positive as possible, so choosing $\mathbf{X}_2 = \mathbf{1}$ is a valid choice for this X_4 and will minimize the cost function. Checking that $(\mathbf{X}_2^*, \mathbf{X}_3^*, \mathbf{X}_4^*) = (\mathbf{1}, -\mathbf{1}, \frac{-0.31}{1.017})$ is a solution, we have:

$$\begin{aligned} \text{Powder in Product} &= 1.109 - .038X_2^* - .095X_4^* \\ &= 1.099958 \leq 1.1 \\ \text{Yield} &= 8.19 - 1.017X_4^* \\ &= 8.5 \geq 8.5 \\ \text{Cost} &= 240 - 4.07X_2^* + 10.87X_3^* + 2.23X_2^*X_3^* \\ &= 222.83 \end{aligned}$$

And Cost is clearly minimized, since we chose the best values for X_2 and X_3 . (Great problem!)

3.3 Sensitivity

It is correctly noted that the coefficients used in the above optimization were produced from carefully-set X_i values, and that the same level of care might not be taken when setting X_i values on a day-to-day basis. By considering the above optimization procedure, we can examine how sensitive the various conditions are to variations in X_i values.

X_3 is easiest to consider, since it's only involved in the expression for Cost. Assuming that the operator is able to make X_2 positive (which is reasonable, since we found that X_2 should be as positive as possible), and that X_3 can't be set lower than -1 (assumed by the problem), X_3 slightly larger than -1 will simply increase the Cost. Though this will void the goal of the exercise, the constraints Yield and Powder in Product won't be affected.

There's a tiny margin for error associated with X_4 , however. The simultaneous constraints

$$-1 \leq X_4 \leq \frac{-0.31}{1.017} \quad \text{and} \quad \left(\frac{.009}{.038} - \frac{.095}{.038}X_4 \right) \leq X_2 \leq 1$$

force $-\frac{.029}{.095} \leq X_4 \leq -\frac{.31}{1.017}$, which is $X_4 \in [-0.30526, -0.30482]$ to five decimal places. If X_4 is above this interval, then Yield will fall below the prescribed value of 8.5. If X_4 is below this interval, then no choice for $X_2 \in [-1, 1]$ will exist to keep Powder in Plant below 1.1. Thankfully, X_4 represents the compression length of the die used in extrusion, which seems like something that can be measured and calibrated quite precisely. The engineers at the plant also mentioned that this value was difficult to change, so perhaps once it was set carefully, it could be monitored occasionally but remain constantly within this slim interval.

3.4 Alternative Optimization

In both previous sections, it was assumed that the coefficients relating the X_i to the Y_i given at the beginning of Section 3.2 are the correct values for the linear models. Of course, these are only fitted values based on a relatively small number of observations, and there are errors associated with these values.

The best idea I had for finding optimal values given probable variability in the coefficients was to put bounds on the probability that the criteria aren't met. For example, the problem could become, for some small α :

$$\begin{aligned} \text{minimize:} & \quad \text{Cost} \\ \text{subject to:} & \quad \mathbb{P}[\text{Powder in Product} \leq 1.1] \geq 1 - \alpha \\ & \quad \mathbb{P}[\text{Yield} \geq 8.5] \geq 1 - \alpha \\ & \quad X_2, X_3, X_4 \in [-1, 1] \end{aligned}$$

We could assume that the linear models specified by Prat and Tort are correct but that the coefficients are random variables instead of fixed numeric values. Then Powder in Product and Yield, sums of random variables, will themselves be random, and we can try to bound the probability of obtaining an extreme unacceptable value. Assuming that the X_i can be set consistently and are therefore not random would simplify matters.

For distributional information about the linear coefficients, one could appeal to asymptotics, though that would be questionable in this case, given the small sample size and the difficulties in obtaining new data. A more reasonable approach might be to find a way to bootstrap means, variances, and/or confidence intervals for the coefficients based on available data, then construct bounds using those estimated quantities.

4 Effects of Class Sizes on Skull Classification

4.1 Introduction

In Lab 4, we built models to classify the time period of Egyptian male human skulls using measurements of various skull dimensions as features. The data and features are originally given and described in [3] and [4]. Classifiers used in Lab 4 included linear discriminant analysis (LDA), Fisher’s discriminant analysis (FDA), and support vector machines (SVM). There were four time periods, and multiclass issues with SVM were handled using a One Versus One strategy [5]. Comparable LDA analyses were performed. Of the three classifiers, SVM far outperformed the discriminant analyses, though at a greatly increased computational cost.

In my analysis of classifier performances, I noted that many of the classification mistakes made by all algorithms involved incorrectly assigning a skull to Period II. Indeed, considering my results of LDA and SVM in the four-class problem from Lab 4 (Table 2), we see that 50%, 51%, and 50%, of misclassifications were of this type, respectively. In Lab 4, I hypothesized that this occurred because Period II has a significantly larger class size: class sizes are 88, 139, 70, and 64 for Periods I, II, III, and IV. I didn’t have time to explore this artifact in Lab 4, but I do now!

Multiclass LDA					One Versus One LDA					One Versus One SVM				
Truth \ Pred.	I	II	III	IV	Truth \ Pred.	I	II	III	IV	Truth \ Pred.	I	II	III	IV
I	26	40	12	10	I	22	43	10	13	I	47	30	5	6
II	12	101	12	14	II	17	97	11	14	II	14	110	5	10
III	16	37	11	6	III	13	39	11	7	III	12	26	27	5
IV	15	23	5	21	IV	15	25	4	20	IV	13	16	3	32

Table 2: Confusion matrices for the classification of all skulls using two types of LDA and SVM, reproduced from my Lab 4 report. For all classifiers, about half of misclassifications involved incorrectly assigning a skull to Period II. This phenomenon is due to Period II containing almost 40% of all skulls in the sample.

In this question, I attempt to answer two questions about the skull data:

1. How are overall classification errors affected by skewed class distribution?
2. How are the distributions of misclassifications affected by skewed class distribution?

After a brief description of the classifiers used, I describe the initial analysis I performed to answer these questions. Unfortunately, my ability to answer Question 1 was hampered by poor initial choices of resampling schemes, so I performed additional resamplings to remedy the matter.

4.2 Classifiers

Three classifiers were considered: (multiclass) LDA, CART, and random forests (RFs). LDA is a simple linear classifier which assumes that the class densities are multivariate Normal with separate mean vectors but the same covariance matrix. As discussed in Problem 1, CART grows a classification tree that partitions the feature space into disjoint regions of constant classification. The random forest algorithm reduces the instabilities of CART trees by growing many partially de-correlated classification trees and aggregating across the forest. Random forests are more computationally intensive than LDA and CART, but they’re also stronger classifiers.

Accurately assessing prediction error is always a consideration in classification settings. The most popular method for estimating prediction error is cross-validation. In K -fold cross-validation, the data are randomly partitioned into K approximately equally-sized groups. For each partition k ($k \in 1, \dots, K$), all data except partition k are used to train the classifier. Then the observations in partition k , which the classifier didn’t get to see, are classified. This is performed for each partition, and the errors are combined. By separating the points used to build the classifier and the points being classified, more correct estimates of prediction

error are obtained. Cross-validation should also be used when choosing tuning parameters for classifiers, much for the same reason: the tuning parameter values used when assessing prediction error on a ‘fresh’ subset of the data shouldn’t be based on observations on the fresh observations.

Random forest has cross-validation element built into it. Each tree is grown from a sample *with replacement* from all observations. Once the tree is grown, those observations not selected (the “out-of-bag data”) are classified by the tree, exactly the same idea as cross-validation. These classifications provide unbiased of estimated error. On average, about one-third of observations will be out-of-bag (actually e^{-1}), so this mechanism is approximately equal to 3-fold cross-validation. So that results across classifiers would be comparable, I used 3-fold CV for LDA and CART. Though the number of variables randomly selected as candidates for each node of each tree in a random forest is technically a parameter that should be tuned via CV, I used the recommended value of \sqrt{p} [6]. The cost parameters for all CART trees grown were selected via 3-fold CV. All random forests were grown with 1,000 trees.

4.3 Sampling Schemes, First Attempt

My initial resampling schemes were as follows:

1. **No resampling (NR)**. Use all observations as is, for comparison purposes. Class sizes are most skewed in this setting, given that as much data as possible are to be included.
2. **Less skewed resampling (LSR)**. Move towards more equal class sizes by reducing the number of skulls in Period i by $(n_i - 64)/2$, $i = 1, 2, 3$, where n_i is the number of skulls in Period i and $n_4 = 64$. This halves the discrepancies in class sizes, relative to the smallest class.
3. **Equal size resampling (ESR)**. The smallest class, Period IV, has 64 skulls in it. Randomly draw subsets of size 64 from the other periods, then run classifiers on these 256 skulls.
4. **Reduced equal size resampling (RESR)**. To inject some randomness into Period IV, randomly draw subsets of size 50 from all periods, then run classifiers on these 200 skulls.

The sample sizes produced by these resampling schemes are given in Table 3, where m_i is the class size for Period i under the corresponding resampling scheme and numbers in bold entail resampling. In instances where resampling was involved ($m_i < n_i$), I performed 50 resamplings. Even in instances where the resampling scheme called for using all observations, I still performed the technique 50 times and averaged the results. This was to average across various partitions of the data obtained when performing cross-validation.

Resampling Scheme	m_1	m_2	m_3	m_4
NR	88	139	70	64
LSR	76	101	67	64
ESR	64	64	64	64
RESR	50	50	50	50

Table 3: Class sizes under the various class size schemes considered. Bold values imply resampling, and 50 resamplings were performed in these cases. In cases where $m_i = n_i$, 50 partitions of the data were considered when cross-validating.

For each (sampling scheme, classifier) combination, I calculated an average cross-validated classification error rate. These error rates are given in Figure 7a. All three classifiers show a slightly increasing error rate as the class sizes become more equal, though **this is confounded by decreasing amounts of data**: in the NR setting, there are 361 skulls total, while the RESR setting has only 200 skulls. As soon as I saw this plot, I realized that I should have designed the various resampling schemes to keep the total number of skulls being considered constant and change only the distributions of class sizes; I address this situation in the next section. Regardless, the decreasing amount of data is consistent across classifiers, and we see that CART is the weakest classifier, with LDA and RF performing comparably. That LDA actually beats RF in the RESR setting is interesting.

Given the cross-validated confusion matrix for each (sampling scheme, classifier) combination, I calculated the proportion of incorrectly classified skulls assigned to each class. For example, I noted earlier than in the confusion matrices given in Table 2, about half of the misclassifications involved incorrectly assigning a skull to Period II. These distributions for LDA, CART, and RF are given in Figures 7b, 7c, and 7d, respectively, with the dotted lines representing the actual proportion of each class in the sample. Interestingly, each classifier has a unique pattern: LDA misclassification distributions converge nicely to approximate equality for ESR and RESR, while the random forest has Period II crossing over Periods I and IV for ESR and RESR. CART was most variable in the NR setting, and we see that, unlike LDA and RF, there is a noticeable change in distribution between ESR and RESR. In all cases, the skew of class sizes is exaggerated in terms of proportion misclassified.

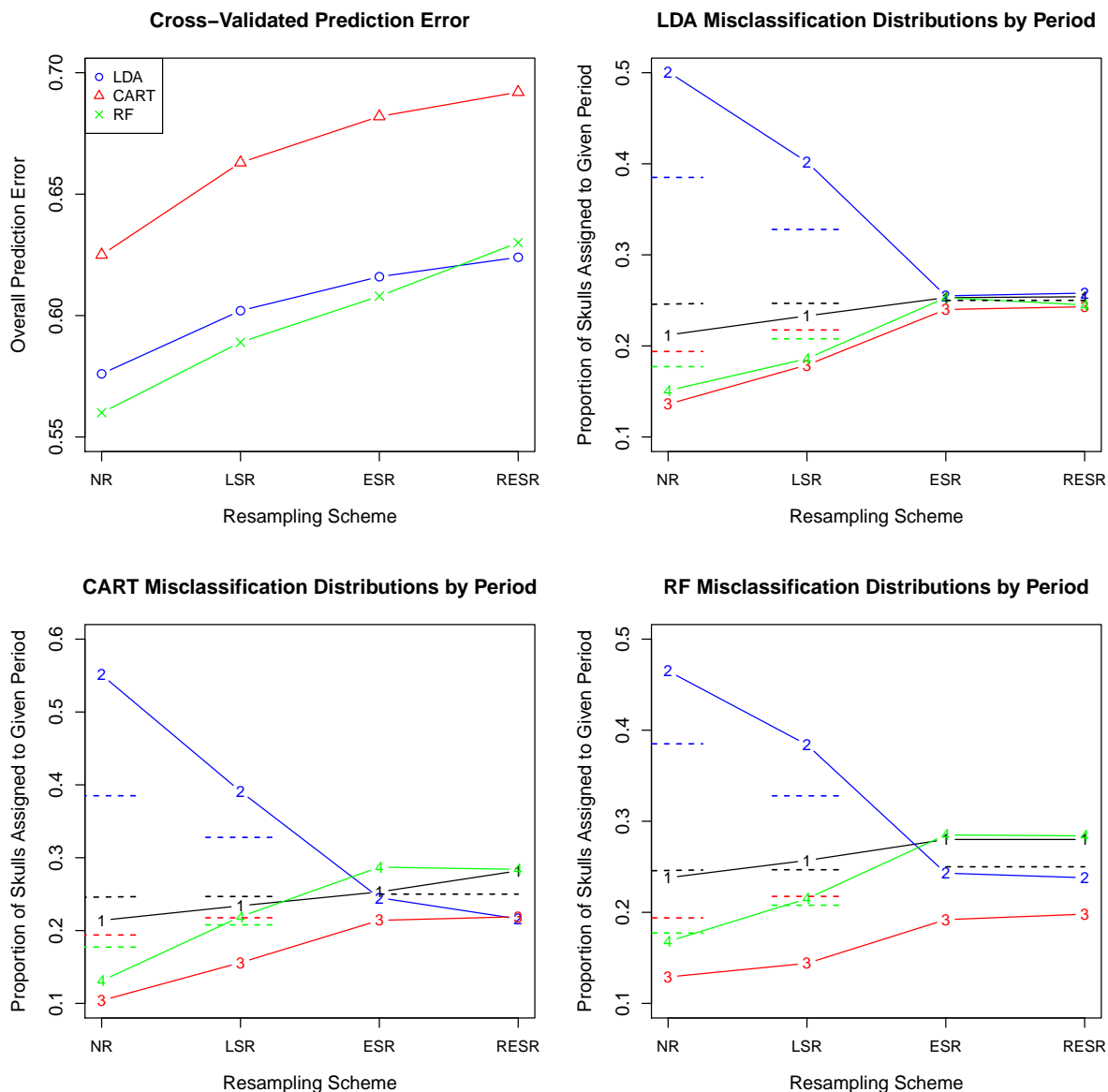


Figure 7: Classification accuracies and distributions of misclassifications from initial resamplings. (a) LDA and RF outperform CART significantly in terms of raw classification. The increasing error is misleading, however. (b-d) Proportion misclassified to each group for each classifier, with dotted lines representing the true proportion of each class in the sample. In all cases, the skew of class sizes is exaggerated in terms of proportion misclassified.

4.4 Sampling Schemes, Second Attempt

When analyzing the results in the previous section, I noted that classification error was increasing as the distribution of class sizes became less skewed, but that this was probably an artifact of the decreasing overall sample size implied by my initial design. To follow up this idea, I designed a new resampling strategy that kept the total number of skulls constant across schemes. These schemes are given in Table 4. Note that the total number of skulls available for the classes limited my choices for which period could be emphasized in the OneSkew and TwoSkew settings: for example, only Period II had more than 120 skulls, and Period I having less than 90 meant using 85 instead in the TwoSkew setting.

Resampling Scheme	m_1	m_2	m_3	m_4	Total
Equal	60	60	60	60	240
OneSkew	40	120	40	40	240
TwoSkew	85	85	35	35	240

Table 4: Class sizes under additional resampling schemes, where the total number of skulls in each scheme is the same.

For each resampling scheme, I again ran cross-validated LDA, CART, and RF. 50 resamplings were performed for each (scheme, classifier) combination. Prediction errors are shown in Figure 8a. We see that when the total number of skulls in each resampling scheme is kept constant, the trend changes: OneSkew error rates are consistently lowest, while Equal error rates are slightly higher than TwoSkew error rates. This makes sense: given that half of the observations are in one class in OneSkew, a classifier could classify everything as belonging to Period II and obtain a 50% accuracy. In Equal and TwoSkew, no such simple option exists. Random forests are again a slightly better predictor than LDA, and CART isn't as relatively bad as it was previously. The patterns of misclassification for the three classifiers in the equal-class-sizes case are even more similar than they were previously, with equal distribution in the Equal setting, a huge preference towards Period II when half the skulls are a member, and split preference when two periods are dominant.

4.5 Conclusions

This was an interesting follow-up to the 'overloading of Period II' phenomenon I noticed during Lab 4. The large number of misclassifications into Period II does, in fact, seem to have been due to the dominating proportion of that class, as seen Figures 7b-d and Figures 8b-d. The effects tend to be exaggerated, too: in all of these figures, Period II's proportion of misclassifications is larger than its proportion of skulls, with a proportion of 0.5 in the OneSkew setting entailing 70-80% of misclassifications in the OneSkew setting. In both resampling strategies I used, random forests emerged as the strongest classifier, with LDA slightly behind and single classification trees trailing. Though this is generally believed to be the case, it was reassuring to see that this hierarchy is maintained across a few data distributions.

The behavior of classifiers when class distributions are skewed is an important behavior to understand. I think first of medical situations, where the low prevalence of a rare disease means that the number of controls far exceeds the number of cases available, though most datasets will have skewed distributions to a certain extent. Based on the very preliminary results seen here, it seems that stronger classifiers perform consistently better than weaker classifiers across a variety of class distributions, and that imbalances between class sizes are exaggerated in terms of misclassification proportions. Examining this effect in different datasets and with different classifiers would be fruitful.

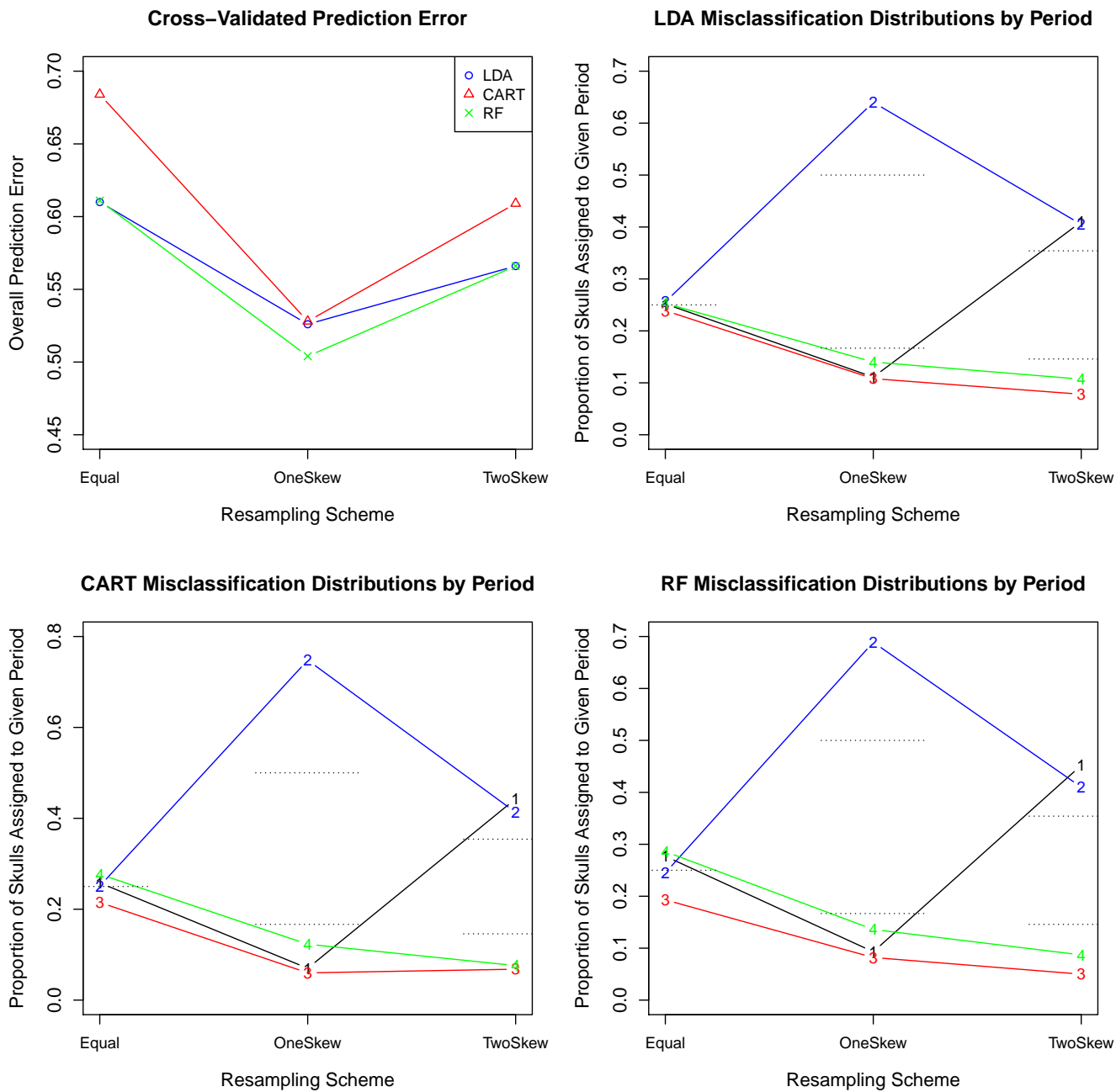


Figure 8: Classification accuracies and distributions of misclassifications. (a) The increasing trend seen previously disappears when equal amounts of data are used in all resampling schemes. Classifiers obtain lowest error when half the data belong to one class, a la OneSkew. (b-d) Almost identical misclassification distributions for the classifiers, where performance was as would be expected.

References

- [1] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.
- [2] A. Prat and X. Tort. Case study: Experimental design in a pet food manufacturing company. Technical Report 37, University of Madison-Wisconsin, 1989.
- [3] A. Thomas and R. Randall-Maciver. *Ancient Races of the Thebaid*. Oxford, 1905.
- [4] M. M. Barnard. The secular variations of skull characters in four series of Egyptian skulls. *Annals of Eugenics*, 6(4), 1935.
- [5] K. Duan and S. S. Keerthi. Which is the best multiclass SVM method? An empirical study. Technical report, Proceedings of the Sixth International Workshop on Multiple Classifier Systems, 2005.
- [6] L. Breiman and A. Cutler. Random Forests. http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Online; accessed May 6 2011].