Reducing Experimenter Burden in Mechanism of Action Determination Experiments

Jacob Tiede

Introduction/Background

Toward the mid-1900s it had become increasingly important in the world of medicine to have a biological basis for the treatments given. With the flourishing study of physiology and the discarding of outdated systems such as humoral medicine [13], it became a necessity that we have a rational and evidence for how our treatments affected the body. In the study of pharmacology this understanding came in the form of the study of mechanisms of action (MOAs) of compounds. This branch of academia revolves, essentially, around how to determine how a particular compound affects a population of cells. The primary problem with making such a determination is that these experiments can be extremely complex due to the number of parameters that must be studied separately [4]. A reduction of the number of experimental parameters has the potential to significantly reduce the burden on experimenters seeking to better understand MOAs. This is the topic of this paper: finding a reduced set of predictors that will not be detrimental to prediction of MOAs, but first we will introduce the study of MOAs in depth as well as further motivate the need for reducing complexity in these experiments.

A natural question that an empiricist might ask is why does the biological action of a molecule matter? Why not just do experiments and show that the compound has the intended effect? A useful example for extrapolating clinically useful information from MOAs is with antihistamines. These are a class of compounds that all affect the histamine system in the body, usually by blocking histamine receptors on cell surfaces [1] with the goal of reducing allergy symptoms. Histamine has a well-known role in local immune response, so inhibiting its action will reduce allergy symptoms [3], but because we know this is the MOA of these compounds, we can also extrapolate potential side effects. Histamine also has a role as a neurotransmitter [3] which can affect wakefulness (which explains why most antihistamines have drowsiness as a side effect [2]). In isolation it may be very hard to determine if drowsiness is causally related to antihistamine use, but with the MOA this connection is quite natural. This explains why the study of MOAs has become such a focus in pharmacological science, because we can use this information to extrapolate many properties about these compounds, vastly beyond what one could reasonably expect from a single experiment to gauge efficacy.

The main experiment type used to study MOAs are ‘microscopy-based methods’ [4]. These broadly involve looking at two populations of cells which are kept as similar to each other as possible. One population is exposed to the compound which we would like to determine the MOA, and the other is exposed to a control ‘vehicle’ (most compounds require a shell to be able to diffuse into cells, so a control vehicle is this shell, but with no compound inside). Since cells are quite complex many markers are recorded. These markers fit into two broad categories: gene traits and cell traits. Gene traits are data related to what genes are being expressed, and cell traits typically have to do with the ‘well-being’ of the cell. To determine the MOA a researcher may need to record hundreds or thousands of these variables [4], and to make accurate predictions about what the MOA is the researcher would need to replicate their experiments many times to get an idea of how each marker reacts to different compounds. However, we do have some power to potentially ween the number of parameters recorded down to a more manageable level using statistical methods.

There is a lot of biological reasons to believe that both gene and cell traits may have correlations with other gene and cell traits. These can range from direct known causal relationships to each other, to less well-known correlations that may reduce the utility of various experimental parameters. For example, [5] gives details on at least one potential collinearity in the gene trait predictors: promoters and suppressors. These are proteins that can promote or suppress the expression of a gene. This means that if our data keeps track of both a promoter/suppressor and the gene product that it promotes or suppresses these will be directly related.

Cell viability traits may also have hidden relationships that could be trimmed from the data. [6] details the role of the mitochondria in apoptosis and shows several potentially related variables. For instance, one sign of apoptosis is the concentration of activated caspase proteases (catalytic proteins that dismantle the cell) in the cytosol. These normally exist in an inactive form (a zymogen) and must be activated by a special protein that is normally is trapped within the inner membrane of the mitochondria. This means that if the data keeps track of both mitochondria permeability and activated caspase proteases then we would see a direct relationship between those two.

While these may not be the specific collinearities we see (indeed it would be strange if researcher kept track of two known correlated quantities), but these examples illustrate that biologically relevant covariates exist. Given the complexity of this study it is almost surely the case that correlations between variables exist that we have not yet found in any meaningful studies. It is in these unknown relationships that we may be able to find new ways to reduce the complexity of this field of study.

Methods/Results

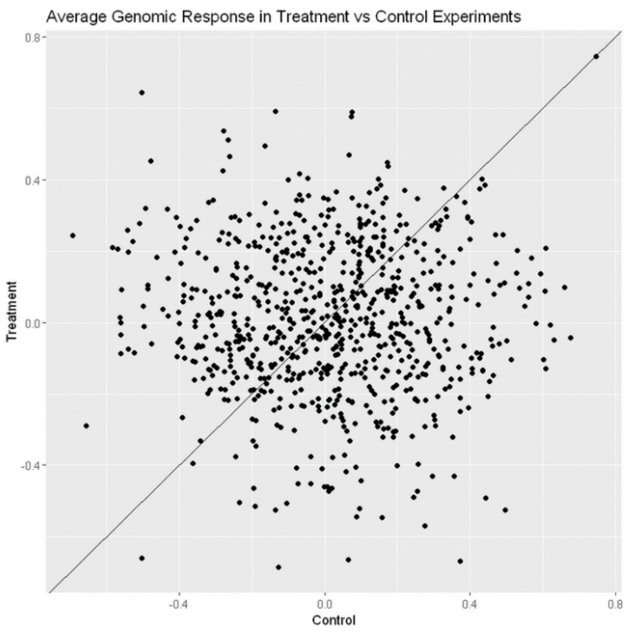
Data Description

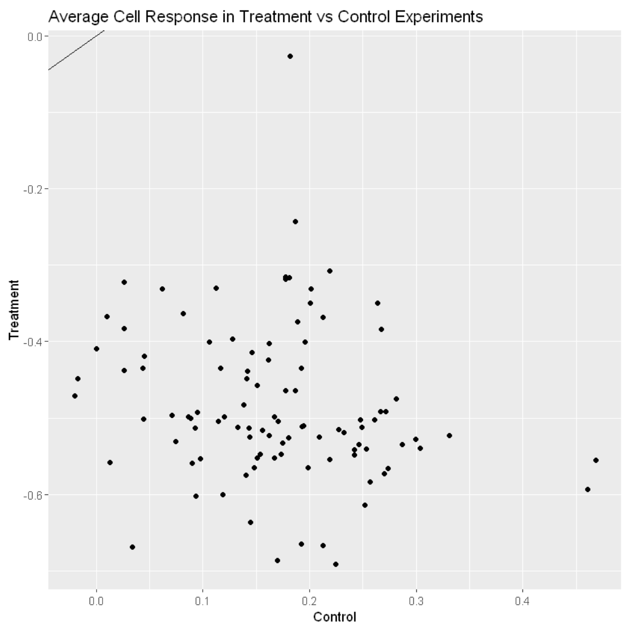
The design matrix is a 23814 by 876 matrix of features (not including the bias column), but one of these is an id column which will be dropped before analysis is performed. Each row corresponds to an experiment with a particular compound, but each row is not necessarily a unique compound (in fact there are only 3289 unique compounds). The most important division of predictive features is, as we have mentioned before, the cell characteristics and the gene characteristics. Assuming that all the gene and cell characteristics come from two different distributions (ie the first gene measure has the same distribution as the second gene characteristic, etc, and this distribution is different from the distribution for all the cell characteristics) we can talk about aggregated summary statistics of these distributions.

The grand mean of all the gene characteristics is around .009 with a standard deviation of 1.236. The median of this distribution was 0 and the values range from -10 to 10. It looks rare to get values close to 10 or -10 meaning that increases in magnitude of these characteristics may be predictive of large changes from baseline (and hence predictive of some unexpected change indicative of the MOA of the administered compound).

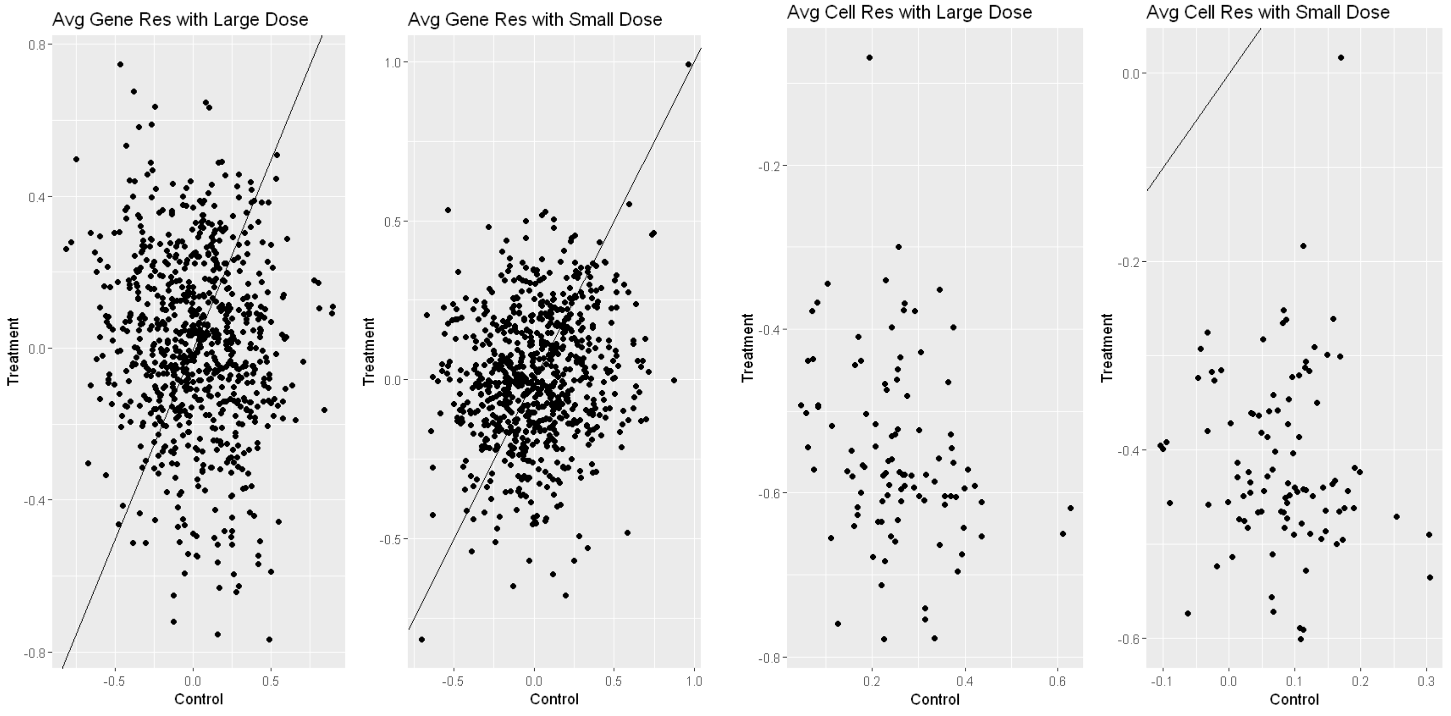
Similarly, the grand mean of the cell characteristics is -.432 with a standard deviation of 1.905. The median of this distribution is -.0054 (suggesting some skew is present), and these values range from -10 to 6.412. Again, values that are large in magnitude seem to be most predictive for this distribution. Note that no significant missing values were appreciated in this dataset.

These summary stats are not very informative about the research question. To gauge the efficacy of our predictors we need to consider the treatment vs control variable (one of the few variables in the data that is not a gene nor cell characteristic). We would like to see if there are any features that are identical for treatment and control (on average) to see if there is any reason to believe that some may be unnecessary. To do this we can make a plot where each data point is a feature in the data, and one axis is the treatment mean of that feature and the other axis is the control mean for those features. Then, any deviance from the line y=x is a response due to the treatment, and if any points lay on the line, then they are less useful features since they do not change on average between treatment and control. Making these plots for the gene and cell characteristics:



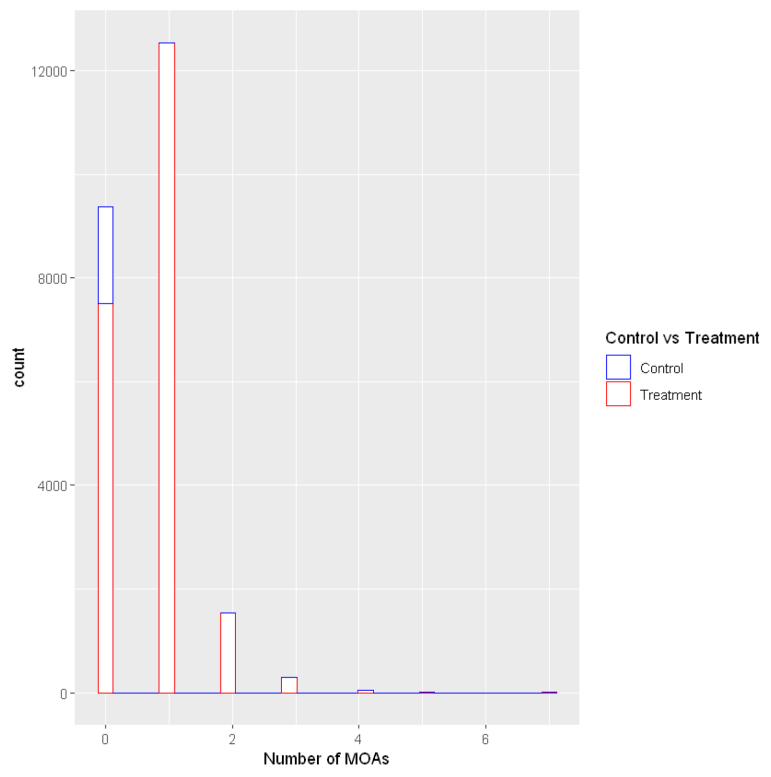


Based on this we can see that all the cell characteristics change on average between treatment and control, but many of the gene responses do not respond on average between treatment and control, so they may be less useful overall. Another potentially important factor that the experimenters' record is for whether a large or small dose of the compound was given. We can remake the above graphs, but conditioning on dose:



From these graphs one can see that higher dose creates a larger effect on average in both the cell characteristics and the gene characteristics.

Turning to the response we see that it is somewhat problematic for how linear/logistic regression is typically defined since the response is a matrix with 207 columns. Each column corresponds with a possible MOA which a compound could be classified as. Interestingly, many compounds in this data have multiple MOAs which further complicates the situation. Here is a histogram showing the number of MOAs (coloring based off control/treatment):



Interestingly, we can also see that a significant number of treatment group compounds have no listed MOA. This structure presents a problem both for defining the "efficacy" of a model as well as the training procedure for a model.

Training the Model:

A General Approach:

To train this model we will need to make use of "multivariate multiple linear regression" [7]. This is a reframing of linear regression that allows for the estimation of a linear relationship to a matrix of predictors to a matrix of output variables. [7] offers an approach that treats each output column as a separate linear/logistic regression to be trained using the normal matrix inversion and multiplication required by these methods. In symbols this is:

$$\hat{\beta}_j = (X^T X)^{-1} X^T Y_j$$

Where $\hat{\beta}_j$ is the set of predictors for the jth column of Y, X is the design matrix and Y_j is the jth column of the response matrix. Note this is for linear regression, but for logistic regression we will do the same thing (treating each column of the response as a separate logistic regression problem). There are problems, however, with this approach when applied to large data. First, if X is large than the inversions and multiplications required to find the least squares solution become time consuming or even computationally intractable. Second, as the problem is set up, we cannot perform any of the required regressions simultaneously, meaning that some potential speed is lost by having to create each regression separately. Solutions to these problems have already been found in the form of gradient descent, and a neural network interpretation of linear regression which we will now detail:

Approximating the Least Squares Solution with Gradient Descent:

In general gradient descent is a method by which we can find the minimum of an arbitrary function. This is done by taking the gradient at a point and taking a small step in the opposite direction of that gradient [15]. This can be written algorithmically as follows (for an arbitrary function f):

1.) Initialize a guess for the minimum x_0
2.) Compute the gradient of f at this point $\nabla f|_{x_0}$

3.) Take a small step in the opposite direction of $\nabla f|_{x_0}$ (ie do $x_1 = x_0 - \alpha \nabla f|_{x_0}$, α being a constant representing the step size)
4.) Do this until a stopping criterion is met using the following general formula: $x_n = x_{n-1} - \alpha \nabla f|_{x_{n-1}}$

In general, this update rule will work, but inefficiently. A better rule, and the one we will employ in our optimization, is the so-called Adam optimizer [8]. This is a well-known optimizer in the field of artificial neural networks, and numerical optimization, and it works by making step-size a variable parameter (proposing both a velocity and a momentum variable which is a way of gauging how far or close we are to the solution). This algorithm takes large steps when it is far away from a minima and small steps when it is close to the minima. We will not describe this algorithm here since it is lengthy, but the full description can be found in the original paper [8].

For the specifics of linear regression, we will need to find a function to minimize, and the function we will consider is the same function that the least squares estimator minimizes:

$$Q(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where n is the number of observations in the data. To use gradient descent, we need to think of x_0 as a vector of all the parameters that we want to estimate X. Since we are considering $\hat{y}$ to be linearly related to the parameters we can write the following relation:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_m x_m = \hat{\beta} X$$

Where m is the number of predictors in the model. To successfully apply gradient descent, we need to find the gradient of Q with respect to all the parameters (the $\hat{\beta}$s):

$$\frac{\partial Q}{\partial \hat{\beta}_J} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_i^j$$

Where x_i^j is the jth predictor of the ith row of X. However, we are interested in logistic regression since our problem is one of classification. This is a slightly more involved derivation, but the general procedure is the same. Consider a version of the logistic regression prediction formula:

$$\hat{y} = sigmoid(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_m x_m)$$

Notice that we use a sigmoid link function as opposed to a logit. This is a convenient interpretation of logistic regression for our purposes, and since it only slightly affects the interpretation of the model (we are observing probabilities as opposed to log odds) so it is a fine substitution to make. To minimize this [9] posits a slightly different cost function which makes finding derivatives easier:

$$Q(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$$

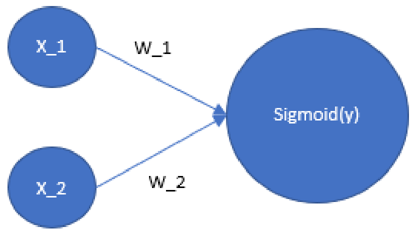
Notice that if we are misclassifying a 0 for a 1 then this function will be extremely large ($\hat{y}_i \approx 1$, and $y_i = 0$, so the first term will be close to zero and the second term will approach infinity). This intuition also holds for misclassifying a 1 for a 0. We can also note that this function is close to zero when the prediction agrees with the observed, so intuitively this reaches a minimum for a perfect classifier. This means that we can safely minimize this function to train a model. Taking the derivative of this is more involved (though it is just an application of the chain rule in reality), we will simply state it (a full derivation can be found in [16]):

$$\frac{\partial Q}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n}(y_i - sigmoid(\beta^T x_i)) * x_i^j$$

Notice that this is almost the same as the linear regression case (the only difference is that we see a sigmoid function appear). This is all that we need to know for this minimization problem. Since this problem is convex (Q is convex) it does not matter want we initialize the parameters to, so simply following the procedure from the first algorithm we should attain convergence (though again we will employ the Adam optimizer since it should converge faster). This will work faster than inverting a matrix, but it still does not allow to parallelize the training process for multivariate multiple linear regression, for this we will consider an equivalent conception of linear regression which uses neural networks.

Improving Training Speed by Using a Neural Network

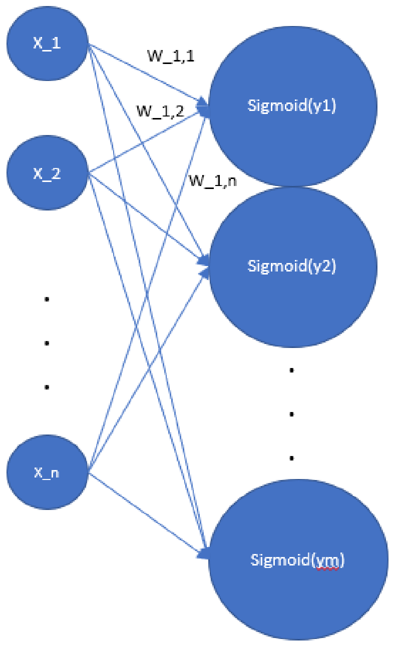
[10] offers a brief explanation on how neural networks works, as well as an equivalence between a specific neural network and linear/logistic regression. First, it is worth understanding the basic operations of a neural network through a picture:



This picture depicts a neural network with 2 input neurons and a single output neuron. The connection weights are denoted w_1 and w_2, and the equation for the output can be read as:

$$y = sigmoid(w_1x_1 + w_2x_2)$$

This is already equivalent to logistic regression in the forward pass. The backward pass (where the weights are updated) is then just updating the weights based off the cost function we defined in the previous section. For the purposes of training our model we propose a slightly different network:



For better visual clarity we have only labeled the weights for the first output neuron, but in general this can be understood as a neural network with n input neurons and m output neurons with full connects between all of the inputs to all of the outputs, and the connections between the input layer to the output layer can be thought of as a single logistic regression to the first output variable, and another separately trained logistic regression to the second output neuron, and so on.

This may not seem like a large departure from simply training separate models using gradient descent, but due to recent advancements in the implementation of neural networks in languages like python (using PyTorch or Tensorflow libraries) there is a lot of potential in this representation for parallelization. For this reason, we will use this representation of logistic regression to take advantage of these prebuilt libraries to speed training up. However, to evaluate and train the model simultaneously we will, again, need to slightly modify the cost function. Since we are now attempting to train many models at the same time the neural network needs a way to evaluate all the models at once. To do this we will simply take the average cross entropy (cross entropy being the cost function that we defined in the previous section for logistic regression) across all the models.

Miscellaneous Preprocessing:
For all the models discussed in the rest of this paper, column-wise normalization was applied to the training features since this improved performance across the board. This normalization was the usual kind defined as:

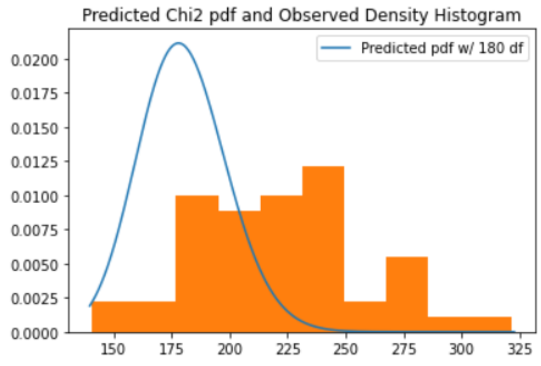
$$Z_j = \frac{X_j - \mu_j}{\sigma_j}$$

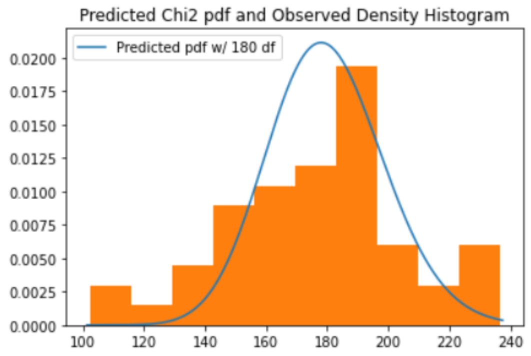
Where X_j is the jth column of the design matrix, μ_j is the mean of the jth column, and σ_j is the standard deviation of the jth column.

Results of training the baseline model:
The first model was trained using all the parameters to give an idea of what predictive accuracy was possible without any feature selection. Using a $K = 10$ fold cross validation scheme to determine if assumptions were met (since checking the assumptions explicitly for over 200 logistic regression models would be too cumbersome) we observed that the test train split used did not significantly affect model performance with all the models attaining around 99.7% accuracy and losses (this is the mean binary cross entropy that we defined previously) of around .017-.018 on the test sets. Note that this K-fold cross was not applied with an additional validation dataset (as the data in K-fold cross validations normally split the data into train, test, and validation). This is because we have determined the exact method that will be used before observing the results, so there is no chance that we will overfit hyperparameters to the test set. This suggests that this problem is sufficiently modeled by our model choice (logistic regression). After this K-fold cross was done a new model was trained with a new test train split which was used to get a final accuracy of 99.7% and a loss of .018 on the test set.

To attain p-values for the parameters in each logistic regression we employed a bootstrapping method to approximate the standard deviations of the model parameters [14] (this method involves resampling the rows of the design matrix/response matrix, fitting a model, and recording the coefficients for use in approximating standard errors, for more details see section 3.1-Random-x Resampling in [14]). From this we found that around 69.0% of the parameters were statistically significant across all models. Almost all model features were significant in at least 10 models (again, keep in mind that we are building a separate logistic regression for each column in the response), but we found that 108 parameters were only significant for 50 or less models. This suggests that some predictors may be more useful than others which is another potential way to motivate the idea that the number of parameters recorded can be reduced.

An interesting question to ask is whether these models sufficiently model the data. For a poisson regression this might be done with a chi-squared test on the residual deviances, however this does not work in this scenario due to the small number of 'replications' in logistic regression (this refers to the fact that we can think of logistic regression as a binomial regression with n = 1). To illustrate this, we will look at density plots for the residual deviances of the boostrapped models that we attained earlier. This shows that model 1 and model 2 (each regressed onto a different response column) have vastly different distributions, neither of which is well approximated by a chi-squared distribution:





To even get the chi-squared distribution to be visible on this plot we needed to set the degrees of freedom to be an unexpected number (180 as opposed to what is predicted when working with a poisson regression). This suggests that the chi-squared test will not be of help here, but we can use an average confusion matrix to see if the model is making reasonable predictions. To get a simple summary of all the logistic regressions we will look at the average confusion matrix which is as follows for this model (on the test set):

$$\begin{bmatrix} 2373.64563107 & 196.67475728 \\ 203.44660194 & 197.95145631 \end{bmatrix}$$

This is somewhat expected since the response is a sparse matrix, so we see that the model predicts (correctly) many zeros. This also shows that our model is possibly not the greatest in practical application since we are really interested in getting true positives, but we can see that if the model predicts a 1 then there is about a 50-50 chance that it is a false positive. For the purposes of this paper this is not concerning since we are simply hoping to show that the reduction of parameters will not affect the model. This does mean that for any dimensionality reduction we will keep a careful eye to make sure that we do not make the true positive rate worse than it is now. One may be able get better accuracy by using techniques like curriculum-based learning or a more complex model, but this is out of the scope of this paper.

The metrics given in this section are mainly to compare the other dimensionally reduced models. In future sections we will be especially concerned with keeping constant or increasing the proportion of parameters that are statistically significant and keeping constant or increasing the true positive rate.

Dimensionality Reduction with Variance Inflation Factors (VIFs):
In this section we will reduce the dimension of the design matrix using VIFs and the condition number of the design matrix. The scheme that we implement is as follows:

1.) Find the VIF for each column in the design matrix.
2.) Remove the column with the highest VIF from the matrix.
3.) Compute the condition number of the matrix if it is less than 30 stop the algorithm.
4.) Recompute the VIFs of the reduced matrix.
5.) Repeat until either 100 columns are removed, or we get a condition number less than 30.

The VIF of a column j of the design matrix is defined as:

$$VIF_j(X) = \frac{1}{1 - R_j^2}$$

Where R_j^2 is the R^2 value of a linear model using all the columns of X other than the jth column to predict the jth column. The condition number is also defined in the canonical fashion as the square root of the ratio of the greatest singular value of X divided by the smallest singular value of X.

At the start of this procedure, we observed that the condition number of the design matrix was 41.5275, and that the largest VIF was 8.2694 both of which suggest that collinearity was most likely present in the design matrix. Performing the algorithm above allowed us to eliminate 70 columns before reaching a final condition number of 29.3518. The columns that this processed eliminated were the following:

['c-83', 'c-13', 'g-50', 'g-100', 'g-75', 'g-441', 'g-651', 'g-369', 'g-64', 'g-534', 'g-300', 'c-18', 'g-392', 'g-635', 'g-629', 'g-253', 'c-52', 'g-502', 'g-37', 'c-73', 'g-195', 'cp_dose', 'g-121', 'g-744', 'c-26', 'g-80', 'g-410', 'g-498', 'g-139', 'g-206', 'g-727', 'g-102', 'g-615', 'c-42', 'g-385', 'g-770', 'c-94', 'g-664', 'g-38', 'c-55', 'c-10', 'g-351', 'g-314', 'c-6', 'g-327', 'g-178', 'g-140', 'g-761', 'g-566', 'g-553', 'g-287', 'g-406', 'g-374', 'g-235', 'g-175', 'g-76', 'g-439', 'g-588', 'g-63', 'g-172', 'c-63', 'g-404', 'g-248', 'g-509', 'g-196', 'g-454', 'g-503', 'g-522', 'g-21', 'c-38']

Note that there are some concerns with causality here based on how we chose which parameter to remove. These concerns will be addressed in the conclusions section.

After training a model with these columns eliminated, we observed little difference between the full model, and this reduced model. A K = 10 fold cross validation scheme revealed that this model attained predictive accuracy of around 99.7%, and losses that ranged from .017 to .018, both of these metrics are identical to that of the full model. Once this was completed a final model was trained on a new test train split and got a final accuracy of 99.7% on the test set and a loss of .018 on the test set.

As far as statistically significant coefficients are concerned, we saw a slight increase in the proportion of statistically significant parameters across all models. In the model trained on the reduced dataset we observed that 72% of model coefficients were statistically significant (up from 69%), and that all coefficients were significant in at least 10 models with only 74 parameters being insignificant in all but 50 or less models (down from 108).

Finally, the average confusion matrix of this model on the test set was:

$$\begin{bmatrix} 2373.66019417 & 196.66019417 \\ 203.49514563 & 197.90291262 \end{bmatrix}$$

Which is nearly identical to the confusion matrix for the full model on the test set. Based off this analysis it appears that, for the task of prediction using a multivariate multiple logistic regression model, the parameters eliminated by using the VIF were not particularly important. This conclusion along with some caveats will be discussed in the conclusions section.

Dimensionality Reduction Using Lasso Regression:

One problem with the use of VIFs for dimensionality reduction is that it is a time-consuming process since many regressions must be fit sequentially. A faster more automatic way might be to use so-called Lasso regression. We will again consider a gradient descent approach to implementing this algorithm, which is approximately the same as the gradient descent detailed earlier in this paper, except for a change to the cost function Q. Under Lasso regression we would also like the model to minimize the magnitude of the parameters (even eliminating some parameters), so we add a penalty on the size of the coefficients of the model (this definition is a slight modification to the one provided by [11]):

$$Q(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} -y_i \log(\hat{y}_i) + (1 - y_i)\log\,(1 - \hat{y}_i) + \lambda \sum_{i=1}^{p} |\beta_i|$$

Where λ is a parameter that, in a sense, determines how aggressively the algorithm will push the weights to zero. The derivative is now much more complicated so for the sake of brevity we will trust the algorithms put in place by Python.

Unfortunately, this method of feature selection is too dependent on the specific logistic regression model that is being scrutinized and represents how methods like stepwise AIC/BIC and other methods that require a trained model might be difficult to apply in this data. The problem lies in consistency. Since each logistic regression's parameters were penalized separately, each converged to solutions that contained zeros in different places. This means that there is no obvious criterion for choosing which columns of the design matrix are or are not helpful for prediction. This is especially problematic since choosing to eliminate columns based on any statistical criteria already makes the determination of causality more difficult. This difficulty would be amplified if we were to decide on an arbitrary criterion for determining the relevance of parameters. For this reason, we advocate the use of methods such as eliminating columns using VIFs instead since they are not swayed by any specific model.

Nevertheless, we were able to perform a K = 10 fold cross using Lasso regression for this data using $\lambda = .001$. These Showed worse results than the other methods attaining an accuracy of about 99.6% on the test set with an associated loss (this loss does not consider the penalty on the weights) of around .24. This may not seem like such a large drop, but since the model was not perfect to begin with it becomes an unacceptable amount. This clearly shows that this feature selection may not be optimal, since we are clearly losing information, but what about for a smaller value of lambda?

Doing another 10 fold cross on the data using Lasso regression (using $\lambda = .000001$, this was the first lambda we tried that had good results for loss and accuracy) show accuracies of around 99.69% and losses of around .018 on the test sets for each fold. These are promising results, but after examining the coefficients we noted that this resulted in only 545 parameters being close to zero across all models which represents .3% of all coefficients in the model. Thus, this is determined to be an ineffective scheme for dimensionality reduction for this dataset.

Conclusions:

Outcomes:

In this study we determined that data on the mechanisms of action could be substantially reduced without loss in performance of linear models (multiple multivariate logistic regression). This was done using variance inflation factors to determine which columns to drop in the design matrix, which we advocate as the most effective method of dimensionality reduction for this dataset. Lasso Regression was also employed to try and expedite the feature selection process but was deemed ineffective due to its lack of consistency across models. In fact, a similar argument can be made against any feature selection that requires a specific trained regression to determine what features to drop, so we would advocate the use of methods that only require the design matrix to do their work.

This could have implications in the practicality of these experiments. Due to the volume of parameters that are recorded in these experiments, many experiments must be performed to get a sense of how each parameter changes. This puts a burden on the experimenter which requires funding and time to overcome. By reducing the number of parameters that need to be recorded these experiments may become less costly and time consuming in the future.

Problems with this Experiment Design

The most substantial problem with this paper is the nature of causality in the reduced dataset. The use of VIFs to determine what columns are unneeded in the design matrix utilizes no domain knowledge for this problem, so we may be extraneously dropping important variables. For example, using the VIF algorithm we drop the column 'cp_dose' or whether a low or high dose was given to a cell population. Beyond the obvious intuition that this is probably an important parameter, it is required for causality in medicine according to the Bradford Hill criteria [12] (see point 5: biological gradient) which determined that a dose dependence relationship must be present for medical causality, so it does not make sense to drop this from the model.

But how can one solve this? Perhaps in future studies one could seek the advice of domain experts to determine if certain columns should remain off limits with regards to what can be eliminated. In general, there is no silver bullet solution to this problem, but it is still worth noting as a weakness that should be more carefully considered in later works.

Future Work:

Aside from problems of causality, there is the issue that our model did not have as high of a true positive rate as we would like. Future works may apply models that can find more complex decision boundaries such as trees to aid in this problem. This would require much more computational power than was available for this study, but it would be worth seeing if our dimensionality reduction is still as effective for different model types.

Cited Sources:

[1] Baroody, F. M., & Naclerio, R. M. (2000). Antiallergic effects of H1-receptor antagonists. *Allergy, 55*, 17-27. doi:10.1034/j.1398-9995.2000.00803.x

[2] Antihistamines: Definition, Types & Side Effects. (n.d.). Retrieved from https://my.clevelandclinic.org/health/drugs/21223-antihistamines

[3] Histamine. (2021, March 05). Retrieved from https://en.wikipedia.org/wiki/Histamine#Vasodilation_and_fall_in_blood_pressure

[4] Mechanism of action. (2021, February 23). Retrieved from https://en.wikipedia.org/wiki/Mechanism_of_action

[5] Hoopes, L. (n.d.). Gene Expression and Regulation. Retrieved from https://www.nature.com/scitable/topic/gene-expression-and-regulation-15/

[6] Wang, C., & Youle, R. J. (2009). The role of mitochondria in apoptosis*. *Annual review of genetics*, *43*, 95–118. https://doi.org/10.1146/annurev-genet-102108-134850

[7] Quick, C. (2013). Multivariate Multiple Regression with Applications to Powerlifting Data. *University of Minnesota Duluth*. Retrieved from https://scse.d.umn.edu/sites/scse.d.umn.edu/files/cassiequickfinalpaper.pdf

[8] arXiv:1412.6980 (link: https://arxiv.org/abs/1412.6980)

[9] Luo, S. (2019, June 07). Optimization: Loss Function Under the Hood (Part II). Retrieved from https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11

[10] Issac, J. (2018, October 12). Understanding Deep Neural Networks from First Principles: Logistic Regression. Retrieved from https://medium.com/@melodious/understanding-deep-neural-networks-from-first-principles-logistic-regression-bd2f01c9e263

[11] Kim, Y., Kim, Y., & Kim, J. (n.d.). Gradient LASSO algoithm. Retrieved from http://www.cs.cmu.edu/afs/cs/project/link-3/lafferty/www/ml-stat2/talks/YondaiKimGLasso-SLIDE-YD.pdf

[12] Hill, A. B. (1965). The Environment and Disease: Association or Causation? *Proceedings of the Royal Society of Medicine, 58*(5), 295-300. doi:10.1177/003591576505800503
[13] Lagay, F. (2002). The Legacy of Humoral Medicine. *AMA Journal of Ethics, 4*(7). doi:10.1001/virtualmentor.2002.4.7.mhst1-0207
[14] Fox, J. (2002). Bootstrapping Regression Models. *Appendix to An R and S-PLUS Companion to Applied Regression*. Retrieved from https://statweb.stanford.edu/~tibs/sta305files/FoxOnBootingRegInR.pdf.
[15] arXiv:1609.04747v2 [cs.LG] (link: https://arxiv.org/pdf/1609.04747.pdf)
[16] Thavanani, S. (2020, July 08). Derivative of Log-Loss function for Logistic Regression. Retrieved from https://medium.com/analytics-vidhya/derivative-of-log-loss-function-for-logistic-regression-9b832f025c2d

Data Source:
This data was from a Kaggle competition which can be found here: https://www.kaggle.com/c/lish-moa/data

Appendix (Code):
NOTE: To run the Python portion of the code a GPU is required. If you do not have a GPU then all instances of .cuda() must be removed.

The code is attached in the same submission as this document in the form of two html documents. The first file is Tiede_Jacob_EDA_In_R.html and this is where I did most of the EDA on the data (this section is in R). The second file is Tiede_Jacob_Linear_Models_In_Python.html and it contains all of the modeling sections (so everything beyond the EDA is in this document) note: this document is in Python.

Files Used in the Code:
All file names are kept consistent with those on the Kaggle page, so in this code the name train_drug.csv has a corresponding file on the Kaggle (as does train_features.csv, and train_targets_scored.csv).