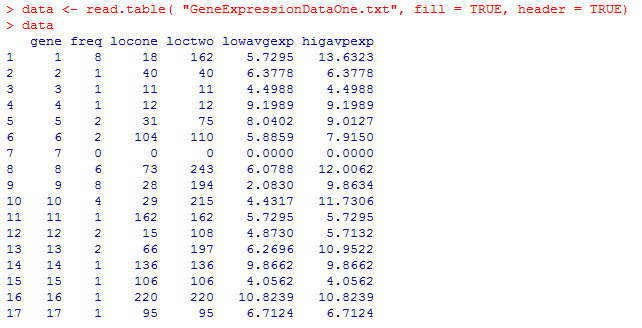
**Neural Network Script of GeneExpressionDataOne – January 28th, 2019 – Darrell Robinson**

**Link:** [**https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/**](https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/)

**Box One:**



***\*\*\*\*\*In order to create the fitted linear model the fill part and the header part of the code MUST ALWAYS EQUAL TRUE in order for the software to identity and match up the numbers to the column labels correctly! It will not work without it!\*\*\*\*\*\*\****

***\*\*\*We created dummy variables for the names of the different genes. The neural network only reads numeric responses, but I know that we can include the names in the R Program script as well. \*\*\*\****

***Gene 1 = Chemokine Ligand Motif Gene***

***Gene 2 = Eukaryotic Translation Initiation Factor Gene***

***Gene 3 = Glutathione. S. Transferase Theta 1 Gene***

***Gene 4 = Ribosomal Protein Gene***

***Gene 5 = Chitinase 3 Like Type Cartilage Glycoprotein 39 Gene***

***Gene 6 = Zinc Finger Protein Family Gene***

***Gene 7 = Cathepsin Family Gene***

***Gene 8 = Cluster of Differentiation Antigen or Molecule Gene***

***Gene 9 = Chromosome Open Read Frame Gene***

***Gene 10 = Solute Carrier Family Gene***

***Gene 11 = Tumor Necrosis Factor Gene***

***Gene 12 = Ankyrin Repeat Domain Gene***

***Gene 13 = DEAD Box Gene***

***Gene 14 Suppressor of Cytokine Signaling 3 Gene***

***Gene 15 = Serpin Peptidase Inhibitor Gene***

***Gene 16 = Guanylate Binding Protein Gene***

***Gene 17 = GTPase Activation Protein Gene***

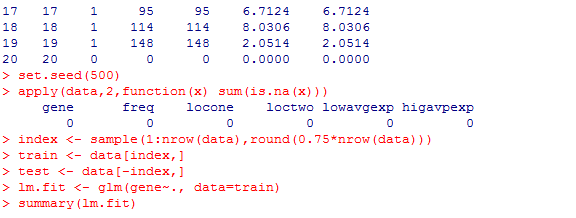
***Gene 18 = Integrin Beta Gene***

***Gene 20 = ATPase Class V Type 10 A Gene***

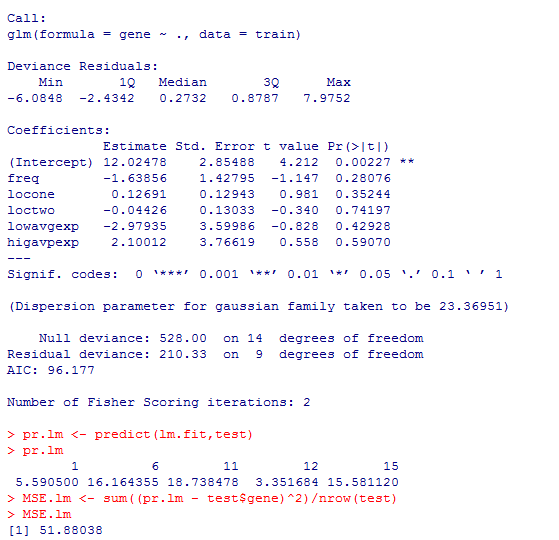
***Gene 21 = Blank Results***

***(This gene did not have a name, but sometimes had other Gene Ontology Characteristics such as Biological Process, Cellular Component, or Molecular Function. They did have any gene expression of any kind. Either manually finding more patterns and tried to estimate what gene this could be or what Gene Ontology Characteristics the Blank Result gene(s) have could be an option. Another option that we could do is to create a Natural Language Processing Neural Network to help predict the Gene Ontology Characteristics any of the Blank Result gene(s) would have. We would be filling in the blanks and seeing how this changes the data as a whole. This could help address the design probe bias that is found in DNA Microarray Technology.)***

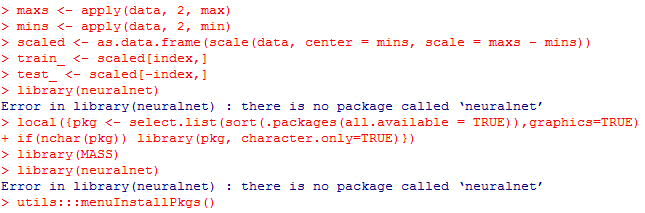
**Box Two:**



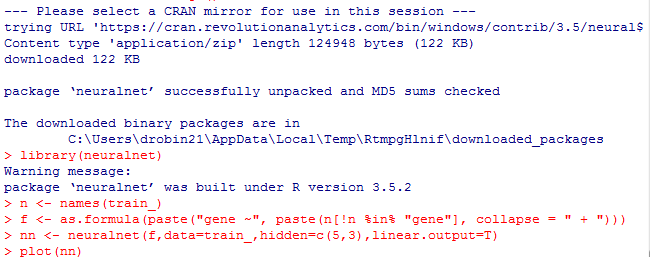
**Box Three:**



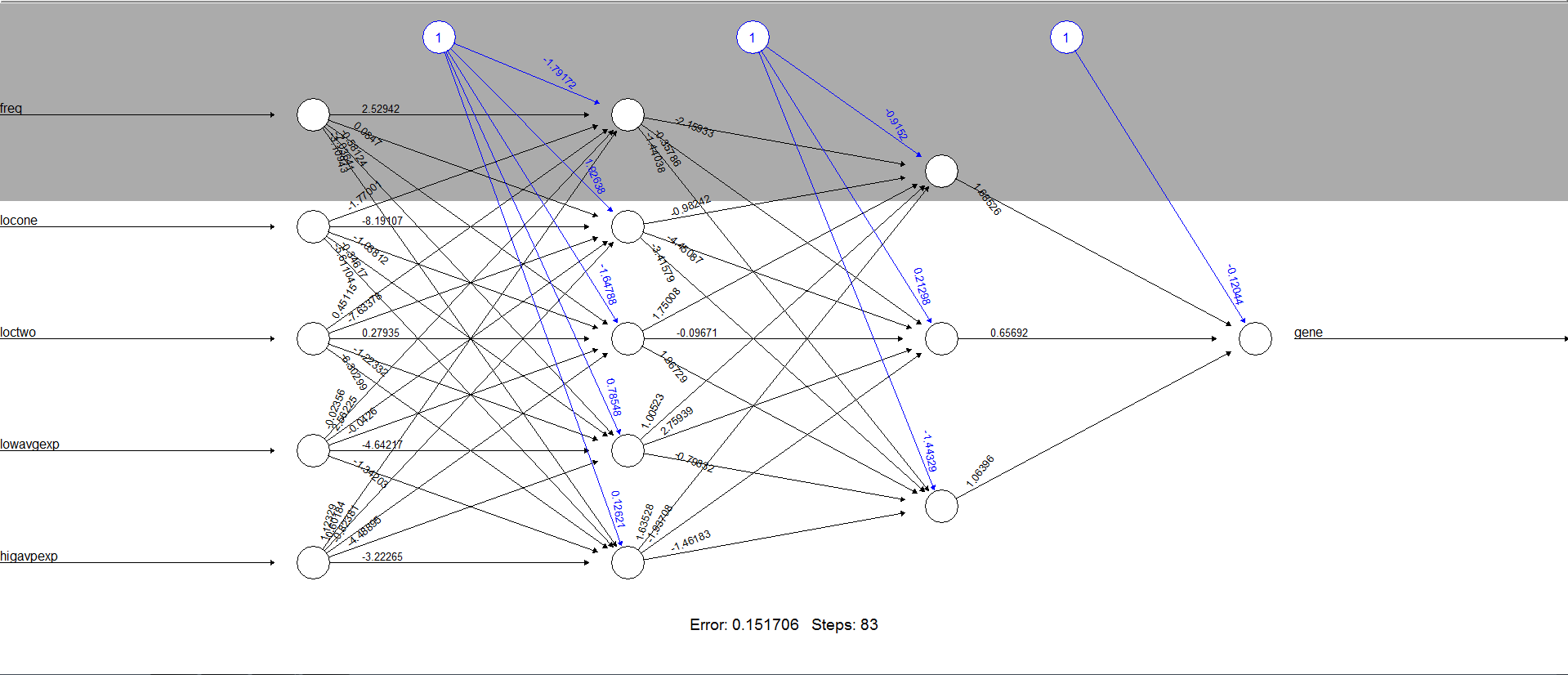
**Box Four:**



**Box Five:**



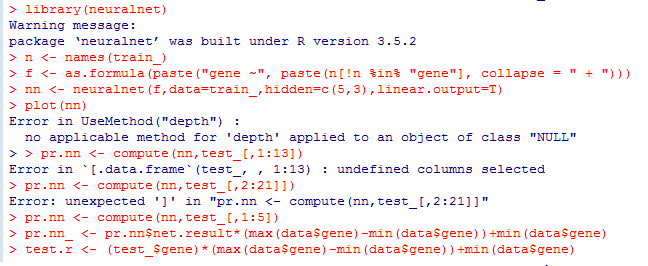
**Neural Network Image One**



**The black lines show the connections between each layer and the weights on each connection while the blue lines show the bias term added in each step. The bias can be thought of as the intercept of a linear model. The net is essentially a black box so we cannot say that much about the fitting, the weights and the model. Suffice to say that the training algorithm has converged and therefore the model is ready to be used.**

**Predicting Gene Using the Neural Network:**

**Box Six:**



**Box Seven:**



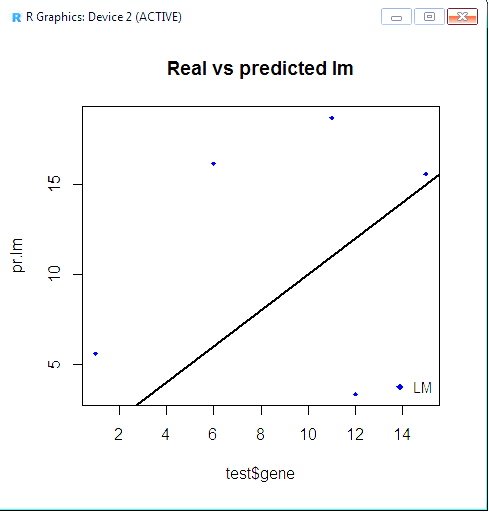
**Apparently the linear model is doing a better job than the neural network at predicting gene. We have to be careful because this result depends on the train-test split performed above. Below, after the visual plot, we are going to perform a fast cross validation in order to be more confident about the results. A first visual approach to the performance of the network and the linear model on the test set is plotted below.**

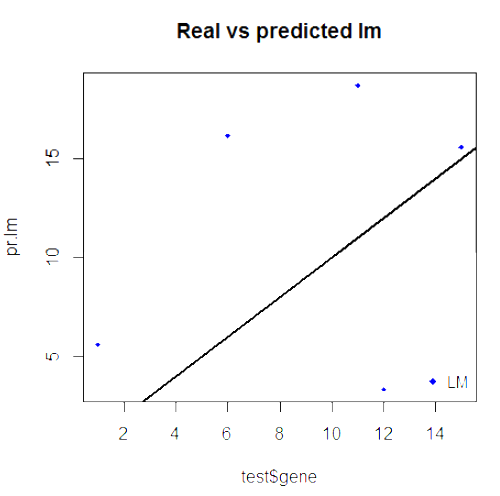
**Visual Approach of the Performance of the Neural Network vs Linear Model on the Test Set**

**Box Eight:**



**Visual Image of Linear Model Performance on the Test Set:**

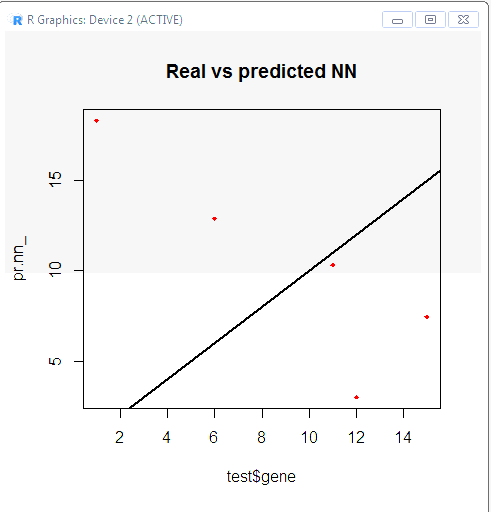




**Box Eight:**

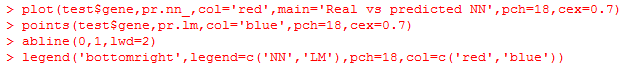


**Visual Image of Neural Network Performance on Test Set**

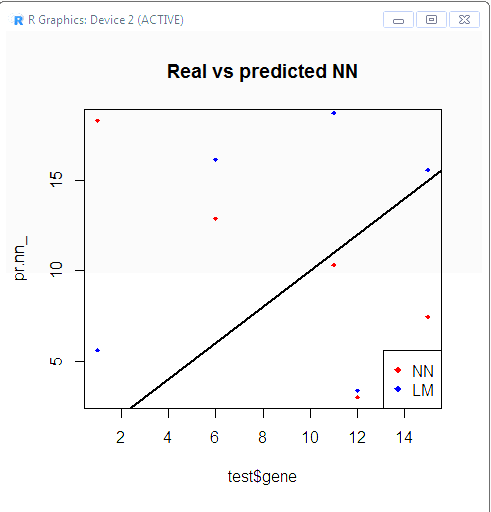


**By visually inspecting the plot we can see that the predictions made by the neural network are (in general) less concentrated or further away from the Mean Squared Error (MSE) line (a perfect alignment with the line indicates a MSE of 0 and thus an ideal perfect prediction) than those made by the linear model.**

**Box Nine:**



**Visual Image of the Neural Network vs Linear Model Performance on Test Set**



**Fast Cross Validation**

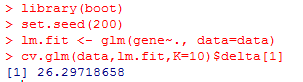
**Cross Validation is another important step of building predictive models. While there are different kind of cross validation methods, the basic idea is repeating the process a certain number of times.**

**Train-Test Split => [1] Do the Train-Test Split, [2] Fit the Model on the Train Set, [3] Test the Model on the Test Set, [4] Calculate the Prediction Error, and [5] Repeat the Process K Times**

**Code to Perform Cross Validation:**

**We will use a loop for the neural network and the cv.glm() function in the boot package for the linear model. Currently, there is no built-in function in R to perform cross validation on this kind of neural network. We will have to either find one or create one. Here we use a 10 Fold Cross Validated MSE for the Linear Model.**

**Box Ten:**



**The Mean Square Error for the Linear Model of the data with the 10 Fold Cross Validation is about 26%, which is good.**

**Below we use a 10 Fold Cross Validated MSE for the Neural Network. We are choosing to split the data in this way: 90% train set and 10% test set in a random way for 10 times. We can also initialize a progress bar using the plyr library to keep an eye on the status of the process since the fitting of the neural network might take a while.**