Computational Genetics

Nicholas Fox, Alexandria Melton, Stephanie Oliva

Loyola University Chicago

Abstract: The goal of this project was to use nonparametric techniques to analyze gene expression data and compare the results to parametric techniques used by Dr. Wheeler. All analyses and code was done using the statistical program, R Studio.

1 Introduction

The focus of this study was to investigate whether non-parametric methods can model gene expression data as well as or better than LASSO (Least Absolute Shrinkage and Selection Operator) or elastic net (alpha=0.5), as researched by Dr. Heather Wheeler, Department of Biology, Loyola University Chicago. In her research, Dr. Wheeler identified that sparse models such as elastic net or LASSO performed better. Using a similar data set, this investigation consists of K-Nearest Neighbors (KNN), hybrid KNN- Support Vector Machine (SVM), Principle Component Analysis (PCA), and non-parametric regression using natural cubic splines (NS).

2 The Data

The dataset consists of expression levels for 498 genes of 107 individuals from the Yoruba population in Nigeria, and the respective Single Nucleotide Polymorphism (SNP) for that gene. Additionally, this data represents just 1 megabase (million base pairs) along chromosome 22, one of the shortest chromosomes. Each gene is then its own dataset, with expression level as the continuous response, and SNPs as the predictors. Thus, each gene has 107 observations, and anywhere between 150 and over 7,000 SNPs. The SNPs take on values of 0, 1, and 2, to identify the number of minor alleles at each location.

3 Methods

The initial request for this investigation was to develop a KNN model to determine if it outperforms Dr. Wheeler's method. KNN was used to first classify by gene expression level and using the fitted values from that model, each gene was then subset into two, and PCA and natural cubic splines were thus fit to each subset. Since KNN is not very powerful in gene studies, a hybrid SVM-KNN model, with PCA and NS components, was introduced to

directly compare to the KNN-PCA-NS model. Lastly, as a direct comparison to Dr. Wheeler's penalized regression, a PCA-NS only model was fit. In the following subsections, each nonparametric technique is explained.

3.1 Principle Component Analysis

Principle Component Analysis (PCA) is a dimension reduction technique used when the number of predictors is larger than the number of observations and uses the covariance between predictors to achieve this reduction. The response variable is not taken into consideration as this is an analysis on the feature space. Each principal component is the linear combination of the eigenvalues of the covariance matrix and the predictors. These linear combinations explain the most variance and are uncorrelated to each other. The model becomes:

$$Y_i = \mathbf{e_i'X} = e_{i1}X_1 + e_{i2}X_2 + \cdots + e_{ip}X_p, \qquad for \quad i = 1, 2, \ldots, p$$

Where Y_i is the i^{th} principal component and $e_{i1}X_1$ is the first value of the i^{th} eigenvector multiplied by the first SNP in one gene. Once the dimensions have been reduced to the principal components that explain the largest percentage of variance, a model can then be fit by using those components as predictors instead of the SNP positions. Since each gene will have their own model, the number of components used will vary for each gene.

3.2 Natural Cubic Splines

Natural cubic splines are a type of non-parametric regression. These splines were chosen because they are smooth and they carry the constrain of linearity at the boundaries. The final model is:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$$

Where, b_1 is the basis function and x_i is the first principal component previously identified. This model would thus be expanded to include the additional X_{p-i} predictors, depending on the number of principal components chosen for each gene. The basis function includes 2 knots, which is specified by the 3 degrees of freedom used in the model.

3.3 K Nearest Neighbors

K nearest neighbors (KNN) is a nonparametric method with no assumptions made on the distribution of the data or errors. Given a test observation, KNN finds the K nearest points in the training data set that are closest to the test observation, this region is defined as $N_k(x)$. Next, it estimates the probability for class A as the proportion of points in $N_k(x)$ which are from class A. KNN then classifies the test observation to the class with the largest probability. The investigator has the option of changing parameter K, the number of nearest neighbors to use, and the distance metric used to calculate the nearest K points. The KNN model is as follows:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

$N_k(x)$ is the neighborhood of x defined by the closest points x_i. "Close" is usually defined by the Euclidean distance, which is the standard straight-line distance between two points in Euclidean space. Euclidian distance is measured using the formula below where $p = 2$.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

The literature [1] however, indicated that other distance metrics might perform better and so the Manhattan or city block distance was also included. The Manhattan distance metric defines the distance between two points as the sum of the absolute difference of their coordinates. You can also write the formula for this metric using the equation listed above where $p = 1$. KNN is usually performed with discrete or categorical data, however gene expression data is a continuous variable.

3.4 Support Vector Machine and K Nearest Neighbors

As an extension of the previous method, a hybrid technique of k-nearest neighbors and support vector machines were used. A support vector machine is a classification machine learning algorithm which finds the way to optimally divide data into two sets. The way it does this is by defining a hyperplane. For a space of n dimensions, a hyperplane is an n-1 dimensional subset that best divides the data into two group. The formula for a hyperplane is defined as:

$$f(x) = \beta_0 + \beta' x$$

Where β_0 is defined as a bias vector and β' is the transpose of the β weight vector. The way that this hyperplane is calculated is by maximizing a margin. The margin is the distance between the hyperplane and the closest points to it, which are referred to as the support vectors. The distance from a point x to the hyperplane is defined as:

$$distance = \frac{|\beta_0 + \beta' x|}{||\beta||}$$

Then, the margin is defined as twice the distance from the x values. The goal is to maximize the margin such that best possible distinction between the groups is found. For the hybrid model, the first method that is employed is the support vector machine. The data is analyzed by the algorithm, and the support vectors are determined. For this method this is all that we are interested in. The support vectors are then used as the training data in a KNN algorithm. Therefore, essentially what is being done is the SVM is choosing the points that are deemed to be the most important points in defining the two groups.

Then once these points are chosen, they are used as the training data for KNN, as opposed to using the complete original data set as was done in previous methods. Once this is done, we repeat the previous procedure whereby principal component analysis is first run in each of the bins (0,1). Then, using these principal components, a natural spline model is fit within each of the two bins and a value of R^2 is calculated for each. The principal components are also used to calculate an overall R2 for each gene, which then can be used for comparison with Dr. Wheeler?s LASSO model.

3.5 Cross Validation Techniques

Our original goal was to use k-fold cross validation in order to determine which k is the best for each gene. We wrote an algorithm that was able to run a 5-fold cross validation on the data to find the best k. However, we were limited by computational time, and the algorithm took an exceptionally long time to run. The kknn R package that we used to run KNN had a built-in function for leave-one-out cross validation (LOOCV), which ran significantly faster than the algorithm we tried to use for the k-fold cross validation. Therefore, in the interest of time, we chose to use LOOCV instead of k-fold cross validation.

4 Model Description

4.1 PCA-NS

We were asked to use non-parametric techniques to analyze the gene expression data. There are several methods you can use. One of the first things we tried on the SNP positions themselves was spline regression. However, with b-splines, natural splines, or smoothing splines there was a problem. These spline functions in R need to be done on each snip positions. In this dataset, some genes had over 7,000 predictors. Since the number of predictors was much larger than the number of observations we had, we used dimension reduction techniques.

Principle component analysis was used on the SNP positions matrix. Then for each gene we found the number of components that reduced a significant amount of the variability. Since not every gene will have the same number of components used in each model, the quick.elbow function in R proved to be very helpful. This function made it very easy to obtain the number of components . The next step was to fit natural cubic splines on each of the components that were extracted from each gene. Once we ran this regression we are then able to compare the R2 values from the nonparametric method to the elastic net regression that Dr. Wheeler used.

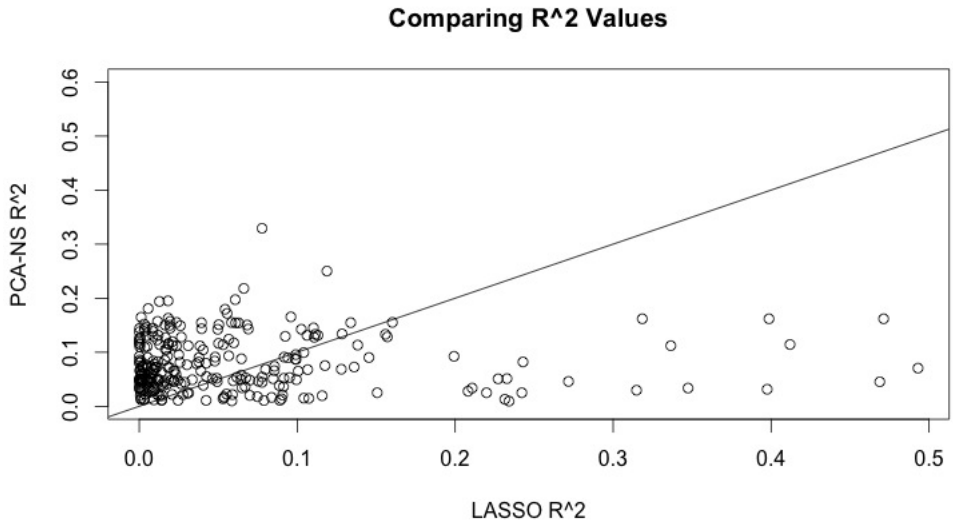

Figure 1: Comparing the R^2 values from the PCA-NS model to the LASSO model

Something to note about comparing R^2 values against models, with the LASSO model there are more than half of the R^2 values missing. That is why you see a large portion of the points at zero for the Lasso model. Below is a table of genes that performed significantly better with the PCA-NS model than the LASSO model.

Gene	Gene Name
ILMN1768580	BCL2L13
ILMN1668907	BCR
ILMN1794599	SNRPD3
ILMN1710675	XBP1
ILMN1697418	RBFOX2
ILMN1663710	PLA2G6
ILMN1735273	CBY1
ILMN1714623	TOMM22
ILMN1679614	SGSM3

Table 1: Genes that performed better with the PCA-NS model

4.2 KNN-PCA-NS

To perform KNN classification, the expression levels had to be classified. Using more than 2 classifications was sub-optimal with only 107 observations. Thus, 0, 1 was used in which expression levels greater than 0 were classified as 1 and 0 otherwise. Leave-one-out cross validation (LOOCV) was implemented to determine the best K parameter for each gene, among a range of K values between 2 and 15. While usually $\sqrt{n}$ is typically used as K, where n is the number of observations, it was necessary to expand the potential values of K since the genes varied widely in terms

of number and location of SNPs. LOOCV was chosen due to its computational efficiency; it vastly out performed k-fold cross validation.

Once the best K was found, and the model was fit on the dataset, the fitted values were used to subset the initial 107 observations per gene. For instance, observations with expression levels that were classified as 0 by the KNN model would then compose a new dataset so that the continuous expression levels could be analyzed using PCA and natural cubic splines. By extension the observations classified as 1 would compose the second dataset. Thus, two models will be created for each gene.

The use of Manhattan and Euclidean distances made a slight difference in the K parameter chosen for classification. As a result, observations were classified differently depending on the metric used. The resulting R^2 values were thus also affected. Below are three histograms illustrating the range of R^2 values for Dr. Wheeler's model, and the KNN-PCA-NS model with either Euclidean or Manhattan distance.

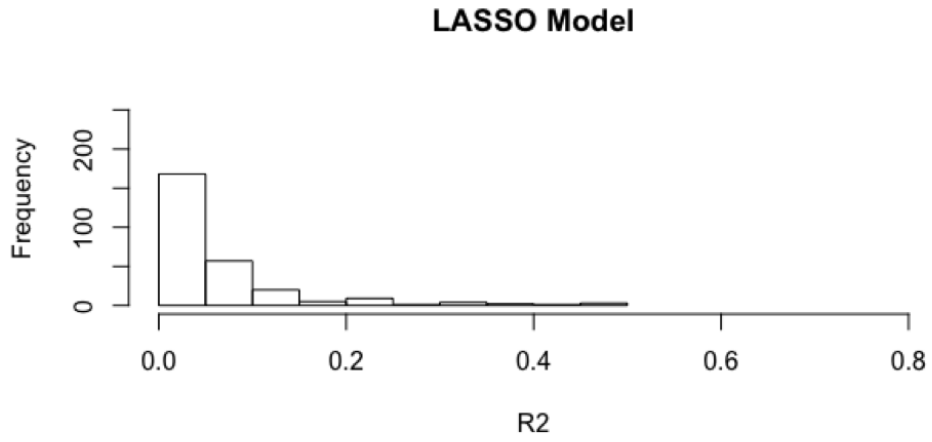

Figure 2: Dr. Wheeler's model has a range of values from 0 to 0.5, with a clear majority between 0 and 0.1

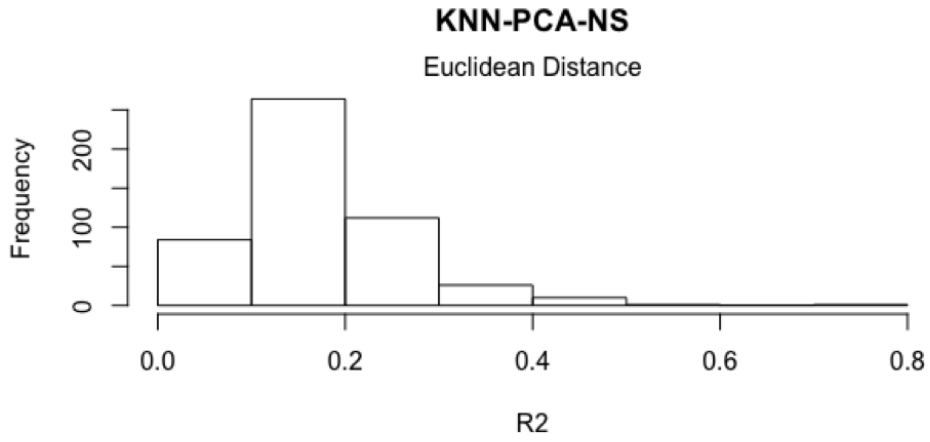

Figure 3: Range of R^2 values using the Euclidian distance

Using Euclidean distance, the range of R^2 is much larger than for the LASSO model. Additionally, much more values are between 0.1 and 0.2, which was not the case for LASSO.

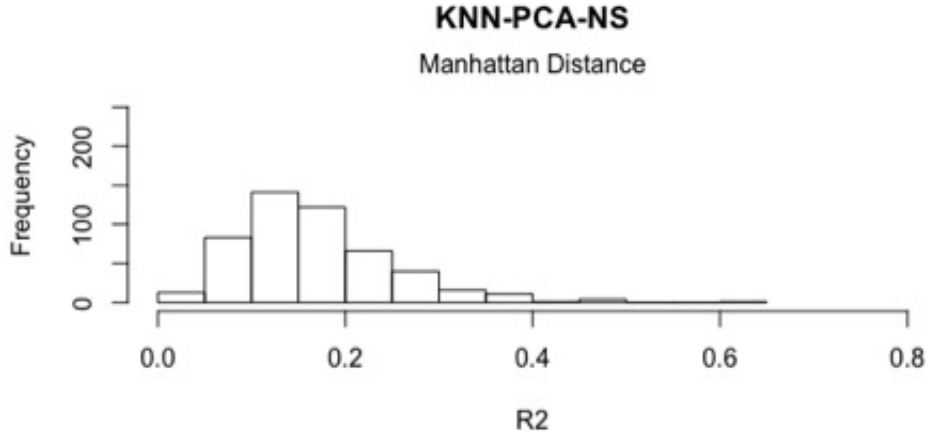

Figure 4: Range of R^2 values using the Manhattan distance

Finally, under Manhattan distance, the histogram has a wider spread and a long tail, indicating a few genes with relatively larger R^2 values. In terms of comparing with Dr. Wheeler's LASSO model, the R^2 value was combined from both individual models to yield one final R^2 value per gene. Below the R^2 values are compared.

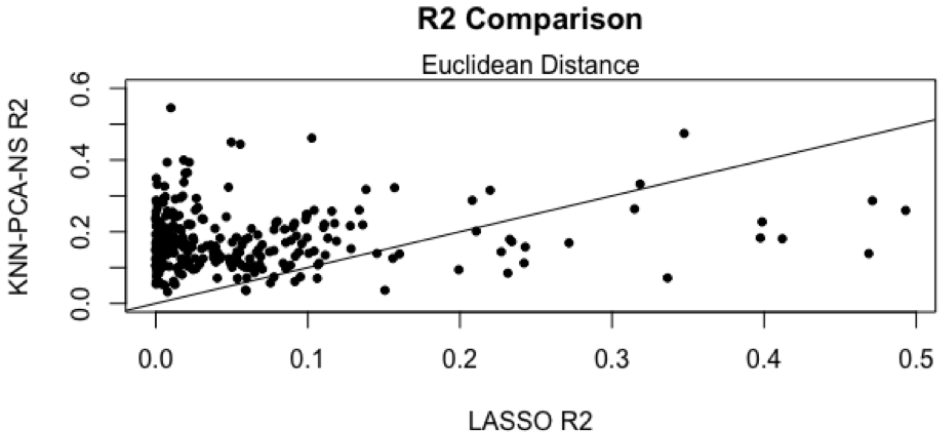

Figure 5: A plot of the R^2 values for the KNN Euclidian Distance against the LASSO values

When Euclidean distance was used, there still appears to be slightly higher performance by KNN-PCA-NS. Though not as high as when Manhattan distance was employed.

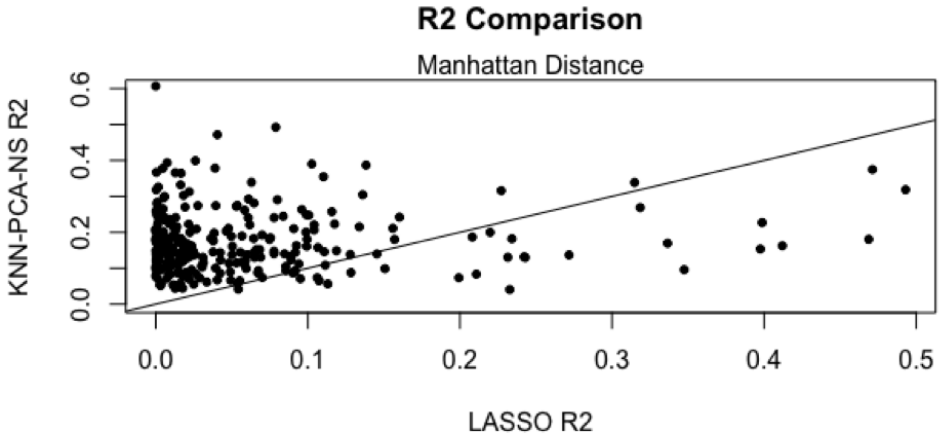

Figure 6: A plot of the R^2 values for the KNN Manhattan Distance against the LASSO values

The KNN-PCA-NS R^2 values under Manhattan distance are largely higher than the lasso model. There is a grouping at the far left that is consistently above the line. Genes with R^2 values from the KNN-PCA-NS model are significantly higher than the lasso model are identified in the following table.

The following genes were identified as having a higher R^2 value than that from the LASSO model. See Appendix to see their location on the plots above.

Genes that perform protein coding had the highest R^2 values. Based on the distance metric, different genes were identified as having higher R^2 values. Ideally, R^2 values would not be used to identify better models, however it was

Gene	Gene Name	Gene Type	Distance
ILMN1809883	2243	protein coding	Manhattan
ILMN1737788	2632	protein coding	Manhattan
ILMN1811736	4329	protein coding	Manhattan
ILMN1745513	7942	protein coding	Manhattan
ILMN1731742	14253	protein coding	Manhattan
ILMN1697957	6923	protein coding	Euclidean
ILMN1737312	12607	protein coding	Euclidean
ILMN1753340	792	pseudogene	Euclidean
ILMN1688178	11934	protein coding	Euclidean

Table 2: List of Genes that performed better with the KNN-PCA-NS model

discussed with the client that this was the desired output. Combined adjusted R^2 values cannot be provided because the model for each subset has a different number of observations and predictors, making it difficult to accurately calculate.

4.3 SVM-KNN-PCA-NS

The final step was to run the entire model and calculate two measures: R^2 for each of the bins (0,1), as well as an overall R^2 for each gene. Once this was done, the R^2 for the entire gene was compared to the R^2 value for the elastic net model. We made comparisons specifically with $\alpha = 1$ for the elastic net (which is equivalent to LASSO), as this was found previously to have the best model for the results. The R^2 values are first shown as histograms to demonstrate the distribution of the values. A plot for LASSO is included for comparison.

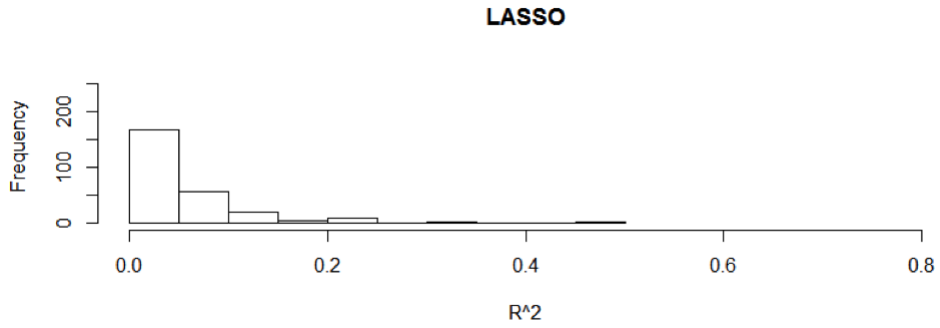

Figure 7: LASSO R^2 Values

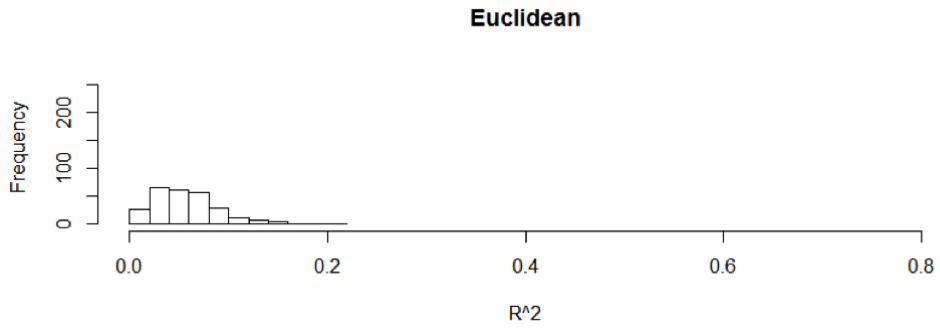

Figure 8: Euclidean R^2 Values

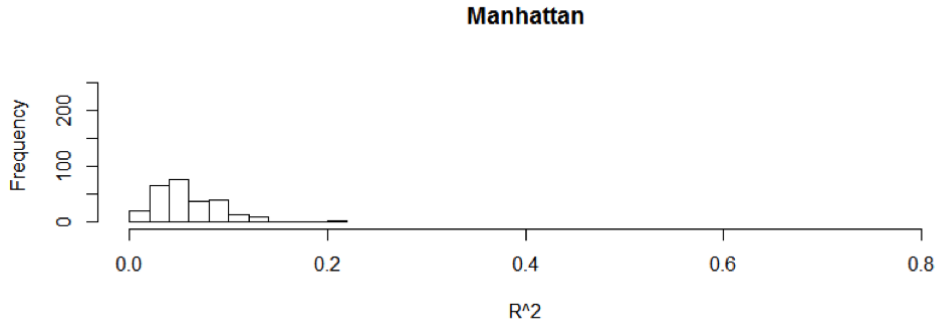

Figure 9: Manhatten R^2 Values

From figure 7, it can be seen that the range of R^2 values for the LASSO model was much greater than the range of values obtained for either Euclidean or Manhattan distances using the hybrid model. However, the vast majority of the LASSO R^2 values fell very close to zero, while the hybrid model had higher values that were more spread out between 0 and 0.1. To compare the model directly, two plots are created which directly compare R^2 values. There is one plot for each distance metric, Euclidean and Manhattan.

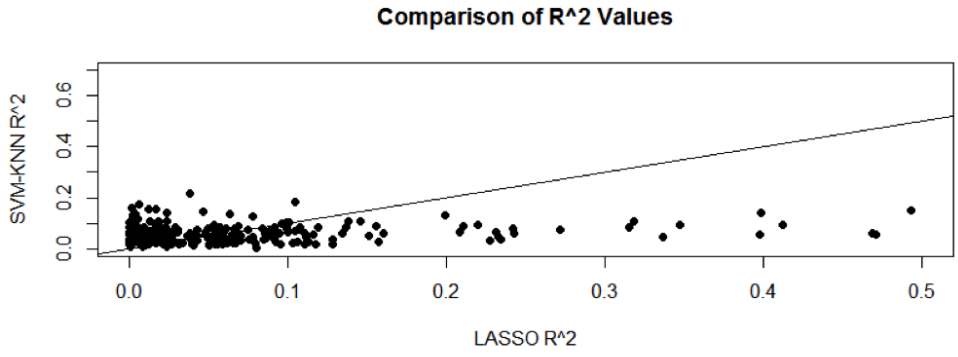

Figure 10: Comparison of R^2 values for LASSO and SVM-KNN-PCA-NS Using Euclidean Distance

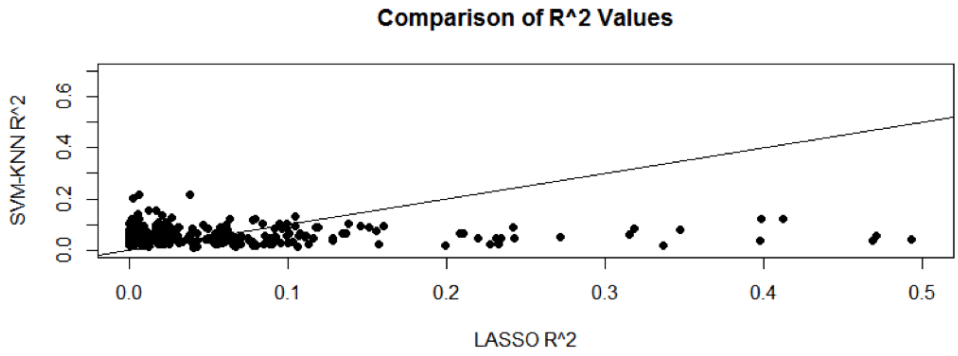

Figure 11: Comparison of R^2 values for LASSO and SVM-KNN-PCA-NS Using Manhatten Distance

What can be seen from these plots further confirms what was illustrated in the histograms. The range of R^2 values that the LASSO model takes on is much greater than the ranges for either distance metric for the hybrid model. That being said, it is also apparent that the SVM-KNN-PCA-NS model also did a better overall job of prediction, as indicated by there being more points above the line than below the line for both distance metrics. It was found that, in fact, 62% of the R^2 values for the hybrid model using Euclidean distance were higher than the values for the LASSO model. Similarly, 63% of the values were higher for the hybrid model using Manhattan distance, as compared to the LASSO model. So while this model did not obtain as high of R^2 values as the LASSO model did, it did a better job in overall prediction of the expression. The following genes were found to have significantly better predictions using the SVM-KNN-PCA-NS model then the LASSO model:

Gene	Gene Name	Gene Type	Distance
ILMN1652486	THAP7	protein coding	Manhattan
ILMN1700147	VPREB3	protein coding	Manhattan
ILMN1759219	ZMAT5	protein coding	Manhattan
ILMN1809147	FAM118A	protein coding	Manhattan
ILMN1681399	CHKB-CPT1B	protein coding	Manhattan
ILMN1667532	MICAL3	protein coding	Euclidean
ILMN1652486	THAP7	protein coding	Euclidean
ILMN1810185	CHADL	protein coding	Euclidean
ILMN1809147	FAM118A	protein coding	Euclidean
ILMN1681399	CHKB-CPT1B	protein coding	Euclidean

Table 3: List of Genes that performed better with the KNN-PCA-NS model

5 Conclusion and Future Research

Overall the model that predicted the gene expression levels the best in terms of R^2 values, is the KNN-PCA-NS model. However when you compare the PCA-NS and SVM-KNN-PCA-NS to the LASSO model, the performance is not much better. In the future trying different distance metrics could improve both the KNN-PCA-NS and the SVM-KNN-PCA-NS models. Since the dataset only has 107 observations the next step would be to obtain more data. With more data, you can create more categories for the KNN steps. For example, since the expression levels for chromosome 22 ranges from about -3 to 3, the categories could be the integers (-3, -2, ..., 2, 3) or if there is significantly more data the bins can be done in 0.5 increments. You would also be able to us k-fold cross validation for all the model and also be able to find the optimal number of knots or degrees of freedom for each spline regression for each gene.

This was a preliminary investigation that requires further analysis. With more data we could create adequate validation and training sets. It is also important to keep in mind that the categorization (0,1) has no interpretable meaning in this situation. In the future, with improved algorithms and more data, KNN could be done using meaningful categories, which could greatly affect the outcome of the analysis.

References

[1] Manor O, Segal E (2013) Robust Prediction of Expression Differences among Human Individuals Using Only Genotype Information. PLoS Genet 9(3): e1003396. doi:10.1371/journal.pgen.1003396

[2] Sivakumar, M et al. ?A Hybrid Text Classification Approach Using KNN and SVM.? International Journal of Innovative Research in Science, Engineering, and Technology Volume 3, Special Issue 3, 2014. `http://www.rroij.com/open-access/a-hybrid-text-classification-approachusing-knn-and-svm.pdf` Accessed 12 December 2016

[3] Sivakumar, M et al. ?A Hybrid Text Classification Approach Using KNN and SVM.? International Journal of Innovative Research in Science, Engineering, and Technology Volume 3, Special Issue 3, 2014. Accessed 12 December 2016

A Appendix

```
###This code takes matrix eQTL input files and runs K-Nearest Neighbors (KNN),
###Principal Component Analysis (PCA), and natural cubic splines (NS).
###By Stephanie Oliva
###12/17/2016

library(kknn)
library(splines)
library(bigpca)
library(dplyr)

################################################
##This section of the code was provided by Dr. Wheeler.
##It has been edited to include only what's needed for the KNN-PCA-NS model.

date <- Sys.Date()
args <- commandArgs(trailingOnly=T)
args <- c('22','1','YRI') #uncomment for testing in RStudio
"%&%" = function(a,b) paste(a,b,sep="")
```

```
##################################################
### Directories & Variables

exp.dir <- "~/Dropbox/STAT_consulting_data/"
snp.dir <- "~/Dropbox/STAT_consulting_data/"
snp.annot.dir <- "~/Dropbox/STAT_consulting_data/"
out.dir <- "~/Dropbox/"

chromosome <- args[1]
alpha <- as.numeric(args[2])
pop <- args[3]

### alter filenames according to how you named them
exp.file <- exp.dir %&% pop %&% "_Expression.txt.gz"
exp.annot.file <- exp.dir %&% "GRCh37_hg19_ILMN_Human-6_v2_gene_annotation_for_elasticNet.txt"
snp.file <- snp.dir %&% pop %&% "_" %&% chromosome %&% ".SNP.txt.gz"
snp.annot.file <- snp.annot.dir %&% pop %&% "_" %&% chromosome %&% ".SNP.Location.txt.gz"


##get gene pos info
gencode <- read.table(exp.annot.file, header=TRUE)
##get snp pos info
snpcode <- read.table(snp.annot.file, header=TRUE)
##get snp allele info (needed for weight output)
allelecode <- read.table(snp.dir %&% "chr" %&% chromosome %&% "_" %&% pop %&% "_alleles.txt.gz")
colnames(allelecode) <- c("CHR","POS","SNP","refAllele","effectAllele")
rownames(allelecode) <- allelecode$POS #name by position b/c we found duplicate rsids

##read exp and chr gt dosages
exp <- read.table(exp.file, header=TRUE)
gt <- read.table(snp.file, header=TRUE)

##join pos info
```

```
popgt <- left_join(snpcode,gt,by=c("snp"="id"))
popgt <- popgt[duplicated(popgt$snp)==FALSE,] #remove duplicated rsids with incorrect pos
popgt <- popgt[duplicated(popgt$pos)==FALSE,] #remove duplicated pos
rownames(popgt) <- popgt[,3] #name by position b/c we found duplicate rsids
popgt <- popgt[popgt[,3] %in% allelecode$POS,] #only keep SNPs in allelecode file (removes esv SNI
##join gene info
popexp <- left_join(gencode,exp,by=c("geneid"="id"))

popsamplelist <- colnames(exp)[-1]

#pull gene info & expression from pop of interest
popexp <- dplyr::filter(popexp,chrom==chromosome)
explist <- as.character(popexp$geneid)

##End of this section of Dr. Wheeler's code.
#####################################################

##vectors to be updated for subset categorized as 0
R2.0 <- NULL #R2
A.R2.0 <- NULL #adj R2
comp0 <- NULL #number of components
n.0 <- NULL #number of obs
p.0 <- NULL #total predictors in NS model

##vectors to be updated for subset categorized as 1
R2.1 <- NULL #R2
A.R2.1 <- NULL #adj R2
comp1 <- NULL #number of components
n.1 <- NULL #number of obs
p.1 <- NULL #total predictors in NS model

k_fin <- NULL #best K per gene
g <- NULL #gene name
r2.fin <- NULL #combined R2 for each gene
```

```
################################################
##First part of this loop written by Dr. Wheeler
################################################

for(i in 1:length(explist)){
  cat(i,"/",length(explist),"\n")
  gene <- explist[i]
  start <- popexp$s1[i] - 1e6 ### 1Mb gene lower bound for cis-eQTLS
  end <- popexp$s2[i] + 1e6 ###  1Mb gene upper bound for cis-eQTLs
  cisgenos <- subset(popgt,popgt[,3]>=start & popgt[,3]<=end) ### pull cis-SNP genotypes
  rownames(cisgenos) <- cisgenos$pos #carry positions along
  cismat <- as.matrix(cisgenos[,4:dim(cisgenos)[2]]) #get dosages only in matrix format for glmnet
  cismat <- t(cismat) #transpose to match previous code
  expmat <- as.matrix(popexp[,9:dim(popexp)[2]]) #make exp only in matrix format for glmnet
  expmat <- t(expmat) #transpose to match previous code
  colnames(expmat) <- popexp$geneid #carry gene IDs along
  if(is.null(dim(cismat))){
    #if(is.null(dim(cismat)) | gene=="ILMN_1740816"){ #special case for GIH alpha=0 to skip Error
    bestbetas <- data.frame() ###effectively skips genes with 0 cis-SNPs
  }else{
    minorsnps <- subset(colMeans(cismat), colMeans(cismat,na.rm=TRUE)>0) ###pull snps with at leas
    minorsnps <- names(minorsnps)
    cismat <- cismat[,minorsnps]
    if(length(minorsnps) < 2){###effectively skips genes with <2 cis-SNPs
      bestbetas <- data.frame() ###effectively skips genes with <2 cis-SNPs
    }else{
      exppheno <- expmat[,gene] ### pull expression data for gene
      exppheno <- scale(exppheno, center=T, scale=T)  ###scale to compare across genes
      exppheno[is.na(exppheno)] <- 0
    }
  }
##########################################
##End of Dr. Wheeler's code
```

```
########################################
colnames(exppheno) <- "expr" #naming response
pe <- cbind(exppheno, cismat) #binding response and predictors for KNN
cvdat <- as.data.frame(pe)
cvdat$expr <- ifelse(cvdat$expr < 0, 0, 1) #categorizing response as 0,1

########################################
############ KNN ########################

cv <- train.kknn(as.factor(expr) ~., data = cvdat, ks=c(2:15),
                 distance=2, kernel = "rectangular") #LOOCV for knn (much faster)
k <- unlist(cv$best.parameters[2])
k_fin[i] <-  k #saving best K
kn1 <- kknn(as.factor(expr)~., cvdat, cvdat, k=k,
            dist=2, kernel = "rectangular") #fitting knn with best K
#####dist=2 is Euclidean, dist=1 was run for Manhattan

fv <- kn1$fitted.values #saving fitted values

###subsetting data by cat 0, 1
gp1 <- which(fv==0)
dat0 <- as.data.frame(pe[gp1,])
dat1 <- as.data.frame(pe[-gp1,])

###################################
##########PCA cat 0###############

pc0 <- prcomp(t(dat0[,-1])) #running PCA
pcom0 <- pc0$rotation #principal components
co0 <- quick.elbow(pc0$sdev^2) #finding principal components that explain the
#most variability

###################################
##########NS Model cat 0#########
```

```
st0 <- lm(paste("dat0$expr ~", paste(paste0("ns(pcom0[,", 1:co0, "],3)"),
                                     collapse = "+"))) #fitting NS model with 3 df
mod.sum0 <-summary(st0)
R2.0[i] <- mod.sum0$r.squared #retrieving R2 values
A.R2.0[i] <- mod.sum0$adj.r.squared #adj R2 values
comp0[i] <- quick.elbow(pc0$sdev^2) #number of principal components

###################################
##########PCA cat 1###############

pc1 <- prcomp(t(dat1[,-1])) #PCA on subset with cat 1
pcom1 <- pc1$rotation #principal components
co1 <- quick.elbow(pc1$sdev^2) #finding principal components that explain the
#highest variability

###################################
##########NS Model cat 1#########

st1 <- lm(paste("dat1$expr ~", paste(paste0("ns(pcom1[,", 1:co1, "],3)"),
                                     collapse = "+"))) #NS model on pc

mod.sum1 <- summary(st1)
R2.1[i] <- mod.sum1$r.squared #R2 values
A.R2.1[i] <- mod.sum1$adj.r.squared #adj R2 values
comp1[i] <- quick.elbow(pc1$sdev^2) #number of pc

###################################
#####Calculating combined R2######

###predicted values
pred0 <- predict(st0)
pred1 <- predict(st1)
```

```
###mean response
ybar.0 <- mean(dat0$expr)
ybar.1 <- mean(dat1$expr)
ybar.fin <- c(rep(ybar.0, nrow(dat0)), rep(ybar.1, nrow(dat1))) #vector of means
dat.p <- rbind(dat0, dat1) #combine data for response
pred.p <- c(pred0, pred1) #combine predictions

###R2
r2.fin[i] <- 1- (sum((dat.p$expr - pred.p)^2) / sum((dat.p$exp-ybar.fin)^2))

####################################
#####Saving values of interest####

g[i] <- gene #keeping gene name

###n obs
n.0[i] <- nrow(dat0)
n.1[i] <- nrow(dat1)

###NS predictors
p.0[i] <- co0 *3
p.1[i] <- co1 *3
}


fin <- cbind(g,k_fin, comp0, n.0, p.0, R2.0, A.R2.0, comp1, n.1,
             p.1, R2.1, A.R2.1, r2.fin) #binding data
write.csv(fin, "~/Dropbox/fin_res_k2_15_dist2_euc_finalized.csv") #saving file


##########################################
###### PCA and Splines for all genes ######
##########################################
```

```
# Using Dr. Wheeler's code,
# PCA was run on every gene then natural splines
# was run using the components found.

Q.E <- rep(NA, length(exppheno))
R.Squared <- rep(NA, length(exppheno))

for(i in 1:length(explist)){
  cat(i,"/",length(explist),"\n")
  gene <- explist[i]
  start <- popexp$s1[i] - 1e6 ### 1Mb gene lower bound for cis-eQTLS
  end <- popexp$s2[i] + 1e6 ###  1Mb gene upper bound for cis-eQTLs
  cisgenos <- subset(popgt,popgt[,3]>=start & popgt[,3]<=end) ### pull cis-SNP genotypes
  rownames(cisgenos) <- cisgenos$pos #carry positions along
  cismat <- as.matrix(cisgenos[,4:dim(cisgenos)[2]]) #get dosages only in matrix format for glmnet
  cismat <- t(cismat) #transpose to match previous code
  expmat <- as.matrix(popexp[,9:dim(popexp)[2]]) #make exp only in matrix format for glmnet
  expmat <- t(expmat) #transpose to match previous code
  colnames(expmat) <- popexp$geneid #carry gene IDs along

  PCAs <- prcomp(t(cismat))
  components <- cbind(PCAs$rotation)
  Q.E[i] <- quick.elbow((PCAs$sdev)^2) # Pulls out the number of components taht explains the most
  Spline.Model <- lm(paste("exppheno ~", paste(paste0("ns(components[,", 1:Q.E[i], "], 3)"), colla
  R.Squared[i] <- summary(Spline.Model)$r.squared

  }


plot(R.Squared, Q.E)
plot(Adj.R.Squared, Q.E)

# This plots our R^2 values against Dr. Wheeler's values
plot( rsq1, R.Squared, ylab = "PCA-NS R^2", xlab = "LASSO R^2", main = "Comparing R^2 Values", yli
```

```
abline(0,1)
identify()


# This combines the two R^2 values into a dataset so we can compare
comp.r.2 <- data.frame(na.omit(cbind(seq(1,498,1), R.Squared, rsq1)))

# Note the adjusted R^2 values were not found because we are
# comparing nested models.


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


####This script takes Matrix eQTL input files and creates a weight file and a predictive performan
####The weight output file can be used to generate a .db file for use in PrediXcan with generate_s
####by Heather E. Wheeler 20160803####

###This is script is based on Heather Wheeler's original code to run an elastic net model
###The script is modified to run SVM-KNN-PCA-NS
date <- Sys.Date()
args <- commandArgs(trailingOnly=T)
args <- c('22','1','YRI') #uncomment for testing in RStudio
"%&%" = function(a,b) paste(a,b,sep="")

##############################################
### Directories & Variables

exp.dir <- "C:/Users/Nick/Dropbox/STAT_consulting_data/"
snp.dir <- "C:/Users/Nick/Dropbox/STAT_consulting_data/"
snp.annot.dir <- "C:/Users/Nick/Dropbox/STAT_consulting_data/"
out.dir <- "C:/Users/Nick/Dropbox/STAT_consulting_data/output/"
```

```
k <- 10 ### k-fold CV
n <- 1 #number of k-fold CV replicates

##alpha = The elasticnet mixing parameter, with 0??? 1.
#alpha=1 is the lasso penalty, and alpha=0 the ridge penalty.

chromosome <- args[1]
alpha <- as.numeric(args[2]) #alpha to test in CV
pop <- args[3]

### alter filenames according to how you named them
exp.file <- exp.dir %&% pop %&% "_Expression.txt.gz"
exp.annot.file <- exp.dir %&% "GRCh37_hg19_ILMN_Human-6_v2_gene_annotation_for_elasticNet.txt"
snp.file <- snp.dir %&% pop %&% "_" %&% chromosome %&% ".SNP.txt.gz"
snp.annot.file <- snp.annot.dir %&% pop %&% "_" %&% chromosome %&% ".SNP.Location.txt.gz"

################################################
### Functions & Libraries
library(glmnet)
library(dplyr)
################################################

##get gene pos info
gencode <- read.table(exp.annot.file, header=TRUE)
##get snp pos info
snpcode <- read.table(snp.annot.file, header=TRUE)
##get snp allele info (needed for weight output)
allelecode <- read.table(snp.dir %&% "chr" %&% chromosome %&% "_" %&% pop %&% "_alleles.txt.gz")
colnames(allelecode) <- c("CHR","POS","SNP","refAllele","effectAllele")
rownames(allelecode) <- allelecode$POS #name by position b/c we found duplicate rsids

##read exp and chr gt dosages
exp <- read.table(exp.file, header=TRUE)
gt <- read.table(snp.file, header=TRUE)
```

```
##join pos info
popgt <- left_join(snpcode,gt,by=c("snp"="id"))
popgt <- popgt[duplicated(popgt$snp)==FALSE,] #remove duplicated rsids with incorrect pos
popgt <- popgt[duplicated(popgt$pos)==FALSE,] #remove duplicated pos
rownames(popgt) <- popgt[,3] #name by position b/c we found duplicate rsids
popgt <- popgt[popgt[,3] %in% allelecode$POS,] #only keep SNPs in allelecode file (removes esv SNI
##join gene info
popexp <- left_join(gencode,exp,by=c("geneid"="id"))

popsamplelist <- colnames(exp)[-1]

#pull gene info & expression from pop of interest
popexp <- dplyr::filter(popexp,chrom==chromosome)
explist <- as.character(popexp$geneid)

set.seed(42)
groupid <- sample(1:10,length(popsamplelist),replace=TRUE) ##need to use same folds to compare alp

resultsarray <- array(0,c(length(explist),8))
dimnames(resultsarray)[[1]] <- explist
resultscol <- c("gene","alpha","cvm","lambda.iteration","lambda.min","n.snps","R2","pval")
dimnames(resultsarray)[[2]] <- resultscol
workingbest <- out.dir %&% "HapMap3_" %&% pop %&% "_exp_" %&% k %&% "-foldCV_elasticNet_alpha" %&%
write(resultscol,file=workingbest,ncolumns=8,sep="\t")

weightcol = c("gene", "rsid", "ref", "alt", "beta", "alpha") #col headers for use with generate_s
workingweight <- out.dir %&% "HapMap3_" %&% pop %&% "_elasticNet_alpha" %&% alpha %&% "_weights_ch
write(weightcol,file=workingweight,ncolumns=6,sep="\t")

################
#Euclidean Loop#
################
```

```
library(e1071)
library(kknn)
library(splines)

#Initializes the matrices for output data
R2.0<-matrix()
A.R2.0<-matrix()
R2.1<-matrix()
A.R2.1<-matrix()
R2.model<-matrix()

for(i in 1:length(explist)){
  cat(i,"/",length(explist),"\n")
  gene <- explist[i]
  start <- popexp$s1[i] - 1e6 ### 1Mb gene lower bound for cis-eQTLS
  end <- popexp$s2[i] + 1e6 ###  1Mb gene upper bound for cis-eQTLs
  cisgenos <- subset(popgt,popgt[,3]>=start & popgt[,3]<=end) ### pull cis-SNP genotypes
  rownames(cisgenos) <- cisgenos$pos #carry positions along
  cismat <- as.matrix(cisgenos[,4:dim(cisgenos)[2]]) #get dosages only in matrix format for glmnet
  cismat <- t(cismat) #transpose to match previous code
  expmat <- as.matrix(popexp[,9:dim(popexp)[2]]) #make exp only in matrix format for glmnet
  expmat <- t(expmat) #transpose to match previous code
  colnames(expmat) <- popexp$geneid #carry gene IDs along
  if(is.null(dim(cismat))){
    #if(is.null(dim(cismat)) | gene=="ILMN_1740816"){ #special case for GIH alpha=0 to skip Error
    bestbetas <- data.frame() ###effectively skips genes with 0 cis-SNPs
  }else{
    minorsnps <- subset(colMeans(cismat), colMeans(cismat,na.rm=TRUE)>0) ###pull snps with at leas
    minorsnps <- names(minorsnps)
    cismat <- cismat[,minorsnps]
    if(length(minorsnps) < 2){###effectively skips genes with <2 cis-SNPs
      bestbetas <- data.frame() ###effectively skips genes with <2 cis-SNPs
    }else{
      exppheno <- expmat[,gene] ### pull expression data for gene
```

```
exppheno <- scale(exppheno, center=T, scale=T)  ###scale to compare across genes
exppheno[is.na(exppheno)] <- 0

#Creats a new data set for the gene
colnames(exppheno) <- "exp"
pe <- cbind(exppheno, cismat)
pe <- as.data.frame(pe)
pe.best.k <- cbind(exppheno,cismat)
pe.best.k <- as.data.frame(pe.best.k)
pe.best.k$exp <- as.factor(ifelse(pe.best.k$exp > 0, "1", "0"))

#Determines the best k
train.k<-train.kknn(exp~.,pe.best.k,ks=c(1:15),distance=2)
best.k<-train.k$best.parameters$k

#Runs the SVM-KNN model with the best k
svm.data<-pe[,-max(dim(pe.best.k)[2])]
model.svm<-svm(exp~.,data=svm.data,scale=FALSE)
final.train<-pe.best.k[model.svm$index,-max(dim(pe.best.k)[2])]
final.test<-pe.best.k[-max(dim(pe.best.k)[2])]
model.knn<-kknn(exp~.,final.train,final.test,k=best.k,kernel="rectangular",dist=2)

#Runs the PCA and NS in each bin, and calculates overall R^2
fv <- model.knn$fitted.values
gp1 <- which(fv==0)
dat0 <- as.data.frame(pe[gp1,])
dat00<-as.data.frame(pe[gp1,-1])
dat1 <- as.data.frame(pe[-gp1,])
dat01<-as.data.frame(pe[-gp1,-1])

pc0 <- prcomp(t(dat00))
pcom0 <- pc0$rotation
z<-summary(pc0)
l<-z$importance[2,]
```

```
        m<-which(l>=.1)
        co0<-max(m)

        st0 <- lm(paste("dat0$exp ~", paste(paste0("ns(pcom0[,", 1:co0, "],3)"), collapse = "+")))
        mod.sum0 <-summary(st0)

        pc1 <- prcomp(t(dat01))
        pcom1 <- pc1$rotation
        z1<-summary(pc1)
        l1<-z1$importance[2,]
        m1<-which(l1>=.1)
        co1<-max(m1)

        st1 <- lm(paste("dat1$exp ~", paste(paste0("ns(pcom1[,", 1:co1, "],3)"), collapse = "+")))

        mod.sum1 <- summary(st1)

        ybar0<-mean(dat0$exp)
        ybar1<-mean(dat1$exp)
        SSE<-sum(sum((st0$fitted.values-dat0$exp)**2),sum((st1$fitted.values-dat1$exp)**2))
        SST<-sum(sum((dat0$exp-ybar0)**2),sum((dat1$exp-ybar1)**2))
        R2.f<-1-(SSE/SST)
      }
    }
    #Outputs R^2 and R^2 adjusted for each bin and R^2 for the entire gene
    R2.0[i] <- mod.sum0$r.squared
    A.R2.0[i] <- mod.sum0$adj.r.squared

    R2.1[i] <- mod.sum1$r.squared
    A.R2.1[i] <- mod.sum1$adj.r.squared

    R2.model[i]<-R2.f
}
```

```
################
#Manhattan Loop#
################

library(e1071)
library(kknn)
library(splines)

#Initializes the matrices the the variables that will be output
R2.01<-matrix()
A.R2.01<-matrix()
R2.11<-matrix()
A.R2.11<-matrix()
R2.model<-matrix()

for(i in 1:length(explist)){
  cat(i,"/",length(explist),"\n")
  gene <- explist[i]
  start <- popexp$s1[i] - 1e6 ### 1Mb gene lower bound for cis-eQTLS
  end <- popexp$s2[i] + 1e6 ###  1Mb gene upper bound for cis-eQTLs
  cisgenos <- subset(popgt,popgt[,3]>=start & popgt[,3]<=end) ### pull cis-SNP genotypes
  rownames(cisgenos) <- cisgenos$pos #carry positions along
  cismat <- as.matrix(cisgenos[,4:dim(cisgenos)[2]]) #get dosages only in matrix format for glmnet
  cismat <- t(cismat) #transpose to match previous code
  expmat <- as.matrix(popexp[,9:dim(popexp)[2]]) #make exp only in matrix format for glmnet
  expmat <- t(expmat) #transpose to match previous code
  colnames(expmat) <- popexp$geneid #carry gene IDs along
  if(is.null(dim(cismat))){
    #if(is.null(dim(cismat)) | gene=="ILMN_1740816"){ #special case for GIH alpha=0 to skip Error
    bestbetas <- data.frame() ###effectively skips genes with 0 cis-SNPs
  }else{
    minorsnps <- subset(colMeans(cismat), colMeans(cismat,na.rm=TRUE)>0) ###pull snps with at leas
    minorsnps <- names(minorsnps)
    cismat <- cismat[,minorsnps]
```

```
if(length(minorsnps) < 2){###effectively skips genes with <2 cis-SNPs
  bestbetas <- data.frame() ###effectively skips genes with <2 cis-SNPs
}else{
  exppheno <- expmat[,gene] ### pull expression data for gene
  exppheno <- scale(exppheno, center=T, scale=T)  ###scale to compare across genes
  exppheno[is.na(exppheno)] <- 0

  #Creates new data set for the gene
  colnames(exppheno) <- "exp"
  pe <- cbind(exppheno, cismat)
  pe <- as.data.frame(pe)
  pe.best.k <- cbind(exppheno,cismat)
  pe.best.k <- as.data.frame(pe.best.k)
  pe.best.k$exp <- as.factor(ifelse(pe.best.k$exp > 0, "1", "0"))

  #Trains to find the best k
  train.k<-train.kknn(exp~.,pe.best.k,ks=c(1:15),distance=1)
  best.k<-train.k$best.parameters$k


  #Runs the SVM-KNN model with the best k
  svm.data<-pe[,-max(dim(pe.best.k)[2])]
  model.svm<-svm(exp~.,data=svm.data,scale=FALSE)
  final.train<-pe.best.k[model.svm$index,-max(dim(pe.best.k)[2])]
  final.test<-pe.best.k[-max(dim(pe.best.k)[2])]
  model.knn<-kknn(exp~.,final.train,final.test,k=best.k,kernel="rectangular",dist=1)

  #Runs PCA then NS with the bins established from KNN
  fv <- model.knn$fitted.values
  gp1 <- which(fv==0)
  dat0 <- as.data.frame(pe[gp1,])
  dat00<-as.data.frame(pe[gp1,-1])
  dat1 <- as.data.frame(pe[-gp1,])
  dat01<-as.data.frame(pe[-gp1,-1])
```

```
      pc0 <- prcomp(t(dat00))
      pcom0 <- pc0$rotation
      z<-summary(pc0)
      l<-z$importance[2,]
      m<-which(l>=.1)
      co0<-max(m)

      st0 <- lm(paste("dat0$exp~", paste(paste0("ns(pcom0[,", 1:co0, "],3)"), collapse = "+")))
      mod.sum0 <-summary(st0)

      pc1 <- prcomp(t(dat01))
      pcom1 <- pc1$rotation
      z1<-summary(pc1)
      l1<-z1$importance[2,]
      m1<-which(l1>=.1)
      co1<-max(m1)

      st1 <- lm(paste("dat1$exp~", paste(paste0("ns(pcom1[,", 1:co1, "],3)"), collapse = "+")))

      mod.sum1 <- summary(st1)

      ybar0<-mean(dat0$exp)
      ybar1<-mean(dat1$exp)
      SSE<-sum(sum((st0$fitted.values-dat0$exp)**2),sum((st1$fitted.values-dat1$exp)**2))
      SST<-sum(sum((dat0$exp-ybar0)**2),sum((dat1$exp-ybar1)**2))
      R2.f<-1-(SSE/SST)
    }
}
#Outputs R^2 and R^2 adjusted for each bin and R^2 for the entire gene
R2.0[i] <- mod.sum0$r.squared
A.R2.0[i] <- mod.sum0$adj.r.squared

R2.1[i] <- mod.sum1$r.squared
```

```
  A.R2.1[i] <- mod.sum1$adj.r.squared

  R2.model[i]<-R2.f
}

table<-read.table("C:/Users/Nick/Documents/results.txt",header = TRUE)
table1<-read.table("C:/Users/Nick/Dropbox/STAT_consulting_data/output/HapMap3_YRI_elasticNet_alpha
table2<-read.table("C:/Users/Nick/Dropbox/STAT_consulting_data/GRCh37_hg19_ILMN_Human-6_v2_gene_an
final.dat<-read.csv("C:/Users/Nick/Dropbox/STAT_consulting_data/svm_final.csv")
finale.dat<-read.csv("C:/Users/Nick/Dropbox/STAT_consulting_data/svm_euc_final.csv")

###################
#Euclidean Results#
###################

R2.en<-table$R2
R2.en<-as.character(R2.en)
R2.en<-as.numeric(R2.en)

R2.euc<-data.frame(ID=table$gene,EN=R2.en,SK=finale.dat$R2.model)
R2.euc1<-na.omit(R2.euc)
SK.better<-R2.euc1[R2.euc1$SK>R2.euc1$EN,]

#The number of R^2 that were higher than the LASSO R^2 values
dim(SK.better)[1]/dim(R2.euc1)[1]

#Histograms comparing the LASSO model to the SVM-KNN-PCA-NS model
hist(R2.euc1$EN, xlab="R^2",main="LASSO",xlim=c(0,.8),ylim=c(0,275))
hist(R2.euc1$SK, xlab="R^2",main="Euclidean",xlim=c(0,.8),ylim=c(0,275))

#Plot comparing R^2 between the two models
plot(R2.euc1$EN,R2.euc1$SK,xlim=c(0,.5),ylim=c(0,.7),pch=16,xlab="LASSO_R^2",ylab="SVM-KNN_R^2",ma
abline(0,1)
```

```
#Identifies the important points in the plot
identify(R2.euc1$EN,R2.euc1$SK)
R2.euc1[c(12,38,197,237,264),]


####################
#Manhattan Results#
####################

R2.man<-data.frame(ID=table$gene,EN=R2.en,SK=final.dat$R2.model)
R2.man<-na.omit(R2.man)
SK.better1<-R2.man[R2.man$SK>R2.man$EN,]

#The number of R^2 that were higher than the LASSO R^2 values
dim(SK.better1)[1]/dim(R2.man)[1]

#histograms comparing the LASSO model to the SVM-KNN-PCA-NS model
hist(R2.man$EN, xlab="R^2",main="LASSO",xlim=c(0,.8),ylim=c(0,275))
hist(R2.man$SK, xlab="R^2",main="Manhattan",xlim=c(0,.8),ylim=c(0,275))

#Plot comparing R^2 values
plot(R2.man$EN,R2.man$SK,xlim=c(0,.5),ylim=c(0,.7),pch=16,xlab="LASSO R^2",ylab="SVM-KNN R^2",main
abline(0,1)

#Identifies the most important points on th eplot
identify(R2.man$EN,R2.man$SK)
R2.man[c(38,61,103,237,264),]


#####################################################################
#For loop for k-fold cross validation, not used for the paper#
#####################################################################
#SVM-KNN Hybrid
library(e1071)
```

```
library(kknn)
library(plyr)
library(splines)
#Classifies expressions as 0 or 1
pred <- read.csv("D:/pred_fin.csv")
exp <- read.csv("D:/expression_fin.csv")
pred.exp <- merge(pred,exp, by="X", all.y = T)
pred.exp$V1 <- as.factor(ifelse(pred.exp$V1 > 0, "1", "0"))
write.csv(pred.exp, file="D:/predexp.csv")

#Creates random subsets
k<-3
pred.exp$id<-sample(1:k,nrow(pred.exp),replace=TRUE)
groups<-1:k
tk<-train.kknn(V1~.,pred.exp,ks=c(1:15),distance=2,scale=FALSE)
best.k<-tk$best.parameters$k

#Euclidean Distance
means<-vector()
accs<-vector()

pb<-create_progress_bar("text")
pb$init(15)

for (i in 1:15){
  for (j in 1:3){
    set.seed(0226)
    train.temp<-subset(pred.exp,id %in% groups[-j])
    test.temp<-subset(pred.exp,id %in% c(j))

    train<-train.temp[,-4874]
    test<-test.temp[,-4874]

    ge.svm<-svm(V1~.,data=train,scale=FALSE)
```

```
      ge.knn.train<-train[ge.svm$index,]

      preds<-kknn(V1~.,train,test,k=i,kernel="rectangular",distance=2)
      meansk<-mean(preds$fitted.values==test$V1)
      means[j]<-meansk
      acc<-mean(means)
    }
    accs[i]<-acc
    pb$step()
}
#Plots the accuracy of each K after 3 fold cross validation
plot(1:15,accs,xlab="K",ylab="Accuracy",main="Accuracy for Varying Levels of K")
lines(1:15,accs,type="l")

#Chooses the best K
best.acc.pos<-which(accs==max(accs))
best.k<-max(best.acc.pos)

#Predicts the entire data set for the chromosome using the best K found
svm.data<-pred.exp[,-4874]
model.svm<-svm(V1~.,data=svm.data,scale=FALSE)
final.train<-pred.exp[model.svm$index,-4874]
final.test<-pred.exp[-4874]
model.knn<-kknn(V1~.,final.train,final.test,k=best.k,kernel="rectangular",dist=2)
mean(model.knn$fitted.values==pred.exp$V1)
```