University of Wisconsin - Madison
Department of Computer Science
Computer Science 540: Introduction to Artificial Intelligence

Using random forest for identification of additive and epistatic SNPs associated with residual feed intake in dairy cattle

Group: Elif Gunal, Ligia Moreira, Michael Ore, Sam Lemley

August 2017

Summary

Feed efficiency is highly relevant for beef and dairy production considering that feed represents 60% or more of the expenses in the farm and it can represent low rates of return for the investments in livestock. According to Sainz & Paulino (2004), residual feed intake (RFI) is defined as actual feed intake minus the expected feed intake of each animal. RFI is an important phenotype, but it is difficult and expensive to accurately measure considering this is an individual measurement for each animal. Considering these factors, Genomic Selection (GS) became an appealing alternative because of the possibility of the use of a reference population. This means not the whole population, but rather a selection of the population with the measurement for the target trait. After genotyping the members of this reference population, this information can be used in a prediction equation based on the effect of the markers on the trait. Then with this equation the predicted breeding values can be calculated and used for future selection of the other animals.

Machine learning (ML) algorithms, such as Random Forest (RF), grew out of the quest for artificial intelligence and became very popular for various ML tasks. Several works have applied ML algorithms to genomic selection, but ML and animal breeding share a very important objective predictions. Specifically, genomic selection involves big data sets, and handling them requires efficient algorithms, making ML methods particularly relevant for this field. RF algorithms generate importance scores for potential explanatory variables, and the algorithm takes into account both the main effect and interactions between variables. In genomic analysis this means additive and epistatic interactions between SNPs.

The objective of this study was to use Random Forests (RF) algorithm to identify additive and epistatic single nucleotide polymorphism (SNP) associated with RFI in dairy cattle. Genomic data included 57,348 SNP genotypes for 3,432 Holstein cows, and phenotypic measurements were daily RFI. The regression on SNP genotypes was implemented using the ranger package in R. The performance of each tree in the RF can be evaluated using the mean squared error (MSE) of the “out-of-bag” (OOB) data, which are the data points not included in the bootstrap sample. Both additive effects and interactions with other SNP will contribute to increasing the importance scores of SNP. All 57,348 SNP were ranked according to their importance scores generated by the RF and the 25 most frequent SNP were reported as possible candidates to predict RFI. The Genomic BLUP (rrBLUP) method, which assumes that all SNP have a small effect and these effects are normally distributed, was also performanced for comparison reasons using BGLR package in R.

Table of Contents

Problem Definition and Scope

Our main goal in planning this project is to evaluate genetic data in order to predict the importance of certain genetic codes in connection to desirable physical traits in livestock. We limit our scope in working towards this goal by specifically evaluating a set of single nucleotide polymorphism (SNP) data for cattle to determine importance of certain SNPs in predicting residual feed intake (RFI). Researchers sometimes use statistical models to predict RFI from SNP data, but incorporating artificial intelligence techniques as a predictive tools in this area of research is an interesting additional option that holds the potential of being more accurate or identifying new connections between SNP data and RFI (Yao et al. 2013). We have access to a large set of SNP and RFI data, which presents us with the opportunity to make use of artificial intelligence techniques to evaluate this data. We considered evaluating SNP data using a variety of machine learning algorithms, but keeping our scope in mind, we decided to focus on a single method. We decided to process our data using a random forest technique, specifically using the ranger library in R. For comparison, we made use of Genomic BLUP (rrBLUP), specifically using the BGLR package in R.

Technical Review

According to Botta et al. (2014), Random Forest (RF) is a Machine Learning method that has been applied in genome-wide association studies (GWAS) for many advantages such as building a predictive model without making any assumption about the underlying relationship between genotype and phenotype identifying other variables' importance using these models, This approach is computationally efficient considering the high dimension of GWAS datasets. Compared to classical Bayesian regression methods proposed in genomic selection, RF has much better performance in detecting interactions between single nucleotide polymorphisms (SNPs).

The broad application in the genetics and genomic analysis made RF one of the most popular machine learning methods used in these fields. This approach is a nonparametric tree-based ensemble approach that merges the ideas of adaptive nearest neighbors with bagging for effective data adaptive inference (Chen and Ishwaran, 2012).

Nguyen et al. (2015) mentioned that RF has been successfully applied to genetic data in many ways, and one of the most important uses is to rank SNPs to find the predictors for one or more variables and identify the effect related with them. The interaction between SNPs is also taken into account for RF which helps to get the most relevant SNPs associated with a variable in high dimension dataset.

In particular, we are basing much of our efforts on the paper *Random Forests approach for identifying additive and epistatic single nucleotide polymorphisms associated with residual feed intake in dairy cattle* from Yao et al. (2013). This project provides us with a significant amount of guidance as it involves processing the same sort of SNP data to predict RFI using random forests, which matches our proposed method of achieving our goal exactly.

Design Requirements

To accomplish our goal of predicting RFI given SNP data using a random forest technique, we need to design a program that:

1. Is able to accept a large, clean SNP dataset as input
2. Incorporates that data into a random forest structure
3. Presents meaningful information using the resulting random forest

To implement this design, we planned from the start to make use of the R language as it can readily handle large data sets, includes a free random forest library, and provides many avenues to present the resulting random forest information. We specifically made use of the randomForest, ranger and BGLR packages in R. We additionally considered using the Python language and the scikit-learn package to implement the random forest program, but decided against this because Elif had existing statistical analysis programs to process this data in R.

Design Description

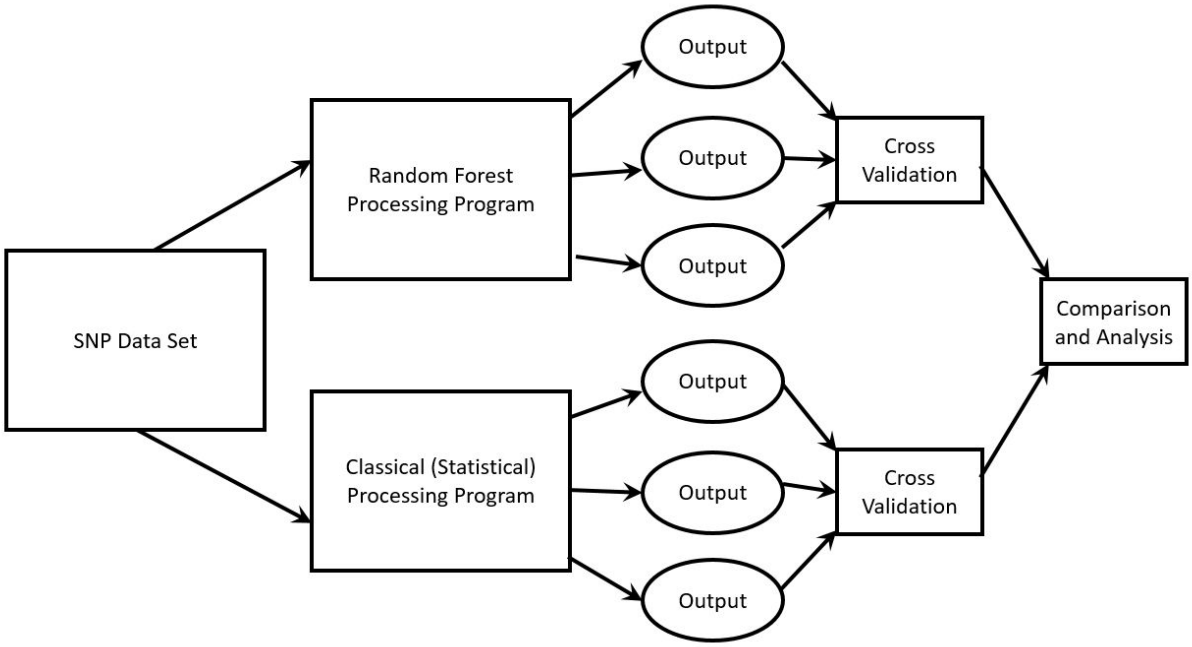
The output of our program provides a predicted importance ranking of SNPs in regards to their individual influence over RFI levels. SNPs represent certain genetic markers, and RFI represents the efficiency in which an animal converts feed into some desirable trait such as weight or milk production. As such, the type of information we generate would be useful in identifying livestock with important traits to pursue or avoid in breeding. However, since we are generating this information over a relatively limited dataset, we anticipate our output will not be immediately actionable in itself, but it will indicate interesting SNPs that would be worth examining in more depth before potentially acting upon that information in the future.

Using the predictive techniques which we implement in our project, farmers and breeders can make better decisions to maximize desirable traits in their livestock. In practice, this would involve an iterative process of an institution, possibly a university or group of large farms, gathering SNP data, processing that data using our programs, making decisions based on that output, and then beginning the process again by gathering and submitting data based on the next generation of livestock. As we cannot claim that random forest is the “best” way of generating these sorts of predictions, we intend for this data to be processed via statistical functions and other artificial intelligence methods, and we suggest that the results of all of these methods should be compared to identify the most important connections between SNPs and RFI.

Description of How Design Components Interact

The data set we are using is from the USDA National Institute of Food and Agriculture feed efficiency project, and was recorded from a population of 3,432 Holstein dairy cows. The phenotypes recorded in this data set are RFI, dry matter intake, and milk yield. The Illumina BovineSNP50 BeadChip was used for genotyping included 57,348 SNP. Cows with more than 5% missing genotypes were excluded, and SNPs with more than 1% missing values or minor

allele frequencies less than 5% were removed. The SNP genotypes at each locus were coded as 0, 1, or 2, according to the number of copies of the minor allele.



Following the process outlined in our design, we begin with this large dataset. That contains a large variety of recorded SNPs and their calculated RFI values. We input that large dataset into two different programs, one program to process the data into a random forest structure, and one program processing that data via a Bayesian statistical method. We run each program over the data multiple times to generate varying output to cross validate. We then compare the prediction accuracy of the separate genomic prediction methods, using repeated five-fold cross-validation, where we predict a random 20% of the phenotypes (testing set) from the remaining 80% of the data (training set), and we repeat this five times. Correlation between predicted and actual phenotypes in the testing set (20% withheld data) is the measure of prediction accuracy, and we can use slope and intercept of a simple linear regression of actual phenotype on predicted phenotype as well. Finally, we compare the information across the two methods. The rrBLUP method was performanced by fitting the model below:

$$\mathbf{y} = 1\mu + \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \mathbf{X}_3\boldsymbol{\beta}_3 + \boldsymbol{\varepsilon},$$

Where:

y is the phenotype vector (dry matter intake, RFI: residual feed intake and milk yield; fitted as single trait regression for each phenotype), $\boldsymbol{\mu}$ is an intercept, $\mathbf{X}_1$ is a design matrix for the effects of parity and block (contemporary group), $\boldsymbol{\beta}_1$ is the corresponding vector of effects, $\mathbf{X}_2$ is the design matrix for the effects of days in milk (dim), $\boldsymbol{\beta}_2$ is a vector of dim effects, $\mathbf{X}_3$ is the Kernel matrix (K) with marker genotypes, and $\boldsymbol{\beta}_3$ is the corresponding vector of marker effects. We treat $\boldsymbol{\beta}_1$ as “fixed” and the other two vectors of effects as random; $\boldsymbol{\beta}_2$ is treated as Gaussian and

marker effects, $\boldsymbol{\beta}_3$, are assigned Gaussian-Kernel priors and $\boldsymbol{\varepsilon}$ is the vector of residual errors which corresponds to the prior used in the Bayesian RKHS model (Perez et al. 2014; Gustavo et al. 2009). The rrBLUP analysis was done using the BGLR package in R software.

Evaluation Overview

The evaluation of our project can be divided into two categories - evaluation of our programming, and evaluation of the results generated by our programs. To assess the quality of our programs, we reviewed and tested our code incrementally as we wrote it. As for the results we generated, we analyze our findings through various methods detailed below. Despite our continued efforts, we did not achieve as high of a correlation within our test data as we intended. As we continue to refine our model and data, the findings we ultimately achieve with higher accuracy will point to interesting SNPs that have a likely impact on RFI, and which would be worthwhile to continue comparing with existing studies and would benefit from further targeted analysis.

Prototype Testing and Results

We tested the program itself incrementally as we built it. We ran into some trouble when attempting to load our large dataset in the step of specifying which columns corresponded to predictors for our input and which column corresponded to our output response. R did not handle the large number (57347) of predictor columns, and would crash. Investigating further, Michael was able to determine an alternative syntax to specify the predictor columns, which avoided this issue and ultimately led to our random forest program processing our full dataset.

Despite our continual investigation and refinement of our model, as of writing this paper, our results do not meet what we consider to be an acceptable level of accuracy. This is illustrated in Table 1. In refining our Bayesian Lasso program, we were able to achieve a correlation result of 0.71 in our training data. Similarly, we were able to achieve a high correlation of 0.996 over our training data using our Random Forest program. After training using both methods, we ran our programs over our test data. In doing so, we encountered significantly lower correlations, suggesting low accuracy of our results. Additionally, the mean squared error (MSE) of both our Bayesian Lasso and Random Forest programs matched at 2.38. This is very close to the overall sample variance of 2.42, which suggests that our ordered results are not a significant improvement over the dataset prior to processing.

Table 1: Correlation and MSE of BL and RF results

	Bayesian Lasso	Random Forest
Training Correlation	0.71	0.996
Training MSE	2.30	0.39
Test Correlation	0.20	0.13
Test MSE	2.38	2.38

Tables 2 and 3 illustrate portions of our results, specifically the top 25 most important SNPs as ranked by the RF and Bayesian methods respectively. As noted above, the correlation of our test samples were low, so we do not expect that these results are reliable representations of a true importance ranking of our data set, but we are including them as examples of our output and for discussion as to what we can do once we achieve a higher correlation. Reflecting the low correlation of these results, we can note that the top 25 SNPs in each tables have no overlap. RF and BL returned completely different results in this respect. With higher accuracy, these results would suggest that these SNPs are likely to have significant additive effect on RFI.

Our tables 2 and 3 mirror table 2 from the paper by Yao et al. (2013). Similarly to this paper, we used the SNP name which we determined as important and queried the Bovine Genome Database. In doing so, we were able to determine the chromosome associated with the specific important SNP, and if available, the nearest known genes to those SNPs.

Table 2: Top 25 important SNPs as determined by random forest ranking (genes identified using the Bovine Genome Database)

RF Rank	**SNP name**	**Chromosome**	**Gene**
1	BTB-01196519	1	
2	Hapmap23168-BTA-75567	5	TMTC2
3	ARS-BFGL-NGS-106596	6	
4	BovineHD2500004380	25	
5	ARS-BFGL-NGS-108499	12	
6	BTA-105578-no-rs	X	PCDH19
7	BTB-01296573	12	
8	ARS-BFGL-NGS-35499	18	HNRNPUL1
9	ARS-BFGL-NGS-25510	11	
10	BTB-01343372	29	NELL1

11	BovineHD2900000621	29	FAT3
12	BovineHD2300012956	23	
13	BovineHD1800017706	18	
14	Hapmap48716-BTA-31374	12	
15	Hapmap34845-BES7_Contig520_696	11	
16	ARS-BFGL-NGS-23292	19	
17	ARS-BFGL-NGS-51613	12	FREM2
18	UA-IFASA-9064	18	BCAM
19	Hapmap49810-BTA-96248	26	
20	BovineHD0200037987	2	
21	BovineHD1000006764	10	EMC9
22	ARS-BFGL-NGS-101154	7	NLRP3
23	BTB-00040563	1	
24	ARS-BFGL-NGS-101359	10	
25	BTA-28961-no-rs	8	

Table 3: Top 25 important SNPs as determined by Bayesian Lasso ranking (genes identified using the Bovine Genome Database)

BL Rank	**SNP name**	**Chromosome**	**Gene**
1	ARS-BFGL-NGS-1747	10	
2	BovineHD3000028329	30	KDM6A
3	ARS-BFGL-NGS-101151	15	OLFML1
4	BovineHD4100017231	25	MKL2
5	BovineHD0500000260	5	LGR5
6	BovineHD1800001913	18	WWOX
7	ARS-BFGL-NGS-74477	20	SLIT3
8	BTB-01916852	21	
9	Hapmap31968-BTC-056754	14	
10	BovineHD0700015218	7	NRG2
11	BovineHD1000028205	10	
12	ARS-BFGL-NGS-116076	20	CDH10
13	BovineHD0700016562	7	
14	BovineHD1800014339	18	RYR1
15	BovineHD1200011158	12	
16	ARS-BFGL-NGS-27933	8	
17	BTA-116497-no-rs	21	

18	ARS-BFGL-NGS-32374	26	
19	ARS-BFGL-NGS-108429	8	
20	BovineHD2100003624	21	
21	BovineHD1900012391	19	
22	ARS-BFGL-NGS-7711	5	PHF21B
23	BTB-00014074	1	LOC534842
24	ARS-BFGL-NGS-111283	1	SLC12AB
25	BovineHD1600006176	16	SPATA17

Overall Assessment of the Design Idea

As described in detail above, our results do not yet meet our correlation standards. We intend to continue investigating the cause of this. With further refinement, we will be able to use our programs to generate more accurate results, which we can use to identify SNPs with larger effects on RFI, but at the moment, our focus is identifying why our test correlation is lower than expected.

We have several ideas of what areas to inspect and what modifications to try next. With further development, we could improve our dataset by including chromosome information with the SNPs, which would help us better limit our data set to SNPs which already have some measured amount of importance. We would like to include other effects which are related to RFI phenotypes in our programmatic evaluation of SNPs. We will consider recalculating RFI via another method as our current method potentially could be blocking direct relationships between genomic information with measured phenotypes. At the moment, we are using a pre-calculated RFI value, but we could calculate RFI using information that we are most concerned with or which we have better data for. At a more general level, we will continue to review our program in depth with the intention of identifying any related issues which we have missed in our prior review. And finally, if these efforts have no effect on the correlation of our results, we could identify a different data set to use. If a new dataset has a higher correlation, it is possible that there is some unexpected or problematic data buried in our current data set.

Next Steps

Our immediate next step is to improve our accuracy and correlation as described above. Worth considering in conjunction with these efforts or after we establish a higher accuracy, Random Forest and Genomic BLUP analysis require time and computational performance. We would like to identify and make use of a selection of established important SNPs using GWAS. This would be helpful for analysis because we could process only the SNPs that have an established effect over RFI and then run RF analysis to rank these SNPs and classify them as the better predictor for this phenotype. This approach would decrease the size of the dataset, which would save time and improve computer performance, as well as assist us in further refining our programs. We would like to compare our results with other statistical methods to

analyze the advantages and disadvantages of each method compared to RF. We are also interested in further developing our programs to assess potential epistatic effects of SNPs which we identify as important. Our programs currently reveal additive effects, so this would require additional development and processing of our data, We estimate that we would require approximately 3 additional weeks to implement these improvements.

Additionally, we are interested in analyzing specific SNPs which we predicted would have a high impact on RFI in depth. To perform an extensive review after limiting our data set as discussed above would likely take another two weeks. Ultimately, we would like to publish our findings, which would require further analysis on our part and could take around 6 months of additional effort.

Intellectual Property Considerations

The intellectual property that emerged from this project are the R code generated to deal with large genomic datasets. When we have improved the accuracy of our programs, the results we generate and the findings we obtain from our analysis will also be included as intellectual property we created through this project. We intend to incorporate these findings into a future publication. We have protected and will continue to protect this property by keeping it confidential between our group members, as well as keeping the records for everything such as dataset, R code, and results.

References

Chen, X., Ishwaran, H.2012. Random forests for genomic data analysis. Genomics 99, 323–329.

Botta, V.; Louppe, G.; Geurts, P.; Wehenkel, L. 2014. Exploiting SNP Correlations within Random Forest for Genome-Wide Association Studies. Plos One. 9(4): e93379. doi:10.1371/journal.pone.0093379.

Elsik CG, Unni DR, Diesh CM, Tayal A, Emery ML, Nguyen HN, Hagen DE. Bovine Genome Database: new tools for gleaning function from the Bos taurus genome. Nucleic Acids Res. 2016 Jan 4;44(D1):D834-9. doi: 10.1093/nar/gkv1077. Epub 2015 Oct 19. PubMed PMID: 26481361.

Liaw, A., and M. Wiener. 2011. randomForest: Breiman and Cutler's Random Forests for classification and regression, R package version 4.6-3.

Nguyen et al. 2015. Genome-wide association data classification and SNPs selection using two-stage quality-based Random Forests. BMC Genomics. 16(Suppl 2):S5.

Pérez, P., & de los Campos, G. (2014). Genome-Wide Regression and Prediction with the BGLR Statistical Package. *Genetics,* 198(2), 483–495. http://doi.org/10.1534/genetics.114.164442.

Sainz, R. D., and P. V. Paulino. 2004. Residual feed intake. Sierra Foothill Research and Extension Center, University of California, Davis

Yao et al. 2013. Random Forests approach for identifying additive and epistatic single nucleotide polymorphisms associated with residual feed intake in dairy cattle. J. Dairy Sci. 96 :6716–6729 http://dx.doi.org/10.3168/jds.2012-6237.

Appendix A: RF Program

```
library(ranger)
library(ggplot2)

######################
## Load data
######################

# Load phenotype data
orig_pheno <- read.csv("rfiusalength42.csv")
unik <- !duplicated(orig_pheno$anim)  ## logical vector of unique values
orig_pheno=orig_pheno[unik,] ## the unique values

# Load genotype data
load("Cleaned_Genotype.RData")

orig_geno <- geno
orig_geno_tb <- data.frame(geno)

# Join phenotype and genotype data
my_merged <- merge(orig_pheno, orig_geno_tb, by.x="anim", by.y="row.names")
my_merged <- my_merged[,19:ncol(my_merged)]

save(my_merged, file="my_merged.RData")
#load("my_merged.RData")

data_c <- my_merged
```

```
nObs = nrow(data_c)

# Restrict to most significant SNPs, from single marker regression
#imp_snp <- readRDS("imp_snp.Rdata")

#orig_data_c <- data_c
#data_c <- data_c[c("RFI", as.character(imp_snp))]

# Scramble RFI (to see if that makes a difference)
#orig_data_c <- data_c
#data_c$RFI <- sample(data_c$RFI)

# Clean column names
colnames(data_c) <- make.names(colnames(data_c))

######################
## Function definitions
######################

# Create folds for cross-validation
set.seed(123)
permute = sample(nObs)

nFold = 5
tstSize = floor(nObs/nFold)

folds = matrix(NA, nFold, tstSize)

for (i in 0:(nFold-1)) {
  start = i*tstSize+1
  end = (i+1)*tstSize
  folds[i+1,] = permute[start:end]
}

# Function that returns predictions and model for given train and test set
default_ranger_model <- function(train, tst) {
  rf <- ranger(data = train, dependent.variable.name = "RFI", importance = "impurity")
  list(train.pred = predict(rf, train)$predictions,
       tst.pred = predict(rf, tst)$predictions,
       model = rf)
}

# Function that returns function that predictions and model for given train
```

```
# and test set, with specific parameters
create_ranger_model <- function(num.trees, mtry, min.node.size) {
  function(train, tst) {
    rf <- ranger(data = train, dependent.variable.name = "RFI", importance = "impurity",
                 num.trees = num.trees, mtry = mtry, min.node.size = min.node.size)
    list(train.pred = predict(rf, train)$predictions,
         tst.pred = predict(rf, tst)$predictions,
         model = rf)
  }
}

# Calculates MSE and correlation via n-fold cross-validation for given model function
calc_mse <- function(model_func) {
  train.mse.tot = 0
  tst.mse.tot = 0

  train.cor.tot = 0
  tst.cor.tot = 0
  for (i in 1:nFold) {
    train.i = c(folds[1:nFold != i,])
    tst.i = folds[i,]

    train = data_c[train.i,]
    tst = data_c[tst.i,]

    preds = model_func(train, tst)

    train.pred = preds$train.pred
    tst.pred = preds$tst.pred

    train.mse = mean((train.pred-train$RFI)^2)
    tst.mse = mean((tst.pred-tst$RFI)^2)

    train.cor = cor(train.pred, train$RFI)
    tst.cor = cor(tst.pred, tst$RFI)

    train.mse.tot = train.mse.tot + train.mse
    tst.mse.tot = tst.mse.tot + tst.mse

    train.cor.tot = train.cor.tot + train.cor
    tst.cor.tot = tst.cor.tot + tst.cor
  }
  list(train.mse.avg = train.mse.tot/nFold,
```

```
      tst.mse.avg = tst.mse.tot/nFold,
      train.cor.avg = train.cor.tot/nFold,
      tst.cor.avg = tst.cor.tot/nFold)
}

# Cross-validate RF model with a range of parameter values, for tuning
try_params <- function(param_df) {
  results <- data.frame(train.mse = rep(0, nrow(param_df)),
                        tst.mse = rep(0, nrow(param_df)))
  for (i in 1:nrow(param_df)) {
    num.trees = param_df$num.trees[i]
    mtry = param_df$mtry[i]
    min.node.size = param_df$min.node.size[i]
    mses <- calc_mse(create_ranger_model(num.trees, mtry, min.node.size))
    results$train.mse[i] <- mses$train.mse
    results$tst.mse[i] <- mses$tst.mse
  }
  results
}

######################
## Model evaluation
######################

# Evaluate base model
def_mse <- calc_mse(default_ranger_model)

# Tune mtrys paramter
mtrys = floor(32*2^(1:5))

param_df <- expand.grid(num.trees = c(256), mtry = mtrys, min.node.size = c(5))

mtrys.results <- try_params(param_df)
save(mtrys.results, file="mtrys_results.RData")

# Tune num.trees parameter
num.trees.vec = floor(64*2^(1:5))

param_df <- expand.grid(num.trees = num.trees.vec, mtry = floor(sqrt(ncol(data_c))))

num.trees.results <- try_params(param_df)
save(num.trees.results, file="num_trees_results.RData")
```

```
# Tune min.node.sizes parameter
min.node.sizes = 2*2^(3:7)

param_df <- expand.grid(num.trees = c(500), mtry = floor(sqrt(ncol(data_c))),
                        min.node.size = min.node.sizes)

min.node.results <- try_params(param_df)
save(min.node.results, file="min_node_results.RData")

# Compute significant SNPs
perm_rf <- ranger(data = data_c, dependent.variable.name = "RFI", importance = "permutation")

perm_imports = perm_rf$variable.importance
save(perm_imports, file = "perm_imports.RData")
#load("perm_imports.RData")

threshold = -sort(perm_imports)[1]

signif_snps = perm_imports[perm_imports > threshold]
```

Appendix B: Bayesian LASSO & G-BLUP Program in R BGLR package

```
install.packages("BGLR")
rm(list=ls())
library("BGLR")

orig_pheno <- read.csv("rfiusalength42.csv")
unik <- !duplicated(orig_pheno$anim)  ## logical vector of unique values
orig_pheno=orig_pheno[unik,] ## the unique values
orig_pheno$Block<- with(orig_pheno, paste0(Exp, rationsequence))#made a block from intloc,
exp and rationsequence
orig_pheno <- orig_pheno[,-which(names(orig_pheno) %in% c("instloc", "rationsequence",
"Exp"))] # delete those block elements just keep block column for them


load("Cleaned_Genotype.RData")

orig_geno <- geno
orig_geno_tb <- data.frame(geno)
```

```
my_merged <- merge(orig_pheno, orig_geno_tb, by.x="anim", by.y="row.names")
my_merged <- my_merged[,16:ncol(my_merged)]

save(my_merged, file="my_merged.RData")
load("mymerged.rdata")
##############################################################################
##### Cross-validation with using different parameters and regularization ##############
##### According to Prof. Kent Weigel advice ###################################
##############################################################################
train <- as.matrix(sample(1:3432,(3432*8)/10),replacement=FALSE)
test <- setdiff(1:3432,train)

train_splitted <- split(train, sample(1:5, nrow(train), replace=T))
tst1 <- train_splitted$`1`
tst2 <- train_splitted$`2`
tst3 <- train_splitted$`3`
tst4 <- train_splitted$`4`
tst5 <- train_splitted$`5`

geno<-as.matrix(my_merged[4:57350])
Y<-my_merged[,1:3]
X<-scale(geno,center=TRUE,scale=TRUE)
G<-tcrossprod(X)/ncol(X)

nFold <- 5
#1.Fits the G-BLUP model as H0.
H0<-list(MRK=list(K=G,model="RKHS"))
MSE<-matrix(nrow=nFold,ncol=1,NA)
COR<-matrix(nrow=nFold,ncol=1,NA)
COR_TRN<-matrix(nrow=nFold,ncol=1,NA)
C_TEST<-matrix(nrow=nFold,ncol=1,NA)
C_VAL<-matrix(nrow=nFold,ncol=1,NA)
##########################################################################
TRIED 6 DIFFERENT MODELS WITH DIFFERENT TEST AND TRAINING PARTITION
WITHIN THE FIRST TRAINING PARTITIONING SET (5-FOLD)
#### 1 ###################################################################
yNA<-Y; yNA[tst1]<-NA; yNA[test]<-NA
fm.1<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10, saveAt="RKHS_")
COR[1,1]<-cor(my_merged$RFI[tst1],fm.1$yHat[tst1])
COR_TRN[1,1]<-cor(my_merged$RFI[-tst1],fm.1$yHat[-tst1])
MSE[1,1]<-mean((fm.1$yHat[tst1]-my_merged[tst1,]$RFI)^2)
C_TEST[1,1]<-cor(my_merged$RFI[test],fm.1$yHat[test])
C_VAL[1,1]<-cor(my_merged$RFI[train],fm.1$yHat[train])
```

```
####### 2 ##############################################
yNA<-Y; yNA[tst2]<-NA; yNA[test]<-NA
fm.2<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10, saveAt="RKHS_")
COR[2,1]<-cor(my_merged$RFI[tst2],fm.2$yHat[tst2])
COR_TRN[2,1]<-cor(my_merged$RFI[-tst2],fm.2$yHat[-tst2])
MSE[2,1]<-mean((fm.2$yHat[tst2]-my_merged[tst2,]$RFI)^2)
C_TEST[2,1]<-cor(my_merged$RFI[test],fm.2$yHat[test])
C_VAL[2,1]<-cor(my_merged$RFI[train],fm.2$yHat[train])
######## 3 ######################
yNA<-Y; yNA[tst3]<-NA; yNA[test]<-NA
fm.3<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10, saveAt="RKHS_")
COR[3,1]<-cor(my_merged$RFI[tst3],fm.3$yHat[tst3])
COR_TRN[3,1]<-cor(my_merged$RFI[-tst3],fm.3$yHat[-tst3])
MSE[3,1]<-mean((fm.3$yHat[tst3]-my_merged[tst3,]$RFI)^2)
C_TEST[3,1]<-cor(my_merged$RFI[test],fm.3$yHat[test])
C_VAL[3,1]<-cor(my_merged$RFI[train],fm.3$yHat[train])
######### 4 ######################
yNA<-Y; yNA[tst4]<-NA; yNA[test]<-NA
fm.4<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10, saveAt="RKHS_")
COR[4,1]<-cor(my_merged$RFI[tst4],fm.4$yHat[tst4])
COR_TRN[4,1]<-cor(my_merged$RFI[-tst4],fm.4$yHat[-tst4])
MSE[4,1]<-mean((fm.4$yHat[tst4]-my_merged[tst4,]$RFI)^2)
C_TEST[4,1]<-cor(my_merged$RFI[test],fm.4$yHat[test])
C_VAL[4,1]<-cor(my_merged$RFI[train],fm.4$yHat[train])
######### 5 ######################
yNA<-Y; yNA[tst5]<-NA; yNA[test]<-NA
fm.5<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10, saveAt="RKHS_")
COR[5,1]<-cor(my_merged$RFI[tst5],fm.5$yHat[tst5])
COR_TRN[5,1]<-cor(my_merged$RFI[-tst5],fm.5$yHat[-tst5])
MSE[5,1]<-mean((fm.5$yHat[tst5]-my_merged[tst5,]$RFI)^2)
C_TEST[5,1]<-cor(my_merged$RFI[test],fm.5$yHat[test])
C_VAL[5,1]<-cor(my_merged$RFI[train],fm.5$yHat[train])
############## G-BLUP ###################
H0<-list(Block=list(~factor(Block),data=Y, model="BRR"),
    MRK=list(K=G,model="RKHS"))
yNA<-Y$RFI; yNA[tst5]<-NA; yNA[test]<-NA
fm.6<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10, saveAt="RKHS_")
COR[5,1]<-cor(my_merged$RFI[tst5],fm.6$yHat[tst5])
COR_TRN[5,1]<-cor(my_merged$RFI[-tst5],fm.6$yHat[-tst5])
MSE[5,1]<-mean((fm.6$yHat[tst5]-my_merged[tst5,]$RFI)^2)
C_TEST[5,1]<-cor(my_merged$RFI[test],fm.6$yHat[test])
C_VAL[5,1]<-cor(my_merged$RFI[train],fm.6$yHat[train])
```

```
################## bayesian LASSO ####################################
geno<-as.matrix(my_merged[4:57350])
Y<-my_merged[,1:3]
X<-scale(geno,center=TRUE,scale=TRUE)

H0<-list(FIXED=list(~factor(BCS), data=Y,model="FIXED"),
        Block=list(~factor(Block),data=Y, model="BRR"),
        MRK=list(X=X,  model="BL"))
yNA<-Y$RFI;
yNA[tst5]<-NA; yNA[test]<-NA

fm.6<-BGLR(y=yNA,ETA=H0,nIter=10000, burnIn=5000,thin=10)
save(fm.6,file="fm_6.rda")
COR[3,1]<-cor(my_merged$RFI[tst5],fm.6$yHat[tst5])
COR_TRN[3,1]<-cor(my_merged$RFI[-tst5],fm.6$yHat[-tst5])
MSE[3,1]<-mean((fm.6$yHat[tst5]-my_merged[tst5,]$RFI)^2)
C_TEST[3,1]<-cor(my_merged$RFI[test],fm.6$yHat[test])
C_VAL[3,1]<-cor(my_merged$RFI[train],fm.6$yHat[train])


H0<-list(Block=list(~factor(Block),data=Y, model="BRR"),
        MRK=list(X=X,  model="BL"))

C_TRAIN <- C_VAL

output[1,1] <- "RHKS"
output[1,2] <- COR_TRN
output[1,3] <- COR
output[1,4] <- C_TEST
output[1,5] <- C_TRAIN
output[1,6] <- MSE



write.table(output, paste0('output.txt',i), row.names = F, quote = F)
```