Analysis & Prediction of Complex Traits using Whole-Genome Regression Methods

Iowa State University, Sept, 2018

Labs by:

Gustavo de los Campos

(gustavoc@msu.edu)

INTRODUCTION

Modern genotyping and sequencing technologies can produce large amounts of genomic data. This information, combined with phenotypic records, can be used to learn about the underlying genetic architecture of traits and diseases and for prediction of un-observed outcomes (e.g., genetic values or disease risk). However, incorporating high dimensional genomic data into modes poses several statistical and computational challenges. Recent advance in statistical learning methods, coupled with improvements in computational power, allow implementing high-dimensional genomic regressions in parametric and semi-parametric settings. In this short course, we will discuss several parametric and non-parametric procedures for implementing Whole Genome Regressions using high dimensional genomic data.

Approach. The workshop will consist of 5 modules. Each module will provide: (a) a brief introduction to the methods and to the underlying statistical theory, (b) examples that illustrate of the use of the methodology based on simulated or real data, and (c) a discussion. The main focus of the workshop is on the use of Bayesian methods, including both parametric and semi-parametric genomic regressions. The examples are all implemented using R.

Pre-requisites. Familiarity with linear regression and basic experience with the R language is required. Background on Bayesian and Penalized regressions will be beneficial but not strictly required.

Software. Please have R and R-libraires BGLR and brrn installed. For details about installation of R see: . Once you have installed R you can install the two required libraries running

```
install.packages("BGLR", repos=" http://cran.us.r-project.org/")
install.packages("brnn", repos=" http://cran.us.r-project.org/")
```

in the R-console.

CONTENTS

LAB1: Variable Selection Using Single Marker Regression

Variable selection is commonly used in genomic analysis. We begin this lab with a discussion of the statistical properties of Ordinary Least Squares (OLS) estimates. Subsequently we study the effects of variable selection on goodness of fit and prediction accuracy.

1.1. Linear models and ordinary least squares

Consider the following model

$$y_i = \mu + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i \qquad i = 1,\dots,n$$

where y_i is the phenotype of the *i*th individual, μ is an effect common to all individuals (an "intercept"), x_{ij} are covariates (e.g., marker genotypes), β_j is the effect of the *j*th covariate and ε_i is a model residual. In matrix notation the model is expressed as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad [1]$$

where: $\mathbf{y} = \{y_i\}$ is a vector of phenotypes, $\mathbf{X} = \{\mathbf{1}, \mathbf{x}_1, \dots, \mathbf{x}_p\}$ is an incidence matrix for the vector of regression coefficients, $\boldsymbol{\beta} = (\mu, \beta_1, \dots, \beta_p)'$ and $\boldsymbol{\varepsilon} = \{\varepsilon_i\}$ is a vector of model residuals.

The ordinary least squares estimate of $\boldsymbol{\beta}$ is the solution to the following optimization problem:

$$\hat{\boldsymbol{\beta}}_{OLS} \underset{\arg\min}{=} \sum_{i}\left(y_i - \sum_j x_{ij}\beta_j\right)^2$$

where $\sum_{i}\left(y_i - \sum_j x_{ij}\beta_j\right)^2$ is a residual sum of squares. The first order conditions of the OLS optimization problem are satisfied by following system of equations: $[\mathbf{X}'\mathbf{X}]\hat{\boldsymbol{\beta}}_{OLS} = \mathbf{X}'\mathbf{y}$.

1.2. Sampling Properties of Ordinary Least Squares Estimates

The mean-squared error (MSE) of an estimator is defined as: $MSE(\hat{\theta}) = E\left[(\theta - \hat{\theta})^2\right]$ where θ is the true value of the parameter and $\hat{\theta}$ is the estimator, which is a function of the data (**X** and **y** in the regression example discussed above). The MSE can be decomposed in two components: $MSE(\hat{\theta}) = [\theta - E(\hat{\theta})]^2 + Var(\hat{\theta})$, where $[\theta - E(\hat{\theta})]^2$ and $Var(\hat{\theta})$ are the squared-bias and variance of the estimator. The expected value of OLS estimates of regression coefficients is:

$$\begin{aligned}
E[\hat{\beta}_{OLS}|\mathbf{X}] &= [\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'E[\mathbf{y}|\mathbf{X}] \\
&= [\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'E[\mathbf{X}\beta + \varepsilon|\mathbf{X}] \\
&= [\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'\mathbf{X}\beta + [\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'E[\varepsilon|\mathbf{X}] \\
&= \beta + [\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'E[\varepsilon|\mathbf{X}]
\end{aligned}$$

When model [1] holds $E[\varepsilon|\mathbf{X}]=\mathbf{0}$ and $E[\hat{\boldsymbol\beta}_{OLS}|\mathbf{X}]=\boldsymbol\beta$; therefore, if the model holds, OLS gives unbiased estimates of regression coefficients. The second term of the MSE formula, $Var(\hat\theta)$, is a frequentist measure of uncertainty and reflects variability of the estimator over repeated sampling. The asymptotic (co)variance matrix of OLS estimates of regression coefficients, given **X**, is, $Var(\hat{\boldsymbol\beta})=[\mathbf{X}'\mathbf{X}]^{-1}\sigma^2$, where σ^2 is the variance of model residuals. Therefore, the MSE of the estimate of the jth regression coefficient is $C^{jj}\sigma^2$ where C^{jj} is the jth diagonal entry of the inverse of the matrix of coefficients, that is $\mathbf{C}^{-1}=[\mathbf{X}'\mathbf{X}]^{-1}$. The variance of OLS estimates is affected by four factors: (*a*) sample size, n, (*b*) the number of predictors, *p*, (*c*) the degree of linear dependency among predictors and (*d*) the residual variance. In the following example we study how MSE of estimates of regression coefficients changes with *n* and *p*.

Example 1.1. Effects of *n* and *p* on Mean-Squared Error of OLS estimates

```r
rm(list=ls())
n<-seq(from=100,to=300,by=10) # vector defining sample size
p<-seq(from=5,to=80,by=4)     # vector defining number of predictors
x<-rbinom(prob=.5,n=max(p)*max(n),size=1) # sample predictors
X<-matrix(nrow=max(n),ncol=max(p),data=x)
varE<-1
VAR<-matrix(nrow=length(n),ncol=length(p),NA)
colnames(VAR)<-p
rownames(VAR)<-n
for(i in 1:length(n)){ # loop over sample size
   for(j in 1:length(p)){ # loop over number of predictors
     tmpX<-X[1:n[i],1:p[j]]
     C<-crossprod(tmpX)
     CInv<-chol2inv(chol(C)) # equivalent to solve(X'X)
     VAR[i,j]<-mean(diag(CInv))*varE  #average variance of estimates
   }
}
# plot Variance (equal to MSE in this case) Vs. n and p
persp(z=VAR,x=n,y=p,xlab="Sample Size",
        ylab="Number of Predictors",zlab="MSE(bj)",col=2)

## If you install library rgl can use the following, to rotate the plot
## with the mouse
library(rgl)
persp3d(z=VAR,x=n,y=p,xlab="Sample Size",
        ylab="Number of Predictors",zlab="MSE(bj)",col=2)
```

In genomic models the number of predictors (p) is large, relative to sample size (n). This induces high variance and consequently large MSE of estimates. This problem, the so-called 'curse of dimensionality' can be confronted either by reducing the number of predictors or with use of regularized regression methods. Next, we study how variable selection affects prediction accuracy. Regularized regressions are discussed later on.

1.3. Variable Selection Using Single Marker Regressions

In a Genome-Wide Association Study (GWAS) the association of each marker with the phenotype is assessed, one marker at a time, using some form of single-marker regression or association test. This information can be used to select markers to be used in prediction models. In this section we illustrate this approach using a dataset from CIMMYT's Global Wheat Breeding Program. This data set, available with BGLR package (de los Campos and Pérez 2013), contains 4 phenotypes evaluated in 599 wheat lines that were genotyped for 1,279 markers. In the examples, we use 450 lines for variable selection and model fitting (the 'training' dataset, XTRN and yTRN in the R-code below) and evaluate the prediction accuracy of each of the methods using data from 149 lines (the 'testing' dataset, XTST and yTST in the examples below).

The problem of selecting *k* out of *p* (*k*<*p*) predictors can be viewed as a model comparison problem. Ideally, we would fit all possible models and select the one that is best according to some model comparison criterion (e.g., AIC, Akaike Information Criterion, Akaike 1973). However, when *p* is large fitting all possible models is not feasible. Instead model search algorithms are used. A very simple search algorithm consists of regressing the response in each of the predictors one at a time ('single marker regression'). Each of these regressions yields a measure of association between markers and phenotypes (e.g., a p-value). Then, we can form our final model by using the first k predictors ranked according to the association measure. This approach is commonly used in Genome Wide Association Studies (GWAS). The following example fits models with *k* predictors (k=1,…,300) chosen based on the marginal association between markers and phenotypes.

Example 1.2. Variable selection using p-values derived from single-marker regressions

```
rm(list=ls())
##### DATA ##############################################
 library(BGLR)
 data(wheat); X=wheat.X; Y=wheat.Y
 objects()
 N<-nrow(X) ; p<-ncol(X)
 y<-Y[,2]
 set.seed(1235)
 tst<-sample(1:N,size=150,replace=FALSE)
 XTRN<-X[-tst,] ; yTRN<-y[-tst]; XTST<-X[tst,] ; yTST<-y[tst]
###### SINGLE MARKER REGRESSIONS ########################
 pValues<-numeric()
 for(i in 1:p){
     fm<-lm(yTRN~XTRN[,i])
     pValues[i]<-summary(fm)$coef[2,4]
     print(paste('Fitting Marker ',i,'.',sep=''))
 }
 plot(-log(pValues,base=10),cex=.5,col=2)
####### VARIABLE SELECTION ##############################
 myRanking<-order(pValues); sqCorTRN<-numeric(); sqCorTST<-numeric()
 for(i in 1:300){
     tmpIndex<- myRanking[1:i]
     fm<-lm(yTRN~XTRN[,tmpIndex])
    sqCorTRN[i]<-cor(yTRN,predict(fm))^2
     bHat<-coef(fm)[-1] ; bHat<-ifelse(is.na(bHat),0,bHat)
     yHat<-as.matrix(XTST[,tmpIndex])%*%bHat
     sqCorTST[i]<-cor(yTST,yHat)^2
     print(paste('Fitting Model with ',i,' markers!',sep=''))
 }
 plot(sqCorTRN,type='o',col=2,ylab='Squared Correlation-Training',
        xlab='Number of markers',ylim=c(0,1))
 plot(sqCorTST,type='o',col=2,ylab='Squared Correlation-Testing',
        xlab='Number of markers',ylim=c(0,.28))
```

References

Akaike, H. 1973. “Information Theory and an Extension of the Maximum Likelihood Principle.” In *Second International Symposium on Information Theory*, 1:267–281.

de los Campos, G., and P. Pérez. 2013. *BGLR=Bayesian Generalized Linear Regression*. R Package Version 1.0. https://r-forge.r-project.org/R/?group_id=1525.

Hoerl, A. E, and R. W Kennard. 1970. “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” *Technometrics* 12 (1): 55–67.

LAB2: Shrinkage Estimation

Shrinkage plays a central role in the development of models for genome-enabled prediction. In this lab we study one of the simplest and oldest shrinkage estimator: ridge regression (RR). We: (2.1) discuss the estimation procedure; (2.2) study how shrinkage controls the trade-offs between goodness of fit and model complexity; (2.3) discuss the relationship between RR and certain type of Bayesian models; and (2.4) introduce one of the most commonly used genomic models: the Genomic Best Linear Unbiased Predictor (G-BLUP), a RR-type method.

2.1. Penalized Estimates

Ordinary least squares (OLS) and Maximum likelihood (ML) estimates are derived by maximizing the fitness of the model to the training data. When the number of predictors (p) is large relative to sample size (n) these methods can large sampling variance, and consequently high mean-squared error. Penalized estimation methods are commonly used to confront some the problems emerging in highly dimensional regressions. These estimates are obtained as the solution to an optimization problem that balances goodness of fit and model complexity. The general form of the optimization problem is:

$$\hat{\boldsymbol{\beta}} \underset{\substack{\arg\min \\ \boldsymbol{\beta}}}{=} \left\{ L(\mathbf{y},\boldsymbol{\beta}) + \lambda J(\boldsymbol{\beta}) \right\} \quad [1]$$

where, $L(\mathbf{y},\boldsymbol{\beta})$ is a loss function that measure lack of fit of the model to the data, $J(\boldsymbol{\beta})$ is a measure of model complexity and $\lambda \geq 0$ is a regularization parameter controlling the trade-offs between fitness and model complexity.

Ridge Regression (Hoerl and Kennard 1970) is a particular case of [1] and is obtained by setting $L(\mathbf{y},\boldsymbol{\beta})$ to be a residual sum of squares $L(\mathbf{y},\boldsymbol{\beta}) = \sum_{\mathrm{i}} \left(y_i - \sum_j x_{ij}\beta_j \right)^2$ and $J(\boldsymbol{\beta})$ to be the sum of square of the regression coefficients; typically, some of the regression coefficients (e.g., the intercept) are not penalized; therefore, $J(\boldsymbol{\beta}) = \sum_{\mathrm{j \in S}} \beta_j^2$ where *S* define the set of coefficients to be penalized.

$$\hat{\boldsymbol{\beta}} \underset{\substack{\arg\min \\ \boldsymbol{\beta}}}{=} \left\{ \sum_{\mathrm{i}} \left(y_i - \sum_j x_{ij}\beta_j \right)^2 + \lambda \sum_{\mathrm{j \in S}} \beta_j^2 \right\} \quad [2]$$

When $\lambda \to \infty$ the solution is $\hat{\boldsymbol{\beta}}_{RR} = \mathbf{0}$. On the other extreme, as $\lambda = 0$ the solution is the OLS estimates of $\boldsymbol{\beta}$. In matrix notation problem [2] can be represented as:

$$\hat{\boldsymbol{\beta}}_{RR} \underset{\arg\min}{=} \left\{ (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right\}$$

where: $(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})=\sum_{\mathrm{i}}\left(y_i-\sum_j x_{ij}\beta_j\right)^2$ is a RSS and $\boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta}=\sum_{\mathrm{j}\in\mathrm{S}}\beta_j^2$ is a sum of squares of the regression coefficients. Here, $\mathbf{D}=Diag\{d_j\}$ is a diagonal matrix whose entries are 1 for $j\in S$ and zero otherwise. The first order conditions of the above optimization problem are satisfied by the following system of linear equations:

$$[\,\mathbf{X}'\mathbf{X}+\lambda\mathbf{D}\,]\hat{\boldsymbol{\beta}}_{RR}=\mathbf{X}'\mathbf{y} \qquad [3]$$

Relative to OLS, RR adds a constant (λ) to the diagonal entries of the coefficient matrix. This shrink estimates towards zero. Shrinkage induces bias but reduces the variance of the estimates, potentially reducing the MSE of estimates.

A simplified example. Let's consider a simple example in which each subject was assigned to one of two possible treatments (treatments 1 and 2). The treatment-means parameterization of this model is: $y_i=x_{1i}\beta_1+x_{2i}\beta_2+\varepsilon_i$ where y_i is the response, x_{1i} is a dummy variable indicator of treatment 1, $x_{2i}=(1-x_{i1})$ is a dummy variable indictor of treatment 2, β_1 and β_2 the means of treatments 1 and 2, respectively, and ε_i is a model residual. The OLS estimates of regression coefficients in this model are:

$$\begin{bmatrix}\sum_{\mathrm{i}}x_{1i}^2 & \sum_{\mathrm{i}}x_{1i}x_{2i}\\ \sum_{\mathrm{i}}x_{1i}x_{2i} & \sum_{\mathrm{i}}x_{2i}^2\end{bmatrix}\begin{pmatrix}\hat{\beta}_1\\ \hat{\beta}_2\end{pmatrix}=\begin{pmatrix}\sum_{\mathrm{i}}x_{1i}y_i\\ \sum_{\mathrm{i}}x_{2i}y_i\end{pmatrix}$$

Moreover, $\sum_{\mathrm{i}}x_{1i}^2$ and $\sum_{\mathrm{i}}x_{2i}^2$ equal the number of individuals in treatment 1 and 2 (denoted as n_1 and n_2 respectively), since x_{1i} and x_{2i} are orthogonal $\sum_{\mathrm{i}}x_{1i}x_{2i}=0$. Finally, $\sum_{\mathrm{i}}x_{1i}y_i$ and $\sum_{\mathrm{i}}x_{2i}y_i$ give simply the sum of the response variable for subjects assigned to treatments 1 and 2, respectively. Therefore,

$$\begin{bmatrix}n_1 & 0\\ 0 & n_2\end{bmatrix}\begin{pmatrix}\hat{\beta}_1\\ \hat{\beta}_2\end{pmatrix}=\begin{pmatrix}\sum_{\mathrm{i:x_{1i}=1}}y_i\\ \sum_{\mathrm{i:x_{2i}=1}}y_i\end{pmatrix}$$

, from where we conclude that the OLS estimate of the treatment mean are simply the average of the phenotypes observed in each treatment, that is $\hat{\beta}_1=\frac{\sum_{\mathrm{i:x_{1i}=1}}y_i}{n_1}$ and $\hat{\beta}_2=\frac{\sum_{\mathrm{i:x_{2i}=1}}y_i}{n_2}$. Now, considering the RR estimates, according to [3] these will be will be

$$\begin{bmatrix}n_1+\lambda & 0\\ 0 & n_2+\lambda\end{bmatrix}\begin{pmatrix}\hat{\beta}_1\\ \hat{\beta}_2\end{pmatrix}=\begin{pmatrix}\sum_{\mathrm{i:x_{1i}=1}}y_i\\ \sum_{\mathrm{i:x_{2i}=1}}y_i\end{pmatrix}$$

; therefore the RR estimates are $\hat{\beta}_1 = \frac{\sum_{i:x_{1i}=1} y_i}{n_1 + \lambda}$ and $\hat{\beta}_2 = \frac{\sum_{i:x_{2i}=1} y_i}{n_2 + \lambda}$. Therefore, adding λ to the diagonal entries of the matrix of coefficients will shrink estimates towards zero The extent of shrinkage will depend on the value of λ relative to sample size. If we fix λ, the extent of shrinkage will decrease as the number of individuals in treatment 1 and 2 (n_1 and n_2, respectively) increases. Asymptotically, if we fix λ and let the number of individuals in each treatment approach infinity, RR estimates converge to OLS estimates.

Other penalized estimators. Several alternative penalized estimation procedures have been proposed, and they differ on the choice of penalty function, $J(\boldsymbol{\beta})$. As we discussed above, in RR, the penalty is proportional to the sum of squares of the regression coefficients or L2 norm, $J(\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j^2$. A more general formulation, known as **Bridge regression** (Frank and Friedman 1993), uses $J(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left\| \beta_j \right\|^{\gamma}$ with $\gamma > 0$. RR is a particular case with $\gamma = 2$. **Subset selection** occurs as a limiting case with $\gamma \to 0$, this penalizes the number of non-zero effects regardless of their magnitude, $J(\boldsymbol{\beta}) = \sum_{j=1}^{p} 1(\beta_j \neq 0)$. Another special case, known as **LASSO** (Least Absolute Shrinkage and Selection Operator; Tibshirani 1996) occurs with $\gamma = 1$, yielding the L1 penalty: $J(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left\| \beta_j \right\|$. Using this penalty induces a solution that may involve zeroing-out some regression coefficients and shrunken estimates of the remaining effects; therefore LASSO combines features of subset selection with those of shrinkage estimation. LASSO has become very popular in several fields of applications. However LASSO and subset selection approaches have two important limitations. First, by construction, in these methods the solution admits at most n non-zero estimates of regression coefficients. In genomic models for complex traits there is no reason to restrict the number of markers with non-zero effect to be limited by sample size. Second, when predictors are correlated, a common feature of genomic data, methods performing variable selection such as the LASSO are usually outperformed by RR (Hastie, Tibshirani, and Friedman 2009). Therefore, in an attempt to combine the good features of RR and of Lasso in a single estimation framework (Zou and Hastie 2005) proposed to use as penalty a weighted average of the L1 and L2 norm, that is, for $0 \le \alpha \le 1$, $J(\beta) = \alpha \sum_{j=1}^{p} \left\| \beta_j \right\| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2$ and termed the method the **Elastic Net** (EN). This model involves then two tuning parameters which need to be specified, the regularization parameter (λ) and α.

2.2. Effect of regularization on estimates, goodness of fit and model DF

In penalized regressions, the regularization parameter (λ) controls the trade-offs between model goodness of fit and model complexity. This affects parameter estimates (their value, and the statistical properties) model goodness of fit to the training dataset and the ability of the model to predict un-observed phenotypes.

Model complexity. The complexity of a linear model can be measured by the degree of freedom of the model. In RR, predictions are computed as $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}_{RR} = \mathbf{X}[\mathbf{X}'\mathbf{X} + \lambda \mathbf{D}]^{-1}\mathbf{X}'\mathbf{y} = \mathbf{H}_{RR}\mathbf{y}$ where $\mathbf{H}_{RR} = \mathbf{X}[\mathbf{X}'\mathbf{X} + \lambda \mathbf{D}]^{-1}\mathbf{X}'$ is the Hat matrix. If we set $\lambda = 0$ we obtain the Hat matrix of OLS: $\mathbf{H}_{OLS} = \mathbf{X}[\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'$. In linear models degree of freedom are equal to the sum of the diagonal entries of **H**. In OLS this just equals the rank of **X** (the number of predictors, if **X** is full-column rank). In RR λ also affects DF. The following R-code fits RR over a grid of values of λ and evaluates the impact that λ

has on goodness of fit to the training data, prediction accuracy, and model degree of freedom.

Example 2.1. Effects of regularization on goodness of fit and model DF

```
 rm(list=ls())
  ##### DATA ##############################################
  library(BGLR)
  data(wheat); X=wheat.X; Y=wheat.Y
  objects()
  N<-nrow(X) ; p<-ncol(X)
  y<-Y[,2]
  set.seed(12345)
  tst<-sample(1:N,size=150,replace=FALSE)
  XTRN<-X[-tst,]
  yTRN<-y[-tst]
  XTST<-X[tst,]
  yTST<-y[tst]

  ## FITTING MODEL OVER A GRID OF VALUES OF lambda
  lambda<-c(5,10,50,100,200,500,700,1000, 2000, 5000,20000)
  ZTRN<-cbind(1,XTRN) ; ZTST<-cbind(1,XTST)
  sqCorTRN<-numeric();  sqCorTST<-numeric(); DF<-numeric()
  BHat<-matrix(nrow=ncol(XTRN),ncol=length(lambda),NA)

  C0<-crossprod(ZTRN)
  rhs<-crossprod(ZTRN,yTRN)

  for(i in 1:length(lambda)){ #loop over values of lambda
    C<-C0
    # adds lambda to the diagonal of C (starts at 2nd diagonal entry)
    for(j in 2:ncol(C)){   C[j,j]<-C[j,j]+lambda[i]    }
    CInv<-chol2inv(chol(C))
    sol<-crossprod(CInv, rhs)
    BHat[,i]<-sol[-1]
    yHatTRN<-ZTRN%*%sol
    sqCorTRN[i]<-cor(yTRN,yHatTRN)^2
    yHatTST<-ZTST%*%sol
    sqCorTST[i]<- cor(yTST,yHatTST)^2
    H<-ZTRN%*%CInv%*%t(ZTRN)
    DF[i]<-sum(diag(H))
    print(i)
   }
   write(sqCorTST,file="sqCorTST.txt")
   write(lambda,file="lambda.txt")
# (Plots in next page)
```

Example 2.1. (from previous page)

```
## PLOT 1: Model Degree of freedom
  plot(DF~log(lambda),type="o",col=2,
       xlab= expression(paste(log(lambda))),
       ylab="DF",ylim=c(0,max(DF)));abline(h=1,lty=2)

  ## PLOT 2: Estimates (shrinkage by marker)
  marker<-1000 # (choose a number between 1 and 1279, try 150…)
  plot(BHat[marker,]~log(lambda),type="o",col=2,main=marker,
       xlab=expression(paste(log(lambda))),ylab="Estimate")
  abline(h=0)
  tmp<-range(BHat[,c(1,5)])

  ## PLOT 3: Estimates (shrinkage all markers)
  y<-as.vector(BHat)
   x<-rep(lambda,each=nrow(BHat))
  boxplot(y~x,col=4,xlab=expression(lambda),
               ylab='Ridge Regression Estimates of Effects')

  ## PLOT 4: Goodness of fit to TRN dataset
  plot(sqCorTRN~log(lambda),type="o",col=2,main="Training data",
          xlab=expression(paste(log(lambda))),ylab="Squared Corr.")

  ## PLOT 5 Prediction Accuracy
  plot(sqCorTST~log(lambda),type="o",col=2,main="Testing data",
          xlab=expression(paste(log(lambda))),ylab="Squared Corr.")
```

2.3. Bayesian View of Ridge Regression

Most penalized estimators can be viewed as posterior modes in certain class of Bayesian models. For instance, RR estimates are equivalent to the posterior mode of the vector of regression coefficients in a Bayesian model with a Gaussian likelihood and a Gaussian prior for the vector of regression coefficients. To see this, recall that that estimates in RR are obtained as the solution to the following optimization problem:

$$\hat{\boldsymbol{\beta}}_{RR} \underset{\arg\min}{=} \left\{ (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right\}$$

Multiplying the objective function by -1/2 and switching from minimization to maximization do not affect the solution; therefore,

$$\hat{\boldsymbol{\beta}}_{RR} \underset{\arg\max}{=} \left\{ -\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda \frac{1}{2} \boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right\}$$

Let $\lambda = \dfrac{\sigma_\varepsilon^2}{\sigma_\beta^2}$ where, σ_ε^2 and σ_β^2 are non-negative constants. Replacing above and dividing the objective function by σ_ε^2 preserves the solution, with this we get:

$$\hat{\boldsymbol{\beta}}_{RR} \underset{\arg\max}{=} \left\{ -\frac{1}{2\sigma_\varepsilon^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \frac{1}{2\sigma_\beta^2} \boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right\}$$

Finally, applying the exponential function to the objective function maintains the solution unchanged, therefore:

$$\hat{\boldsymbol{\beta}}_{RR} \underset{\arg\max}{=} \left\{ \exp\left[-\frac{1}{2\sigma_\varepsilon^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \frac{1}{2\sigma_\beta^2}\boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right] \right\}$$

$$\underset{\arg\max}{=} \left\{ \exp\left[-\frac{1}{2\sigma_\varepsilon^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \right] \exp\left[-\frac{1}{2\sigma_\beta^2}\boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right] \right\}$$

The first component of the objective function, $\exp\left[-\frac{1}{2\sigma_\varepsilon^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \right]$, is proportional to a Gaussian likelihood, centered at $\mathbf{X}\boldsymbol{\beta}$ and with (co)variance matrix $\mathbf{I}\sigma_\varepsilon^2$. The second component, $\exp\left[-\frac{1}{2\sigma_\beta^2}\boldsymbol{\beta}'\mathbf{D}\boldsymbol{\beta} \right]$, is proportional a Gaussian prior for the regression coefficients, centered at zero and with (co)variance matrix $\mathbf{D}^{-1}\sigma_\beta^2$. Therefore, estimates obtained with RR are equivalent to the posterior mode of regression coefficients in the following Bayesian model.

$$\begin{cases} \text{Likelihood}: [\mathbf{y} \mid \boldsymbol{\beta}, \sigma_\varepsilon^2] \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{I}\sigma_\varepsilon^2) \\ \text{Prior}: \quad [\boldsymbol{\beta} \mid \sigma_\beta^2] \sim N(\mathbf{0}, \mathbf{D}^{-1}\sigma_\beta^2) \end{cases} \quad [4]$$

The posterior distribution of $\boldsymbol{\beta}$ in the above model is multivariate normal with a mean (co-variance matrix) equal to the solution (inverse of the coefficient matrix) of the following system: $[\mathbf{X}'\mathbf{X} + \lambda\mathbf{D}]\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$; this is just the RR equations. This is also the Best Linear Unbiased Predictor (BLUP) of $\boldsymbol{\beta}$ given **y**.

Recall that the ratio $\frac{\sigma_\varepsilon^2}{\sigma_\beta^2}$ is equivalent to λ in RR. In a fully-Bayesian models we assign priors to each of these variance parameters, this allow inferring these unknowns from the same training data that is used to estimate marker effects. The following example fits a Bayesian RR using the R-package BGLR (de los Campos and Pérez 2013), after you run the model:

- type str(fm) in the R-console and inspect what BGLR returns (see help(BGLR) for further details,
- Check the posterior mean of σ_ε^2 and σ_β^2 (fm$varE and fm$ETA[[1]]$varB, respectively), remember the ratio of these variances is interpretable as λ in RR,
- Examine trace plots (see code in lines 31-34 in the example below)
- Compare prediction accuracy of the fully-Bayesian method versus RR.

Example 2.2. Bayesian Ridge Regression Using BGLR

```r
rm(list=ls())
##### DATA (same as Example 2.1) ########################################
 library(BGLR)
 data(wheat); X=scale(wheat.X)/sqrt(ncol(wheat.X)); Y=wheat.Y
 objects()
 N<-nrow(X) ; p<-ncol(X)
 y<-Y[,2]
 set.seed(12345)
 tst<-sample(1:N,size=150,replace=FALSE)
 XTRN<-X[-tst,]
 yTRN<-y[-tst]
 XTST<-X[tst,]
 yTST<-y[tst]

## Fits the model
  ETA<-list(list(X=XTRN,model="BRR"))
  fm<-BGLR(y=yTRN,ETA=ETA,nIter=52000,burnIn=2000)

## trace plots
  ## residual variance
   varE<- scan("varE.dat")
   plot(varE,type="o",col=2,cex=.5)
   abline(h=fm$varE,col=4, lwd=2)
   abline(v=200,col=4,lwd=2)

  ## variance of marker effects
   varB<- scan("ETA_1_varB.dat")
   plot(varB,type="o",col=2,cex=.5)
   abline(h=fm$ETA[[1]]$varB,col=4, lwd=2)
   abline(v=200,col=4,lwd=2)

  ## ratio of variance components
   lambda<-varE/varB
   plot(lambda,type="o",col=2)
   abline(h=mean(lambda[-c(1:200)]),col=4, lwd=2,cex=.5)
   abline(v=200,col=4,lwd=2)

## Posterior mean of Bayesian model Vs RR-estimate
    # Computing Ridge regression estimate
     C=crossprod(X)
     diag(C)=diag(C)+mean(lambda[-c(1:200)])
     rhs=crossprod(X,y)
     bRR=chol2inv(chol(C))%*%rhs
     plot(y=bRR,x=fm$ETA[[1]]$b);abline(a=0,b=1,col=4,lwd=2)


## Prediction Accuracy: Bayesian vs grid search with CV
  x<-scan(file="lambda.txt")
  y<-scan(file="sqCorTST.txt")
  plot(y~log(x),type="o",col=2,
         xlab=expression(paste(log(lambda))),ylab="Squared Corr.",
         ylim=c(0.1,.3))
  abline(v= log(fm$varE/fm$ETA$X$varB),col=4)
  abline(h=cor(yTST,XTST%*%fm$ETA[[1]]$b)^2,col=4)
```

2.4. The Genomic Best Linear Unbiased Predictor (G-BLUP)

The G-BLUP ('Genomic Best Linear Unbiased Predictor', e.g., VanRaden, 2008) is one of the most widely used models in genomic prediction. This model is equivalent to the Bayesian Ridge Regression model of expression [4]. The G-BLUP is also equivalent to the canonical pedigree model (the so-called Animal Model) when the (pedigree-based) numerator relationship matrix is replaced with a genomic relationship matrix. In this section we demonstrate the equivalence and provide an example where the G-BLUP model is implemented in a Bayesian context using BGLR.

In the Bayesian Ridge Regression Model of expression [4] the data equation is: $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$; in this model marker effects are assigned IID normal priors, $\boldsymbol{\beta} \sim N\left(\mathbf{0}, \boldsymbol{I}\sigma_\beta^2\right)$. To show the equivalence of this model with the G-BLUP we make of change of variables using $\boldsymbol{u} = \boldsymbol{X\beta}$; here u is a vector whose entries are the evaluations of the conditional expectation function, $u_i = \sum_{j=1}^{p} x_{ij}\beta_j$. Using **u** in place of $\boldsymbol{X\beta}$ the data equation becomes $\boldsymbol{y} = \boldsymbol{u} + \boldsymbol{\varepsilon}$. The vector of random effects **u** is a linear combination of IID normal random variables (the marker effects). The multivariate normal density is closed under linear transformations; therefore, $\boldsymbol{u}$ follows a multivariate normal density. The mean and co-variance matrix of **u** are given by: $E(\boldsymbol{u}) = E(\boldsymbol{X\beta}) = \boldsymbol{X}\boldsymbol{E}(\boldsymbol{\beta}) = \mathbf{0}$ and $Cov(\boldsymbol{u}) = Cov(\boldsymbol{X\beta}) = \boldsymbol{X}Cov(\boldsymbol{\beta})\boldsymbol{X}' = \boldsymbol{XX}'\sigma_\beta^2$; therefore $\boldsymbol{u} \sim N\left(\mathbf{0}, \boldsymbol{XX}'\sigma_\beta^2\right)$. Usually the matrix $\boldsymbol{XX}'$ is scaled by dividing its entries by the sum of the variance of the markers (K); therefore, $\boldsymbol{u} \sim N(\mathbf{0}, \boldsymbol{G}\sigma_u^2)$ where $\boldsymbol{G} = \boldsymbol{XX}'K^{-1}$, and $\sigma_u^2 = K\sigma_\beta^2$. With this the G-BLUP model becomes

$$\begin{cases} \text{Data} - \text{equation: } \boldsymbol{y} = \boldsymbol{u} + \boldsymbol{\varepsilon} \\ \text{Probability assumptions: } p(\boldsymbol{u}, \boldsymbol{\varepsilon} | \sigma_\varepsilon^2, \sigma_u^2) = N(\boldsymbol{\varepsilon} | \mathbf{0}, \boldsymbol{I}\sigma_\varepsilon^2) N(\boldsymbol{u} | \mathbf{0}, \boldsymbol{G}\sigma_u^2) \end{cases}$$

The structure of this model is essentially that of the Animal model with the pedigree-based numerator relationship matrix replaced with a genomic relationship matrix. Usually the variance components are unknown; these can be estimated using likelihood (e.g., maximum likelihood or REML) or Bayesian methods. The following example provides an application of the G-BLUP model using the wheat data set.

Example 2.3. Ridge Regression and G-BLUP

```
rm(list=ls())
### DATA #######################################################
  library(BGLR); data(wheat); X=wheat.X; Y=wheat.Y; y<-Y[,1]
### Computing the genomic relationship matrix (let's discuss alternatives)
  X<-scale(X)/sqrt(ncol(X))
  G<-tcrossprod(X)

### GBLUP & BRR (two parametrizations of the same model)
 fmGBLUP<-BGLR(y=y,ETA=list(list(K=G,model="RKHS")),
                nIter=12000,burnIn=2000,saveAt="GBLUP_")

 fmBRR<-BGLR(y=y,ETA=list(list(X=X,model="BRR")),
                nIter=12000,burnIn=2000,saveAt="BRR_")
  fmBRR$varE; fmBRR$ETA[[1]]$varB
  fmGBLUP$varE; fmGBLUP$ETA[[1]]$varU

 plot(fmBRR$yHat,fmGBLUP$yHat,col=2);abline(a=0,b=1,col=4)


```

References

de los Campos, G., and P. Pérez. 2013. *BGLR=Bayesian Generalized Linear Regression*. R Package Version 1.0. https://r-forge.r-project.org/R/?group_id=1525.

Frank, I.E., and J.H. Friedman. 1993. "A Statistical View of Some Chemometrics Regression Tools." *Technometrics*: 109–135.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. 2009. Corr. 3rd printing 5th Printing. Springer.

Hoerl, A. E, and R. W Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 12 (1): 55–67.

Tibshirani, R. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1): 267–288.

Zou, H., and T. Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2): 301–320.

LAB3: The ‘Bayesian Alphabet’

In LAB2 we discussed Ridge Regression (RR) and a Bayesian equivalent method. In RR shrinkage tends to be homogeneous across marker effects. This approach may not be optimal if some markers are in LD with QTL and others are in complete LE with QTL. Dealing with these types of problems requires implementing models that perform variable selection and shrinkage simultaneously. In a penalized regression this can be attained using penalties such as the L1 norm (e.g., the Lasso). In a Bayesian setting variable selection and differential shrinkage of estimates of effects can be obtained using priors other than the Gaussian (e.g. finite mixtures, or priors from the thick-tailed family). In this LAB we discuss a family of models, hereinafter referred as to the Bayesian Alphabet, that share the same likelihood function but differ on the prior used for marker effects. We briefly review these methods (3.1), illustrate the behavior of these methods in a simplified simulation setting (3.2) and discuss an application to real data (3.2.).

3.1. The Bayesian Alphabet

In standard parametric genomic regressions phenotypes, y_i, are regressed on marker covariates, $\{x_i\}$, using a linear model of the form $y_i = \mu + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i$, where μ is an effect common to all subjects (i.e., an ‘intercept’), $\{x_{ij}\}$ are marker genotypes (usually coded as 0,1,2) , $\{\beta_j\}$ are marker effects and ε_i is a model residuals. A standard practice for continuous traits is to assume that model residuals are IID normal, this yields the following likelihood function:

Likelihood: $p(\mathbf{y}|\mu,\beta,\sigma^2) = \prod_{i=1}^{n} N\left(y_i|\mu + \sum_{j=1}^{p} x_{ij}\beta_j, \sigma^2\right)$ [1]

where, $N\left(y_i|\mu + \sum_{j=1}^{p} x_{ij}\beta_j, \sigma^2\right)$ is a normal density for the random variable y_i centered at $\mu + \sum_{j=1}^{p} x_{ij}\beta_j$ and with variance σ^2.

With dense panels, the number of markers (p) vastly exceeds the number of data points (n) and because of this penalized or Bayesian shrinkage estimation methods are commonly used. In a Bayesian setting, shrinkage of estimates of effects is controlled by the choice of prior density assigned to marker effects. The joint prior density of the unknowns is commonly structured as follows:

Prior:

$$p(\mu,\beta,\sigma^2|df,S,\omega) \propto \left\{\prod_{j=1}^{p} p(\beta_j|\theta_{\beta_j},\sigma^2)p(\theta_{\beta_j}|\omega)\right\}\chi^{-2}(\sigma^2|df,S) \qquad [2]$$

Above, a flat prior was assigned to the intercept, $\chi^{-2}\left(\sigma^2 \middle| df, S\right)$ is a scaled-inverse Chi-squared density assigned to the residual variance and with *df* degree of freedom and scale equal to *S*, $p\left(\beta_j \middle| \theta_\beta, \sigma^2\right)$ denotes the prior density of the jth marker effect, θ_{β_j} is a vector of parameters indexing the prior density assigned to marker effects, $p\left(\theta_{\beta_j} \middle| \omega\right)$ is the prior density assigned to θ_{β_j} and ω are parameters indexing this density. The marginal prior density of marker effects is obtaining by integrating θ_{β_j} out, $p\left(\beta_j \middle| \sigma^2, \omega\right) = \int p\left(\beta_j \middle| \theta_{\beta_j}, \sigma^2\right) p\left(\theta_{\beta_j} \middle| \omega\right) \partial\theta_{\beta_j}$.

Using Bayes rule, the posterior density of model unknowns given the data is proportional to the product of the likelihood, given in eq. [1], and the prior density, eq. [2], that is:

Posterior density:

$$\begin{aligned} p\left(\mu, \beta, \sigma^2 \middle| \mathbf{y}, df, S, \omega\right) &\propto \prod_{i=1}^{n} N\left(y_i \middle| \mu + \sum_{j=1}^{p} x_{ij}\beta_j, \sigma^2\right) \\ &\times \left\{ \prod_{j=1}^{p} p\left(\beta_j \middle| \theta_{\beta_j}, \sigma^2\right) p\left(\theta_{\beta_j} \middle| \omega\right) \right\} \chi^{-2}\left(\sigma^2 \middle| df, S\right) \end{aligned}, \qquad [3]$$

The Bayesian Alphabet. Following the seminal contribution of Meuwissen, Hayes, and Goddard (2001) several linear Bayesian regression methods have been proposed and used for simulation and real data analysis. They differed in the choice of prior density assigned to marker effects. In a **Bayesian Ridge** regression (BRR), the conditional prior assigned of marker effects are IID normal, $p\left(\beta_j \middle| \theta_{\beta_j}, \sigma^2\right) = N\left(\beta_j \middle| 0, \sigma_\beta^2\right)$ and $p\left(\theta_{\beta_j} \middle| \omega\right) = \chi^{-2}\left(\sigma_\beta^2 \middle| df_\beta, S_\beta\right)$.

A second group of models, which includes **BayesA** (Meuwissen, Hayes, and Goddard 2001) and the **Bayesian LASSO** (BL, Park and Casella 2008) use priors from the thick-tail family (scaled-t in BayesA and Double Exponential in the BL). These priors induce a different type of shrinkage than the one induced by the BRR.

A third group of models, which include BayesB (Meuwissen, Hayes, and Goddard 2001) and the spike-slab models (Ishwaran and Rao 2005) use priors that are mixtures of a peak (or a spike) of mass at zero and of a continuous density (e.g., t, or normal). Figure 1 shows the densities of a Gaussian and Double Exponential densities and that of a mixture model with a peak of mass at zero and a Gaussian slab. The three densities have mean equal to zero and variance equal to one.

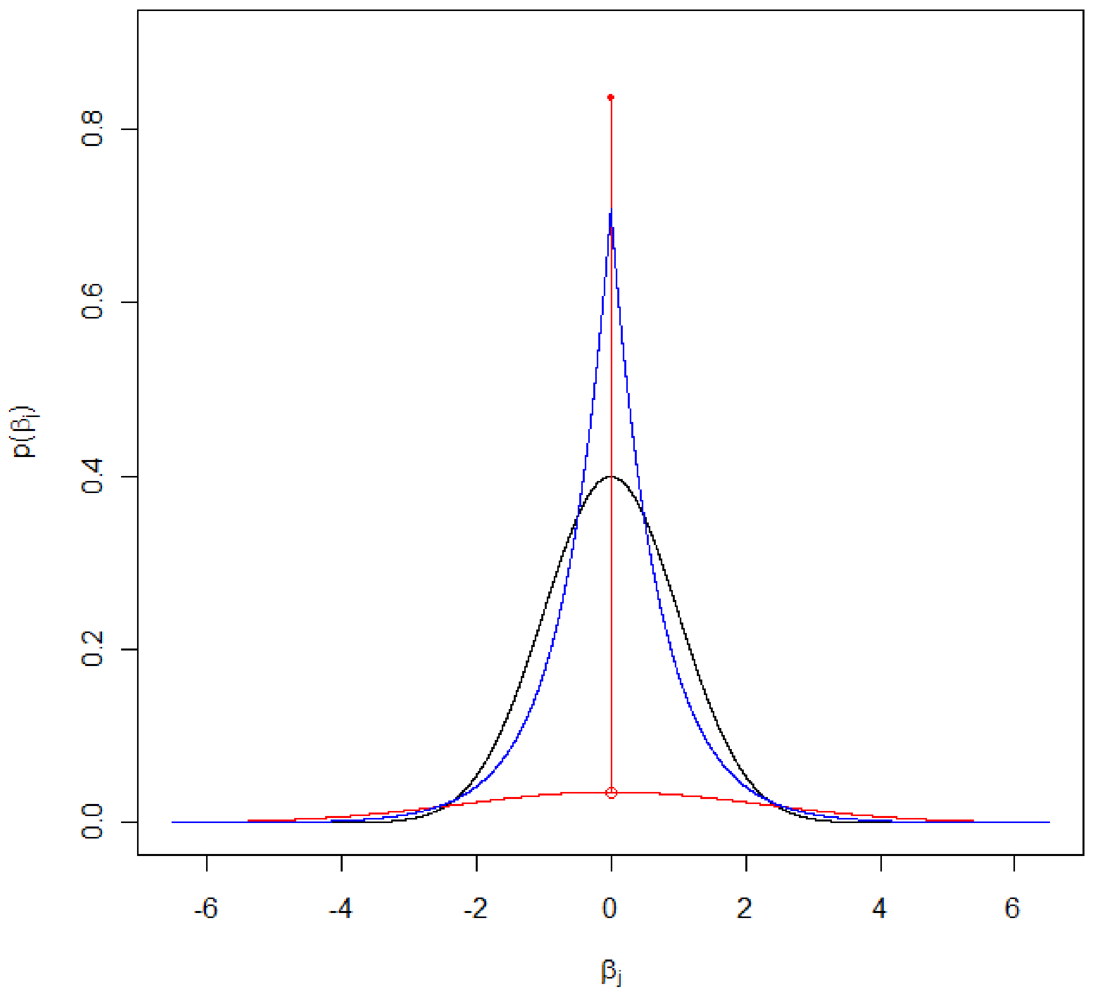

Figure 1. Density of a standard normal random variable (black), of a double-exponential random variable (blue) and of a random variable following a mixture density with a mass point at zero (with probability 0.8) and a Gaussian process with probability 0.2. All variables with zero mean and variance equal to one.

Many of the thick tail distributions, such as the *t* or the double-exponential densities can be represented as infinite mixtures of scaled normal densities. For instance, the t-prior density assigned to marker effects in **BayesA** (Meuwissen, Hayes, and Goddard 2001) can be represented as $t(\beta_j | df_\beta, S_\beta) = \int N(\beta_j | 0, \sigma^2_{\beta_j}) \chi^{-2}(\sigma^2_{\beta_j} | df_\beta, S_\beta) \partial\sigma^2_{\beta_j}$ where df_β and S_β are prior degree of freedom and scale parameters and $\chi^{-2}(\sigma^2_{\beta_j} | df_\beta, S_\beta)$ is a scaled-inverse Chi-squared density.

In the **Bayesian LASSO** (Park and Casella 2008) the Double-exponential prior density is represented as: $DE(\beta_j | \lambda^2, \sigma^2_\varepsilon) = \int N(\beta_j | 0, \sigma^2_\varepsilon \tau^2_j) Exp\left(\tau^2_j \middle| \frac{\lambda^2}{2}\right) \partial\sigma^2_{\beta_j}$. In the fully-Bayesian LASSO, λ^2 is treated as unknown and is assigned a Gamma prior. This prior is indexed by two parameters (rate and shape, see `help(rgamma)`) which are assumed to be known. Alternative priors for the regularization parameter are discussed in de los Campos et al. (2009).

In **BayesB** (Meuwissen, Hayes, and Goddard 2001) marker effects are assumed to be equal to zero with probability π and with probability (1-π) the effect is assumed to be a draw form a t-distribution such as the one described in BayesA. Model **BayesC** (Habier et al. 2011) is similar to BayesB but uses a Gaussian slab instead of the t-density used in BayesB.

For infinitesimal traits, zeroing-out marker effects, such as in BayesB or C, may harm predictive ability. Therefore, an alternative is to replace the peak of mass at zero used in BayesB or C with a continuous density with small variance. This strategy is commonly used in what it is referred as to **Spike-Slab models** (Ishwaran and Rao 2005); for instance one can mix two Gaussian densities, one with very small variance and one with larger variance. The code below presents an example where models BRR, BL, BayesA-B-C are fitted to simulated data.

Example 3.1. The Bayesian Alphabet

```
rm(list=ls())
  library(BGLR)
  set.seed(12345)
  data(mice);
  n<-nrow(mice.X); p<-ncol(mice.X)
  mrkSet<-sort(sample(1:p,size=5000))
  X<-scale(mice.X[,mrkSet],center=TRUE,scale=TRUE)

## Toy simulation example
  nQTL<-10 ; p<-ncol(X); n<-nrow(X) ; h2<-0.5
  whichQTL<-sample(1:p,size=nQTL)
  b0<-rep(0,p)
  b0[whichQTL]<-rnorm(n=nQTL, mean=3,sd=1)
  plot(b0)
  signal<-as.vector(X%*%b0)
  SD<-sqrt(h2/var(signal))
  b0<-b0*SD  ; signal<-signal*SD
  error<-rnorm(n,sd=sqrt(1-h2))
  var(signal)/var(signal+ error)
  y<-signal+error ## note, by construction
  tst<-sample(1:n,size=150)
  yNA<-y
  yNA[tst]<-NA

## Fits various models using BGLR
## Note: we are using only 6000, for real analysis use longer chains
 nIter<-6000 ; burnIn<-1000
 # want to run it without QTL in the marker set?
 #X=X[,-whichQTL]
 #dir.create('withoutQTL')
 #setwd('withoutQTL')

## Bayesian Ridge Regression (Gaussian prior), "equivalent" to G-BLUP
 ETA<-list(MRK=list(X=X,model='BRR'))
 fmBRR<-BGLR(y=yNA,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt='BRR_')

## BayesA(Scaled-t prior)
 ETA$MRK$model<-'BayesA'
 fmBA<-BGLR(y=yNA,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt='BA_')

## Bayesian LASSO(Laplace prior)
 ETA$MRK$model<-'BL'
 fmBL<-BGLR(y=yNA,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt='BL_')

## BayesC (point of mass at zero + Gaussian slab)
 ETA$MRK$model<-'BayesC'
 fmBC<-BGLR(y=yNA,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt='BC_')

## BayesB (point of mass at zero + scaled-t slab)
 ETA$MRK$model<-'BayesB'
 fmBB<-BGLR(y=yNA,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt='BB_')

# Plots in next page
```

Example 3.1. The Bayesian Alphabet (continuation)

```
 # Residual Variance
 tmp<-c(fmBRR$varE,fmBA$varE,fmBL$varE,fmBC$varE,fmBB$varE)
 names(tmp)<-rownames(MSE)
 barplot(tmp)
 #DIC  and other stats
  fmBRR$fit ; fmBA$fit ; fmBL$fit ; fmBC$fit ; fmBB$fit

 ## plots of estimates
 plot(fmBRR$ETA[[1]]$b,ylim=range(b0),col=4,ylab='Estimate',main='BRR')
 points(x=whichQTL,y=b0[whichQTL],col=2); abline(v=whichQTL,col=8)
 plot(fmBA$ETA[[1]]$b,ylim=range(b0),col=4,ylab='Estimate',main='BayesA')
 points(x=whichQTL,y=b0[whichQTL],col=2) ; abline(v=whichQTL,col=8)
 plot(fmBL$ETA[[1]]$b,ylim=range(b0),col=4,ylab='Estimate',main='BL' )
 points(x=whichQTL,y=b0[whichQTL],col=2) ; abline(v=whichQTL,col=8)
  plot(fmBC$ETA[[1]]$b,ylim=range(b0),col=4,ylab='Estimate',main='BayesC')
 points(x=whichQTL,y=b0[whichQTL],col=2) ; abline(v=whichQTL,col=8)
 plot(fmBB$ETA[[1]]$b,ylim=range(b0),col=4,ylab='Estimate',main='BayesB')
 points(x=whichQTL,y=b0[whichQTL],col=2) ; abline(v=whichQTL,col=8)

 ## Correlation between simulated and predicted signal in TST
 tmp<-c(cor(signal[tst],fmBRR$yHat[tst]) ,
        cor(signal[tst],fmBA$yHat[tst]),cor(signal[tst],fmBL$yHat[tst]),
        cor(signal[tst],fmBC$yHat[tst]), cor(signal[tst],fmBB$yHat[tst]))
 names(tmp)<-c('BRR', 'BA', 'BL', 'BC', 'BB')
 barplot(tmp)

# predictions
plot(fmBRR$yHat,fmBB$yHat,xlim=range(fmBB$yHat),
     ylim=range(fmBB$yHat),main='Predicted Genetic Values',
     xlab='BRR',ylab='BayesB');abline(a=0,b=1,lwd=2,col=2)

## Trace plots of parameters (example with BRR)
 plot(scan('BRR_ETA_1_varB.dat'),type='o',
          ylab=expression(paste(sigma[beta]^2)),col=2)
 plot(scan('BRR_varE.dat'),type='o',
          ylab=expression(paste(sigma[epsilon]^2)),col=2)

## Trace plots of parameters (example with BayesB)
## Note: the algorithm has not converged, we need to run
##       many more iterations!!!!
 TMP=read.table('BB_ETA_MRK_parBayesB.dat',header=TRUE)
 plot(TMP[,1],type='o',
          ylab='Scale',col=2)
 plot(TMP[,2],type='o',
          ylab='Probability of inclusion',col=2)
 plot(TMP,type='o',
          xlab='Scale', ylab='Probability of inclusion',col=2)

; list.files()
```

3.2. Joint modeling of genetic and non-genetic effects

In the previous example we have considered regressing phenotypes on markers. In practice, the effects of factors other than markers need to be considered; these can be regarded either as ‘fixed’ or random effects. Using the mice data set included in BGLR (de los Campos and Pérez 2013) we present an example where body-mass index (BMI) is regressed on: sex and litter size as fixed effects, cage as random effect (using a Gaussian prior), an additive effect with co-variance matrix computed from the pedigree (for this we use the standard multivariate normal prior of the infinitesimal model) and markers (using a scaled-t prior).

Example 3.2. Joint modeling of Genetic and Non Genetic Effects

```
rm(list=ls())
library(BGLR)
data(mice);
y<-mice.pheno$Obesity.BMI
y<-(y-mean(y))/sd(y)
tst<-sample(1:length(y),size=200)
yNA<-y
yNA[tst]<-NA
X<-mice.X
A<-mice.A
litterSize<-mice.pheno$Litter
litterSize<-ifelse(is.na(litterSize),
                   mean(litterSize,na.rm=TRUE),litterSize)

sex<-mice.pheno$GENDER

## Model with fixed effects of sex and litter size,
# and random effects of cage, markers and regression on pedigree.

  ETA<-list( FIXED=list(~factor(litterSize)+factor(sex),
                             model='FIXED'),
             CAGE=list(~factor(cage)-1,data=mice.pheno, model='BRR'),
             PED=list(K=A,  model='RKHS'),
             MRK=list(X=X,  model='BayesA')
           )
  fm<-BGLR(y=yNA,ETA=ETA, nIter=7000, burnIn=1000,saveAt='full_')

  cor(y[tst],fm$yHat[tst])
  plot( (fm$ETA$MRK$b^2),xlab='SNP',
        ylab='Squared-Effect',cex=.5,col=2,type='l')

```

References

de los Campos, G., H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel, and J. M Cotes. 2009. “Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree.” *Genetics* 182 (1): 375–385.

de los Campos, G., and P. Pérez. 2013. *BGLR=Bayesian Generalized Linear Regression*. R Package Version 1.0. https://r-forge.r-project.org/R/?group_id=1525.

de los Campos, Gustavo, John M. Hickey, Ricardo Pong-Wong, Hans D. Daetwyler, and Mario P. L. Calus. 2012. “Whole Genome Regression and Prediction Methods Applied to Plant and Animal Breeding.” *Genetics* (June 28). doi:10.1534/genetics.112.143313. http://www.genetics.org/content/early/2012/06/28/genetics.112.143313.

Habier, D., R. Fernando, K. Kizilkaya, and D. Garrick. 2011. “Extension of the Bayesian Alphabet for Genomic Selection.” *BMC Bioinformatics* 12 (1): 186.

Ishwaran, H., and J. S Rao. 2005. “Spike and Slab Variable Selection: Frequentist and Bayesian Strategies.” *The Annals of Statistics* 33 (2): 730–773.

Meuwissen, T H, B J Hayes, and M E Goddard. 2001. “Prediction of Total Genetic Value Using Genome-wide Dense Marker Maps.” *Genetics* 157 (4) (April): 1819–1829.

Park, T., and G. Casella. 2008. “The Bayesian Lasso.” *Journal of the American Statistical Association* 103 (482): 681–686.

LAB4: Semi-parametric Genomic Regression Using Kernel Methods

In previous labs we discussed methods where phenotypes are regressed on genetic markers using linear methods. In LABS 4 and 5 we consider models where the regression function is specified using semi-parametric procedures. Section 4.1 presents an overview of non-parametric regression using Reproducing Kernel Hilbert Spaces (RKHS). The flexibility of this approach is illustrated in section 4.2 with an application for a single predictor (scatter-plot smoothing). In section 4.3 we discuss the relationship between RKHS regressions and Bayesian methods. As before, the Bayesian approach offers a natural way of dealing with regularization parameters. In section 4.4 we discuss how the RKHS regression framework can be used with highly dimensional genomic data. Finally, in section 4.5 we introduce a Bayesian procedure that can be used to infer, from data an 'optimal' degree of smoothness.

In a standard regression model, the response, y_i, is expressed as the sum of a conditional expectation function, $g(\mathbf{x}_i)$, and a model residual, ε_i, that is $y_i = g(\mathbf{x}_i) + \varepsilon_i$. In previous labs we focused on the case where $g(\mathbf{x}_i)$ is a linear function of marker genotypes, that is $g(\mathbf{x}_i) = \sum_{j=1}^{p} x_{ij}\beta_j$. Departures from the linear model could theoretically be captured by extending the regression formula with addition of contrasts between marker genotypes, for instance dominance could be modeled using dummy variables of the form $d_{ij} = \{1 \text{ if } x_{ij} = 1; 0 \text{ otherwise}\}$, and similar contrasts could be used to model interaction of alleles at different loci (i.e., epitasis). However, with large p the number of contrasts needed to model even low degree of interactions (e.g., 1s order epistatic interactions) is extremely large and the problem becomes intractable.

Alternatively, we could try to capture departures from the linear model using semi-parametric procedures. This was first suggested in the context of Genomic Selection (GS) by Gianola, Fernando, and Stella 2(006) who proposed implementing GS using various semi-parametric procedures. Since then, several existing semi parametric procedures have been evaluated in GS. In this lab we focus on Reproducing Kernel Hiblert Spaces (RKHS). We discuss another commonly used non-parametric procedure, the Penalized Neural Networks, in LAB 5.

4.1. Reproducing Kernel Hilbert Spaces (RKHS) regressions

Reproducing kernel Hilbert spaces (RKHS) methods are used for semi-parametric modeling in different areas of application such as scatter-plot smoothing (e.g., smoothing spline, Wahba 1990); spatial smoothing (e.g., Kriging, Cressie 1988); classification problems (e.g., support vector, Vapnik 1998), just to mention a few. Gianola, Fernando, and Stella (2006) suggested using this methodology for semi-parametric genomic enabled prediction. Since then, several authors have discussed and evaluated this me

thodology in a genomic context. Estimates in RKHS can be motivated as solution to a penalized optimization problem in a RKHS of real-valued functions or, simply, as posterior modes in certain class of Bayesian models. Next, we provide an overview of the methodology. A detailed discussions of RKHS regressions in the context of genome-enabled prediction can be found in Gianola and van Kaam (2008); de los Campos, Gianola, and Rosa (2009); de los Campos et al. (2010) and Schaid (2010).

Penalized Regression in Reproducing Kernel Hilbert Spaces

In RKHS regressions we define the space (or set of functions) in which we perform the regression by choosing a reproducing kernel (RK). Technically, the RK can be any positive semi-definite function [1] mapping from pairs of points in input space onto the real line, that is $K(\mathbf{x}_i,\mathbf{x}_{i'}):\{(\mathbf{x}_i,\mathbf{x}_{i'})\rightarrow\Re\}$. Fixing one of the arguments of the RK, for instance *i*, yields a basis function, a map from the second argument, $\mathbf{x}_{i'}$, onto the real line, specifically $K_i(\mathbf{x}_{i'})=K(\mathbf{x}_{i'}|\mathbf{x}_i)$. We can do this for *i=1,…,n* to obtain a set of n basis functions $\{K_1(\mathbf{x}_{i'}),....,K_n(\mathbf{x}_{i'})\}$, or in matrix notation $\mathbf{K}=\{\mathbf{k}_1,....,\mathbf{k}_n\}$where $\mathbf{K}$ is a matrix whose columns are the evaluations of each of the basis functions for *i'=1,…,n*, that is $\mathbf{k}_i=\left[K_i(x_1),....,K_i(x_n)\right]'$.

In RKHS regressions the evaluations of unknown genomic function, $g(\mathbf{x}_i)$, are expressed as linear combinations of the basis functions provided by the reproducing kernel, RK, $K(\mathbf{x}_i,\mathbf{x}_{i'})$, that is $g(\mathbf{x}_i)=\sum_{i'}K(\mathbf{x}_i,\mathbf{x}_{i'})\alpha_{i'}=\sum_{i'}K_{i'}(\mathbf{x}_i)\alpha_{i'}$, and the squared of the norm of the function in the Hilbert space is given by $\|g\|^2=\sum_i\sum_{i'}K(\mathbf{x}_i,\mathbf{x}_{i'})\alpha_{i'}\alpha_i$. Stacking the evaluations of the function into a vector, $\mathbf{g}=\left[g(\mathbf{x}_1),...,g(\mathbf{x}_n)\right]'$ yields: $\mathbf{g}=\mathbf{K}\alpha$ and $\|\mathbf{g}\|^2=\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$.

Estimates in RKHS are usually obtained as the solution to the following penalized residual sum of squares (intercept and non-maker effects omitted for ease of notation):

$$\hat{\boldsymbol{\alpha}}\underset{\arg\min}{=}\left\{(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})'(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})+\lambda\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}\right\} \quad [1]$$

above, $(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})'(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})$is a residual sum of squares, $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$ is a penalty on model complexity, which is taken to be the square of the norm of the function and λ is a regularization parameters.

The solution to the above optimization problem can be shown to be:

$$\hat{\boldsymbol{\alpha}}=[\mathbf{K}'\mathbf{K}+\lambda\mathbf{K}]^{-1}\mathbf{K}'\mathbf{y}. \quad [2]$$

Predictions are then obtained as follows:

$$\mathbf{K}\hat{\boldsymbol{\alpha}}=\mathbf{K}[\mathbf{K}'\mathbf{K}+\lambda\mathbf{K}]^{-1}\mathbf{K}'\mathbf{y}=[\mathbf{I}+\lambda\mathbf{K}^{-1}]^{-1}\mathbf{y}; \quad [3]$$

where, $\mathbf{K}[\mathbf{K}'\mathbf{K}+\lambda\mathbf{K}]^{-1}\mathbf{K}'=[\mathbf{I}+\lambda\mathbf{K}^{-1}]^{-1}$ is the Hat matrix of RKHS.

[1]For $K(\mathbf{x}_i,\mathbf{x}_{i'})$ to be positive semi definite it must satisfy $\sum_i\sum_{i'}\alpha_i\alpha_{i'}K(\mathbf{x}_i,\mathbf{x}_{i'})K(\mathbf{x}_i,\mathbf{x}_{i'})\geq 0$ for every non-null sequence$\{\alpha_i\}$.

Model specification in RKHS regression is defined by two main elements[2]: the choice of the reproducing kernel; this provides the basis functions and the norm, and λ which, as in ridge regression, represents a shrinkage parameter.

4.2. Scatter plot smoothing with a Gaussian kernel

In the following example we will use a RKHS regression to estimate a conditional expectation function non-parametrically. In the example, there is a single predictor, $x_i \in [0, 2\pi]$ and the true conditional expectation function is $g(x_i) = 120 + \sin(x_i)$. Data was generated according to the following model: $y_i = 120 + \sin(x_i) + \varepsilon_i$ where $\varepsilon_i \overset{IID}{\sim} N(0,1)$. With this setting, approximately 1/3rd of the variance of the response is explained by the conditional expectation function and 2/3rd by model residuals.

In this example we use a Gaussian kernel,

$$K(x_i, x_{i'}) = \exp\{-h \times d(x_i, x_{i'})\}$$

where: $d(x_i, x_{i'})$ is the squared-Euclidean distance between x_i and $x_{i'}$, $d(x_i, x_{i'}) = (x_i - x_{i'})^2$, and h is a bandwidth parameter controlling how fast the kernel decay as the two points, $(x_i, x_{i'})$, get further apart. In the example we evaluate the effects of h (which defines the RK) and of the regularization parameter λ.

- Run the code with the values of *h* and λ given in the example.
- Set *h*=1/1000, this makes the kernel extremely global, and run the code.
- Set h=50, this makes the kernel extremely local, and run the code.
- Now fix *h=1* and change lambda, evaluate λ =200, then λ =1/100, evaluate results.

[2] A third element pertains to the choice of the function used to measure model goodness/lack of fit to the training data. Here we focus on the case where lack of fit is measured by the residual sum of squares; other common choices are the negative of the log-likelihood, this allows modeling continuous, binary and other types of outcomes. For binary outcomes another popular choice is the hinge function, the support vector machine (Vapnik 1998) is a special case of RKHS where the loss-function is chosen to be a hinge function (Wahba 1990).

Example 4.1. Scatter-plot smoothing with a Gaussian kernel

```
### SIMULATION############################################
  set.seed(12345)
  N<-200
  x<-seq(from=0,to=2*pi,length=N)
  signal<-sin(x)
  error<-rnorm(N)
  y<-signal+error
  h<-1
  lambda<-10
### DISTANCE FUNCTION AND REPRODUCING KERNEL #######
  D<-as.matrix(dist(x,method="euclidean"))^2
  K<-exp(-h*D)
  diag(K)<-diag(K) +.001

### FITTING THE MODEL #############################
  yStar<-y-mean(y)
  #f eq. [3]
    KInv<-chol2inv(chol(K))
    C<-KInv*lambda
    diag(C)<-diag(C)+1
    H<-chol2inv(chol(C))  # the Hat matrix
   #end
uHat<-H%*%(y-mean(y))

plot(y~x, main=paste("lambda=",lambda," h=",h,sep=""))
lines(x=x,y=signal,col=2,lwd=2)
lines(x=x,y=uHat+mean(y),col=4,lwd=2)

## want to make the function less local? set h=1/1000,
## want to make it extremely local? set h=1000
## Now fix h=100 and set lambda= 1/100
```

4.3. Bayesian view of RKHS

The solution to the penalized RKHS regression (see eq. [1]) can be shown to be equal to the posterior mode of the vector of regression coefficients in the following Bayesian model:

$$\begin{cases} \mathbf{y} = \mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\varepsilon} \\ \left(\begin{matrix} \boldsymbol{\varepsilon} \\ \boldsymbol{\alpha} \end{matrix} \middle| \sigma_\varepsilon^2, \sigma_g^2 \right) \sim N\left[\mathbf{0}, \begin{pmatrix} \mathbf{I}\sigma_\varepsilon^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}^{-1}\sigma_\alpha^2 \end{pmatrix} \right] \end{cases} \qquad [4]$$

where $\lambda = \sigma_\varepsilon^2 \sigma_\alpha^{-2}$. The proof of the equivalence between the posterior mode of $\boldsymbol{\alpha}$ in the Bayesian model described in [4] and the solution given in [2] can be obtained following the same steps used in section 2.5 of LAB 2.

Further, changing variables in [4] from $\mathbf{K}\boldsymbol{\alpha}$ to $\mathbf{g} = \mathbf{K}\boldsymbol{\alpha}$, and noting from the properties of the MVN density (see section 2.6 of LAB 2) that $\mathbf{g} \sim MVN\left(\mathbf{0}, \mathbf{K}\sigma_g^2\right)$, where $\sigma_\alpha^2 = \sigma_g^2$, we obtain an equivalent representation of [4],

$$\begin{cases} \mathbf{y} = \mathbf{g} + \boldsymbol{\varepsilon} \\ \left(\begin{matrix} \boldsymbol{\varepsilon} \\ \mathbf{g} \end{matrix} \middle| \sigma_\varepsilon^2, \sigma_g^2 \right) \sim N\left[\mathbf{0}, \begin{pmatrix} \mathbf{I}\sigma_\varepsilon^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}\sigma_g^2 \end{pmatrix} \right] \end{cases} \qquad [5]$$

Therefore, from the Bayesian perspective, the evaluations of functions at points in the input space, $\mathbf{g} = \{g(\mathbf{x}_i)\}$ are viewed as realizations from Gaussian process satisfying: $Cor[g(\mathbf{x}_i), g(\mathbf{x}_{i'})] = \dfrac{K(\mathbf{x}_i, \mathbf{x}_{i'})}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)K(\mathbf{x}_{i'}, \mathbf{x}_{i'})}}$. Here, the RK $K(\mathbf{x}_i, \mathbf{x}_{i'})$ is viewed as a (co)variance function which defines a notion of smoothens of the function with respect to points in the input space (genotypes in our case). A high value of $Cor[g(\mathbf{x}_i), g(\mathbf{x}_{i'})]$ implies that, a-priori, we expect the function to behave smoothly when we jump from $\mathbf{x}_i$ to $\mathbf{x}_{i'}$. At the same time, this means y_i is informative about $g(\mathbf{x}_{i'})$ and that $y_{i'}$ informs us something about $g(\mathbf{x}_i)$.

Special cases. Certain parametric models appear as special cases of RKHS regression. For instance, if our information set consists of a pedigree and $\mathbf{K}$ is a matrix of additive relationship matrix, the model defined by [1] is equivalent to the infinitesimal additive model, the so-called **Animal Model** (de los Campos, Gianola, and Rosa 2009). The Ridge Regression (and consequently, GBLUP, see section 2.6 of LAB 2) is another example of a parametric model that can be represented as a RKHS, in this case obtained by setting $\mathbf{K} = \mathbf{X}\mathbf{X}'$. These are examples where the RK is chosen so as to represent the types of patterns expected under a parametric model. Another alternative is to choose kernels based on their performance (e.g., predictive ability). In this lab we will focus on this second approach.

4.4. Genomic-Enabled Prediction Using RKHS

In this section we use the Gaussian kernel for genomic-enabled prediction. To this end, we replace the distance function by a genomic-distance. For instance, we can set $d(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$; the Gaussian kernel becomes: $K(x_i, x_{i'}) = \exp\{-h \times d(\mathbf{x}_i, \mathbf{x}_{i'})\}$. The function `dist()` of R takes tow arguments: x which should be a numeric vector or matrix, and methods, which should be a character string indicating the method for computing distances. By default the Euclidean distance is computed. Type `help(dist)` for further details. The function returns an object, which can be converted to an n×n matrix, containing pairwise distance between the rows of **X**.

The example below uses the BGLR R-package (de los Campos and Pérez, 2013) to fit RKHS models over a grid of values of the bandwidth parameter (*h*) and evaluates the effect of it on goodness of fit, model complexity and predictive ability.

- Ran the code;
- Evaluate how goodness of fit and predictive ability changes with *h*
- How does $\lambda = \frac{\sigma_\varepsilon^2}{\sigma_g^2}$ changes with *h*?

Example 4.2. RKHS for Genomic Prediction

```
rm(list=ls())
library(BGLR)
data(wheat); X=wheat.X; Y=wheat.Y

### DISTANCE MATRIX #############################
  D<-as.matrix(dist(X,method="euclidean"))^2
  D<-D/mean(D)
  h<-c(1e-2,.1,.4,.8,1.5,3,5)

### GENERATES TESTING SET #######################
 set.seed(12345)
 tst<-sample(1:599,size=150,replace=FALSE)
 y<-Y[,2]
 yNA<-y
 yNA[tst]<-NA

### FITS MODELS #################################

 PMSE<-numeric() ; VARE<-numeric(); VARU<-numeric() ;
 pD<-numeric(); DIC<-numeric()
 fmList<-list()

 for(i in 1:length(h)){
     print(paste('Working with h=',h[i],sep=''))
    # COMPUTES THE KERNEL
    K<-exp(-h[i]*D)
    # FITS THE MODEL
    ETA<-list(list(K=K,model='RKHS'))
    prefix<- paste(h[i], "_",sep="")
    fm<-BGLR(y=yNA,ETA=ETA,
           nIter=5000,burnIn=1000,df0=5,S0=2,saveAt=prefix)
    fmList[[i]]<-fm
    PMSE[i]<-mean((y[tst]-fm$yHat[tst])^2)
    VARE[i]<-fm$varE
    VARU[i]<-fm$ETA[[1]]$varU
    DIC[i]<-fm$fit$DIC
    pD[i]<-fm$fit$pD
 }

 R2<-1-PMSE/mean((y[tst]-mean(y[-tst]))^2)

 ### PLOTS ###############################
 plot(VARE~h,xlab="Bandwidth",
      ylab="Residual Variance",type="o",col=4)

 plot(I(VARE/VARU)~h,xlab="Bandwidth",
      ylab="variance ratio (noise/signal)",type="o",col=4)

 plot(pD~h,xlab="Bandwidth", ylab="pD",type="o",col=2)

 plot(DIC~h,xlab="Bandwidth", ylab="DIC",type="o",col=2)

 plot(R2~h,xlab="Bandwidth", ylab="R-squared",type="o",col=2)
```

4.5. Kernel Averaging

The choice of the RK (its functional form and the values of parameters such as the bandwidth) constitutes the central element of model specification in RKHS regressions. There are several ways of choosing a kernel. In **parametric models**, the RK is chosen to represent the type of patterns expected under a particular parametric model (e.g., additive infinitesimal, **K**=**A**; linear model, **K**=**XX**'). Form a **non-parametric** perspective one can choose kernels based on the performance of the model, e.g., predictive ability; an illustration of this was provided in the previous example where a validation set was used to evaluate predictive ability of RKHS using a Gaussian kernel, over a grid of values of the bandwidth parameter.

A third way is by **inferring the kernel** from the data. For instance, in a Bayesian context one could assign a prior to the bandwidth parameter and infer this parameter jointly with other unknowns. While this is appealing, it is computationally demanding for at least two reasons: (a) the RK must be re-computed every time a new value of the bandwidth parameter is sampled; (b) mixing may be poor. This occurs because, usually, variance parameters and the bandwidth parameter are highly correlated at the posterior distribution. An alternative which we consider next is to offer the algorithm all candidate kernels jointly. For instance, we can make the conditional expectation to be a sum of several random effects, $\{\mathbf{g}_1,\dots,\mathbf{g}_{N_k}\}$ each of which has its own (co)variance function, the model becomes:

$$\begin{cases} \mathbf{y} = \mathbf{1}\mu + \sum_{k=1}^{N_k} \mathbf{g}_k + \boldsymbol{\varepsilon} \\ p\left(\boldsymbol{\varepsilon}, \mathbf{g}_1,\dots,\mathbf{g}_{N_k} \middle| \sigma^2_\varepsilon, \sigma^2_{g_1},..,\sigma^2_{g_{N_k}}\right) = N\left(\boldsymbol{\varepsilon} \middle| \mathbf{0}, \mathbf{I}\sigma^2_\varepsilon\right) \prod_{k=1}^{N_k} N\left(\mathbf{g}_k \middle| \mathbf{0}, \mathbf{K}_k\sigma^2_{g_k}\right) \end{cases}$$

It can be shown that, conditional on variance parameters, the above model is equivalent to one with a single random effect, **g**, whose prior distribution is $N\left(\mathbf{g}\middle|\mathbf{0},\bar{\mathbf{K}}\sigma^2_g\right)$ where: $\bar{\mathbf{K}} = \mathbf{K}_1\alpha_1 + \mathbf{K}_2\alpha_2 + \dots + \mathbf{K}_{N_k}\alpha_{N_k}$ is a weighted sum of the candidate kernels with weight given by $\alpha_k = \frac{\sigma^2_{g_k}}{\sigma^2_g}$ and $\sigma^2_g = \sum_k \sigma^2_{g_k}$. Variance parameter here can then be seen as weights associated to each kernel which can be inferred from the data. The larger the variance associated to a given kernel the larger the contribution of that random effect to the conditional expectation We refer to this approach as kernel averaging (KA, de los Campos et al., 2010).

The following example illustrates the use of KA; the sequence of kernels was generated using the Gaussian kernel and the values of the bandwidth parameter used in our previous example.

- Run the code below.
- What Kernel gets higher weight?
- Is that the Kernel that gave highest predictive ability in our previous example?
- Compare the predictive ability of KA with that of models fitted in our previous example (i.e., single kernel with fixed bandwidth).

Example 4.3. Kernel Averaging

```
#(continues from example 4.3)
### Prepare the kernel list for KA #########################
 PMSE<-numeric()
 VARE<-numeric()
 KList<-list()
 for(i in 1:length(h)){
    KList[[i]]<-list(K=exp(-h[i]*D),model='RKHS')
 }

## Displays entries of different kernels
 plot(KList[[1]]$K[100,],ylim=c(0,1),col=2);abline(v=100)

 plot(KList[[7]]$K[100,],ylim=c(0,1),col=2);abline(v=100)


 fmKA<-BGLR(y=yNA,ETA=KList,thin=10,
            nIter=103000,burnIn=3000 ,saveAt="KA_")

 VARG<-numeric()
 for(i in 1:length(KList)){  VARG[i]<-fmKA$ETA[[i]]$varU }
 weights<-round(VARG/sum(VARG),5)

 PMSE<-mean((y[tst]-fmKA$yHat[tst])^2)
 R2_KA<-1-PMSE/mean((y[tst]-mean(y[-tst]))^2)

## This plot compares the R2 obtained with fixed values of h verus KA.
plot(R2~h,xlab="Bandwidth", ylab="R-squared",type="o",col=2)
abline(h=R2_KA,col=4)

# take a look at the trace plots of variance parameters
tmp<- scan('KA_ETA_1_varU.dat')
plot(tmp,type='o',col=2, ylab='var(u)',main=paste('h=',h[1],sep=''))

tmp<- scan('KA_ETA_5_varU.dat')
plot(tmp,type='o',col=2, ylab='var(u)',main=paste('h=',h[5],sep=''))


tmp<- scan('KA_varE.dat')
plot(tmp,type='o',col=2, ylab='var(e)',main='Residual Variance')
```

References

Cressie, N. 1988. “Spatial Prediction and Ordinary Kriging.” *Mathematical Geology* 20 (4): 405–421.

de los Campos, G., D. Gianola, G. J. M. Rosa, K. A Weigel, and J. Crossa. 2010. “Semi-parametric Genomic-enabled Prediction of Genetic Values Using Reproducing Kernel Hilbert Spaces Methods.” *Genetics Research* 92 (04): 295–308.

de los Campos, G., D. Gianola, and G. J.M Rosa. 2009. “Reproducing Kernel Hilbert Spaces Regression: a General Framework for Genetic Evaluation.” *Journal of Animal Science* 87 (6): 1883.

de los Campos, G., and P. Pérez. 2013. *BGLR=Bayesian Generalized Linear Regression*. R Package Version 1.0. https://r-forge.r-project.org/R/?group_id=1525.

Gianola, D., R.L. Fernando, and A. Stella. 2006. “Genomic-Assisted Prediction of Genetic Value With Semiparametric Procedures.” *Genetics* 173 (3) (July 1): 1761–1776. doi:10.1534/genetics.105.049510.

Gianola, D., and J. B van Kaam. 2008. “Reproducing Kernel Hilbert Spaces Regression Methods for Genomic Assisted Prediction of Quantitative Traits.” *Genetics* 178 (4): 2289.

Schaid, D. J. 2010. “Genomic Similarity and Kernel Methods I: Advancements by Building on Mathematical and Statistical Foundations.” *Human Heredity* 70 (2): 109–131.

Vapnik, V. N. 1998. “Statistical Learning Theory.”

Wahba, G. 1990. “Spline Methods for Observational Data.” *SIAM: Philadelphia*.

LAB 5: Penalized Neural Networks

In this LAB we discuss a second class of non-parametric procedure for genomic regression. Section 5.1 provides a brief introduction to the methodology; in section 5.2 we use a penalized NN for scatter-plot smoothing, and in sections 5.3. and 5.4. we present examples where a NN is used for genomic regression.

5.1. Introduction

In **linear regression** models the conditional expectation is represented as a weighted sum of input variables, $E\left(y_i|\mathbf{x}_i\right)=\sum_{j=1}^{p} x_{ij}\beta_j$. Many **non-linear patterns** can be represented linearly by appropriate choice of **basis functions:** $E\left(y_i|\mathbf{x}_i\right)=\sum_{m=0}^{M}\phi\left(\mathbf{x}_i\right)w_m$ where, $\left\{\phi_m\left(\mathbf{x}_i\right)\right\}_{m=1}^{M}$ are the basis functions, which map from the input variables onto the real line. Examples of these are the polynomial basis functions: $\Phi=\left\{\varphi_m(x_i)=x_i^m\right\}_{m=0}^{M}$. For instance, if M=2 we have $\Phi=\left\{1,x_i,x_i^2\right\}$, therefore, $E(y_i|\mathbf{x}_i)=\beta_0+\beta_1x_i+\beta_2x_i^2$. Other common examples of non-linear basis functions are the power, logarithm and exponential functions. With these types of basis functions each of the regression coefficients affect the behavior of the conditional expectation in the entire input space.

Local basis functions can be used to model a conditional expectation within certain regions of the input space. **Splines** represent an example of this. In a spline**,** polynomial basis functions are used to represent the regression function locally. The **Gaussian kerne**l discussed in LAB4 is another example of a local basis function, here $\varphi_m(\mathbf{x}_i,\mathbf{t}_m,h)=e^{-h\|\mathbf{x}_i-\mathbf{t}_m\|^2}$ where $\mathbf{t}_m$ is a focal point and h is a bandwidth parameter which controls how fast the basis function decay as $\mathbf{x}_i$ gets further apart from the focal point. Model specification in this case pertains to the choice of focal points (how many to use and where to place them in the input space) and of the bandwidth parameter. In the RKHS regressions of LAB4, the strategy was to 'offer' the model a large number of basis functions (one per subject in the sample) generated by setting $\mathbf{t}_1=\mathbf{x}_1$, $\mathbf{t}_2=\mathbf{x}_2$,…, $\mathbf{t}_n=\mathbf{x}_n$; therefore $E(y_i|\mathbf{x}_i)=\sum_{i'=1}^{n}\alpha_{i'}\times e^{-h\|\mathbf{x}_i-\mathbf{x}_{i'}\|^2}$. This strategy may induce over-fitting and this was confronted by using shrinkage estimation procedures. This is approach is also used in smoothing spline (Craven and Wahba 1978; Wahba 1990).

Non-linear basis functions such as the ones described above offer great potential for capturing potentially complex patterns between input and output variables; however, the set of basis functions needs to be defined a-priori. In Neural Networks (NN) the basis functions used for regression are inferred (i.e., are data driven), this gives NN great potential for capturing potentially complex patterns.

One of the simplest NNs is the ***single hidden layer feed-forward NN***. This NN can be thought as non-linear regressions consisting of two steps (Hastie, Tibshirani, and Friedman 2009): in the first one (or hidden layer) the basis functions are inferred, and in the second one (or output layer) the output, y_i , is regressed on the basis function inferred in the hidden layer. A graphical representation of such NN is given in Figure 1. The term feed-forward is used to highlight that in these NNs information flows from inputs (the $\mathbf{x}_i$'s) to output (the y_i's), other NN allow feedbacks.

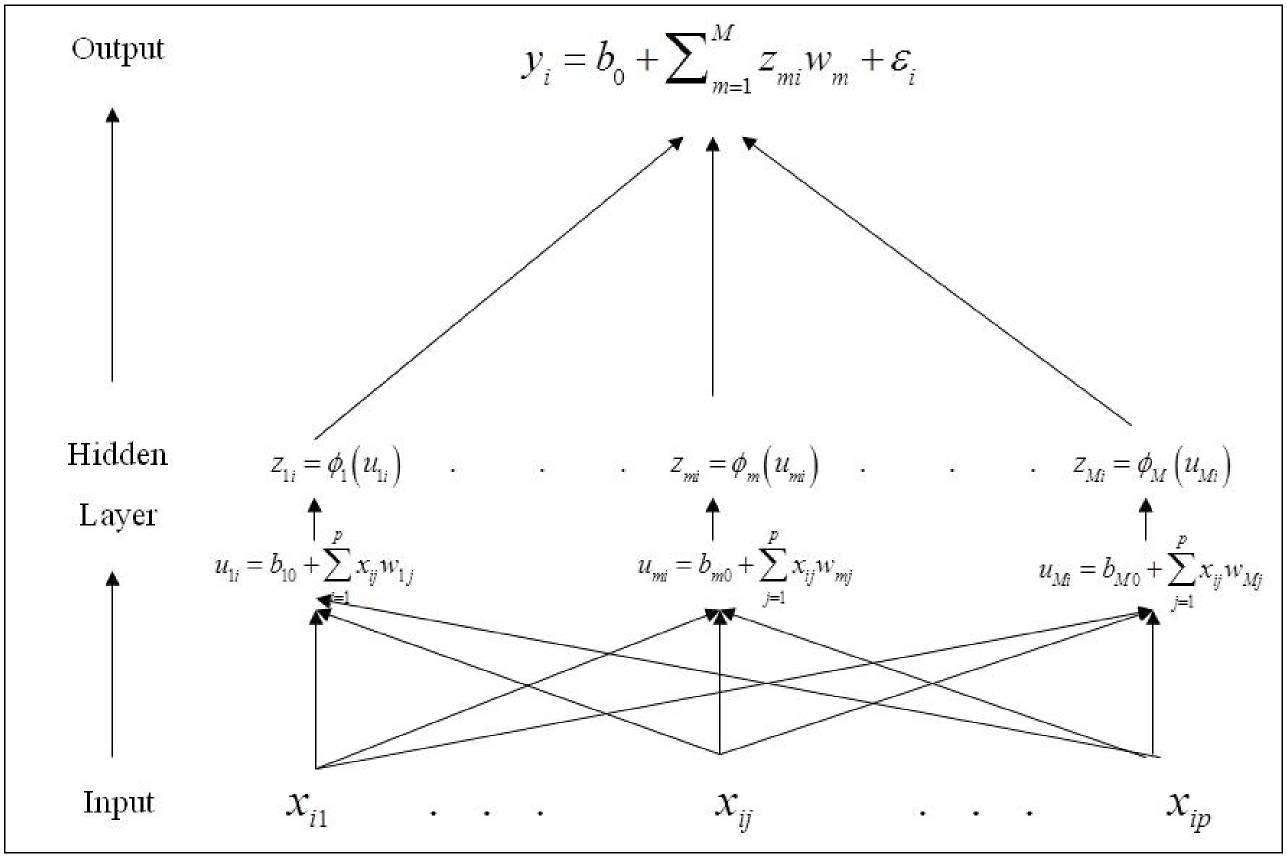
Output
$y_i = b_0 + \sum_{m=1}^{M} z_{mi} w_m + \varepsilon_i$
Hidden Layer
$z_{1i} = \phi_1(u_{1i})$
$z_{mi} = \phi_m(u_{mi})$
$z_{Mi} = \phi_M(u_{Mi})$
$u_{1i} = b_{10} + \sum_{j=1}^{p} x_{ij} w_{1j}$
$u_{mi} = b_{m0} + \sum_{j=1}^{p} x_{ij} w_{mj}$
$u_{Mi} = b_{M0} + \sum_{j=1}^{p} x_{ij} w_{Mj}$
Input
x_{i1}
x_{ij}
x_{ip}

Figure 1. Graphical Representation of Single Hidden Layer Feed-Forward Neural Network for a Continuous Response (y_i) and p predictor variables ($x_{1i},...,x_{ip}$). The network contains M neurons. At each neuron, linear combinations of the predictors ($u_{mi} = b_{m0} + \sum_{j=1}^{p} x_{ij} w_{mj}$) are inferred and subsequently activated $z_{mi} = \phi_m(u_{mi})$. These basis functions are then used in the output layer to regress the output variable using a linear model ($y_i = b_0 + \sum_{m=1}^{M} z_{mi} w_m + \varepsilon_i$).

As illustrated in Figure 1, in the **hidden layer** M basis functions, $\varphi_m\left(b_{m0} + \sum_{j=1}^{p} x_{ij} w_{mj}\right)$ are inferred (one at each **neuron)**. Each of these basis functions consist of a linear score, $u_{mi} = b_{m0} + \sum_{j=1}^{p} x_{ij} w_{mj}$, activated by a non-linear **activation function**, $\varphi_m(.)$.

In the **output layer**, the outcome, y_i , is regressed on the basis functions using an additive model. The example of Figure 1 is for a continuous response; in many applications with NN the outcome is categorical. In those cases an additional activation functions are added in the output layer. Note that, if the activation function of the hidden and output layers are identity functions (i.e., $\varphi_m(u_{im}) = u_{im}$ the model of Figure 1 becomes a standard multiple linear regression model. Moreover, if we set the $\varphi_m(\cdot)$ to be the basis functions of a reproducing kernel (see LAB4), the NN of Figure 1 becomes the RKHS regression. From this perspective the NN can be viewed as a general framework that includes the linear model and the RKHS as special cases.

The **activation functions** of the hidden layers map from the real line onto the [0,1] interval, and a common choice is to set this to be a sigmoid function. For instance we could use $\phi_m(z_{mi}) = \frac{1}{1+e^{-\theta \times z_{mi}}}$ for some $\theta > 0$.

Architecture of a Neural Network. The elements that define model specification in NN are: (a) the choice of input variables, (b) the type of network (e.g., feed-forward), (c) the number of layers, (d) the number of neurons per layer, and (d) the choice of activation functions. In general the term 'architecture' of the network is used to referred to the choices made in (b)-(d).

Penalized Neural Networks. The set of parameters to be estimated in the NN of Figure 1 include: all the intercepts and regression coefficients at each neurons, the parameters of the activation functions, and the intercept and regression coefficients of the output layer. With large *p,* and with several neurons, the total number of parameters to be estimated can be extremely large. This, together with the intrinsic flexibility of the NN, can easily yield over-fitting and poor predictive performance. To prevent this, a common strategy is to fit the neural network using penalized methods such as those discussed in LAB2. Therefore, in a penalized NN, parameters are estimated by minimizing an objective function consisting of a lack-of fit function (e.g., a residual sum of squares) plus a penalty on model complexity. Any of the penalties discussed in LAB 2 can be used; however, a common choice is to set the penalty to be the sum of squares of the regression coefficients (usually intercepts are not penalized).

In what remains of the lab we illustrate the use of penalized NN using a beta version of the R-package brnn (Pérez et al. 2013).

5.2. Scatter-plot smoothing using a penalized NN

The following example illustrates the use of penalized NN for scatter-plot smoothing.

Example 1: Scatter-plot Smoothing Using a Neural Network

```
rm(list=ls()); library(splines); library(brnn)

### SIMULATION (same as the one used in Ex. 1 of LAB4) #####
  set.seed(12345)
  N<-200
  x<-seq(from=0,to=2*pi,length=N)
  signal<-sin(x)
  error<-rnorm(N)
  y<-signal+error

## Various parametric models
  lm1<-lm(y~x)
  poly3<-lm(y~x+I(x^2)+I(x^3))
## Natural spline with 4 knots
   X<-ns(x=x,df=4)
  fmNS<-lm(y~X)
## Neural Networks with 1,2,3 and 5 neurons
  NN1<-brnn(y~as.matrix(x),neurons=1)
  yHatNN_1<-predict(NN1)

  NN2<-brnn(y~as.matrix(x),neurons=2)
  yHatNN_2<-predict(NN2)

  NN3<-brnn(y~as.matrix(x),neurons=3)
  yHatNN_3<-predict(NN3)

  NN4<-brnn(y~as.matrix(x),neurons=4)
  yHatNN_4<-predict(NN4)

  NN5<-brnn(y~as.matrix(x),neurons=5)
  yHatNN_5<-predict(NN5)

## R-Squared ########################################################
  R2_lm<-1-mean((signal-predict(lm1))^2)/var(signal)
  R2_ply3<-1- mean((signal-predict(poly3))^2)/var(signal)
  R2_NS<-1- mean((signal-predict(fmNS))^2)/var(signal)
  R2_NN<-numeric()
  R2_NN[1]<-1-mean((signal-yHatNN_1)^2)/var(signal)
  R2_NN[2]<-1-mean((signal-yHatNN_2)^2)/var(signal)
  R2_NN[3]<-1-mean((signal-yHatNN_3)^2)/var(signal)
  R2_NN[4]<-1-mean((signal-yHatNN_5)^2)/var(signal)
  R2_NN[5]<-1-mean((signal-yHatNN_5)^2)/var(signal)

## Plots ###########################################################
  plot(y~x,col=1,cex=.5);lines(x=x,y=signal,lwd=2,col=2)
  lines(x=x,y=yHatNN_3,col=4,lwd=4,lty=2)
  plot(R2_NN~I(1:5),xlab='Neurons',ylab= 'R-Sq.',type='o', col=2)
  abline(h=R2_NS,col=4,lty=2)

```

Example 1 illustrates the flexibility that NNs have in terms of capturing complex patters: starting from a single predictor, the NN generated complexity by inferring multiple basis functions which were able to capture the non-linear patterns between inputs and outputs very well. The example uses a single predictor, but as illustrated in Figure 1 the method could also be applied to multiple-predictors. However, with large p and with multiple neurons, the computational requirements increase substantially.

5.3. Penalized Neural Network Using Pre-selected Markers

In Example 2 we first select the top *p* (set to 100 markers in line 21) from single marker regressions and subsequently offer these markers to a NN with 4 neurons.

Example 2: Penalized Neural Network Applied to Pre-selected Markers

```
1   rm(list=ls())
2   ### DATA ################################################
3    library(BGLR)
4    library(brnn)
5    data(wheat); X=wheat.X; Y=wheat.Y
6   
7    N<-nrow(X) ; p<-ncol(X)
8    y<-Y[,4]
9    set.seed(12345)
10   tst<-sample(1:N,size=150,replace=FALSE)
11   XTRN<-X[-tst,] ; yTRN<-y[-tst]
12   XTST<-X[tst,] ;  yTST<-y[tst]
13  ### SINGLE MARKER REGRESSIONS ########################
14   pValues<-numeric()
15   for(i in 1:p){
16       fm<-lm(yTRN~XTRN[,i])
17       pValues[i]<-summary(fm)$coef[2,4]
18       print(paste('Fitting Marker ',i,'!',sep=''))
19   }
20   nMarkers<-100
21   selSNPs<-order(pValues)[1:nMarkers]
22   XTRN<-XTRN[,selSNPs]
23   XTST<-XTST[,selSNPs]
24  
25  ### Neural Network ###################################
26  set.seed(1101)
27  NN<-brnn(y=yTRN,x=XTRN,neurons=4, epochs=500,
28        data=subset(data,partitions!=2),verbose=TRUE)
29  
30  #Predicted vs observed values for the training and testing sets
31  yHatTRN<-predict(NN)
32  yHatTST<-predict(NN,newdata=XTST)
33  plot(yHatTRN~yTRN,main="Training",ylab=expression(hat(y)),xlab="y")
34  points(x=yTST,y=yHatTST,col=2,pch=19)
35  cor(yTST,yHatTST)
36  
37  # change the seed in line 27 how did results changed?
```

5.4. Penalized Neural Networks Using Marker-derived Basis Functions as Inputs

In Example 2 we pre-selected markers, another strategy consist of first mapping the input information into some basis functions (e.g., using a reproducing kernel or using genomic relationships) and then applying the NN to these basis functions. For instance, Gianola et al. (2011) suggested using the additive relationships as basis functions, by so doing we reduce the number of input variables of the NN from p to n. In Example 3 we illustrate this approach, and in Example 4 we present another approach based on marker-derived principal components.

Example 3: Penalized Neural Network Applied to Marker-derived Genomic Relationships

```
rm(list=ls())
### DATA ############################################
 library(BGLR);
 library(brnn);
 data(wheat); X=wheat.X; Y=wheat.Y
 for(i in 1:ncol(X)){ (X[,i]<-X[,i]-mean(X[,i]))/sd(X[,i])}
 G<-tcrossprod(X)/ncol(X)
 N<-nrow(X)
 y<-Y[,4]

 set.seed(12345)
 tst<-sample(1:N,size=150,replace=FALSE)
 yTRN<-y[-tst]
 yTST<-y[tst]
 GTRN<-G[-tst,]
 GTST<-G[tst,]

 NN<-brnn(y=yTRN,x=GTRN,neurons=4, epochs=50,verbose=TRUE)
 yHatNN<- predict(NN, newdata=GTST)
 cor(yHatNN,yTST)
```

References

Craven, P., and G. Wahba. 1978. “Smoothing Noisy Data with Spline Functions.” *Numerische Mathematik* 31 (4): 377–403.

Gianola, D., H. Okut, K. Weigel, and G. Rosa. 2011. “Predicting Complex Quantitative Traits with Bayesian Neural Networks: a Case Study with Jersey Cows and Wheat.” *BMC Genetics* 12 (1): 87.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. 2009. Corr. 3rd printing 5th Printing. Springer.

Pérez, P., D. Gianola, G. J. M. Rosa, K. A Weigel, and J. Crossa. 2013. “Technical Note: An R Package for Fitting Bayesian Regularized Neural Networks with Applications in Animal Breeding (forthcoming).” *Journal of Animal Science*.

Wahba, G. 1990. “Spline Methods for Observational Data.” *SIAM: Philadelphia*.