Methods for High Dimensional Regression

G. de los Campos

11/29/2021

Up to now, we considered methods that estimate parameters by either maximizing the likelihood function (ML) or by minimaizing the residual sum of squares (OLS). These methods have reasonably good statistical properties (e.g., OLS is unbiased and has minimum variance among the class of linear unbiased estiamtors, ML is asymptotically unbiased and asymptotically efficient). However, the performace of these methods can be sub-optimal when the number of parameters to be estimated (e.g., the number of regression coefficients) is large relative to sample size.

In this last module of the course we will consider methods that are tailored for problems involving a large number of predictors. We will cover the following approaches:

- **1) Independent screening**: This approach selects predictors using a marginal association test (i.e., testing the association of the response and each predictor, one predictor at a time) and builds models using the top-q (q=1,2,. . .). An optimal model DF can be chosen by evaluating thea ability of the model to predict testing data for models with 1DF, 2DF,. . .
- **2) Forward Regression**: This approach builds a sequence of models, starting from a null model (e.g., intercept only), adding one predictor at a time, each time adding to the model the predictor that produces the largest reduction in the RSS (alternatives to this that we won't discuss are backgward elimination and methods that combine forward and backward methods).
- **3) Penalized Regressions**: This approach estimates effects using a penalized sum of squares. We will consider three methods: Ridge Regression, Lasso, and Elastic Net. The extent of regularization will be controled by a parameter (λ) which will be chosen to maximize prediction accuracy in testing data. We will also discuss best subset selection briefly.
- **4) Bayesian Srhinkage and Variable Selection methods**: In Bayesian regression the choice of prior will determine whether the model performs shrinkage, variable selection, or a combination of the two. We will present examples using shrinkage and variable selection priors.

Data

To illustrate the application of the methods listed above, we will use a data set available in the BGLR R-package. This data set provides four phenotypes (see object `wheat.Y`) for 599 wheat lines that were genotyped at 1,279 genetic markers (see object `wheat.X`).

Loading the data

This code below loads the data, center, and scales the the genotypes. While centering and scaling is not strictly needed, it is often a good practice when using penalized (e.g., Lasso) or Bayesian regressions.

```r
library(BGLR)
data(wheat)
head(wheat.Y)
```

```
##               1           2           4          5
## 775   1.6716295 -1.72746986 -1.89028479  0.0509159
## 2166 -0.2527028  0.40952243  0.30938553 -1.7387588
## 2167  0.3418151 -0.64862633 -0.79955921 -1.0535691
## 2465  0.7854395  0.09394919  0.57046773  0.5517574
## 3881  0.9983176 -0.28248062  1.61868192 -0.1142848
## 3889  2.3360969  0.62647587  0.07353311  0.7195856
```

```
  dim(wheat.X)
```

```
## [1]  599 1279
```

```
  X=scale(wheat.X,center=TRUE,scale=TRUE)
  y=wheat.Y[,2] # picks one phenotype

  N<-nrow(X) ; p<-ncol(X)
```

We will compare models based on their ability to predict data that was not used to fit the models. The following code produces a training-testing partition that we will use for all mehtods.

Creating a Training-Testing partition

```
set.seed(12345)
tst<-sample(1:N,size=150,replace=FALSE)
XTRN<-X[-tst,]
yTRN<-y[-tst]
XTST<-X[tst,]
yTST<-y[tst]
```

1) Indenpendent screening

To implement this approach, we will first rank predictors based on the marginal association of each predictor with the response. This is done using the training data only.

```
pValues<-numeric()
for(i in 1:p){
   fm<-lsfit(y=yTRN,x=XTRN[,i])
   pValues[i]<-ls.print(fm,print.it=F)$coef[[1]][2,4] # extracts p-value, similar to lm() but a bit fa
}
```

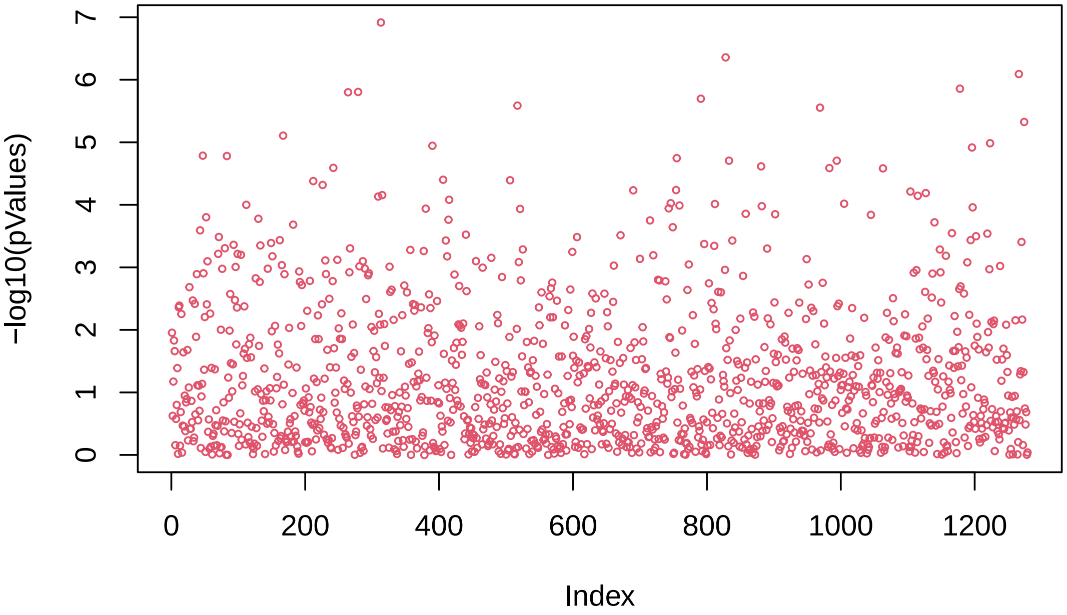


Figure 1: Marginal association p-value

Let's now build models using the top-q (q=1,. . . ,300) markers. The script: - Ranks markers based on p-values (from smallest to largest). - Fit models using the top 1, top 2, . . . , top-q markers using data from the training set. - For each of the fitted model the script computes the correlation between phenotype and predictions within the training data and in the testing data.

Building prediction models using the top-q markers

```
mrk_rank<-order(pValues); corTRN<-numeric(); corTST<-numeric()
for(i in 1:300){
    tmpIndex<- mrk_rank[1:i]
    ZTRN=XTRN[,tmpIndex,drop=F]
    ZTST=XTST[,tmpIndex,drop=F]

    fm<-lm(yTRN~ZTRN)
    bHat=coef(fm)[-1]
    bHat<-ifelse(is.na(bHat),0,bHat)

    yHatTRN=ZTRN%*%bHat
  corTRN[i]<-cor(yTRN,yHatTRN)

    yHatTST=ZTST%*%bHat
    corTST[i]<-cor(yTST,yHatTST)

}
```

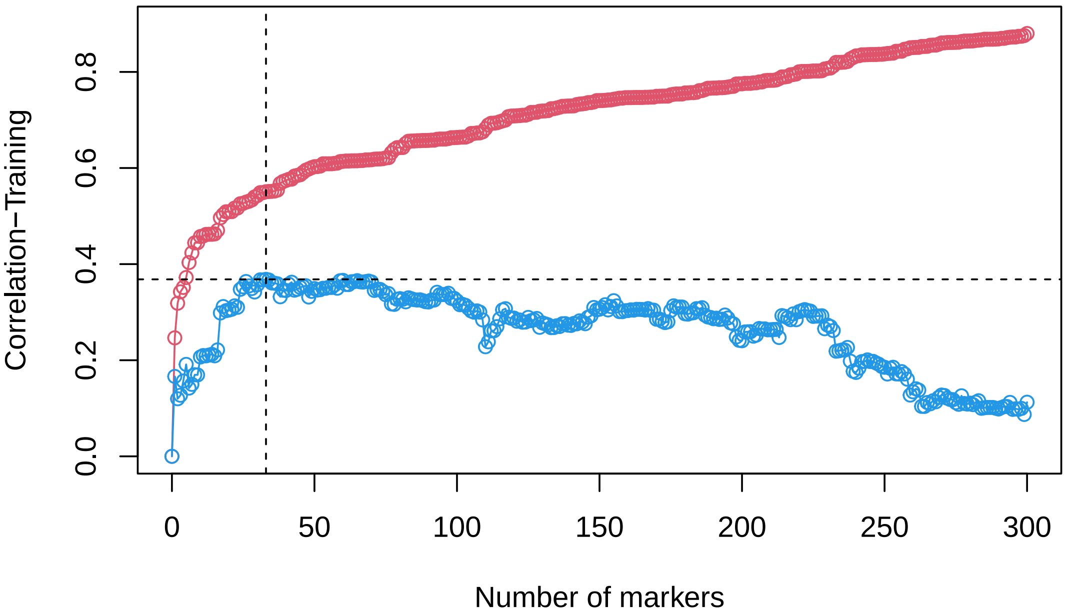


Figure 2: Correlation between predictions and phenotypes in training and testing set by model DF.

Remarks

- Goodness of fit in the training data set (`corTRN`, blue) increases with DF
- However prediction accuracy in testing data (`corTST`, red) increases, reaches a plateau, and then decreases.

The curves presented in the previous figure are estimates subject to sampling variability. To quantify this and to reduce the variance of estimtes we can conduct many training-testing partitions and average across them. This is illustrated in the following figure, each of the skyblue lines is an estiamte derived from a training-testing partition. The solid red line is an averag across partitions.

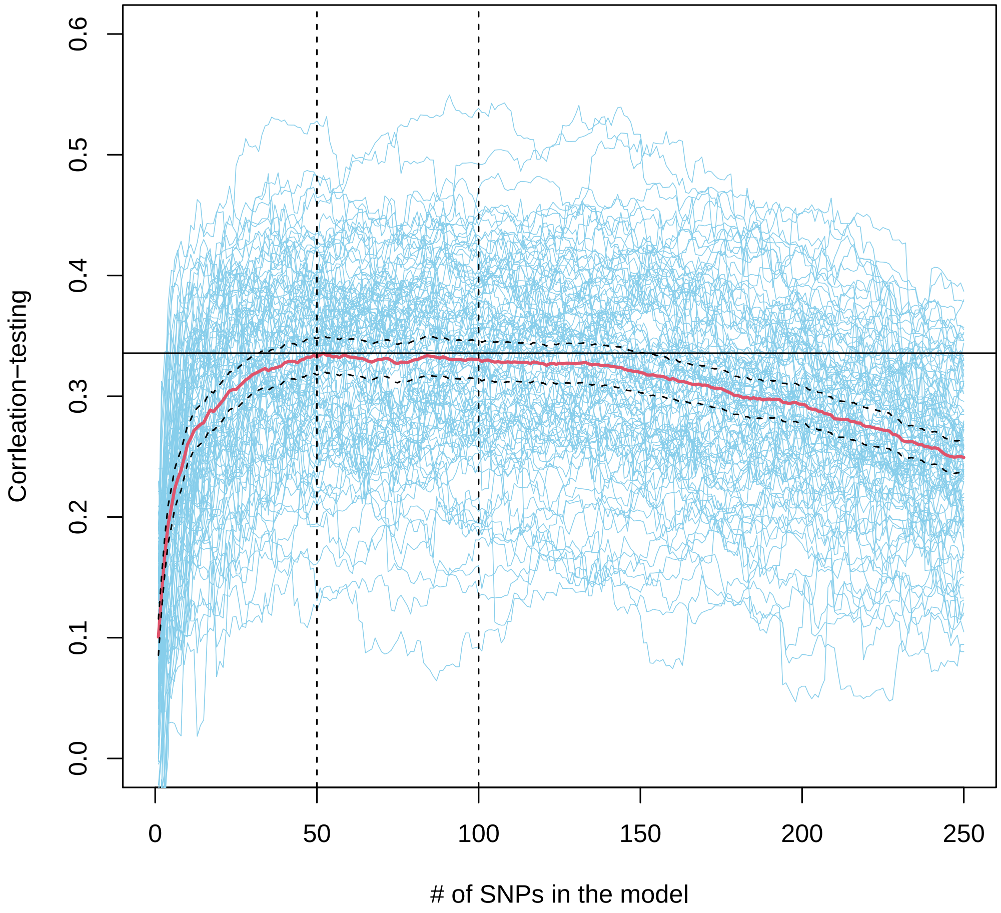


Figure 3: Correlation between predictions and phenotypes in testing data by model DF. (100 training-testing partitions)

2) Forward Regression

One limitation of independent screening is that the predictors' rank is based on the marginal association with the outcome. This does not guarantee that in each step the predictor added to the model is the one that produces the best improvement to the model. To see this, imagine a situation where two predictors (say X1 and X2) are almost perfectly correlated. If they are almost perfectly correlated, they will receive a similar rank based on their marginal association with the outcome. Imagine X2 was ranked before X1, and that you are at a step where X2 was already included in the model. Would the inclusion of X1 in the model, after X2 was added, improve the model's ability to fit the training data? The answer is not because X1 is almost perfectly correlated with another variable (X1) already in the model; therefore, X1 provides very little additional information.

The problem above-described suggests that perhaps we should evaluate predictors based on their potential contribution to the model, conditional on all the predictors that have already ented into the model. This is what forward regression does. Schematically:

1) We assess the marginal association between the outcome and each predictor (say we have p candidate predictors, X1, . . . , Xp.
2) We include the top-ranked predictor, assuming it is X2 the model becomes $y_i = \mu + x_{2i}\beta_2 + \varepsilon_i$.
3) Subsequently, we evaluate the all possible models with two predictors: $y_i = \mu + x_{2i}\beta_2 + x_{3i}\beta_j + \varepsilon_i$ for $j = 1, 3, ..., p$. We chose among these models the one with the smallest residual sum of squares.
4) We repeat #3, each time adding to the current model the one with the smallest RSS.

The base package of R includes the `step()` function which can be used for forward regression, backward elimination, and stage-wise approaches that combine forward and backward regression. These functions can be used with objects from multiple regression methods (e.g., lm, glm). They perform very well for problems involving a limited number of candidate predictors. However, these functions can be very slow if the number of predictors is large. The `BGData` provides a function (FWD) for forward regression that is optimized for big data with large number of predictors. The following code uses this function and compares the results with that of independent screening.

```r
library(BGDataExt)
```

```
##
## Attaching package: 'BGDataExt'

## The following object is masked from 'package:graphics':
##
##     segments
```

```r
FM=FWD(y=yTRN,X=XTRN,df=100,verbose=FALSE)

FM$path$variable # gives the order with which predictors entered the model
```

```
##   [1] "Int"       "wPt.2151" "c.305387" "wPt.7160" "c.344090" "c.375371"
##   [7] "c.312155" "c.381717" "wPt.2406" "wPt.1313" "wPt.3533" "wPt.1554"
##  [13] "c.305115" "c.347702" "c.306153" "c.372528" "c.346343" "c.343999"
##  [19] "c.378173" "wPt.8463" "c.378765" "wPt.4706" "wPt.7024" "wPt.8616"
##  [25] "c.303848" "c.304782" "c.375127" "c.373172" "wPt.0164" "c.304454"
##  [31] "wPt.0413" "wPt.4569" "wPt.0245" "wPt.3116" "wPt.9780" "wPt.3566"
##  [37] "c.343821" "c.343777" "c.408424" "wPt.1085" "wPt.7101" "c.373080"
##  [43] "wPt.2291" "c.344517" "c.349495" "wPt.1048" "wPt.6853" "wPt.9454"
##  [49] "wPt.9951" "wPt.8226" "wPt.1100" "wPt.9859" "wPt.8752" "wPt.0312"
##  [55] "c.348774" "c.117419" "wPt.8721" "c.377823" "c.344260" "c.344062"
##  [61] "c.379269" "wPt.5128" "wPt.5067" "c.304317" "wPt.0105" "c.345254"
##  [67] "wPt.0205" "c.348141" "c.377416" "c.376040" "wPt.7068" "wPt.5231"
##  [73] "wPt.9796" "wPt.8043" "wPt.4725" "wPt.0008" "c.376406" "wPt.2260"
##  [79] "wPt.3605" "wPt.0832" "c.304172" "c.378892" "c.373816" "c.304691"
##  [85] "c.346852" "c.344082" "c.372567" "wPt.6904" "c.378083" "wPt.3611"
##  [91] "c.305238" "wPt.1482" "wPt.9103" "wPt.5590" "wPt.9668" "c.343987"
##  [97] "wPt.8006" "c.379865" "c.372669" "wPt.2614" "wPt.7004"
```

```
#FM$path$RSS, $path$AIC, $path$BIC, $path$LogLik give the corresponding statistics at each step in the

dim(FM$B) # gives the estimated effects at each step in the forward path
```

```
## [1] 1280  101
```

```
COR=rep(NA,100)

for(i in 2:101){ # first model is the intercept-only
  COR[i-1]=cor(yTST,cbind(1,XTST)%*%FM$B[,i])
}

plot(COR,x=1:100,type='o')
```

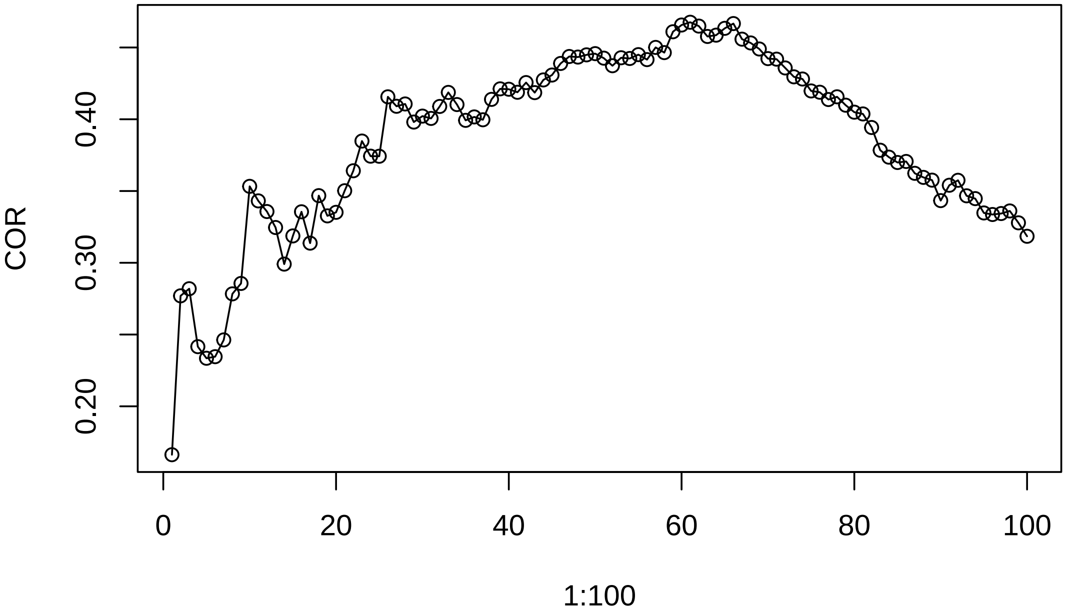


Figure 4: Prediction correlations obtained with forward regression.

We see that with ~60 predictors, the forward regression achieves a slighlty higher prediction correlation than any of the correlations obtained with indpeendent screening.

Best subset selection

The problem we are trying to tackle can be described as follows:

- We have an outcome (Y) and a large number of predictors (X1,. . . ,Xp).
- We want to find the best model among all the models that can be formed using these *p* predictors.

The problem is known as *best subset selection*. One difficulty is that the number of possible models can be extrremely large; thus, evaluating all possible models is usually not doable. Independent screening, forward regression (and related procedures such as backward elimination) can be seen as approaches that evaluate a subset of all possible models (the subest is defined by the path produced by these algorithms). While none of these procedures guarantee that we can achieve the best model, the hope is that the path develped by these algorithms will include a model with a perfromance (e.g., prediction accuracy) close to the best possible model. It turns out that best subset selection can be seen as a penalized regression problem–a approach that we discuss below.

Regularized Regression

A groundbreaking study by James & Stein (1961) showed that in some settings the ML estimator could be indmisible; that is, they showed that in some cirumbstances there was another estimator that had lower Mean-Squared Error (MSE) over the entire parameter space. Their estimator shrunks the least square estimates towards zero; thus reducing the variance of estimates.

Recall that the MSE of an estimator can be decomposed as the sum of the variance plus the squared of the bias of the estimator $MSE = E[(\hat{\theta} - \theta)^2] = Variance + Bias^2$. Shrinkage reduces the variance of estimates at the expense of some bias. However, when the number of parameters to be estimated is large, the reduction in variance overcomes the increase in bias; thus leding to a reduction in MSE.

There are many ways to obtain regularized estimates; two commonly used approaches are penalized and Bayesian methods. We discussed each of them in the next two sections. In most cases there is a duality between the two approaches by which a penalized estimator can be viewed as the posterior mode from a Bayesian model.

3) Penalized regressions using glmnet

In a penalized regression, estimates are obtained by minimizing a penalized log-likelihood or, in the case of linear models, a penalized residual sum of squares:

$\hat{\beta} = argmin\{(y - X\beta)'(y - X\beta) + \lambda J(\beta)\}$

where $J(\beta)$ is a penalty function. Common choices for the penalty function are the

- L2-norm, $J(\beta) = \sum_j \beta_j^2$ (aka Ridge Regression, Hoerl and Kennard 1970),
- L-1 norm $J(\beta) = \sum_j |\beta_j|$ (aka Lasso, Tibshirani, 1996), and,
- Linear combinations of the two $J(\beta) = (1-\alpha)\sum_j \beta_j^2 + \alpha \sum_j |\beta_j|$ (aka Elastic Net, Zhou and Hastie, 2005), for some $\alpha \in [0, 1]$.

Choosing $\lambda = 0$ leads Ordinary Least Squares estimates. Ridge regression shrunk OLS estimates towards zero, without making variable selection. Lasso and Elastic Net combine variable selection and shrinkage.

Commonly, these models are fitted over a grid of values of the regularization parameter (λ); an optimal value for that parameter is often chosen by evaluating the ability of the fitted models to predict data that was not used to train the models (i.e., testing data).

Note: Best subset selection is obtained using as a penality a function that counts the number of non-zero effects: $J(\beta) = \Sigma_j 1(\beta_j \neq 0))$.

Ridge Regression (RR)

In the RR (Hoerl and Kennard 1970) $J(\beta) = \sum_j \beta_j^2 = \beta'\beta$; thus, the objective function becomes

$\hat{\beta} = argmin\{(y - X\beta)'(y - X\beta) + \lambda\beta'\beta\} = argmin\{y'y + \beta'(X'X + I\lambda)\beta - 2\beta'X'y\}$

The solution can be shown to be

$\hat{\beta} = (X'X + I\lambda)^{-1}X'y$

Adding λ to the diagonal entries of $X'X$ srhinks estimates towards zero. This is illustrated in the following simplified example.

```
  set.seed(195021)
  # Toy simulation
   n=50
   p=3  # number of predictors
   W=matrix(nrow=n,ncol=p,rnorm(n*p))
   b=c(-1,1,2) # true effects
   signal=W%*%b
   error=rnorm(sd=sd(signal),n=length(signal))
   y=signal+error
   # centering to avoid the need of including an intercept
   W=scale(W,center=T,scale=F)
   y=y-mean(y)

  # OLS
  WW=crossprod(W)
  Wy=crossprod(W,y)
  bOLS=solve(WW,Wy)

  # Ridge regression
  lambda=3
  C=WW
  diag(C)=diag(C)+lambda
  bRR_3=solve(C,Wy)

  lambda=10
  C=WW
  diag(C)=diag(C)+lambda
  bRR_10=solve(C,Wy)


  lambda=100
  C=WW
  diag(C)=diag(C)+lambda
  bRR_100=solve(C,Wy)
  round(cbind('true_effect'=b,'ols'=bOLS,'RR_3'=bRR_3,'RR_10'=bRR_10,'RR_100'=bRR_100),3)
```

```
##      true_effect
## [1,]          -1 -1.073 -1.040 -0.965 -0.461
## [2,]           1  0.964  0.871  0.707  0.192
## [3,]           2  2.509  2.350  2.051  0.797
```

Let's compare estimates in just one Monte Carlo (MC) replicate (to estimate MSE we should average over many MC replicates).

```
 sum((b-bOLS)^2)
```

```
## [1] 0.2656491
```

```
 sum((b-bRR_3)^2)
```

```
## [1] 0.140734
```

```
 sum((b-bRR_10)^2)
```

```
## [1] 0.08950725
```

```r
sum((b-bRR_100)^2)
```

```
## [1] 2.389746
```

In this example suign $\lambda = 3$ or $\lambda = 10$ improved the estimates (smaller distnace to the true parameter values), but using $\lambda = 100$ induced too much shrinkage towards zero, thus increasing the squared-difference between esitmates and true parameter values.

Unlike the Ridge Regression, Lasso and Elastic Net estimates do not have a closed form; however, estimates can be derived using iterative algorithms (e.g., a coordiante-descent gratient) such as the ones implemented in the `glmnet` R-package.

Fitting Penalized Regressions using the glmnet R-package

The following code shows how to implement Ridge Regression, Lasso, and Elastic Net using the `glmnet` package. By default, `glmnet` fits models over a grid of 100 values of the regularization parameter λ. The plots produced at the end of the script display, for each of the models, the correlation between predictions and observations in testing data, by value of λ.

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-2
```

```r
# alpha 0 gives Ridge Regression
fmRR=glmnet(y=yTRN,x=XTRN,alpha=0)
dim(fmRR$beta)
```

```
## [1] 1279  100
```

```r
length(fmRR$lambda)
```

```
## [1] 100
```

```r
range(fmRR$lambda)
```

```
## [1]   2.477649 247.764898
```

```r
# alpha 1 gives Lasso
fmL=glmnet(y=yTRN,x=XTRN,alpha=1)

# alpha between 0 and 1 gives elastic net
fmEN=glmnet(y=yTRN,x=XTRN, alpha=0.5)

COR.RR=rep(NA,100)
COR.L=rep(NA,100)
COR.ENet=rep(NA,100)

# evaluating correlation in TST set
for(i in 1:100){
  COR.RR[i]=cor(yTST,XTST%*%fmRR$beta[,i])
  COR.L[i]=cor(yTST,XTST%*%fmL$beta[,i])
  COR.ENet[i]=cor(yTST,XTST%*%fmEN$beta[,i])
}
```

```
## Warning in cor(yTST, XTST %*% fmL$beta[, i]): the standard deviation is zero

## Warning in cor(yTST, XTST %*% fmEN$beta[, i]): the standard deviation is zero
```

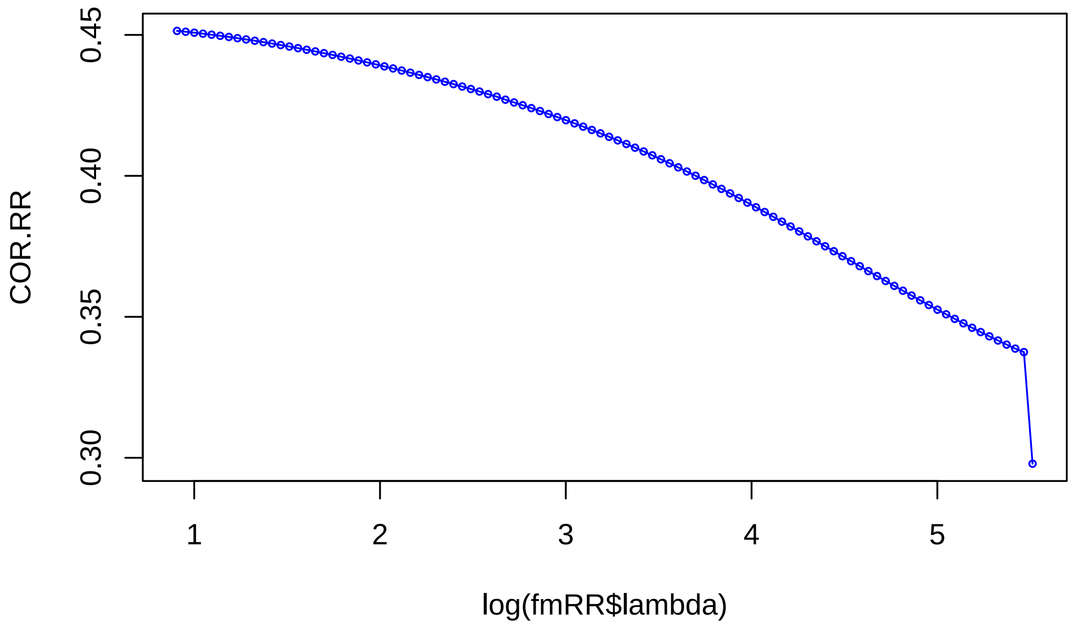


Figure 5: Correlation between predictions and phenotypes in testing, Rdige Regression

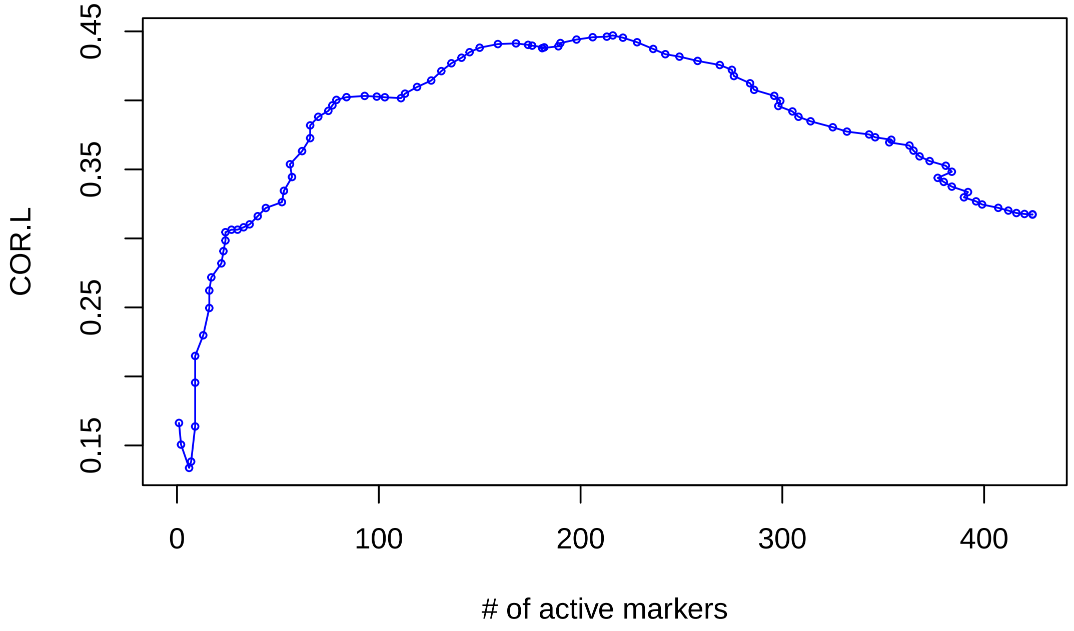


Figure 6: Correlation between predictions and phenotypes in testing, Lasso

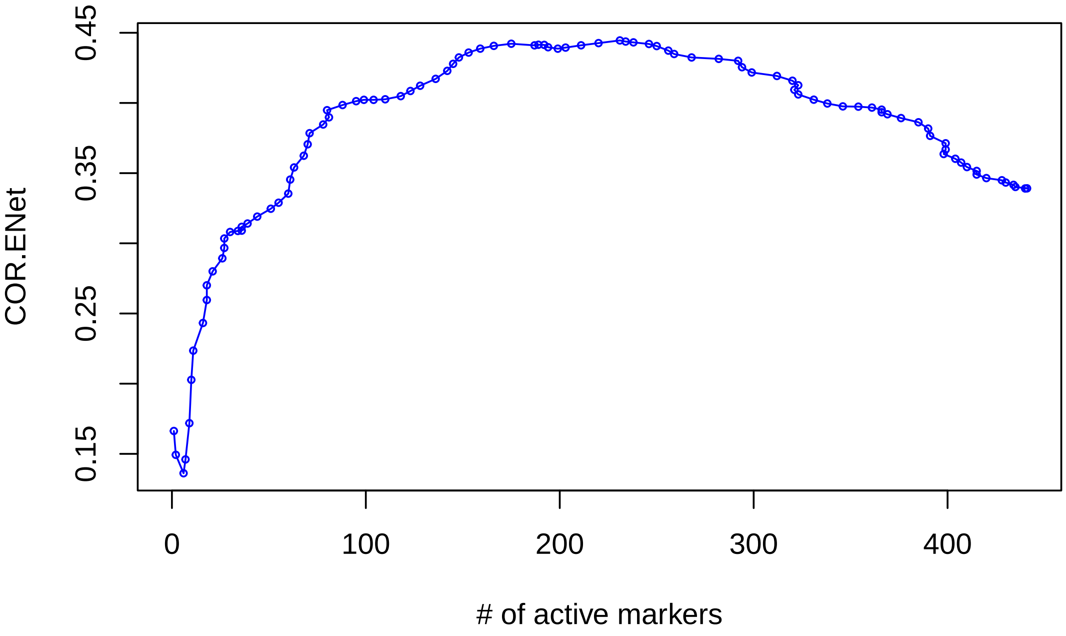


Figure 7: Correlation between predictions and phenotypes in testing, Elastic Net

Remarks

- The `glmnet` function fits each of the models over a grid of values of λ, the rules used to choose those values are described in Friedman, Hastie, and Tibshirani (2010).
- The matrix `$beta` has the solutions (estimated effects) obtained for each value of λ‘
- After fitting the model we evaluate prediction accuracy by correlating the testing phenotypes `yTST` with predictions (see loop above).
- In the case of Lasso and Elastic Net the default values of lambda led to an internal maxima, i.e., an internal region with maximum correlation. This is not the case for the Ridge Regression, the grid of values of λ used in that case may need to use smaller values of λ. The following example produce new fits using a user-provided grid of values for the regularization parameter.

```
lambda=c(fmRR$lambda,min(fmRR$lambda)*seq(from=0.9,to=0.01,length=50))
fmRR=glmnet(y=yTRN,x=XTRN,alpha=0,lambda=lambda)

# evaluating correlation in TST set
COR.RR=rep(NA,length(fmRR$lambda))
for(i in 1:length(fmRR$lambda)){
 COR.RR[i]=cor(yTST,XTST%*%fmRR$beta[,i])
}
```

- The three models achieve correlations of about 0.45 (Ridge Regression does slightly better)
- This is much better than what we obtained selecting markers based on their marginal association (correlation ~0.37, Example 1).
- Remember that estimates of accuracy, such as the ones discussed above, are point estimates subject to sampling variability; there is sampling variance emerging from the sampling of training and testing data. In the In-class assignment you will be asked to repeat the examples using many training-testing partitions; we will use those results to assess sampling variability on these estimates and also to get a more precise estimate.

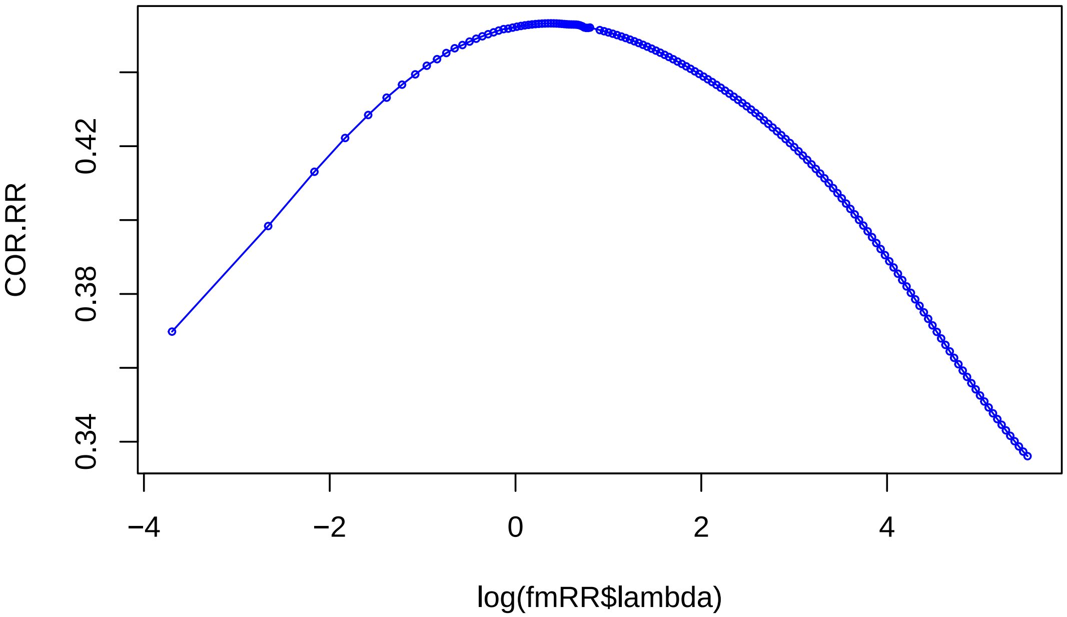


Figure 8: Correlation between predictions and phenotypes in testing, Ridge Regression

4) Bayesian Regressions

In a Bayesian model, estimates are obtained from the posterior distribution of the model uknowns (e.g., regression coefficients).

Recall that from Baye's rule we have that $p(A|B) = \frac{p(A,B)}{p(B)} = \frac{p(B|A)P(A)}{p(B)}$. Taking A to be the model parameters (e.g., β in a regression model), and B to be the data (y), we have that

$p(\beta|y) = \frac{p(y|\beta)p(\beta)}{p(y)}$

Above,

- $p(\beta|y)$ is the posterior distribution of the parameters given the data (the object we use to summarize knowledge and uncertainty about model parameters),
- $p(y|\beta)$ is the conditional distribution of the data given the parameters, the likelihood function when viewed as a function of β,
- $p(\beta)$ is the prior distribution of the model uknowns, the object we use to summarize *prior knowledge*, and
- $p(y) = \int p(y|\beta)p(\beta)\, d\beta$.

The last object, $p(y)$, is the marginal distribution of the data. This object does not involve the uknown parmaeters; therefore, the postrior distribution is proportional to the product of the likelihood times the prior distribution

$p(\beta|y) \propto p(y|\beta)p(\beta)$

This makes evident how the posterior distribution (and inferences from it, e.g., the posterior mean, or the posterior mode) depends on both evidence provided by the data, quantified via the *likelihood function*, and *prior knowledge* summarized by the prior distribution.

3.1) A Bayesian model with a Gaussian likelihood and a Gaussian prior

Let's consider a linear model $y = X\beta + \varepsilon$ with a Gaussian likelihood,

$p(y|X,\beta) = N(X\beta, I\sigma^2_\varepsilon)$

The ML estimator can be shown to be the OLS estimator

$\hat{\beta} = (X'X)^{-1}X'y$

Conisder now using a Gaussian IID prior with zero-mean and variance σ^2_β, that is

$p(\beta) = N(0, I\sigma^2_\beta)$

The posterior distribution becomes

$p(\beta|y,\sigma^2_\varepsilon,\sigma^2_\beta) \propto N(X\beta, I\sigma^2_\varepsilon) \times N(0, I\sigma^2_\beta)$

This can be shown to be proportional to a Multivariate Normal distribution with mean $\tilde{\beta} = (X'X + I\lambda)^{-1}X'y$ and variance-covariance matrix $V = (X'X + I\lambda)^{-1}\sigma^2$, where $\lambda = \sigma^2/\sigma^2_\beta$. Note that $\tilde{\beta} = (X'X + I\lambda)^{-1}X'y$ is the Ridge-regression estimator. Thus, Ridge Regression estiamtes can be seen as the posterior mean (also the posterior mode) of the vector of effects in a Gaussian regression model with IID Gaussian prior.

The regularization parameter, $\lambda = \sigma^2/\sigma^2_\beta$, is a noise-to-signal ratio. In penalize regressions, this parameter is often chosen using cross-validation (see section on penalized regressions, above). In a Bayesian context, the variances can be treated as unknown (e.g., by assigning them a scaled-inverse chi-square prior); thus inferring effects and variances jointly from the training data. This is illustrated in the following example wich uses the BGLR R-package.

```
library(BGLR)
 nIter=6000 # I set this to small value that way it will run quickly, for more serious analyses use lo
 burnIn=1000 # and longer burnin
# Gaussian prior ("Bayesian Ridge-Regression")
 LP=list( list(X=XTRN,model='BRR') ) # 2-level list, allows specifying different types of random and f
 fmBRR=BGLR(y=yTRN,ETA=LP,nIter=nIter,burnIn=burnIn,saveAt='BRR_',verbose=FALSE)
 # Retriving samples from the variance parameters

 vE=scan('BRR_varE.dat',quiet=TRUE)
 vB=scan('BRR_ETA_1_varB.dat',quiet=TRUE)
 lambda=vE/vB
```

Trace plots (left) are used to assess convergence to the posterior distribution; density plots (right) are used to summarize knowledge and uncertainty about parameters given the data

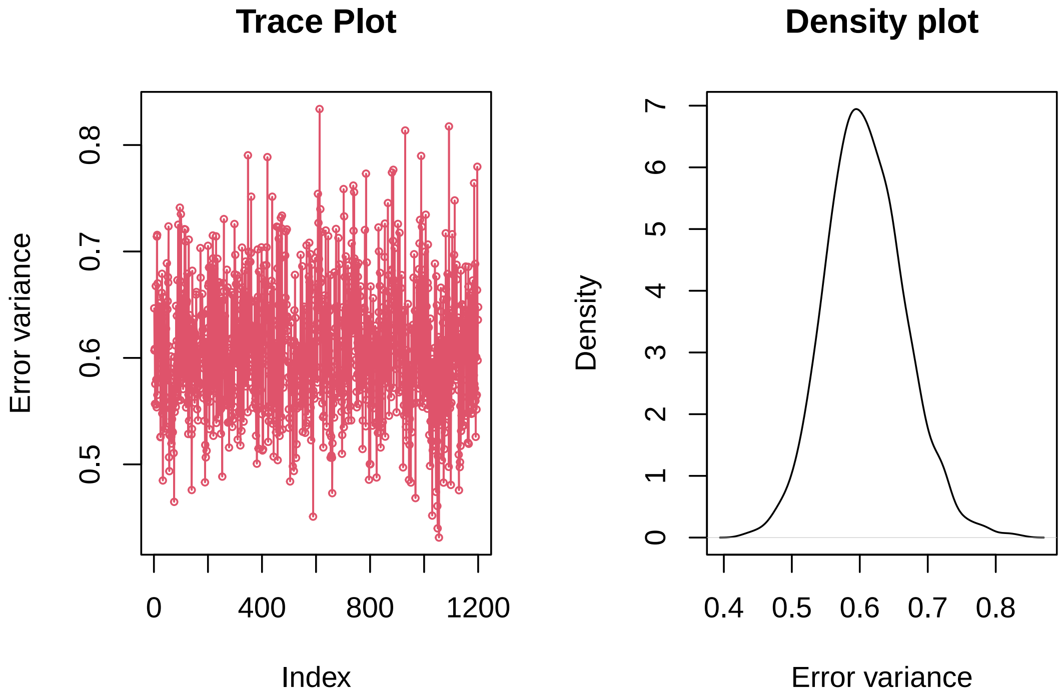


Figure 9: Trace density plots of the error variance.

```
## Prediction accuracy in the testing set
cor( yTST, XTST%*%fmBRR$ETA[[1]]$b)
```

```
##            [,1]
## [1,] 0.4502181
```

```
max(COR.RR)
```

```
## [1] 0.4532439
```

While the RR appears to outperform slightly the Bayesian model in prediction accuracy, the estimate of prediction accuracy for the RR is likely upwardly biased because, below, in the comparison we choose lambda based on the same testing data that is used to evalaute accuracy. On the other hand, predictions from the Bayesian model were derived using the training data only.

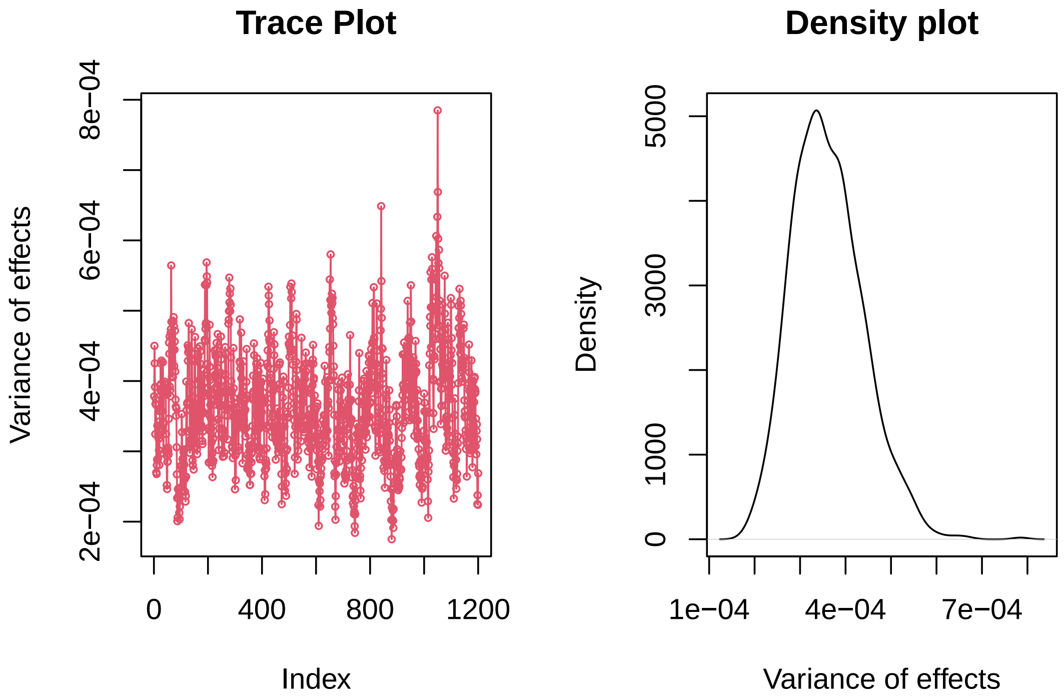


Figure 10: Trace and density plots of the variance of effects.

3.2) Shrinkage and variable selection priors

The Gaussian prior used in the previous section induces shrinkage without performing any variable selection. In the last decade a pletora of Bayesian models using different priors have been developed. These priors can be classified in three main groups: (i) Gaussian, (ii) Thick-tailed priors, this group includes the double-exponential (used in the Bayesian Lasso) and the sclaed-t prior, and (ii) finite mixture priors with a point of mass at zero, these models perform both variable selection and srhinkage. The following figure displays these priors.

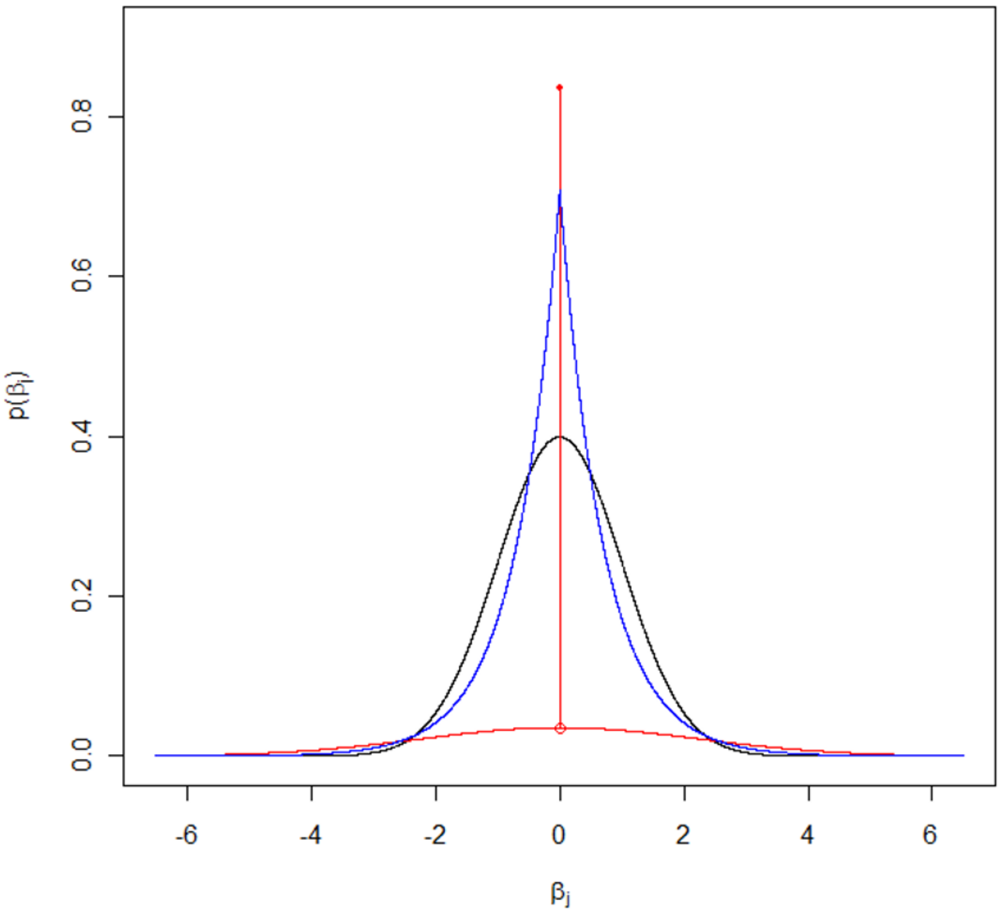


Figure 11: Prior distributions of effects commonly used in Bayesian models

These priors are scaled by multiple *hyper-parameters* (e.g., degree of freedom, scale, prior probability of non-null effects). Fortunately, some of these parameters can be treated as uknown and can be inferred by data.

The BGLR R-package implements some of these priors. The following example illustrates how to fit Bayesian models with different prior distributions of effects. For additional examples and a description of the methods implemented you can check the following GitHub repository (include multiple examples) and the following manuscript.

Model fitting

```
# Scaled-t
 LP[[1]]$model='BayesA'
 fmBA=BGLR(y=yTRN,ETA=LP,nIter=nIter,burnIn=burnIn,saveAt='BA_',verbose=FALSE)

# Double-Exponential
 LP[[1]]$model='BL'
 fmBL=BGLR(y=yTRN,ETA=LP,nIter=nIter,burnIn=burnIn,saveAt='BL_',verbose=FALSE)
```

```
# Spike-slab (Gaussian)
 LP[[1]]$model='BayesC'
 fmBC=BGLR(y=yTRN,ETA=LP,nIter=nIter,burnIn=burnIn,saveAt='BC_',verbose=FALSE)

# Spike-slab (Scaled-t)
 LP[[1]]$model='BayesB'
 fmBB=BGLR(y=yTRN,ETA=LP,nIter=nIter,burnIn=burnIn,saveAt='BB_',verbose=FALSE)
```

Evaluation of Prediction Accuracy

```
bayes=c(
   'BRR'   =cor( yTST, XTST%*%fmBRR$ETA[[1]]$b),
   'BL'    =cor( yTST, XTST%*%fmBL$ETA[[1]]$b),
   'BayesA'=cor( yTST, XTST%*%fmBA$ETA[[1]]$b),
   'BayesB'=cor( yTST, XTST%*%fmBB$ETA[[1]]$b),
   'BayesC'=cor( yTST, XTST%*%fmBC$ETA[[1]]$b)
)
round(bayes,3)
```

```
##    BRR     BL BayesA BayesB BayesC
##  0.450  0.450  0.451  0.455  0.448
```

In general we see no big difference in prediction accuracy, all the Bayesian models achieved in this example a prediction correlation close than the one achieved by penalized regressions.

These models offer more than just predictions. This code illustrates how to extract other parameters (for more details follow the links provided above).

Retrieving samples and estimates

```
##         775        2166        2465        3881        3889        4248
## -0.76327204  0.40523461  0.09598448 -0.43601817  0.39317259 -0.05501539
```

```
## [1] -0.007336212
```

```
## [1] 0.6099191
```

```
## $logLikAtPostMean
## [1] -468.7343
##
## $postMeanLogLik
## [1] -524.2074
##
## $pD
## [1] 110.9462
##
## $DIC
## [1] 1159.361
```

```
## [1] 6000
```

Estimates (posterior means of effects) and posterior standard deviation of effects

```
# Gaussian prior
 head(fmBRR$ETA[[1]]$b)  # posterior means of effects
```

```
##      wPt.0538      wPt.8463      wPt.6348      wPt.9992      wPt.2838
##  6.897426e-04 -3.197053e-03 -1.624006e-03 -5.027481e-05 -1.239100e-03
##      wPt.8266
##  1.392390e-03
```

```
   head(fmBRR$ETA[[1]]$SD.b) # posterior SDs
```

```
##   wPt.0538   wPt.8463   wPt.6348   wPt.9992   wPt.2838   wPt.8266
## 0.01763717 0.01804330 0.01781049 0.01855480 0.01793263 0.01848906
```

```
  # Bayes A
    head(fmBA$ETA[[1]]$b)
```

```
##      wPt.0538      wPt.8463      wPt.6348      wPt.9992      wPt.2838
##  4.787594e-04 -1.945393e-03  9.506763e-05  1.217393e-04 -1.006084e-03
##      wPt.8266
##  4.190577e-04
```

```
    head(fmBA$ETA[[1]]$SD.b) # posterior SDs
```

```
##   wPt.0538   wPt.8463   wPt.6348   wPt.9992   wPt.2838   wPt.8266
## 0.01550164 0.01576625 0.01645243 0.01629077 0.01745095 0.01594865
```

```
  # Bayesian Lasso
    head(fmBL$ETA[[1]]$b)
```

```
##      wPt.0538      wPt.8463      wPt.6348      wPt.9992      wPt.2838
##  0.0007948838 -0.0024648168 -0.0003986224 -0.0009162976 -0.0008811480
##      wPt.8266
##  0.0010594661
```

```
    head(fmBL$ETA[[1]]$SD.b) # posterior SDs
```

```
##   wPt.0538   wPt.8463   wPt.6348   wPt.9992   wPt.2838   wPt.8266
## 0.01662410 0.01621366 0.01510588 0.01760440 0.01669351 0.01506010
```

```
  # BayesC
    head(fmBC$ETA[[1]]$b)
```

```
##      wPt.0538      wPt.8463      wPt.6348      wPt.9992      wPt.2838
##  0.0002725590 -0.0014128833 -0.0011997475  0.0001290981 -0.0010514157
##      wPt.8266
## -0.0007484630
```

```
    head(fmBC$ETA[[1]]$SD.b) # posterior SDs
```

```
##   wPt.0538   wPt.8463   wPt.6348   wPt.9992   wPt.2838   wPt.8266
## 0.02679801 0.02468063 0.02663396 0.02624973 0.02642326 0.02585304
```

```
  # BayesB
    head(fmBB$ETA[[1]]$b)
```

```
##      wPt.0538      wPt.8463      wPt.6348      wPt.9992      wPt.2838
## -0.0003210662 -0.0026599218 -0.0017579226  0.0002400352 -0.0012135738
##      wPt.8266
## -0.0002733823
```

```
    head(fmBB$ETA[[1]]$SD.b) # posterior SDs
```

```
##   wPt.0538   wPt.8463   wPt.6348   wPt.9992   wPt.2838   wPt.8266
## 0.03163931 0.03017243 0.03050227 0.03106210 0.03578673 0.02951179
```

Posterior probability of non-zero effect

Models `BayesB` and `BayesC` use priors with a point of mass at zero. In these models, at each iteration of the sampler, only a fraction of the predictors have non-zero effects. We can use these models to estimate the overall proportion of non-zero effects, and also the posterior probability of inclusion for each of the predictors.

Overall proportion of non-zero effects

```
fmBC$ETA[[1]]$probIn
```

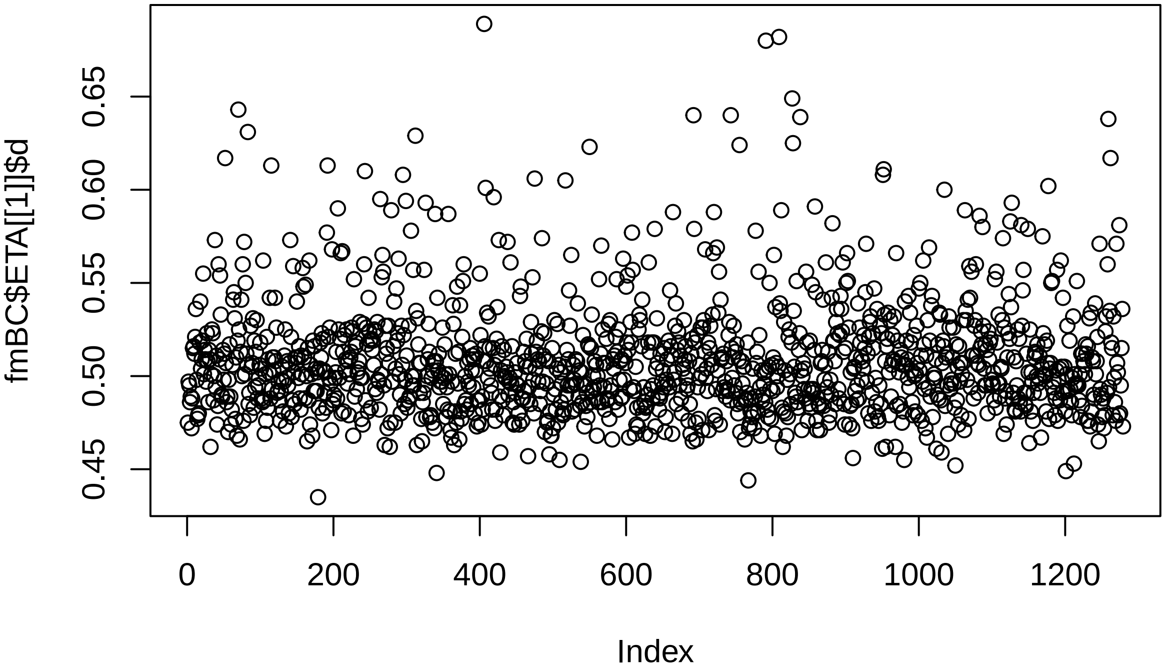
```
## [1] 0.5092388
```

```
fmBB$ETA[[1]]$probIn
```

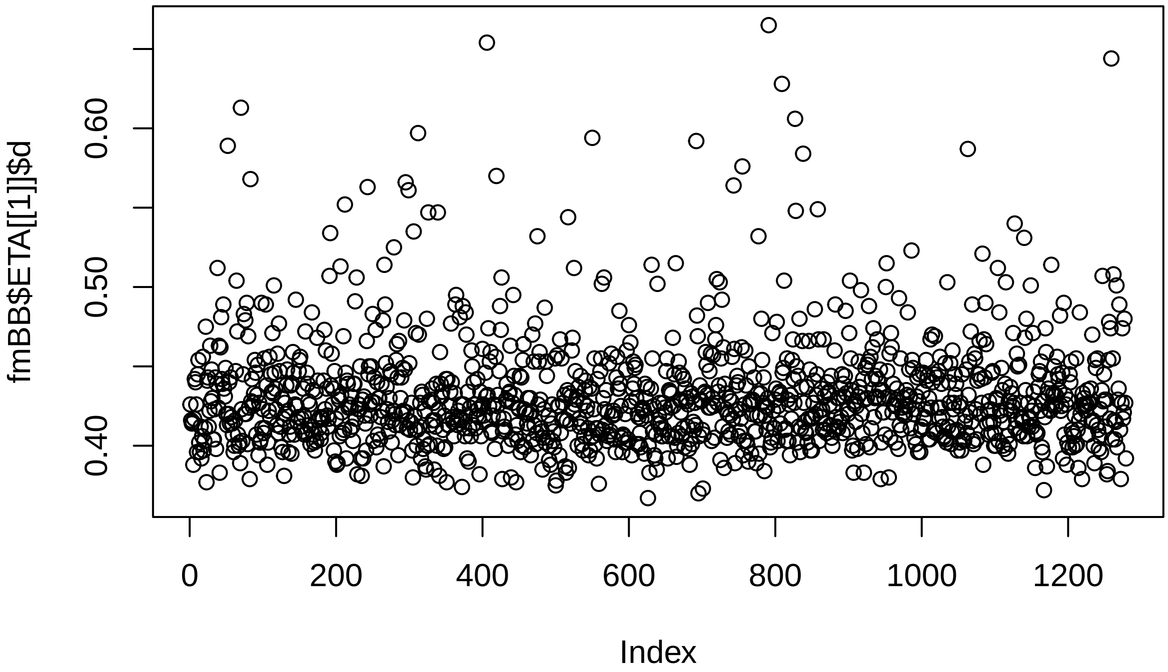
```
## [1] 0.4327404
```

Probability of inclusion by predictor

```
  plot(fmBC$ETA[[1]]$d) # posterior probability of inclussion
```



```
  plot(fmBB$ETA[[1]]$d) # posterior probability of inclussion
```


fmBB$ETA[[1]]$d
0.40
0.50
0.60
0
200
400
600
800
1000
1200
Index