Regularized selection indices for breeding value prediction using hyper-spectral image data

Marco Lopez-Cruz[1], Eric Olson[1], Gabriel Rovere[2,3,4], Jose Crossa[6], Susanne Dreisigacker[6], Suchismita Mondal[6], Ravi Singh[6], and Gustavo de los Campos[3,4,5,*]

{[1]Department of Plant, Soil and Microbial Sciences, [2]Department of Animal Science, [3]Department of Epidemiology and Biostatistics, [4]Institute for Quantitative Health Science and Engineering, [5]Department of Statistics and Probability}, Michigan State University, USA
[6]International Maize and Wheat Improvement Center (CIMMYT), Mexico

*Corresponding author (e-mail: gustavoc@msu.edu)

Contents

1 Selection index

Recall the basic genetic model that decomposes phenotypic observations, $\boldsymbol{y} = (y_1, \ldots, y_n)'$, as the sum of breeding values, $\boldsymbol{u}_y = (u_{y_1}, \ldots, u_{y_n})'$, and environmental deviations, $\boldsymbol{e}_y = (e_{y_1}, \ldots, e_{y_n})'$, as

$$y_i = u_{y_i} + e_{y_i} \tag{1}$$

Both $\boldsymbol{u}_y$ and $\boldsymbol{e}_y$ are assumed to have null means and $\text{var}(\boldsymbol{u}_y) = \sigma^2_{u_y}\mathbf{K}$, $\text{var}(\boldsymbol{e}_y) = \sigma^2_{e_y}\mathbf{I}$, and $\text{cov}(\boldsymbol{u}, \boldsymbol{e}') = \mathbf{0}$, where $\sigma^2_{u_y}$ and $\sigma^2_{e_y}$ are the common genetic and residual variances, respectively; $\mathbf{K}$ is the **kinship relationship matrix** and $\mathbf{I}$ is an identity matrix.

A **selection index** (SI) (Hazel, 1943; Smith, 1936) is used to predict the genetic merit for a the selection goal (u_{y_i}, e.g., the genetic merit for grain yield of the i^{th} genotype) as a linear combination of p measured phenotypes, $\boldsymbol{x}_i = (x_{i1}, ..., x_{ip})'$, of the form

$$\mathcal{I}_i = \boldsymbol{x}_i'\boldsymbol{\beta} \tag{2}$$

where $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)'$ is a vector of regression coefficients whose entries define the weights of each of the measured phenotypes in the SI.

1.1 Standard selection index

In a **standard SI** the weights are derived by minimizing the expected squared deviation between the genetic merit for the selection goal and the index, that is

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \frac{1}{2}\mathbb{E}\left(u_{y_i} - \boldsymbol{x}_i'\boldsymbol{\beta}\right)^2$$

Assuming that $\boldsymbol{x}_i$ follows also model (1) and that their environmental effects ($\boldsymbol{e}_{x_i}$) are orthogonal to u_{y_i}, the above problem is equivalent to

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \left[-\mathbf{G}'_{x,y}\boldsymbol{\beta} + \frac{1}{2}\boldsymbol{\beta}'\mathbf{P}_x\boldsymbol{\beta}\right] \tag{3}$$

where $\mathbf{G}_{x,y} = \mathbb{E}(u_{y_i}\boldsymbol{x}_i)$ is a vector containing the genetic covariances between the selection target and each of the measured phenotypes, and $\mathbf{P}_x = \mathbb{E}(\boldsymbol{x}_i\boldsymbol{x}_i')$ is the phenotypic (co)variance matrix of $\boldsymbol{x}_i$. The solution to this optimization problem is

$$\hat{\boldsymbol{\beta}} = \mathbf{P}_x^{-1}\mathbf{G}_{x,y} \tag{4}$$

The SI is by construction the **best linear predictor** (BLP) of the genetic merit for the selection goal; this property holds when $\mathbf{G}_{x,y}$ and $\mathbf{P}_x$ are known.

2 Regularized selection indices

2.1 Principal components-based selection indices

Principal components (PC) can be obtained from the singular value decomposition of the real-valued matrix, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]'$ (the matrix containing all measured traits) which takes the form $\mathbf{X} = \mathbf{UDV}'$. Matrix $\mathbf{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_p]$ contains the left-singular vectors, $\mathbf{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_p]$ is the matrix with the right-singular vectors, and $\mathbf{D} = \text{diag}(d_1, \ldots, d_p)$ is a diagonal matrix with positive or zero elements. The PCs $\mathbf{W} = \mathbf{XV} = \mathbf{UD}$ are then linear combinations of the measured phenotypes. A **principal components-based SI** (**PC-SI**) uses the first q PCs ($\tilde{\mathbf{W}} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_q]$, $q \leq p$) as "measured phenotypes" in the SI, as

$$\mathcal{I}_i = \tilde{\boldsymbol{w}}_i'\boldsymbol{\gamma}^{(q)}$$

where $\tilde{\boldsymbol{w}}_i'$ is a vector containing the scores for the i^{th} observation on the first q PCs and $\boldsymbol{\gamma}^{(q)}$ are their corresponding regression coefficients. The solution to the optimization problem takes the form $\hat{\boldsymbol{\gamma}}^{(q)} = (n-1)(\tilde{\mathbf{D}}^2)^{-1}\mathbf{G}_{\tilde{w},y}$, where $\tilde{\mathbf{D}}$ contains only the first q elements of $\mathbf{D}$, and $\mathbf{G}_{\tilde{w},y}$ is a vector containing the genetic covariances between each of the top q PCs and the selection objective. This solution can be mapped to coefficients for the measured traits, as in expression (2), using

$$\hat{\boldsymbol{\beta}}^{(q)} = (n-1)\tilde{\mathbf{V}}(\tilde{\mathbf{D}}^2)^{-1}\mathbf{G}_{\tilde{w},y} \tag{5}$$

where $\tilde{\boldsymbol{V}}$ is the matrix containing only the first q right-singular vectors.

2.2 Penalized selection indices

In a penalized regression, regularization is achieved by including in the objective function a penalty on model complexity. A **penalized selection index** (**PSI**) is obtained by considering an optimization problem as in (3), as follows:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta}\left[-\mathbf{G}_{x,y}'\boldsymbol{\beta} + \frac{1}{2}\boldsymbol{\beta}'\mathbf{P}_x\boldsymbol{\beta} + \lambda \cdot J(\boldsymbol{\beta})\right] \tag{6}$$

where λ is a penalty parameter and $J(\boldsymbol{\beta}_i)$ is a penalty function. Commonly used penalties include the L2 ($||\boldsymbol{\beta}_i||_2^2 = \sum_{j=1}^{n}\beta_{ij}^2$) and L1 ($||\boldsymbol{\beta}_i||_1 = \sum_{j=1}^{n}|\beta_{ij}|$) norms (Fu, 1998) either alone or in a combination of both. We considered a penalty function as

$$J(\boldsymbol{\beta}) = \frac{1}{2}(1-\alpha)\sum_{j=1}^{n}\beta_j^2 + \alpha\sum_{j=1}^{n}|\beta_j|$$

where α is a weighting factor. This penalty is used in an **elastic-net-type** regression (Zou and Hastie, 2005) that combines the shrinkage-inducing feature of a **ridge-regression** (Hoerl and Kennard, 1970), that uses the L2-norm alone, and the variable selection feature of a **LASSO** regression (Tibshirani, 1996), that uses the L1-norm alone. With no penalization ($\lambda = 0$), the solution for (6) is equivalent to that of the standard SI in (4).

The ridge-regression (**L2-PSI**) and LASSO (**L1-PSI**) types are special cases of the elastic-net-type PSI (**EN-PSI**) when $\alpha = 0$ and $\alpha = 1$, respectively. A closed-form solution for the regression coefficients can be found only for the L2-PSI (Hastie et al., 2009). In this case, the solution is

$$\hat{\boldsymbol{\beta}}^{L2} = \left(\mathbf{P}_x + \lambda\mathbf{I}\right)^{-1}\mathbf{G}_{x,y}$$

where $\mathbf{I}$ is a $p \times p$ identity matrix. For the EN-PSI ($0 < \alpha \leq 1$) solutions are obtained using iterative algorithms such as **least angle regression** (LARS, Efron et al., 2004) or **coordinate descent** (Friedman et al., 2007) for different values of the parameters α and λ. These combinations of the values of α and λ will result in different EN-PSI from which an optimal index can be obtained such as the prediction accuracy is maximum.

3 Phenotypic and genetic parameters

The population phenotypic (co)variance matrix $\mathbf{P}_x$ among the measured phenotypes is estimated using the unbiased sample (co)variance matrix given by

$$\hat{\mathbf{P}}_x = \frac{1}{n-1}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)'$$

where $\bar{\boldsymbol{x}}$ is the vector containing the sample mean of of the measured phenotypes. For centered data, this reduces to $\hat{\mathbf{P}}_x = \frac{1}{n-1}\mathbf{X}'\mathbf{X}$, where $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]'$ is the matrix containing all measured traits.

The genetic covariance ($\mathbf{G}_{y,x_j}$) between the target trait and the j^{th} measured trait ($j = 1, ..., p$) is estimated using a sequence of uni-variate genetic models as in equation (1). First, the model is fitted for the target trait as response, then for each of the measured traits, and then for the sum of the target trait and each of the measured traits. The genetic covariances between the target trait and the measured traits are then estimated using

$$\hat{\mathbf{G}}_{y,x_j} = \frac{1}{2}\left(\hat{\sigma}^2_{u_{y+xj}} - \hat{\sigma}^2_{u_y} - \hat{\sigma}^2_{u_{xj}}\right) \tag{7}$$

where $\hat{\sigma}^2_{u_y}$, $\hat{\sigma}^2_{u_{xj}}$ and $\hat{\sigma}^2_{u_{y+xj}}$ are the estimated genetic variances for the target trait, the j^{th} measured trait, and the sum of the target trait and the j^{th} measured trait, respectively.

4 Accuracy of indirect selection

The accuracy of the index to be used for indirect selection, $\text{Acc}(\mathcal{I})$, is defined as the correlation between the index ($\mathcal{I}_i$) and the breeding values (u_i). This parameter is equal to the product of the square root of the heritability of the SI, $h_{\mathcal{I}}$, times the genetic correlation between the SI and the selection target, $\text{cor}(u_{\mathcal{I}_i}, u_{y_i})$, this is

$$\text{Acc}(\mathcal{I}) = \text{cor}(\mathcal{I}_i, u_i) = \text{cor}(u_{\mathcal{I}_i}, u_{y_i})h_{\mathcal{I}}$$

The efficiency of indirect selection relative to mass phenotypic selection (RE) is calculated as the ratio between the accuracy of indirect selection and the accuracy of direct selection, h_y, this is

$$RE = \frac{h_{\mathcal{I}}}{h_y}\text{cor}(u_{\mathcal{I}_i}, u_{y_i})$$

where h_y is the square root of the heritability of the selection goal (y), calculated as

$$h^2_y = \frac{\sigma^2_{u_y}}{\sigma^2_{u_y} + \sigma^2_{e_y}}$$

5 Cross validation

To avoid bias in the estimation, the accuracy of indirect selection must be estimated using data that was not used to derive the coefficients of the index. To this end, a **training-testing partition** procedure is implemented as follows: (i) data is partitioned into training (TRN) and testing (TST) sets, (ii) the coefficients of the SI, $\boldsymbol{\beta}$, are derived in the training set, (iii) these coefficients are applied to data of the testing set to derive an index (using (2)), and (iv) parameters $\text{cor}(u_{\mathcal{I}_i}, u_{y_i})$, $h_{\mathcal{I}}$, and $\text{Acc}(\mathcal{I})$ are estimated in the testing set.

6 Experimental data

The data set consists of 1,092 inbred wheat lines grouped into 39 trials and grown during the 2013- 2014 season at the Norman Borlaug experimental research station in Ciudad Obregon, Sonora, Mexico. Each trial consisted of 28 breeding lines that were arranged in an alpha-lattice design with three replicates and six sub-blocks. The trials were grown in four different environments:

Env	Name	Planting conditions Date	Planting conditions System	Number of irrigations	N genotypes (records)	Average (SD) YLD
E1	Flat-Drought	Optimum	Flat	Minimal drip	1,092 (3,274)	2.06 (0.58)
E2	Bed-2IR	Optimum	Bed	2	1,090 (3,267)	3.67 (0.43)
E3	Bed-5IR	Optimum	Bed	5	1,092 (3,274)	6.11 (0.61)
E4	Bed-EHeat	Early	Bed	5	1,013 (2,858)	6.43 (0.73)

Records of grain yield (YLD, $ton\ ha^{-1}$) were collected as the total plot yield after maturity. Measurements for days to heading (DTH), days to maturity (DTM), and plant height (PH, cm) were recorded only in the first replicate at each trial. Reflectance phenotypic data were collected from the fields using both infrared (A600 series Infrared camera, FLIR, Wilsonville, OR) and hyper-spectral (A-series, Micro-Hyperspec, VNIR Headwall Photonics, Fitchburg, MA) cameras mounted on a Piper PA-16 Clipper aircraft on 9 different dates (time-points) between January 10^{th} and March 27^{th}, 2014. During each flight, data from 250 wavelengths ranging from 392 to 850 nm were collected for each pixel in the pictures. The average reflectance of all the pixels for each wavelength was calculated from each of the geo-referenced trial plots and reported as each line reflectance.

Grain yield and image data were pre-adjusted using mixed-effects model that accounted for genotype, trial, replicate, and sub-block. No pre-adjusting was made for DTH, DTM, and PH traits. Pre-adjusted phenotypes were obtained by subtracting from the phenotypic record the mean plus BLUPs of trial, replicate, and sub-block effects.

7 Implementation

The above described data will be used to generate selection indices for grain yield (as selection target) using all the wavelengths (as measured phenotypes). Analyses will be implemented in `R` software (R Core Team, 2019) using the `SFSI` R-package. The package can be installed from the `GitHub` (development version) repository at https://github.com/MarcooLopez/SFSI.

7.1 Data downloading

Both phenotypic and reflectance data are available through the `SFSI` R-package in its `GitHub` version. Each file corresponds to one environment and contains objects `Y`, `WL`, and `VI`. Matrix `Y` contains pre-adjusted observations for YLD, DTH, DTM, and PH. Object `WL` is a list type containing the hyper-spectral image data for all 9 time-points. Element `WL[[j]]`, $j = 1, ..., 9$, is a matrix with 250 columns (wavelengths) and rows matched with those of matrix `Y`. Likewise, the object `VI` is a list with 9 elements being two-columns matrices containing green and red NDVI. **Box 1** shows how to prepare data for a single environment which will be used in later analyses.

Box 1. Data downloading

```
rm(list = ls())
setwd("/Users/epidemiology/Dropbox/MSU/PBGB/projects/PSI/pipeline")
site <-  "https://github.com/MarcooLopez/SFSI/blob/master/data"
filename <- "wheat.E1.RData"

# Download file
dir.create("data", recursive=TRUE)
download.file(paste0(site,"/",filename,"?raw=true"),paste0("data/",filename),mode="wb")
```

7.2 Heritability and variance components

Heritability can be calculated from variance components obtained by fitting the model (1) for the whole data. Code below illustrates how to fit this model using the `solveMixed` function from the `SFSI` R-package.

Box 2. Variance components calculation

```
library(SFSI)

# Load data
load(paste0("data/",filename))
Y$gid <- factor(as.character(Y$gid))
y <- as.vector(scale(Y[,"YLD"]))

# Fit model
Z <- model.matrix(~0+gid,data=Y)
evdZZt <- eigen(tcrossprod(Z))   # Decomposition of ZZ'
fm0 <- solveMixed(y,U=evdZZt$vectors,d=evdZZt$values,BLUP=FALSE)
c(fm0$varU,fm0$varE,fm0$h2)

dir.create("output", recursive=TRUE)
save(fm0,evdZZt,file="output/varComps.RData")
```

7.3 Training-testing partitions

Code in **Box 3** below illustrates how to split data into TRN and TST sets. In this example, $\frac{2}{3}$ of the 29 trials (26 trials, $n_{TRN} \approx 2,184$ observations) will be randomly assigned to the training set and the remaining $\frac{1}{3}$ (13 trials, $n_{TST} \approx 1,092$) to the testing set.

The output will be a 'list' type of length `nPart` containing indices indicating which observations are assigned to testing sets. The object will be saved in the file `partitions.RData` and will be used later for single time-point and across time-points analyses. For demonstration purposes, only `nPart=5` will be run throughout all the analyses.

Box 3. Create testing set partitions

```
trials <- as.numeric(as.character(Y$trial))
nTrial <- length(unique(trials))
nTST <- ceiling((1/3)*nTrial)  # Number of trials in TST set

nPart <- 5     # Number of partitions
seeds <- round(seq(1E3, .Machine$integer.max, length = nPart))

partitions <- vector("list",length=nPart)   # Object to store partitions

for(k in 1:length(partitions))
{   set.seed(seeds[k])
    tst <- sample(1:nTrial,nTST,replace=FALSE)
    partitions[[k]] <- which(trials %in% tst)
}
save(partitions,file="output/partitions.RData")
```

7.4 Genetic covariances estimation

The genetic covariances between the grain yield and each of the wavelengths will be calculated using a sequence of models as described in (7) implemented in the `getGenCov` function from the

SFSI package. Code in **Box 4** below shows how to calculate genetic and phenotypic covariances within time-point. The time-point can be specified through variable `tp` thus it can be changed to calculate covariances for all the 9 time-points. Calculations are performed using data from training set only for each partition. Results are stored in matrices `gencov` and `phencov`, and saved in the file `covariances.RData`.

Box 4. Genetic covariances within time-point

```
tp <- 9      # Time-point
load("output/partitions.RData")

gencov <- phencov <- c()   # Matrices to store covariances

for(k in 1:length(partitions))
{   cat("  partition = ",k,"of",length(partitions),"\n")
    indexTST <- partitions[[k]]
    indexTRN <- (1:nrow(Y))[-indexTST]

    # Training set
    xTRN <- scale(WL[[tp]][indexTRN,])
    yTRN <- as.vector(scale(Y[indexTRN,"YLD"]))

    genotype <- factor(as.character(Y[indexTRN,"gid"]))
    Z <- model.matrix(~ 0 + genotype)   # Design matrix for genotypes
    ZZt <- tcrossprod(Z)

    # Get genetic variances and covariances
    fm <- getGenCov(y1=yTRN,y2=xTRN,K=ZZt,mc.cores=5,scale=FALSE)

    gencov <- cbind(gencov,fm$covU)
    phencov <- cbind(phencov,fm$covU + fm$covE)
}
dir.create(paste0("output/timepoint_",tp), recursive=TRUE)
save(gencov,phencov,file=paste0("output/timepoint_",tp,"/covariances.RData"))
```

7.5 Single time-point selection indices

Standard, L1-penalized and PC-based SIs will be fitted using data from a single time-point. The regression coefficients are to be obtained from training data within each partition previously created.

7.5.1 Regression coefficients

Code in **Box 5a** can be used to estimate regression coefficients using training data within each partition following equations (4) (for the standard SI), and (5) (with $1, 2, ..., 250$ PCs for the PC-SI). Coefficients for L1-PSI are estimated as in expression (6) whose solutions are found using the `lars2` function from the `SFSI` R-package. This function implements the LARS algorithm that gives solutions for an entire path of values of parameter λ.

Time-point can be specified through variable `tp` thus it can be changed to get results for all time-points $(1, 2, ..., 9)$. Results are stored in matrices `betaSI`, `betaPCSI` and `betaL1PSI`, and saved in the file `coefficients.RData`. Calculations are performed across all partitions previously created.

Box 5a. Regression coefficients within time-point

```
tp <- 9      # Time-point
load(paste0("output/timepoint_",tp,"/covariances.RData"))
load("output/partitions.RData")

# Objects to store regression coefficients
betaSI <- betaPCSI <- betaL1PSI <- vector("list",length(partitions))

for(k in 1:length(partitions))
{   cat("  partition = ",k,"of",length(partitions),"\n")
    indexTST <- partitions[[k]]
    indexTRN <- (1:nrow(Y))[-indexTST]

    # Training set
    xTRN <- scale(WL[[tp]][indexTRN,])
    VARx <- var(xTRN)
    EVDx <- eigen(VARx)

    # -- Standard SI --
    VARinv <- EVDx$vectors %*% diag(1/EVDx$values) %*% t(EVDx$vectors)
    betaSI[[k]] <- VARinv %*%  gencov[,k]

    # -- PC-SI --
    VARinv <- diag(1/EVDx$values)
    gamma <- as.vector(VARinv %*% t(EVDx$vectors) %*%  gencov[,k])
    beta <- apply(EVDx$vectors %*% diag(gamma),1,cumsum)
    betaPCSI[[k]] <- data.frame(I(beta),df=1:nrow(beta),lambda=NA)

    # -- L1-PSI --
    fm <- lars2(VARx, gencov[,k])
    beta <- as.matrix(fm$beta)[-1,]
    betaL1PSI[[k]] <- data.frame(I(beta),df=fm$df[-1],lambda=fm$lambda[-1])
}
save(betaSI,betaPCSI,betaL1PSI,file=paste0("output/timepoint_",tp,"/coefficients.RData"))
```

7.5.2 Accuracy of the index

The regression coefficients above calculated can be applied to data from testing set to derive selection indices (as in equation (2)). Code in **Box 5b** can be used to (i) compute the SIs for the testing data, and then (ii) calculate the accuracy of the index, the heritability of the index, and genetic correlation of the index with grain yield. The genetic covariances are obtained using again the approach in (7) implemented in the `getGenCov` function.

Results for the three indices (standard SI, PC-SI and L1-PSI) are stored in the object matrix `accSI` and saved in the file `accuracies.RData`. Calculations are performed across all partitions previously created.

Box 5b. Accuracy of selection within time-point

```
tp <- 9      # Time-point
load(paste0("output/timepoint_",tp,"/coefficients.RData"))
load("output/partitions.RData")

accSI <- c()    # Object to store accuracy components

for(k in 1:length(partitions))
{   cat("  partition = ",k,"of",length(partitions),"\n")
    indexTST <- partitions[[k]]

    # Testing set
    xTST <- scale(WL[[tp]][indexTST,])
    yTST <- as.vector(scale(Y[indexTST,"YLD"]))
    genotype <- factor(as.character(Y[indexTST,"gid"]))
    Z <- model.matrix(~ 0 + genotype)    # Design matrix for genotypes

    # Calculate the indices
    SI <- xTST %*% betaSI[[k]]
    PCSI <- tcrossprod(xTST, betaPCSI[[k]]$beta)
    L1PSI <- tcrossprod(xTST, betaL1PSI[[k]]$beta)

    # Fit genetic models
    ZZt <- tcrossprod(Z)
    fitSI <- as.matrix(data.frame(I(SI),I(PCSI),I(L1PSI)))
    fm <- getGenCov(y1=yTST,y2=fitSI,K=ZZt)

    # Retrieve accuracy components
    h <- sqrt(fm$varU2/(fm$varU2 + fm$varE2))
    gencor <- fm$covU/sqrt(fm$varU1*fm$varU2)
    accuracy <- gencor*h

    df <- c(max(betaL1PSI[[k]]$df),betaPCSI[[k]]$df,betaL1PSI[[k]]$df)
    lambda <- c(min(betaL1PSI[[k]]$lambda),betaPCSI[[k]]$lambda,betaL1PSI[[k]]$lambda)
    accSI <- rbind(accSI,data.frame(rep=k,SI=colnames(fitSI),h,gencor,accuracy,df,lambda))
}
save(accSI,file=paste0("output/timepoint_",tp,"/accuracies.RData"))
```

7.5.3 Effect of regularization on the accuracy of the index

The regularized selection indices were calculated for $q = 1, 2, ..., 250$ PCs for the PC-SI and for decreasing values, $\lambda_1, ..., \lambda_{250}$, of the path of the penalization for the L1-PSI. In the later case, decreasing λ will give increasing number of active number of predictors in the index. Code in **Box 6** below can be used to create a plot depicting the evolution of accuracy (and its components) over values of regularization (number of PCs or number of active predictors). Results for the standard SI corresponds to the case of a PC-SI with 250 PCs and a L1-PSI with 250 active predictors. These cases are shown at the right-most results in the plots.

Box 6. Effect of regularization

```
library(ggplot2); library(reshape); library(ggpubr)

tp <- 9      # Time-point
load(paste0("output/timepoint_",tp,"/accuracies.RData"))

accSI <- split(accSI,as.character(accSI$SI))
accSI <- data.frame(do.call(rbind,lapply(accSI,function(x)apply(x[,-c(1,2)],2,mean))))
accSI$SI <- unlist(lapply(strsplit(rownames(accSI),"\\."),function(x)x[1]))

# Plot of PC-SI
dat <- melt(accSI[accSI$SI=="PCSI",],id=c("df","lambda","SI"))
plot1 <- ggplot(dat[dat$df>1,],aes(df,value,group=variable,color=variable)) + theme_bw() +
  geom_line(size=1) + labs(title="PC-SI",y="correlation",x="Number of PCs")

# Plot of L1-PSI
dat <- melt(accSI[accSI$SI=="L1PSI",],id=c("df","lambda","SI"))
plot2 <- ggplot(dat[dat$df>1,],aes(df,value,group=variable,color=variable)) + theme_bw() +
  geom_line(size=1) + labs(title="L1-PSI",y="correlation",x="Number of active predictors")

ggarrange(plot1,plot2,nrow=1,common.legend=TRUE,legend="right")
```

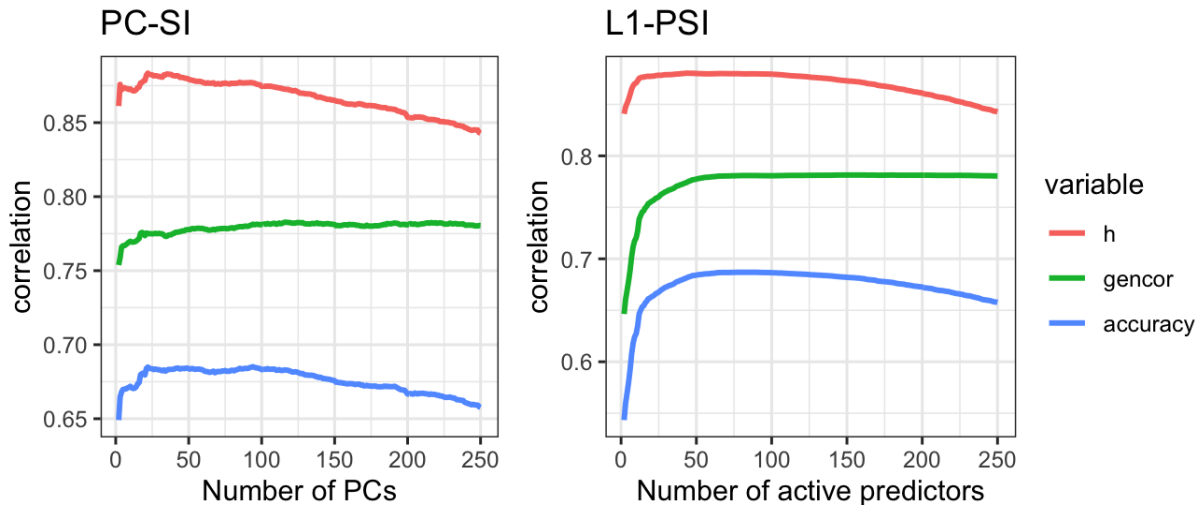

Figure 1: Accuracy of selection (average across 5 TRN-TST partitions) and its components of the index versus the number of principal components entering in the PC-SI (left) and the number of active predictors in the L1-PSI (right). Time-point 9. Environment E1

7.5.4 Optimal regularized vs standard selection indices

An optimal regularized selection index can be chosen by selecting an optimal regularization (number of PCs or value of λ) such that the accuracy of selection of the resulting index in the testing set is maximum. Code in **Box 7** below can be used to select the optimal PC-SI and L1-PSI, and to compare the accuracy of selection with that of the standard SI (non-regularized). Results are reported as an average across all partitions along with a 95% confidence interval.

Box 7. Optimal selection index

```
tp <- 9      # Time-point
load(paste0("output/timepoint_",tp,"/accuracies.RData"))

accSI$SI <- unlist(lapply(strsplit(as.character(accSI$SI),"\\."),function(x)x[1]))
accSI <- split(accSI,paste(accSI$rep,"_",accSI$SI))
accSI <- do.call(rbind,lapply(accSI,function(x)x[which.max(x$accuracy),]))

dat = aggregate(accuracy ~ SI, mean, data=accSI)
dat$sd = aggregate(accuracy ~ SI,sd,data=accSI)$accuracy
dat$n = aggregate(accuracy ~ SI,length,data=accSI)$accuracy
dat$se = qnorm(0.975)*dat$sd/sqrt(dat$n)

ggplot(dat,aes(SI,accuracy,ymin=accuracy-se,ymax=accuracy+se)) + theme_bw() +
   geom_bar(stat="identity",width=0.5,fill="orange") + geom_errorbar(width=0.2) +
   geom_text(aes(label=sprintf("%.3f",accuracy)),y=min(dat$accuracy)*0.75)+labs(x="")
```

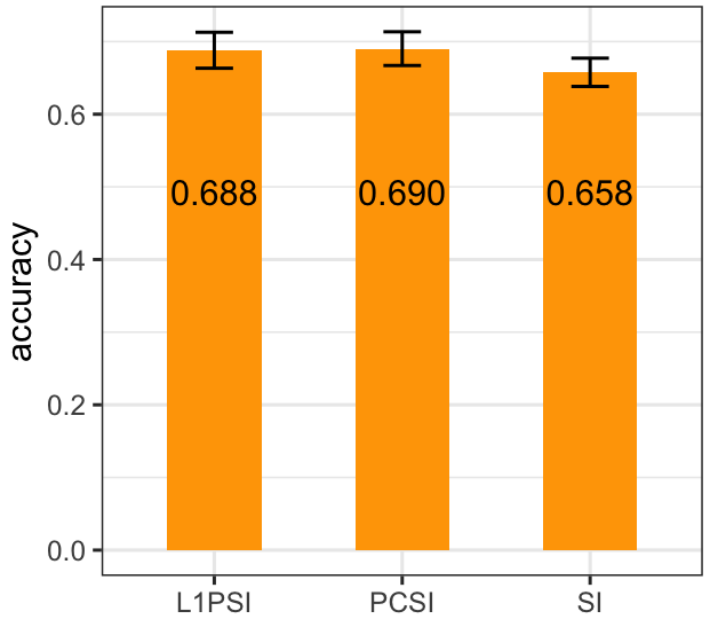

Figure 2: Accuracy of selection (average across 5 TRN-TST partitions) of the standard SI, and the optimal PC-SI and L1-PSI. Time-point 9. Environment E1

7.5.5 Comparison of the indices across time-points

The accuracy of indirect selection for the optimal (i.e., the one with the highest accuracy of indirect selection) L1-PSI and PC-SI can be assessed across time-points and compared with a standard SI. Code in **Box 8** below can be used to calculate the optimal PC-SI and L1-PSI, and to compare their accuracy of selection with that of the standard SI across all time-points.

To this end, all the within time-point analyses in **Box 4b**, **Box 5a** and **Box 5b** must be previously performed for all the 9 time-points. This can be achieved using, for instance, a loop as: `for(tp in 1:9){ }`.

Box 8. Accuracy across time-points

```
AccSI <- c()
for(tp in 1:9)
{ load(paste0("output/timepoint_",tp,"/accuracies.RData"))
  AccSI <- rbind(AccSI,data.frame(Timepoint=tp,accSI))
}

AccSI$SI <- unlist(lapply(strsplit(as.character(AccSI$SI),"\\."),function(x)x[1]))
AccSI <- split(AccSI,paste(AccSI$Timepoint,"_",AccSI$rep,"_",AccSI$SI))
AccSI <- do.call(rbind,lapply(AccSI,function(x)x[which.max(x$accuracy),]))
AccSI$Timepoint <- factor(as.character(AccSI$Timepoint))

dat = aggregate(accuracy ~ SI + Timepoint,mean,data=AccSI)
dat$sd = aggregate(accuracy ~ SI + Timepoint,sd,data=AccSI)$accuracy
dat$n = aggregate(accuracy ~ SI + Timepoint,length,data=AccSI)$accuracy
dat$se = qnorm(0.975)*dat$sd/sqrt(dat$n)

ggplot(dat,aes(Timepoint,accuracy,color=SI,group=SI,ymin=accuracy-se,ymax=accuracy+se))+
   geom_line() + geom_point() + theme_bw() + geom_errorbar(width=0.2)
```

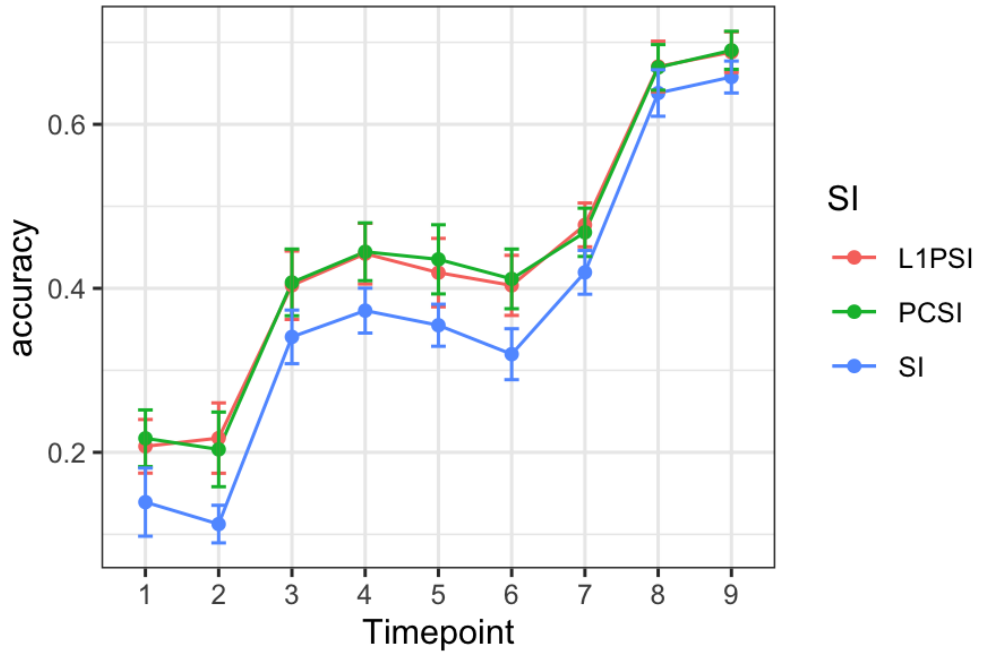

Figure 3: Accuracy of selection (average across 5 TRN-TST partitions) of the standard SI, and the optimal PC-SI and L1-PSI. Time-point 9. Environment E1

7.6 Multi time-point selection indices

Selection indices can be calculated to include reflectance data from multiple time-points. In this case, the vector of measured phenotypes $\boldsymbol{x}_i$ will contain $2,250$ traits corresponding to 250 wavebands measured at each of 9 time-points.

7.6.1 Regression coefficients

Code in **Box 9a** shows how to calculate coefficients for the PC-SI and L1-PSI using data from all 9 time-points. To this end, the genetic covariances of all $2,250$ traits with grain yield are needed, thus, all the within time-point genetic covariances must be previously calculated for all

the 9 time-points using code in **Box 4b**. Results are saved in file `coefficients.RData` in folder `output/multi_timepoint`.

PC-SIs will be calculated including $1, 2, ..., 500$ PCs only (the maximum possible is $2,250$ PCs). The `solveEN` function (included in the `SFSI` R-package) will be used for convenience to calculate the regression coefficients for the L1-PSI. `solveEN` function will yield solutions for a given grid of values of λ (covering all the space of possible values) rather than all the $2,250$ solutions for the entire λ path provided by `lars2` function.

Box 9a. Regression coefficients multi time-points

```
Gencov <- c()
for(tp in 1:9)
{ load(paste0("output/timepoint_",tp,"/covariances.RData"))
  Gencov <- rbind(Gencov,gencov)
}
load("output/partitions.RData")

# Objects to store regression coefficients
betaPCSI <- betaL1PSI <- vector("list",length(partitions))

for(k in 1:length(partitions))
{   cat("  partition = ",k,"of",length(partitions),"\n")
    indexTST <- partitions[[k]]
    indexTRN <- (1:nrow(Y))[-indexTST]

    # Training set
    xTRN <- scale(do.call(cbind,WL)[indexTRN,])
    VARx <- var(xTRN)
    EVDx <- eigen(VARx)

    # -- PC-SI --
    gamma <- t(sweep(EVDx$vectors,2,1/EVDx$values,FUN="*")) %*%  Gencov[,k]
    beta <- apply(sweep(EVDx$vectors,2,as.vector(gamma),FUN="*"),1,cumsum)[1:500,]
    betaPCSI[[k]] <- data.frame(I(beta),df=1:nrow(beta),lambda=NA)

    # -- L1-PSI --
    fm <- solveEN(VARx, Gencov[,k],nLambda=100,maxIter=200,tol=5E-4)
    beta <- as.matrix(fm$beta)[-1,]
    betaL1PSI[[k]] <- data.frame(I(beta),df=fm$df[-1],lambda=fm$lambda[-1])
}
dir.create("output/multi_timepoint", recursive=TRUE)
save(betaPCSI,betaL1PSI,file="output/multi_timepoint/coefficients.RData")
```

7.6.2 Accuracy of the index

As for the single time-point, the regression coefficients are applied to data from testing set to derive selection indices. Code in **Box 9b** can be used to (i) compute the multi time-point PC-SI and L1-PSI for the testing data, and then (ii) calculate the accuracy of the index, the heritability of the index, and genetic correlation of the index with grain yield. The genetic covariances are obtained using the `getGenCov` function.

Results for the two indices (PC-SI and L1-PSI) are stored in the object matrix `AccSI` and saved in the file `accuracies.RData` in folder `output/multi_timepoint`. Calculations are performed across all partitions previously created.

Box 9b. Accuracy of selection multi time-points

```
load("output/partitions.RData")
load("output/multi_timepoint/coefficients.RData")

AccSI <- c()     # Objects to store accuracy components

for(k in 1:length(partitions))
{   cat("  partition = ",k,"of",length(partitions),"\n")
    indexTST <- partitions[[k]]

    # Testing set
    xTST <- scale(do.call(cbind,WL)[indexTST,])
    yTST <- as.vector(scale(Y[indexTST,"YLD"]))
    genotype <- factor(as.character(Y[indexTST,"gid"]))
    Z <- model.matrix(~ 0 + genotype)

    # Calculate the indices
    PCSI_multi <- tcrossprod(xTST, betaPCSI[[k]]$beta)
    L1PSI_multi <- tcrossprod(xTST, betaL1PSI[[k]]$beta)

    # Fit genetic models
    ZZt <- tcrossprod(Z)
    fitSI <- as.matrix(data.frame(I(PCSI_multi),I(L1PSI_multi)))
    fm <- getGenCov(y1=yTST,y2=fitSI,K=ZZt)

    # Retrieve accuracy components
    h <- sqrt(fm$varU2/(fm$varU2 + fm$varE2))
    gencor <- fm$covU/sqrt(fm$varU1*fm$varU2)
    accuracy <- gencor*h

    df <- c(betaPCSI[[k]]$df,betaL1PSI[[k]]$df)
    lambda <- c(betaPCSI[[k]]$lambda,betaL1PSI[[k]]$lambda)
    AccSI <- rbind(AccSI,data.frame(rep=k,SI=colnames(fitSI),h,gencor,accuracy,df,lambda))
}
save(AccSI,file="output/multi_timepoint/accuracies.RData")
```

7.6.3 Optimal multi vs single time-point selection indices

As for the single time-point case, one can obtain an optimal regularized selection index such that the accuracy of selection in the testing set is maximum. Code in **Box 10** below can be used to select the optimal PC-SI and L1-PSI, and to compare the accuracy of selection with that of the single time-point standard SI, PC-SI and L1-PSI. Results are reported as an average across all partitions along with a 95% confidence interval.

Box 10. Optimal selection index

```
tp <- 9      # Time-point
load(paste0("output/timepoint_",tp,"/accuracies.RData"))
load("output/multi_timepoint/accuracies.RData")

AccSI <- rbind(accSI,AccSI)
AccSI$SI <- unlist(lapply(strsplit(as.character(AccSI$SI),"\\."),function(x)x[1]))
AccSI <- split(AccSI,paste(AccSI$rep,"_",AccSI$SI))
AccSI <- do.call(rbind,lapply(AccSI,function(x)x[which.max(x$accuracy),]))

dat = aggregate(accuracy ~ SI, mean,data=AccSI)
dat$sd = aggregate(accuracy ~ SI, sd, data=AccSI)$accuracy
dat$n = aggregate(accuracy ~ SI, length, data=AccSI)$accuracy
dat$se = qnorm(0.975)*dat$sd/sqrt(dat$n)

ggplot(dat,aes(SI,accuracy,ymin=accuracy-se,ymax=accuracy+se)) + theme_bw() +
   geom_bar(stat="identity",width=0.65,fill="orange") + geom_errorbar(width=0.2) +
   geom_text(aes(label=sprintf("%.3f",accuracy)),y=min(dat$accuracy)*0.75)+labs(x="")
```

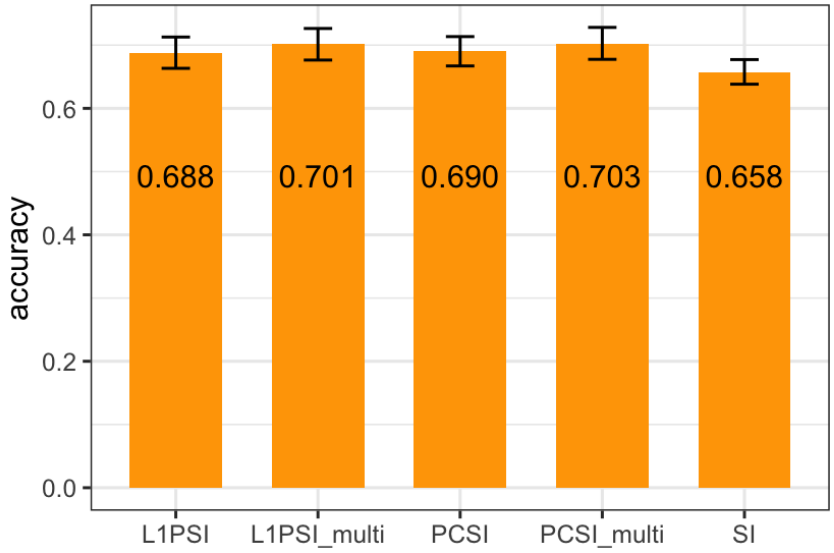

Figure 4: Accuracy of selection (average across 5 TRN-TST partitions) of the standard SI, and the optimal PC-SI and L1-PSI for the time-point 9 and multi time-point. Environment E1

7.7 Sparsity of the index

The value of λ in (6) for the L1-PSI define the number of predictors with non-zero coefficient in the index. Code in **Box 11** below can be used to estimate the regression coefficients for a multi time-point L1-PSI for the whole dataset. To achieve this, the genetic covariances will be calculated for the $2,250$ wavelengths using whole data by means of the `getGenCov` function. The coefficients are then obtained using the `solveEN` with a value of λ obtained from the partitions.

A heatmap of the coefficients will be generated to show the sparsity and regions of the electromagnetic spectrum that were selected by the method.

Box 11. Sparsity of the optimal index

```
load("output/varComps.RData")    # Load the SVD of ZZ' to speed computation
load("output/multi_timepoint/accuracies.RData")

# Get genetic covariances
x <- scale(do.call(cbind,WL))
fm <- getGenCov(y1=Y$YLD,y2=x,U=evdZZt$vectors,d=evdZZt$values, mc.cores=5)
gencov <- fm$covU

# Get a value of lambda across partitions
AccSI$SI <- unlist(lapply(strsplit(as.character(AccSI$SI),"\\."),function(x)x[1]))
AccSI <- split(AccSI,paste0(AccSI$SI,"_",AccSI$rep))
AccSI <- do.call(rbind,lapply(AccSI,function(x)x[which.max(x$accuracy),]))
lambda <- mean(AccSI$lambda[AccSI$SI=="L1PSI_multi"])

# Get regression coefficients
VARx <- var(x)
beta <- solveEN(VARx,gencov,lambda=lambda)$beta

wl <- factor(rep(gsub("wl","",colnames(WL[[1]])),9))
Timepoint <- factor(rep(1:9,each=ncol(WL[[1]])))
dat <- data.frame(beta=as.vector(beta),Timepoint,wl)
dat$beta[abs(dat$beta) < .Machine$double.eps] <- NA

ggplot(dat, aes(x=wl,y=Timepoint,fill = abs(beta))) + geom_tile(height=0.8) +
  theme_bw() + labs(x="Wavelength (nm)") +
  scale_x_discrete(breaks = levels(wl)[seq(1,nlevels(wl),25)]) +
  scale_fill_gradientn(na.value = 'gray35', colours = c(rev(heat.colors(15))))
```

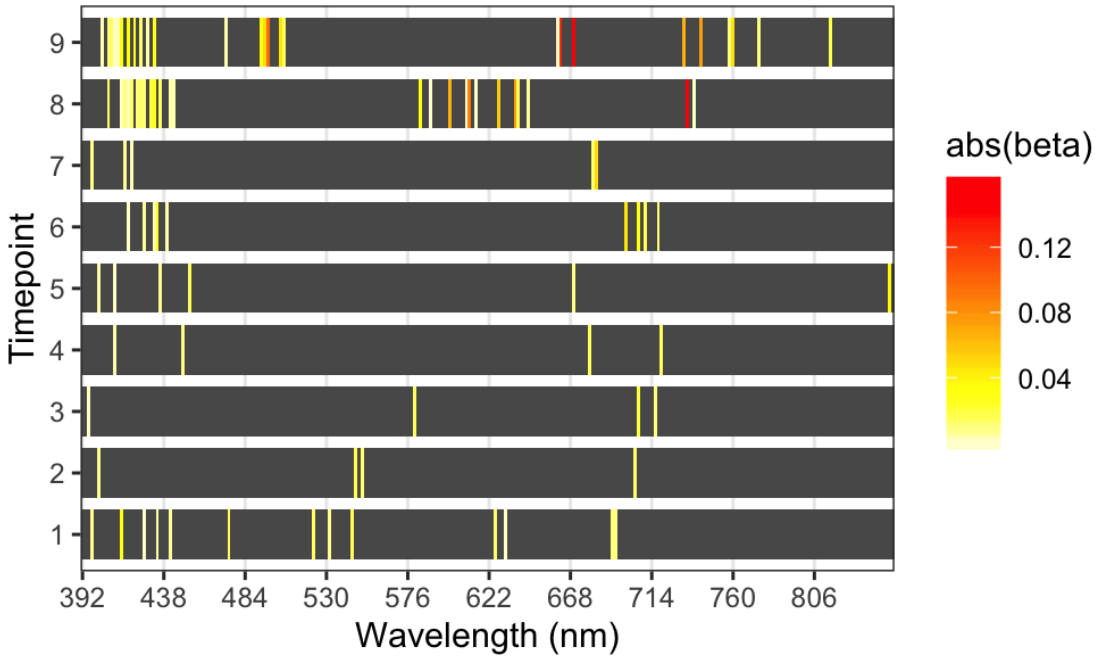

Figure 5: Heatmap of regression coefficients for L1-penalized SI using multi time-point data

References

Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani (2004). Least angle regression. *The Annals of Statistics 32*(2), 407–499.

Friedman, J., T. Hastie, H. Höfling, and R. Tibshirani (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics 1*(2), 302–332.

Fu, W. J. (1998). Penalized regressions: the Bridge versus the LASSO. *Journal of Computational and Graphical Statistics 7*(3), 397–416.

Hastie, T., R. Tibshirani, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction* (2nd ed.). New York, USA: Springer.

Hazel, L. N. (1943). The genetic basis for constructing selection indexes. *Genetics 28*(6), 476–490.

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

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Smith, H. F. (1936). A discrimant function for plant selection. *Annals of Eugenics 7*, 240–250.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B 58*(1), 267–288.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B 67*(2), 301–320.