

Empowering Genomic Prediction: Two Essential Tools for Democratizing the Access to Statistical Machine Learning Methods

Talk given to the Center for Quantitative Genetics and Genomics (CQGG), Aarhus University, Denmark.

Dr. Osval Antonio Montesinos López

oamontes1@ucol.mx

Facultad de Telemática

Universidad de Colima

México

April, 20, 2023.

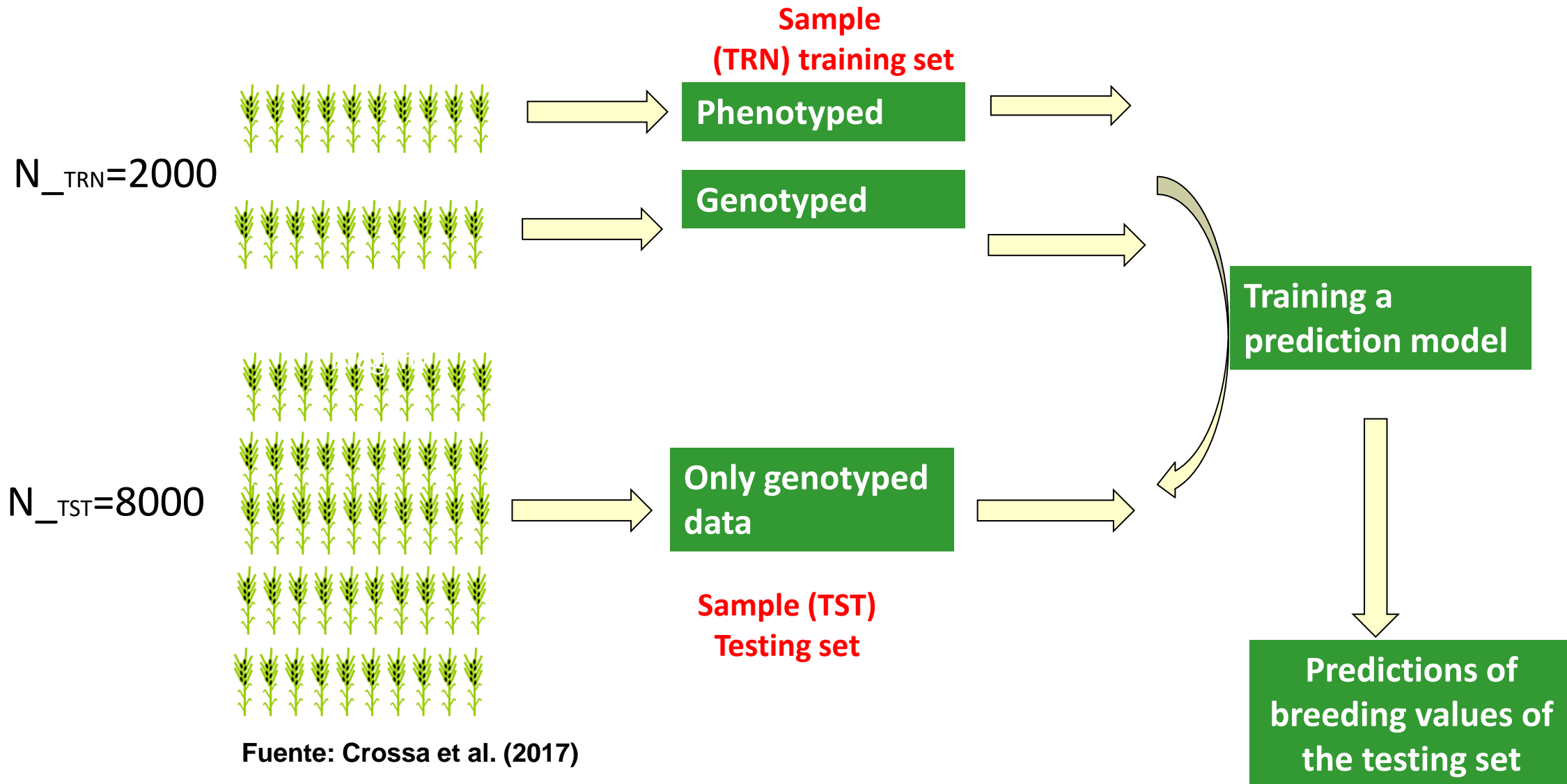
Outline

- Introduction
- SKM library
- Book: Multivariate Statistical Machine Learning
Methods for genomic prediction

Introduction

- Genomic selection (GS) is revolutionizing plant breeding (Meuwissen, et al., 2001). But, because it is a predictive methodology, a basic understanding of statistical machine learning methods is necessary for its successful implementation.
- However, due to lack of time and appropriate training, it is difficult for breeders and scientists of related fields to learn all the fundamentals of prediction algorithms.
- Fortunately, with smart or highly automated software, it is possible for these professionals to appropriately implement any state-of-the-art statistical machine learning method.

Figure 1. Diagram about the logic of genomic selection



SKM library

Sparse kernel methods (SKM) library

- SKM is an R library that allows us to implement seven of the most powerful state-of-the-art algorithms.
 - Bayesian methods
 - Random Forest (RF)
 - Support vector Machine (SVM)
 - Generalized linear models (GLM)
 - Gradient boosting machine (GBM)
 - Partial Least Squares (PLS)
 - Feed-forward artificial neural networks

Sparse kernel methods (SKM) library

- SKM not implement the algorithms itself but internally uses some already popular libraries for this purpose, which are known to have efficient and complete implementations of the algorithms.
- SKM use the same format for implementing 6 out of the 7 state of the art statistical machine learning algorithms.
- For algorithms with hyperparameters, SKM has an easy way to specify the conditions for hyperparameter tuning.

Sparse kernel methods (SKM) library

- Also, has many easy-to-use functions for computing metrics for evaluation of the prediction accuracy.
- Additionally provides many function for implementing many cross-validation strategies and for making summaries of prediction accuracy.

Hyperparameter tuning in statistical machine learning algorithms

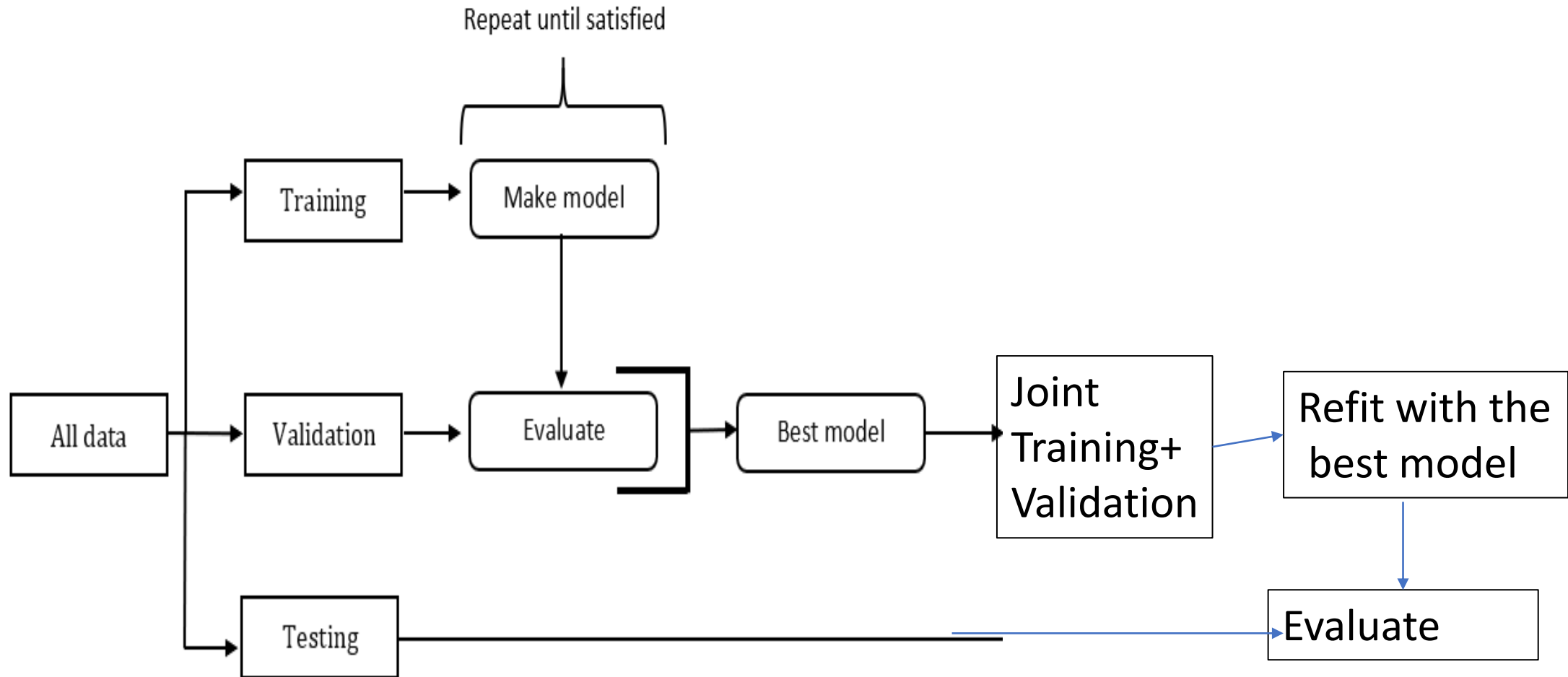


Figure 5.9. Schematic representation of the training, validation and testing sets adapted by Cook (2017).

Hyperparameter tuning in statistical machine learning algorithms

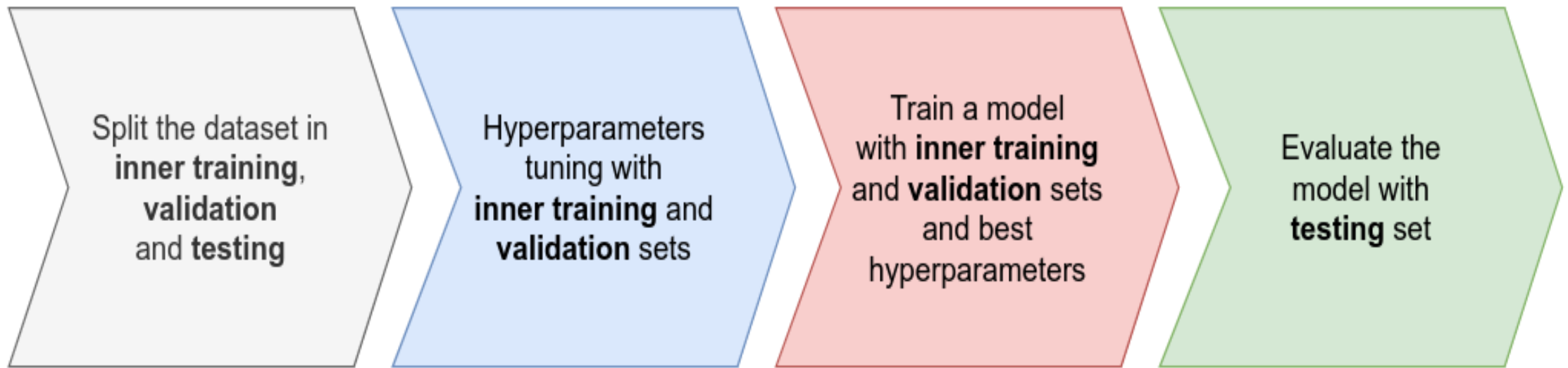


Figure 1: Machine learning algorithm evaluation in one iteration of cross validation.

Installation of the SKM library

Script to install all the required packages for SKM library

```
is_installed <- function(package) {  
  
  return(all(package %in% rownames(installed.packages())))  
  
  }  
  
is_windows_os <- function() {  
  
  return(.Platform$OS.type == "windows")  
  
  }  
  
ok <- TRUE  
  
if (is_windows_os()) {  
  
  if (!is_installed("installr")) {  
  
    install.packages("installr")  
  
  }  
  
  ok <- install.Rtools(check = TRUE, check_r_update = FALSE)  
  
  }
```

Installation of the SKM library

```
if (ok) {  
  if (!is_installed("devtools")) {  
    install.packages("devtools")  
  }  
  
  devtools::install_github("gdlc/BGLR-R", upgrade = "default")  
  devtools::install_github("rstudio/tensorflow", upgrade = "default")  
  devtools::install_github("rstudio/keras", upgrade = "default")  
  devtools::install_github("brandon-mosqueda/SKM", upgrade = "default")  
}
```

Installation of the SKM library

If devtools can not be installed try:

<https://cran.r-project.org/bin/windows/Rtools/rtools42/files/rtools42-5355-5357.exe>

Kernel implementation on the SKM library

- One important aspect of the SKM library is the availability of kernels to capture non-linear patterns in data, giving the library its name.
- In the SKM library, you can find seven kernels that can be implemented: Linear, Polynomial, Sigmoid, Gaussian, Exponential, Arc-Cosine 1 and Arc-Cosine L (with $L = 2, 3, \dots$).

Evaluation of Prediction accuracy

Metrics for regression problems available in SKM

Mean Squared Error (**mse**),

Root Mean Squared Error (**rmse**),

Normalized Root Mean Squared Error (**nrmse**),

Mean Absolute Error (**mae**) and

Mean Arctangent Absolute Percentage Error (**maape**)

numeric_summary() provides all the above numeric metrics

Evaluation of Prediction accuracy

Metrics for classification problems available in SKM

Accuracy (**accuracy**),

precision (**precision**),

recall (**recall**),

sensitivity (**sensitivity**),

specificity (**specificity**),

Brier's score (**brier_score**).

Details of most of these metrics can be found in Montesinos-López et al. (2022a)

Evaluation of Prediction accuracy

Metrics for classification problems available in SKM

F1 score (**f1_score**),

Kappa coefficient (**kappa_coeff**),

Matthews coefficient (**matthews_coeff**),

ROC area under the curve (**roc_auc**) and

precision-recall area under curve (**pr_auc**)

categorical_summary() provides all the above categorical metrics

Evaluation of Prediction accuracy

Specific functions for genomic prediction

gs_summaries is a function specific for the context of genomic prediction

gs_summaries expects a data frame with 5 columns: *Fold*, the fold number; *Line*, the line; *Env*, the environment; *Observed* and *Predicted*.

The type of response variable is automatically inferred so numeric or categorical metrics are computed and reported depending on that.

This function returns three different summaries computed by line, by environment and by fold.

Evaluation of Prediction accuracy

The following block of code shows how we can use the function `gs_summaries` with the predictions in the data frame generated.

```
# Errors metrics for prediction performance  
Summaries <- gs_summaries(Predictions)  
# Print summaries by line, environment and fold  
Summaries$line  
Summaries$env  
Summaries$fold
```

Types of cross-validation

K-fold: `cv_kfold()` is the function in SKM

Stratified k-fold: `cv_kfold_strata()` is the name of the function in SKM

Random partition: `cv_random()` is the function to implement this cross-validation in SKM.

Random stratified partitions: `cv_random_strata()`

Types of cross-validation

Leave-one-group-out: `cv_leave_one_group_out()`

Random line: `cv_random_line()`

Missing: `cv_na()`

Hyperparameters tuning

Five algorithms in SKM are of models with hyperparameters (with the exception of bayesian_model and partial_least_squares that require hyperparameters but they are computed internally efficiently).

- tune_type: It can be “Grid_search” or “Bayesian_optimization”.
- tune_cv_type: Cross validation for tuning, it can be “K_fold” or “Random” that call a random partition.

Hyperparameters tuning

Grid search

```
model <- SKM::random_forest(  
  X_training,  
  y_training,
```

Tunable hyperparameters

```
trees_number = c(100, 200, 500),  
node_size = c(5, 2),  
sampled_x_vars_number = c(0.1, 0.5, 0.8),
```

```
tune_type = "Grid_search",  
tune_cv_type = "k_fold",  
tune_folds_number = 5,  
tune_grid_proportion = 0.5,  
tune_loss_function = "mae"
```

```
)
```


Hyperparameters tuning

Bayesian optimization

```
model <- SKM::random_forest(  
  X_training,  
  y_training,
```

Tunable hyperparameters

```
trees_number = list(min = 100, max = 500),  
node_size = list(min = 2, max = 5),  
sampled_x_vars_number = list(min = 0.2, max = 0.8),
```

```
tune_type = "Bayesian_optimization",  
tune_cv_type = "k_fold",  
tune_folds_number = 5,  
tune_bayes_samples_number = 10,  
tune_bayes_iterations_number = 20,
```

```
tune_loss_function = "mae"
```

```
)
```

Hyperparameters tuning

Show all the evaluated combinations with its loss value

```
model$hyperparams_grid
```

Show the best combination used for fitting the final model

```
model$best_hyperparams
```

Bayesian models

- In SKM we refer to Bayesian models to all those models of the BGLR library (Perez & de los Campos, 2014).
- To implement these models can be used the function `bayesian_model()` of SKM, but, in contrast with the other six algorithms in SKM, the expected format of the predictor consist of a list with the different components of the model.
- Furthermore, instead of partitioning the data in training and testing sets, all the data is provided and the positions used for testing are set to NA or specified in a parameter included for this purpose.

Bayesian models

$$Y_{ij} = \mu + L_i + g_j + gL_{ij} + \epsilon_{ij}$$

where $L = (L_1, \dots, L_I)^T \sim \mathcal{N}_J(\mathbf{0}, \sigma_L^2 \mathbf{H})$, where \mathbf{H} denotes the environmental relationship matrix, that is, $\mathbf{H} = \frac{X_E X_E^T}{r}$, $\mathbf{g} = (g_1, \dots, g_J)^T \sim \mathcal{N}_J(\mathbf{0}, \sigma_g^2 \mathbf{G})$, $\mathbf{gL} = (gL_{11}, \dots, gL_{1J}, \dots, gL_{IJ})^T \sim \mathcal{N}_{JJ}(\mathbf{0}, \sigma_{gL}^2 \mathbf{H} \odot \mathbf{Z}_g \mathbf{G} \mathbf{Z}_g^T)$, where \mathbf{G} is the genomic relationship matrix as computed by (VanRaden, 2008), \odot denotes the Hadamard product and \mathbf{H} is the environmental relationship matrix of size I .

Bayesian models

```
X <- list(  
  list(x = K_e, model = "BGLUP"),  
  list(x = K_g, model = "BGBLUP"),  
  list(x = K_ge, model = "BGBLUP")  
)  
# Sample of 20% of the observations used as testing set  
testing_indices <- sample(nrow(X), nrow(X) * 0.2)
```

```
SKM::bayesian_model(  
  X,  
  y,
```

Bayesian models

```
iterations_number = 100,  
burn_in = 50,  
thinning = 5,
```

```
testing_indices = testing_indices)
```

Bayesian models

Example:

Ex_S0. Univariate_Continuous_GBLUP

All code are available at:

https://github.com/osval78/SKM_Genomic_Selection_Example

Random forest

Random Forest

Bayesian optimization

```
model <- SKM::random_forest(  
  X_training,  
  y_training,
```

Tunable hyperparameters

```
trees_number = list(min = 100, max = 500),  
node_size = list(min = 2, max = 5),  
sampled_x_vars_number = list(min = 0.2, max = 0.8),
```

```
tune_type = "Bayesian_optimization",  
tune_cv_type = "k_fold",  
tune_folds_number = 5,  
tune_bayes_samples_number = 10,  
tune_bayes_iterations_number = 20,
```

```
tune_loss_function = "mae"
```

```
)
```

Random Forest

Example:

Ex_S1.Univariate_Continuous

Support Vector Machine

- The `support_vector_machine()` in the SKM library can be used to implement SVM models.
- For example, a SVM with radial kernel $\left(K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})\right)\right)$ with γ a positive constant can be implemented using the following code:

Support Vector Machine

```
SKM::support_vector_machine(  
  X_training, y_training,  
  
  # Tunable hyperparameters  
  kernel = "radial",  
  gamma = 1/NCOL(x),  
  coef0 = 0,  
  cost = 1)
```

Support Vector Machine (SVM)

Example:

Ex_S2. Univariate_Binary

Generalized linear models (GLM)

Under generalized linear models, we use a set of p predictor variables and a response variable to fit a model of the form:

Distribution: y_i is distributed among those which are considered exponential families of probability distributions (like normal, binomial, Poisson, etc.)

Linear predictor: $\eta_i = \beta_0 + X_{i1}\beta_1 + \cdots + X_{ip}\beta_p$

Link function: $g(\mu_i) = \eta_i$

where y_i is the response variable for the i -th individual (sample), x_i is the predictor variable, with $i=1, 2, \dots, 3$, β_0 denotes and intercept term, β_i is the beta coefficient corresponding to the predictor x_i .

Generalized linear models (GLM)

The loss function that is minimized is

$$\text{minimize}\{-\ell(\boldsymbol{\beta}; \mathbf{y}) + 0.5\lambda \sum_{j=1}^P \{(1 - \alpha)\beta_j^2 + \alpha|\beta_j|\}\}$$

Where λ is the penalization parameter, when $\alpha=0$ a Ridge penalization is implemented, while when $\alpha=1$ a Lasso penalization is implemented and when $0 < \alpha < 1$ the Elastic Net penalization is implemented. Can be implemented prediction models for continuous response variables (identity link function) for binary (logit link function) for categorical (generalized logit link function) and for counts (exponential logit link function) (Friedman, Hastie, & Tibshirani, 2010).

Generalized linear models (GLM)

- Under the SKM library for continuous response variables, it is possible to train multi-trait models.
- The function to implement all these models is the `generalized_linear_model()` function.
- The required parameters for this function are shown in the following block of code:

Generalized linear models (GLM)

```
SKM::generalized_linear_model(  
  X_training, y_training,  
  
  # Tunable hyperparameters  
  alpha = 1, ###Lasso Regression  
  
  # Tune configuration parameters  
  tune_type = "Grid_search",  
  tune_folds_number = 5,  
  tune_grid_proportion = 1,  
  tune_bayes_samples_number = 10,  
  tune_bayes_iterations_number = 10,
```

Generalized linear models (GLM)

Other algorithm's parameters

lambdas_number = 100,

Seed for reproducible results

seed = NULL,

verbose = TRUE

)

Generalized linear models (GLM)

Example:

Ex_S3. Univariate_Categorical

Gradiente boosting machine

The following block of code provides the specification of `generalized_boosted_machine()` function:

```
SKM::generalized_boosted_machine(  
  X_training, y_training,  
  # Tunable hyperparameters  
  trees_number = 500,  
  max_depth = 1,  
  node_size = 10,  
  shrinkage = 0.1,  
  sampled_records_proportion = 0.5)
```

Gradiente boosting machine

Example

Ex_S7.Univariate_Continous_GBM

Partial Least Squares (PLS)

- Partial Least Square (PLS) regression is one of the most popular in biological sciences, because it can model complex biological events, it is flexible for considering different factors, and it is unaffected by data collinearity.
- For this reason, authors suggest that the PLS is a potentially valuable method for modeling high-dimensional biological data (as derived from genomics, proteomics and peptidomics) ([Palermo et al., 2009](#)).
- PLS can model multiple responses, while efficiently dealing with multicollinearity.

Partial Least Squares (PLS)

Example:

Ex_S8.Univariate_Continuous_PLS

Book

State-of-the-art material for genomic
prediction

Osva Antonio Montesinos López
Abelardo Montesinos López
José Crossa

Multivariate Statistical Machine Learning Methods for Genomic Prediction

Foreword by Fred van Eeuwijk

OPEN ACCESS

 Springer

1	General Elements of Genomic Selection and Statistical Learning	1
1.1	Data as a Powerful Weapon	1
1.2	Genomic Selection	3
1.2.1	Concepts of Genomic Selection	4
1.2.2	Why Is Statistical Machine Learning a Key Element of Genomic Selection?	6
1.3	Modeling Basics	8
1.3.1	What Is a Statistical Machine Learning Model?	8
1.3.2	The Two Cultures of Model Building: Prediction Versus Inference	9
1.3.3	Types of Statistical Machine Learning Models and Model Effects	11
1.4	Matrix Algebra Review	17
1.5	Statistical Data Types	25
1.5.1	Data Types	25
1.5.2	Multivariate Data Types	28
1.6	Types of Learning	28
1.6.1	Definition and Examples of Supervised Learning	29
1.6.2	Definitions and Examples of Unsupervised Learning	32
1.6.3	Definition and Examples of Semi-Supervised Learning	33
	References	33
2	Preprocessing Tools for Data Preparation	35
2.1	Fixed or Random Effects	35
2.2	BLUEs and BLUPs	36
2.3	Marker Depuration	43
2.4	Methods to Compute the Genomic Relationship Matrix	49

2.5	Genomic Breeding Values and Their Estimation	52
2.6	Normalization Methods	57
2.7	General Suggestions for Removing or Adding Inputs	58
2.8	Principal Component Analysis as a Compression Method	63
	Appendix 1	68
	Appendix 2	68
	References	69
3	Elements for Building Supervised Statistical Machine Learning Models	71
3.1	Definition of a Linear Multiple Regression Model	71
3.2	Fitting a Linear Multiple Regression Model via the Ordinary Least Square (OLS) Method	71
3.3	Fitting the Linear Multiple Regression Model via the Maximum Likelihood (ML) Method	75
3.4	Fitting the Linear Multiple Regression Model via the Gradient Descent (GD) Method	76
3.5	Advantages and Disadvantages of Standard Linear Regression Models (OLS and MLR)	80
3.6	Regularized Linear Multiple Regression Model	81
	3.6.1 Ridge Regression	81
	3.6.2 Lasso Regression	93
3.7	Logistic Regression	98
	3.7.1 Logistic Ridge Regression	100
	3.7.2 Lasso Logistic Regression	102
	Appendix 1: R Code for Ridge Regression Used in Example 2	104
	References	107
4	Overfitting, Model Tuning, and Evaluation of Prediction Performance	109
4.1	The Problem of Overfitting and Underfitting	109
4.2	The Trade-Off Between Prediction Accuracy and Model Interpretability	111
4.3	Cross-validation	115
	4.3.1 The Single Hold-Out Set Approach	115
	4.3.2 The k -Fold Cross-validation	116
	4.3.3 The Leave-One-Out Cross-validation	117
	4.3.4 The Leave- m -Out Cross-validation	117
	4.3.5 Random Cross-validation	118
	4.3.6 The Leave-One-Group-Out Cross-validation	118
	4.3.7 Bootstrap Cross-validation	119
	4.3.8 Incomplete Block Cross-validation	120
	4.3.9 Random Cross-validation with Blocks	121
	4.3.10 Other Options and General Comments on Cross-validation	122

4.4	Model Tuning	124
4.4.1	Why Is Model Tuning Important?	126
4.4.2	Methods for Hyperparameter Tuning (Grid Search, Random Search, etc.)	127
4.5	Metrics for the Evaluation of Prediction Performance	128
4.5.1	Quantitative Measures of Prediction Performance	129
4.5.2	Binary and Ordinal Measures of Prediction Performance	131
4.5.3	Count Measures of Prediction Performance	137
	References	138
5	Linear Mixed Models	141
5.1	General of Linear Mixed Models	141
5.2	Estimation of the Linear Mixed Model	142
5.2.1	Maximum Likelihood Estimation	142
5.3	Linear Mixed Models in Genomic Prediction	148
5.4	Illustrative Examples of the Univariate LMM	148
5.5	Multi-trait Genomic Linear Mixed-Effects Models	152
5.6	Final Comments	157
	Appendix 1	158
	Appendix 2	158
	Appendix 3	159
	Appendix 4	159
	Appendix 5	160
	Appendix 6	163
	Appendix 7	165
	References	168
6	Bayesian Genomic Linear Regression	171
6.1	Bayes Theorem and Bayesian Linear Regression	171
6.2	Bayesian Genome-Based Ridge Regression	172
6.3	Bayesian GBLUP Genomic Model	176
6.4	Genomic-Enabled Prediction BayesA Model	178
6.5	Genomic-Enabled Prediction BayesB and BayesC Models	180
6.6	Genomic-Enabled Prediction Bayesian Lasso Model	184
6.7	Extended Predictor in Bayesian Genomic Regression Models	186
6.8	Bayesian Genomic Multi-trait Linear Regression Model	188
6.8.1	Genomic Multi-trait Linear Model	190
6.9	Bayesian Genomic Multi-trait and Multi-environment Model (BMTME)	195
	Appendix 1	198
	Appendix 2: Setting Hyperparameters for the Prior Distributions of the BRR Model	199
	Appendix 3: R Code Example 1	200

	Appendix 4: R Code Example 2	202
	Appendix 5	204
	R Code Example 3	204
	R Code for Example 4	206
	References	207
7	Bayesian and Classical Prediction Models for Categorical and Count Data	209
	7.1 Introduction	209
	7.2 Bayesian Ordinal Regression Model	209
	7.2.1 Illustrative Examples	216
	7.3 Ordinal Logistic Regression	221
	7.4 Penalized Multinomial Logistic Regression	225
	7.4.1 Illustrative Examples for Multinomial Penalized Logistic Regression	228
	7.5 Penalized Poisson Regression	232
	7.6 Final Comments	235
	Appendix 1	236
	Appendix 2	238
	Appendix 3	240
	Appendix 4 (Example 4)	242
	Appendix 5	244
	Appendix 6	246
	References	248
8	Reproducing Kernel Hilbert Spaces Regression and Classification Methods	251
	8.1 The Reproducing Kernel Hilbert Spaces (RKHS)	251
	8.2 Generalized Kernel Model	253
	8.2.1 Parameter Estimation Under the Frequentist Paradigm	253
	8.2.2 Kernels	255
	8.2.3 Kernel Trick	257
	8.2.4 Popular Kernel Functions	260
	8.2.5 A Two Separate Step Process for Building Kernels	269
	8.3 Kernel Methods for Gaussian Response Variables	269
	8.4 Kernel Methods for Binary Response Variables	271
	8.5 Kernel Methods for Categorical Response Variables	274
	8.6 The Linear Mixed Model with Kernels	274
	8.7 Hyperparameter Tuning for Building the Kernels	278
	8.8 Bayesian Kernel Methods	280
	8.8.1 Extended Predictor Under the Bayesian Kernel BLUP	283
	8.8.2 Extended Predictor Under the Bayesian Kernel BLUP with a Binary Response Variable	286

8.8.3	Extended Predictor Under the Bayesian Kernel BLUP with a Categorical Response Variable	287
8.9	Multi-trait Bayesian Kernel	288
8.10	Kernel Compression Methods	289
8.10.1	Extended Predictor Under the Approximate Kernel Method	294
8.11	Final Comments	297
	Appendix 1	298
	Appendix 2	301
	Appendix 3	305
	Appendix 4	306
	Appendix 5	308
	Appendix 6	311
	Appendix 7	316
	Appendix 8	320
	Appendix 9	325
	Appendix 10	329
	Appendix 11	331
	References	334
9	Support Vector Machines and Support Vector Regression	337
9.1	Introduction to Support Vector Machine	337
9.2	Hyperplane	338
9.3	Maximum Margin Classifier	340
9.3.1	Derivation of the Maximum Margin Classifier	343
9.3.2	Wolfe Dual	346
9.4	Derivation of the Support Vector Classifier	354
9.5	Support Vector Machine	357
9.5.1	One-Versus-One Classification	361
9.5.2	One-Versus-All Classification	361
9.6	Support Vector Regression	369
	Appendix 1	371
	Appendix 2	373
	Appendix 3	375
	References	377
10	Fundamentals of Artificial Neural Networks and Deep Learning	379
10.1	The Inspiration for the Neural Network Model	379
10.2	The Building Blocks of Artificial Neural Networks	382
10.3	Activation Functions	387
10.3.1	Linear	388
10.3.2	Rectifier Linear Unit (ReLU)	388
10.3.3	Leaky ReLU	389
10.3.4	Sigmoid	390
10.3.5	Softmax	390
10.3.6	Tanh	391

10.4	The Universal Approximation Theorem	392
10.5	Artificial Neural Network Topologies	393
10.6	Successful Applications of ANN and DL	396
10.7	Loss Functions	399
10.7.1	Loss Functions for Continuous Outcomes	400
10.7.2	Loss Functions for Binary and Ordinal Outcomes	401
10.7.3	Regularized Loss Functions	402
10.7.4	Early Stopping Method of Training	405
10.8	The King Algorithm for Training Artificial Neural Networks: Backpropagation	407
10.8.1	Backpropagation Algorithm: Online Version	412
10.8.2	Illustrative Example 10.1: A Hand Computation	413
10.8.3	Illustrative Example 10.2—By Hand Computation	418
	References	424
11	Artificial Neural Networks and Deep Learning for Genomic Prediction of Continuous Outcomes	427
11.1	Hyperparameters to Be Tuned in ANN and DL	427
11.1.1	Network Topology	427
11.1.2	Activation Functions	428
11.1.3	Loss Function	428
11.1.4	Number of Hidden Layers	428
11.1.5	Number of Neurons in Each Layer	430
11.1.6	Regularization Type	431
11.1.7	Learning Rate	432
11.1.8	Number of Epochs and Number of Batches	433
11.1.9	Normalization Scheme for Input Data	434
11.2	Popular DL Frameworks	435
11.3	Optimizers	436
11.4	Illustrative Examples	438
	Appendix 1	459
	Appendix 2	462
	Appendix 3	466
	Appendix 4	467
	Appendix 5	471
	References	476
12	Artificial Neural Networks and Deep Learning for Genomic Prediction of Binary, Ordinal, and Mixed Outcomes	477
12.1	Training DNN with Binary Outcomes	477
12.2	Training DNN with Categorical (Ordinal) Outcomes	482
12.3	Training DNN with Count Outcomes	486
12.4	Training DNN with Multivariate Outcomes	490
12.4.1	DNN with Multivariate Continuous Outcomes	490
12.4.2	DNN with Multivariate Binary Outcomes	493

	12.4.3	DNN with Multivariate Ordinal Outcomes	498
	12.4.4	DNN with Multivariate Count Outcomes	501
	12.4.5	DNN with Multivariate Mixed Outcomes	504
	Appendix 1		507
	Appendix 2		512
	Appendix 3		517
	Appendix 4		521
	Appendix 5		526
	References		531
13	Convolutional Neural Networks		533
	13.1	The Importance of Convolutional Neural Networks	533
	13.2	Tensors	534
	13.3	Convolution	539
	13.4	Pooling	542
	13.5	Convolutional Operation for 1D Tensor for Sequence Data	545
	13.6	Motivation of CNN	546
	13.7	Why Are CNNs Preferred over Feedforward Deep Neural Networks for Processing Images?	548
	13.8	Illustrative Examples	554
	13.9	2D Convolution Example	560
	13.10	Critics of Deep Learning	566
	Appendix 1		568
	Appendix 2		572
	References		576
14	Functional Regression		579
	14.1	Principles of Functional Linear Regression Analyses	579
	14.2	Basis Functions	584
		14.2.1 Fourier Basis	584
		14.2.2 B-Spline Basis	585
	14.3	Illustrative Examples	589
	14.4	Functional Regression with a Smoothed Coefficient Function	598
	14.5	Bayesian Estimation of the Functional Regression	604
	Appendix 1		611
	Appendix 2 (Example 14.4)		617
	Appendix 3 (Example 14.5)		621
	Appendix 4 (Example 14.6)		626
	References		631
15	Random Forest for Genomic Prediction		633
	15.1	Motivation of Random Forest	633
	15.2	Decision Trees	634
	15.3	Random Forest	637

15.4	RF Algorithm for Continuous, Binary, and Categorical Response Variables	639
15.4.1	Splitting Rules	641
15.5	RF Algorithm for Count Response Variables	650
15.6	RF Algorithm for Multivariate Response Variables	655
15.7	Final Comments	660
	Appendix 1	662
	Appendix 2	664
	Appendix 3	665
	Appendix 4	669
	Appendix 5	672
	Appendix 6	676
	References	680
	Index	683

Open Source:

<https://link.springer.com/book/10.1007/978-3-030-89010-0#toc>

Conclusions

Due to the fact that the GS is a predictive methodology still is challenging its implementation in many breeding programs since many factors affect its accuracy.

For this reason, a lot of research is being conducted to improve some of the factors that affects its accuracy.

However, as pointed out before statistical machine learning methods are key for the successful implementation of the GS methodology.

Conclusions

The SKM library provides an opportunity to utilize various statistical machine learning techniques for Genomic prediction without the need for extensive expertise in statistics and programming.

Additionally, SKM offers the advantage of incorporating other types of omics data, such as genomic, phenotypic and other omics information, which has been shown to enhance predictive accuracy based on empirical evidence.

Conclusions

In order to democratize the implementation of genomic prediction, there is a need for software that is easier to use, more automatic, and open source.

The book provides fundamental concepts, illustrative examples, and open-source code for various statistical machine learning methods that are useful for genomic selection.

Furthermore, this document contains most of the state-of-the-art algorithms utilized in genomic prediction, presenting a comprehensive resource for researchers in the field.

Questions ?

Thanks!

REFERENCES

Montesinos-López A, Montesinos-López OA, Montesinos-López JC, Flores-Cortes CA, de la Rosa R, Crossa J. (2021). A guide for kernel generalized regression methods for genomic-enabled prediction. *Heredity* (Edinb). 126(4):577-596. doi: 10.1038/s41437-021-00412-1.

Montesinos-López O. A., Montesinos-López A., Crossa, J. (2022a). *Multivariate Statistical Machine Learning methods for genomic prediction*. Springer Nature, Switzerland, ISBN: 978-3-030-89012-4.

Montesinos-López, O.A., Mosqueda-González, B.A., Palafox-González, A., Montesinos-López, A. and Crossa, J. (2022). A General-Purpose Machine Learning R Library for Sparse Kernels Methods With an Application for Genome-Based Prediction. *Front. Genet.* 13:887643. doi: 10.3389/fgene.2022.887643