Published August 23, 2018

An R Package for Multitrait and Multienvironment Data with the Item-Based Collaborative Filtering Algorithm

Osval A. Montesinos-López,* Francisco Javier Luna-Vázquez, Abelardo Montesinos-López, Philomin Juliana, Ravi Singh, and José Crossa*

O.A. Montesinos-López, F.J. Luna-Vázquez, Facultad de Telemática, Univ. de Colima, 28040, Colima, Colima, México; A. Montesinos-López, Dep. de Matemáticas, Centro Univ. de Ciencias Exactas e Ingenierías, Univ. de Guadalajara, 44430, Guadalajara, Jalisco, México; P. Juliana, R. Singh, J. Crossa, CIMMYT, Apdo. Postal 6-641, 06600, Ciudad de México, México.

ABSTRACT The Item-Based Collaborative Filtering for Multitrait and Multienvironment Data (IBCF.MTME) package was developed to implement the item-based collaborative filtering (IBCF) algorithm for continuous phenotypic data in the context of plant breeding where data are collected for various traits and environments. The main difference between this package and the other available packages that can implement IBCF is that this one was developed for continuous phenotypic data, which cannot be implemented in the current packages because they can implement IBCF only for binary and ordinary phenotypes. In the following article, we will show how to both install the package and use it for studying the prediction accuracy of multitrait and multienvironment data under phenotypic and genomic selection. We illustrate its use with seven examples (with information from two datasets, Wheat_IBCF and Year_IBCF, which are included in the package) comprising multienvironment data, multitrait data, and both multitrait and multienvironment data that cover scenarios in which breeding scientists are interested. The package offers many advantages for studying the genomic-enabled prediction accuracy of multitrait and multienvironment data, ultimately helping plant breeders make better decisions.

Abbreviations: DH, days to heading; GS, genomic selection; GY, grain yield; IBCF, item-based collaborative filtering; ICBF.MTME, Item-Based Collaborative Filtering for Multitrait and Multienvironment Data package; MSEP, mean square error of prediction; PTesting, the percentage of data to be used for the testing dataset; TRN, training dataset; TST, testing dataset.

CORE IDEAS

- We provide software for a recommender system.
- This software will assist plant breeders to make predictions of unobserved primary traits from other observed secondary traits.
- The software could be useful in conventional phenotypic selections or in genomic selection.

GLOBAL FOOD PRODUCTION must increase by 70% by 2050 to feed an additional 2.3 billion people; in developing countries, it needs to almost double. Although production will not need to grow as fast as in previous decades because of a slowdown in population growth rates, it must be pointed out that incomes are growing, and the volume requirements are remarkable. For example, an additional one billion tons of cereals and 200 million tons of meat will need to be produced annually by 2050 (Food and Agriculture Organization of the United Nations, 2009). Therefore, there is an urgent need to increase agricultural productivity to meet the enormous challenges facing humanity. Simply stated, current improvements in crop production through genetics and agronomy are insufficient (less than half of what is

Citation: Montesinos-López, O.A., F.J. Luna-Vázquez, A. Montesinos-López, P. Juliana, R. Singh, J. Crossa, 2018. An R Package for Multitrait and Multienvironment Data with the Item-Based Collaborative Filtering Algorithm. Plant Genome 11:180013. doi: 10.3835/plantgenome2018.02.0013

Received 24 Feb. 2018. Accepted 24 May 2018.
*Corresponding authors (oamontes2@hotmail.com, j.crossa@cgiar.org).

needed) to support the predicted human activities caused by population growth and increased prosperity by 2050. The predicted 9 billion people in 2050 will consume the same amount of agricultural products that 12 billion people would consume today (Godfray et al., 2010).

Consequently, breeding scientists need innovative methods to be able to reach the goal of increasing food production without significantly increasing the land used for agricultural production. Genomic selection (GS) facilitates the rapid selection of superior genotypes and accelerates the breeding cycle (Crossa et al., 2017), which could increase grain production in less time and revolutionize animal breeding and plant breeding. For example, in some beef breeds, GS is now applied on a large scale, as in the United States where more than 52,000 'Angus' cattle (*Bos taurus* L.) have now been genotyped for genomic estimated breeding value evaluation (Lourenco et al., 2015). Furthermore, the results of simulations and applications in some breeding programs show evidence of the power of GS (Meuwissen et al., 2001; Bernardo and Yu, 2007; Lorenzana and Bernardo, 2009; Heffner et al., 2009; 2010), since it has been shown to improve genetic gains (Massman et al., 2013; Asoro et al., 2013; Combs and Bernardo, 2013; Beyene et al., 2015; Rutkoski et al., 2015) and significantly reduce the time needed to release new varieties of plants and animals. If we compare the results of maize (*Zea mays* L.) and wheat (*Triticum aestivum* L.) breeding programs that use traditional selection to programs that use GS, though it is comparable to the traditional scheme, GS produces considerable savings in time and resources (Massman et al., 2013; Asoro et al., 2013; Combs and Bernardo, 2013; Beyene et al., 2015; Rutkoski et al., 2015).

However, the GS approach depends heavily on the quality of the data at hand and on the statistical methods: GS uses statistical models for predicting individuals in the validation set that were genotyped by using only the individuals in the training set that were phenotyped and genotyped. As such, when the data quality is poor, we cannot expect good predictions. Along the same lines, when the training set is very small, the model will most probably not work well for prediction. When the structure and size of the data are complex, it is more difficult to make good predictions because we cannot process the very large datasets that are becoming more common in GS. Last but not least, the type of statistical model used is essential for making good predictions. Accordingly, plant breeders and statisticians are continuously searching for better statistical methods to improve the prediction power of the GS paradigm. Whereas this type of research has increased the number of statistical methods that are currently used in GS, there is still a need for better statistical methods to increase prediction power considerably.

This lack of statistical methods for GS is more evident in the context of multitrait and multienvironment data, since many of today´s breeding programs want to improve lines with multiple traits tested in different environments simultaneously. In this sense, there are some models available, such as the Bayesian multitrait and multienvironment model proposed by Montesinos-López et al. (2016) for continuous data, and its Bayesian counterpart for count data (Montesinos-López et al., 2017). However, the main disadvantage of these models is that they are not at all efficient in the context of the large datasets that are becoming more and more common in the GS context. Therefore, to address the lack of methods for multitrait and multienvironment data, Montesinos-López et al. (2018) proposed the IBCF algorithm, which proved to be comparable to the conventional multitrait and multienvironment models, but had the advantage of being very efficient in terms of the time required for its implementation.

The IBCF is an algorithm attributed to Amazon.com (Linden et al., 2003). It is a type of collaborative filtering method for recommender systems based on the similarity between items calculated from people's ratings of those items (Linden et al., 2003; Sarwar et al., 2001). This method first executes a model-building stage by finding the similarity between all pairs of items. This similarity function can take many forms, such as the correlation between ratings or the cosine of those rating vectors. Afterwards, the system executes a recommendation stage. It uses the items that are most similar to a user's already-rated items to generate a list of recommendations. Usually, this calculation is a weighted sum or linear regression. This form of recommendation is similar to "People who rate Item X highly, like you, also tend to rate Item Y highly, and you haven't rated Item Y yet, so you should try it" (Linden et al., 2003; Sarwar et al., 2001). This algorithm is used for making online recommendations of products like movies, music, news, books, research articles, search queries, social tags, and products in general. Although software for implementing this method is available, the available software for IBCF works only for binary or ordinal responses, since most products are rated on a binary or ordinal scale. The advantage of this method is that it can be successfully implemented in the context of plant breeding for continuous phenotypes to make predictions in the context of multitrait and multienvironment data, as was done by Montesinos-López et al. (2018). However, no software is available for its implementation in the context of plant breeding for continuous traits.

Therefore, we propose an R package called IBCF.MTME for studying the prediction accuracy of datasets that are collected as multitrait and multienvironment data (R Core Team, 2018). However, it can only be used with multitrait or multienvironment data and is only appropriate for continuous phenotypes. We provide many useful tools that facilitate the study of the prediction accuracy of multitrait and multienvironment datasets, which are very common in plant breeding. We illustrate the use of the package by giving six examples that cover many practical scenarios of interest to plant breeders. Additionally, we provide tools for data transformation and summary plots to help breeders prepare the dataset the package requires as input, as well as explain how to interpret its results.

MATERIALS AND METHODS

The IBCF Algorithm

The IBCF algorithm is very popular with electronic commerce websites for recommending items and products, in which they use inputs about a customer's interests to generate a list of recommended items. This algorithm was recently implemented in GS and proved to be comparable to conventional whole-genome prediction models when the correlation between traits and environments was moderate or high (Montesinos-López et al., 2018). The IBCF algorithm basically works by building a database of users' preferences (lines) for items (trait–environment combinations), as can be seen in Table 1, which provides the raw phenotypic data on six lines evaluated in two different environments (E1 and E2) for two different traits (T1 and T2), with both traits measured in different scales. This raw phenotypic dataset has four missing values. By column, we then standardize Eq. [1] for each column in Table 1 (except the first one):

$$([z_{ij} = (y_{ij} - \mu_j)\sigma_j^{-1}]), \quad [1]$$

where i denotes the users (lines) and j denotes the columns (trait–environment combinations). We then use the standardized information to form Table 2. In this example, $i = 1,\ldots,6$, $j = 1,2\ldots,4$, μ_j is the mean of column j, and σ_j denotes the SD of Column j. In addition, for purposes of comparison, the true values of the missing values are: $y_{11} = 5.4387$, $y_{24} = 80.009$, $y_{42} = 6.979$, and $y_{63} = 72.085$. We then calculated Pearson's correlations among the columns of Table 2 (trait–environment combinations), which are given in Table 3. We used the following formula (Sarwar et al., 2001; Montesinos-López et al., 2018) to calculate the predictions for the missing phenotypes of line i in item j:.

$$\hat{y}_{ij} = \mu_j + \sigma_j\, \hat{z}_{ij} \quad [2]$$

where $\hat{z}_{ij} = \dfrac{\sum_{j' \in N_i(j)} z_{ij'} w_{jj'}}{\sum_{j' \in N_i(j)} |w_{jj'}|}$ is the scaled predictive phenotype for user (line) i on item (trait–environment) j, $N_i(j)$ denotes the items rated by user (line) i as being most similar to item j, $w_{jj'}$ is the weight between items j and j'. The weights used in Eq. [2] are obtained from an item-to-item similarity matrix built from Pearson's correlation (given in Table 3), which provides information on how similar an item is to another item.

Next, we illustrate how to calculate the four missing values via Eq. [2]. First, we calculate the scaled predicted value for y_{11}:

$$\hat{z}_{11} = \frac{-1.2988 \times 0.7103 - 0.0127 \times 0.8838 - 0.6769 \times 0.9912}{|0.7103| + |0.8838| + |0.9912|} = -0.6207\,. \quad [3]$$

The predicted value of y_{11} in its original scale is equal to $\hat{y}_{11} = \hat{z}_{11}\sigma_j + \mu_j = -0.6207 \times 1.2836 + 6.9719 = 6.1752$.

This means that the predicted value of Line 1 in trait–environment Combination 1 ($\hat{y}_{11}$) is 6.1752, which is close to the true value of 5.4386873. Next, we show how to calculate the predicted value of the missing value, $\hat{y}_{24}$; first, the scaled predicted value is equal to the following:

$$\hat{z}_{24} = \frac{-0.6815 \times 0.9912 - 0.7465 \times 0.7463 - 0.9406 \times 0.9521}{|0.9912| + |0.7463| + |0.9521|} = -0.7913\,. \quad [4]$$

The predicted value of y_{24} in its original scale is then equal to $\hat{y}_{24} = \hat{z}_{24} \times \sigma_j + \mu_j = -0.7913 \times 1.5606 + 80.4028 = 79.1679$.

Now the predicted value of Line 2 in trait–environment Combination 4 ($\hat{y}_{24}$) is 79.1679, which is close to its true value of 80.0092. We then present the scaled predicted response for Line 4 and trait–environment Combination 2, y_{42}, which is:

$$\hat{z}_{42} = \frac{-1.3122 \times 0.7103 - 1.0395 \times 0.7741 - 1.3437 \times 0.7463}{|0.7103| + |0.7741| + |0.7463|} = -1.2281\,. \quad [5]$$

The predicted value of y_{42} in its original scale is equal to $\hat{y}_{42} = \hat{z}_{42} \times \sigma_j + \mu_j = -1.2281 \times 0.2967 + 8.4276 = 8.0632$.

This means that the predicted value of Line 4 in trait–environment Combination 2 ($\hat{y}_{42}$) is 8.0632, which is close to its true value of 6.9786. Finally, we present

Table 1. Example of item-based collaborative filtering: Raw phenotypic data.

Line†	T1_E1	T1_E2	T2_E1	T2_E2
Line 1	NA‡	8.0423	70.833	79.3463
Line 2	6.0971	8.2061	70.539	NA
Line 3	7.9828	8.7582	71.1034	81.3512
Line 4	5.2876	NA	70.5082	78.3059
Line 5	7.1267	8.6252	71.2019	80.8350
Line 6	8.3658	8.5061	NA	82.1754

† Line denotes the lines; T1 and T2 denote Traits 1 and 2, respectively; E1 and E2 denote Environments 1 and 2, respectively.

‡ This raw phenotypic dataset has four missing values [denoted not available (NA)].

Table 2. Example of item-based collaborative filtering: Standardized phenotypic data.

Line†	T1_E1	T1_E2	T2_E1	T2_E2
Line 1	NA‡	-1.2988	-0.0127	-0.6769
Line 2	-0.6815	-0.7465	-0.9407	NA
Line 3	0.7874	1.1144	0.8407	0.6077
Line 4	-1.3122	NA	-1.0395	-1.3436
Line 5	0.1205	0.6662	1.1522	0.2770
Line 6	1.0858	0.2647	NA	1.1358

† Line denotes the lines; T1 and T2 denote Traits 1 and 2, respectively; E1 and E2 denote Environments 1 and 2, respectively.

‡ This dataset has four missing values [denoted not available (NA)].

Table 3. Example of item-based collaborative filtering: Matrix of correlation.

Line†	T1_E1	T1_E2	T2_E1	T2_E2
T1_E1	1.000	0.7103	0.8838	0. 9912
T1_E2	0.7103	1.0000	0.7741	0.7463
T2_E1	0.8838	0.7741	1.000	0.9521
T2_E2	0.9912	0.7463	0.9521	1.0000

† Line denotes the lines; T1 and T2 denote Traits 1 and 2, respectively; E1 and E2 denote Environments 1 and 2, respectively.

the scaled predicted value of Line 6 in trait–environment Combination 3:

$$\hat{z}_{63}=\frac{1.0858\times 0.8838+0.2646\times 0.7741+1.1359\times 0.9521}{|0.8838|+|0.7741|+|0.9521|}=0.8605\,. \quad [6]$$

The predicted value of y_{63} in its original scale is equal to $\hat{y}_{63}=\hat{z}_{63}\times\sigma_j+\mu_j=0.8605\times 0.3165+70.8373=71.1097$.

This means that the predicted value of Line 6 in trait–environment Combination 3 ($\hat{y}_{63}$) is 71.1097, which is close to its true value, 72.0845.

As this example shows, the calculations are easy but laborious; however, the IBCF.MTME package does this job automatically for us and the dataset required can be on different scales (not standardized) for the traits; in other words, the traits can be measured on different scales. Internally, the IBCF.MTME package standardizes ($\left[z_{ij}=(y_{ij}-\mu_j)\sigma_j^{-1}\right]$) in each column of the dataset given in Table 4 (which shows the format of the type of data required) for the training dataset obtained in each random partition. This implies that to use the formula given in Eq. [2] for making predictions about a particular trait–environment combinations, the data are standardized and the similarity matrix resulting from the corresponding training dataset of a particular partition selected from the whole dataset in Table 4 is computed. Therefore, the predictions are obtained by using Eq. [2] with the parameters estimates obtained from the training dataset corresponding to each partition and the predictions are made for the observations in the testing dataset.

Evaluation of Prediction Accuracy in the IBCF.MTME Package

The IBCF.MTME package can implement a random cross-validation. This method consists of randomly dividing the whole dataset into two subsets: the training (TRN) dataset and the testing (TST) dataset. The percentage of the whole dataset assigned to the TRN and TST datasets is fixed by the user. For example, for each random partition, 80% of the whole dataset can be assigned to the TRN and the remaining 20% to the TST dataset. Random cross-validation is different from *K*-fold cross-validation because the partitions are not mutually exclusive; this means that in the random cross-validation approach, one observation can appear in more than one partition. Consequently, some samples cannot be evaluated, whereas others can be evaluated more than once, meaning that the testing and training subsets can be superimposed.

The random cross-validation procedure explained above is the best option when the data are not stratified. For this reason, the datasets for breeding programs in the context of multienvironment data are stratified by environments. Therefore, the random cross-validation explained above was modified so it could be implemented in the IBCF.MTME package. For multienvironment data on only one trait, the problem of predicting the performance of lines in environments where they have not been evaluated was considered. This validation design mimics the prediction problem faced by breeders in incomplete field trials where lines have been evaluated in some but not all target environments. The TRN–TST partitions for this prediction problem were obtained as follows: since the total number of records available for the dataset with multienvironments and only one trait is $N = J \times I$, to select the lines in the TST data set, the user fixes the percentage of data to be used for TST (PTesting). We then chose PTesting × N (lines) at random; subsequently, one environment per line was randomly picked from the index of environments (i = 1, 2, …, I). The resulting cells (ij) were assigned to the TST dataset and the ones not selected through this algorithm were used for the TRN dataset. Lines were sampled without replacement if $J \geq$ PTesting × N and with replacement otherwise (López-Cruz et al., 2015). The construction of the TRN–TST datasets for multitrait and multienvironment data was similar to the construction of the multienvironment data explained above. However, since the data are now for multiple traits and multiple environments, we assume that the response variable is missing for all the traits under study and not just for the one missing in the multienvironment dataset. On the other hand, for multitrait data with only one environment, the construction of the TRN–TST datasets was exactly the same as the construction of the multienvironment data, with the assumption that environments are taken as traits.

We also provide another type of cross-validation that is only useful when we want to predict certain traits of some lines that are missing in one or more years (or environments) but are present in others. However, in this type of cross-validation, there is only one partition (one testing set and one training set; no subsampling); subsequently, in the output for each year–trait or environment–trait combination for both metrics [Pearson's correlation and mean square error of prediction (MSEP)], it was impossible to estimate the SE. This type of cross-validation is needed when breeders want to predict certain traits for some years (or environments) based on the information for all traits in other years or environments.

Under all types of scenarios for constructing the TRN–TST dataset for each partition, the IBCF model is fitted with the TRN dataset and calculates predictions for the TST dataset. Finally, with the output of each TST

Table 4. Phenotypic information in its original scale for building the rating matrix for multitrait and multienvironment data for J genotypes, I environments, and L traits.

Genotypes	Trait–environment combinations									
	T1_E1	…	T1_EI	T2_E1	…	T2_EI	…	TL_E1	…	TL_EI
G1†	y_{111}	⋯	y_{I11}	y_{112}	⋯	y_{I12}	⋯	y_{11L}	⋯	y_{I1L}
G2	y_{121}	⋯	y_{I21}	y_{122}	⋯	y_{I22}	⋯	y_{12L}	⋯	y_{I2L}
⋮	⋮	…	⋮	⋮	…	⋮	…	⋮	…	⋮
GJ	y_{1J1}	⋯	y_{IJ1}	y_{1J2}	⋯	y_{IJ2}	⋯	y_{1JL}	⋯	y_{IJL}.

† G, genotypes; E, environment; T, trait; y_{ijl}, the phenotype from the j^{th} line in the i^{th} environment for the l^{th} trait.

dataset, two metrics (in this package, they were Pearson's correlation and MSEP) were used to measure prediction accuracy; they were calculated with the information on the observed and predicted values of the testing dataset. The final outputs for both metrics are the arithmetic mean and SE of the random partitions implemented.

A summary of prediction accuracies is given for trait–environment combinations or trait–year combinations, since it is very important in varietal recommendations to evaluate prediction accuracy at the environment level for genotypes and traits because of the importance of genotype × environment and genotype × environment × trait interactions. As such, in this article, we use "environment" in its general connotation—that is, as synonymous to a region—because, when recommending cultivars, it is often more important to evaluate prediction accuracy when modeling the genotype × region interaction rather than the genotype × environment interaction.

About the IBCF.MTME Package

To start using the package, you will need to install the IBCF.MTME R package (R Core Team, 2018) from GitHub via the devtools package, with the command devtools::install_github('frahik/IBCF.MTME'). To adjust the model with the IBCF() function, first, you have to generate the partitions needed to implement the cross-validation with the auxiliary function CV.RandomPart() as follows:

```
CV.RandomPart(Dataset, NPartitions = 10,
PTesting = .35, Traits.testing = NULL,
Set_seed = NULL).
```

By default, the function generates 10 random partitions of the dataset; in each partition, 65% of the data is assigned to the training sample and the rest (35%) is assigned to the validation sample. If you want to modify the default values, you can change both parameters. The NPartitions parameter represents the number of partitions generated to implement the cross-validation, PTesting is the percentage of observations used as a validation sample in each partition, and Traits.testing is null by default and uses all the traits in the TST to fit the model; otherwise, those traits specified in Traits.testing are assumed to be missing. On the other hand, Set_seed is the seed to be used when generating random numbers to form random partitions for reproducible research. In other words, the CV.RandomPart() function is used to generate training and validation samples to study the predictive ability under IBCF for the dataset. In addition, the output of this function is an input of the IBCF() function. The dataset object is a data.frame object that contains four columns in the "Tidy Data" format:

- $Line: The line or genotype identifier; the name of this column can be changed.
- $Env: The name of the evaluated environment(s); the name of this column must be respected.
- $Trait: the name of the evaluated trait (s); this name must be respected.
- $Response: The variable response obtained for the row corresponding to a line, environment, and trait combination; this name should not be modified.

As output of the above function, a cross-validation object is returned with the following data:

- Dataset: This is a data.frame object of size $n \times (m + 1)$, where n is the number of entered lines under study and m is the value that results from multiplying the number of environments under study by the number of evaluated traits. In addition, the first column corresponds to the identifier of each line.
- CrossValidation_list: A list that contains the partitions, NPartitions, where each index contains an $n \times m$ dimension matrix, where n is the number of lines under study, and m is the length of the combination of traits and the environments evaluated; each value should be 1 if the observation will be present in TRN and 2 if the observation will be in TST.
- Environments: The name of the evaluated environment(s).
- Traits.testing: The names of the traits used as part of TST for traits; this means that the other traits are assumed to be known but these can be missing. If Traits.testing is null, all traits can be missing.
- Traits: The name of the evaluated trait(s).
- Observations: Unique identifiers found in the dataset.
- Class: Cross-validation

Once the object generated by the previous function has been successfully obtained, the IBCF method can be implemented as follows: IBCF(CrossValidation_Object, dec = 4), with the following parameters:

- CrossValidation_Object: An object generated by the function CV.RandomPart().
- Dec: The number of decimal places to be displayed in the output.

By using the IBCF() function, an object of the IBCF class is generated, containing the following:

- NPartitions: The number of random partitions evaluated.
- predictions_Summary: A data.frame object with the data summary containing the following columns:
  - Trait_Env: The combination of traits and environments evaluated.
  - Pearson: The mean Pearson's correlation obtained from the number of the implemented random partitions.
  - SE_Cor: The SE of the mean Pearson's correlation,
  - MSEP: The mean square error of prediction calculated from the number of random partitions implemented.
  - SE_MSEP: The SE of the MSEP.
- observed: All values used to fit the model.
- yHat: All predicted values.

- predicted_Partition: A list of matrices with the predicted values for each partition.
- Data.Obs_Pred: A data.frame object in matrix form to obtain the $observed and the predicted values ($yHat) that contains the following columns:
  - Gids: Line or genotype identifier; the name of this column may be changed.
  - Traits_Env: Contains the entered trait–environment combinations under study.
  - Traits_Env.1: Contains the predicted values of the trait–environment combinations evaluated.
- Class: IBCF.

The package also includes a variation of the IBCF() function dedicated to studying predictive ability. For example, when we are interested in predicting all lines in a year or years or if we want to use information from other years as TRN. This function is denoted as: IBCF.Years(Dataset, colYears = 1, Years.testing = " ", Traits.testing = " "). For this, the parameters required as input are as follows:

- Dataset: A data.frame with at least three columns distributed in the matrix format, where the first $Years column must contain the identifier of each year under study, the next column gives the identifier of each line or genotype, and the following columns give the traits evaluated for each line in each year.
- colYears: The name or position of the column containing all years; by default, this column is the first column of the dataset.
- Years.testing: A vector that contains some of the years entered in the first column of the dataset object to use as testing values to fit the model. It is important to mention that the vector Years.testing must contain at least 1 yr and should be less than all the years under study; otherwise, there will be no information for the training model.
- Traits.testing: A vector that contains some of the names corresponding to the columns that identify the traits; this vector indicates the traits that will be used to form the validation sample. It is important for the length of the vector Traits.testing to be less than the total number of traits under study.

The output of IBCF.Years() results in an object that contains the following:

- Years.testing: Contains the names of the years used as testing.
- Traits.testing: Contains the traits used as testing.
- predictions_Summary: A data.frame object that contains the following columns:
  - Year_Trait: The combination of the year with the evaluated trait.
  - Pearson: The mean Pearson's correlation obtained.
  - MSEP: The mean square error of prediction calculated.
- observed: A vector with the observed values;
- predicted: A vector with the predicted values;
- Data.Obs_Pred: A data.frame object that contains the following columns:
  - Years: Contains the years used in the cross-validation as testing.
  - Gids: Line or genotype identifier; the name of this column may be changed.
  - Traits: Contains the entered observations of the evaluated traits.
  - Traits.1: Contains the predictions of the evaluated traits.
- Class: IBCFY.

Useful Functions of the Package

The package includes two other tools for data transformation that make the user's work easier.

The getMatrixForm() Function

The getMatrixForm(TidyDataset, onlyTrait = FALSE) function is used for transforming the tidy data into the matrix form required by the program and requires the following parameters:

- TidyDataset: A data.frame object that contains four columns:
  - $Line: The line or genotype identifier; the name of this column may change.
  - $Env: The name of the evaluated environment(s); the name of this column cannot be changed.
  - $Trait: The name of the evaluated trait(s); the name of this column cannot be changed.
  - $Response: The variable response obtained for the rows corresponding to the combinations of line, environment and trait; the name of this column cannot be changed.
- onlyTrait: Logical value that, by default, is FALSE and uses the $Trait and $Env columns for the matrix format; if true, it only uses the $Trait column.

The output of this function is a data.frame object of $n \times (m + 1)$ dimensions, where n is the number of lines entered (without repetition) and m is the length of the combination of traits and environments plus the first column of the corresponding lines.

The getTidyForm() Function

The getTidyForm(Mat, onlyTrait = FALSE) is the inverse function of getMatrixForm() and converts a data matrix form to a Tidy Data form; to use this function, the following parameters are required:

- Mat: A data.frame object of a $n \times (m + 1)$ dimensions, where n is the number of lines under study and m is the length of the combination of traits and environments under study plus the first column corresponding to the lines.

- onlyTrait: A logical value; by default, it is false and ignores the first column to transform the rest of the columns with the expected 'Trait_Env' names format and summarizes the data in three columns: $Trait, $Env, and $Response. If the value is true, it ignores the first two columns to transform the rest of the columns, considering them as the evaluated traits, and summarizes them in two columns: $Trait and $Response.

The output of this function is a data.frame object with transformed data of $q \times 4$ dimensions, where q results from the product of the number of lines, the number of environments, and the number of traits evaluated and 4 represents the number of columns corresponding to lines, environments, traits, and responses.

Wheat Dataset

The package includes a dataset used in a study by Montesinos-López et al. (2018). This dataset, called *Wheat_IBCF*, consists of 250 wheat lines evaluated in three environments with four traits (i.e., 3000 observations).

You must use the data() function.

```
data('Wheat_IBCF')
```

The data are contained in a data.frame object divided as follows:

- $GID: The genotype identifier (with 250 different identifiers).
- $Trait: The name of the evaluated traits (with four different traits).
- $Env: The name of the evaluated environments (with three different environments).
- $Response: The wheat yield for the corresponding combination of traits, environments, and lines (with 3000 observations).

To see more details on this dataset, we will use the head() function to obtain the first six observations:

```
head(Wheat_IBCF)
##       GID Trait    Env   Response
## 1 6569128    DH Bed2IR -17.565895
## 2 6688880    DH Bed2IR  -4.565895
## 3 6688916    DH Bed2IR  -3.565895
## 4 6688933    DH Bed2IR  -4.565895
## 5 6688934    DH Bed2IR  -7.565895
## 6 6688949    DH Bed2IR  -7.565895
```

The dataset is in the Tidy Data format, meaning that each column represents a variable, each row is an observation, and each cell is the value belonging to the combination of trait, line, and environment. This way of preparing a dataset was proposed by Wickham (2014). To observe the entire structure of the dataset, we use the str() function:

```
str(Wheat_IBCF)
## 'data.frame':    3000 obs. of  4 variables:
##  $ GID     : int  6569128 6688880 6688916 6688933
6688934 6688949 6689268 6689407 6689482 6689550 ...
##  $ Trait   : chr  "DH" "DH" "DH" "DH" ...
##  $ Env     : chr  "Bed2IR" "Bed2IR" "Bed2IR" "Bed2IR" ...
##  $ Response: num  -17.57 -4.57 -3.57 -4.57 -7.57 ...
```

We have 3000 observations for four variables in this dataset, which is consistent with the explanation above.

Year Dataset

The package includes a dataset based on simulated data; Appendix A includes the code used to simulate this dataset. The dataset, called *Year_IBCF*, consists of 60 lines evaluated in the 3 yr under study (20 lines per year) for 12 traits, which gives a total of 720 observations.

The data() function must be used to load the data.

```
data('Year_IBCF')
```

The data are contained in a data.frame object divided as follows:

- Years: Each year where each line was evaluated,
- Gid: The identifier of each line evaluated,
- Trait: The name of each trait evaluated, and
- Response: The response variable corresponding to a line or trait in each year.

Again, to see how this dataset is composed, you can use head(Year_IBCF) to see the first six observations.

Note that this dataset is also in Tidy Data format and the structure of the object is as follows:

```
str(Year_IBCF)
## 'data.frame':    720 obs. of  4 variables:
##  $ Years   : num  2014 2014 2014 2014 2014 ...
##  $ Gids    : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Trait   : chr  "T1" "T1" "T1" "T1" ...
##  $ Response: num  5.14 5.68 4.85 3.57 5.02 ...
```

Note that it is consistent with the explanation above: 720 observations of four variables.

EXAMPLES

This section illustrates the use of the **IBCF.MTME** package and gives several examples of GS to predict continuous traits.

Example 1: Data Transformation

In this example, you will find more details about the two formats established in this package for data processing. First, you will find the Tidy Data format and then the matrix format. The function getMatrixForm() is useful for transforming a dataset from the Tidy Data format to the matrix format, whereas the getTidyForm() transforms a dataset from the matrix format to the Tidy Data format. The two datasets provided in the package are in Tidy Data format, so we will use the Wheat_IBCF dataset to exemplify how to use these two functions.

First, in a clean R environment, the IBCF.MTME library should be loaded and, with it, the Wheat_IBCF dataset through the use of the data() command, as shown in the following code block:

```
rm(list = ls())
library(IBCF.MTME)
data('Wheat_IBCF')
```

Once the data are loaded, the head() function shows that this dataset is in the Tidy Data format, since it only has four columns corresponding to lines [genotypic identifier of lines (GID)], traits (Trait), environments (Env), and the response variables (Response). In this dataset, GID is the identifier of the lines.

```
head(Wheat_IBCF)
##       GID Trait    Env   Response
## 1 6569128    DH Bed2IR -17.565895
## 2 6688880    DH Bed2IR  -4.565895
## 3 6688916    DH Bed2IR  -3.565895
## 4 6688933    DH Bed2IR  -4.565895
## 5 6688934    DH Bed2IR  -7.565895
## 6 6688949    DH Bed2IR  -7.565895
```

The $GID column contains the identifier of each line, days to heading (DH) is the trait evaluated in the first six observations for the "bed planting system with two irrigations" (Bed2IR) environment, and the $Response column stores the responses obtained for each row that corresponds to a specific line, trait, or environment. We can observe the dimension of this dataset through the dim(Wheat_IBCF) command, which indicates that this dataset has 3000 rows and four columns. In addition, if we are looking to transform this format into the matrix format, we can do it through the getMatrixForm() function. The first six observations can be seen with the head() command, as detailed below:

```
M <- getMatrixForm(Wheat_IBCF)
head(M)
##       GID  DH_Bed2IR  DH_Bed5IR   DH_Drip   GY_
Bed2IR   GY_Bed5IR GY_Drip
## 1 6569128 -17.565895  1.6923078  0.945047
-0.35188724 -0.38496851 -0.44365852
## 2 6688880  -4.565895 -0.3076922  0.945047
-0.61107566 -0.29930171 -0.19868885
## 3 6688916  -3.565895 -1.3076922 -2.054953
-0.03534797 -0.30202380 -0.09566196
## 4 6688933  -4.565895 -3.3076922 -1.054953
-0.11721194  0.08545996 -0.03161493
## 5 6688934  -7.565895 -3.3076922 -0.054953
0.05415546 -0.24010219 -0.35510868
## 6 6688949  -7.565895 -6.3076922 -2.054953
-0.48406932  0.19542652 -0.04798615
##     NDVI_Bed2IR  NDVI_Bed5IR    NDVI_Drip  PH_
Bed2IR  PH_Bed5IR  PH_Drip
## 1 -0.016692414 -0.002449404  0.006528357  -6.968964
-1.282471  3.6682234
## 2 -0.016657172 -0.011380990 -0.022684209 -11.030829
-8.725086 -3.6481125
## 3  0.002233731 -0.000877320 -0.030087900 -10.494572
-6.680635 -6.1052408
## 4 -0.009772406 -0.011128989  0.003965353  -4.033027
-5.546853 0.7411906
## 5 -0.002767986 -0.010064999  0.008624755 -12.103778
-11.590284 -6.4430134
## 6 -0.015566957  0.002153392 -0.021943740  -5.316197
-1.954483  3.0296975
```

Note that the first column does not appear to be completely altered; however, in this column, the identifiers of the line now appear without repetition but in subsequent columns, the names are the combinations of the traits and environments. Therefore, each value represents the observation measured in each line for a given trait–environment combination. The data have been considerably reduced, with the dim(M) function are shown to have 250 rows and 13 columns. The size of the dataset is now 250 rows that correspond to the unique identifiers of the lines; the columns correspond to 12 different trait–environment combinations plus the column of the line identifiers. This format is popular among researchers because of the compact way it represents the data. However, the cross-validation function of the package requires the tidy format to work, so if you do not have a set of data in this format, you need to convert them to the tidy format with the getTidyForm() function. In this example, we will obtain the original version of the dataset again, as shown in the following block form:

```
Tidy <- getTidyForm(M)
```

With the head() command, the data are once again in the Tidy Data format:

```
head(Tidy)
##       GID Trait    Env   Response
## 1 6569128    DH Bed2IR -17.565895
## 2 6688880    DH Bed2IR  -4.565895
## 3 6688916    DH Bed2IR  -3.565895
## 4 6688933    DH Bed2IR  -4.565895
## 5 6688934    DH Bed2IR  -7.565895
## 6 6688949    DH Bed2IR  -7.565895
```

This fact can also be verified through the dimensions of the dataset with the dim(Tidy) function, which shows it has 3000 rows and four columns.

Example 2: Predictions with a Single Environment and Multiple Traits

This example shows how to study prediction accuracy when the data only have one environment and four traits. This is done with the Wheat_IBCF dataset included in the package. The following code block is used to load this dataset:

```
rm(list = ls())
library(IBCF.MTME)
data('Wheat_IBCF')
```

With this, we loaded the object Wheat_IBCF, which was described in the wheat dataset section. Although this dataset has information from four environments, we will only work with the Bed5IR environment; therefore, the dataset to use is:

```
Dataset <- Wheat_IBCF[which(Wheat_IBCF$Env ==
'Bed5IR'), ]
```

With dim(Dataset), we get that the dimension of this dataset that is 1000 rows and four columns. The matrix has been reduced from 3000 rows (observations) to 1000. Next, we use the CV.RandomPart() function to obtain the partitions of the sample for cross-validation:

```
CrossV <- CV.RandomPart(Dataset, NPartitions = 10,
PTesting = 0.25, Set_seed = 5)
```

"Dataset" is the dataset in tidy format that contains information on the multitrait and multienvironment experiment; with NPartitions = 10, you are specifying that you want to make a cross-validation with 10 random partitions; and with PTesting = 0.25, you are specifying that 25% of the whole dataset in each partition will be assigned to TST and the remaining 75% to TRN. In addition, you must specify the number of seeds needed to obtain a reproducible work, regardless of the operating system and the person who runs it.

We now have an object of the "CrossValidation" class that has been named CrossV; its structure can be obtained via R with str(CrossV). This object contains the new $Dataset with the matrix format automatically, 10 matrices that are part of each random partition of the required cross-validation, the environments and features that were recognized by the function, and the number of observations.

With this object, it is possible to perform a cross-validation study with the model by simply calling the IBCF() function and inserting this object as a parameter; the predictive model will be adjusted automatically.

```
pm1 <- IBCF(CrossV)
```

To view the results provided by this function, simply use the summary() function on the object where the results were stored, as shown below:

```
summary(pm1)
##     Trait_Env Pearson SE_Cor    MSEP SE_MSEP
## 1   DH_Bed5IR  0.3488 0.0242 33.4035  1.5099
## 2   GY_Bed5IR  0.1637 0.0325  0.1106  0.0062
## 3 NDVI_Bed5IR  0.5794 0.0370  0.0001  0.0000
## 4   PH_Bed5IR  0.0602 0.0241 41.7650  1.4914
```

As a result of the summary() function, five columns are obtained. The first corresponds to the trait–environment combination for which prediction accuracies are provided under two metrics: Pearson's correlation and MSEP. The second column provides the average predictive ability of the partitions implemented in terms of Pearson's correlation and the third column provides the SE of the mean Pearson's correlation given in Column 2. In the fourth column, the average MSEP of the randomly implemented partitions are given, and in the last column, the corresponding SE for the MSEP are provided.

To further understand the structure of this object, we can use the str(pm1) command. Use of this command shows that this object contains six components: the number of partitions evaluated; the summary of the predictions [which is retrieved through the summary() function]; the observed values and predicted values, which also appear in $Data.Obs_Pred ordered by the line identifier; and the trait–environment combination. The predictions obtained in each partition also appear. Below we show the first six observed and predicted values obtained by the model:

```
head(pm1$Data.Obs_Pred)
##       GID  DH_Bed5IR   GY_Bed5IR  NDVI_Bed5IR  PH_
Bed5IR DH_Bed5IR.1 ... PH_Bed5IR.1
## 1 6569128  1.6923078 -0.38496851 -0.002449404
-1.282471 -3.42715638 ... -6.3549369
## 2 6688880 -0.3076922 -0.29930171 -0.011380990
-8.725086 -6.81057189 ... -6.3010208
## 3 6688916 -1.3076922 -0.30202380 -0.000877320
-6.680635 -3.53399864 ... -5.6198338
## 4 6688933 -3.3076922  0.08545996 -0.011128989
-5.546853 -4.36973174 ... -2.6316550
## 5 6688934 -3.3076922 -0.24010219 -0.010064999
-11.590284 -6.49925946 ... -6.3959559
## 6 6688949 -6.3076922  0.19542652  0.002153392
-1.954483 -0.07608169 ... -0.7451675
```

These are the trait–environment combinations of observed and predicted values, those that appear in the output above without .1, after the name of the trait-environment combination correspond to the observed values, while those with .1 after the trait-environment combination correspond to the predicted values. In addition, a plot can be made with the summarized predictions for the trait-environment combinations under study. For example, Fig. 1a was obtained via the plot(…) function by entering the IBCF object, generated with the IBCF() function. The "select" parameter allows you to obtain a graph with a summary of the predictions with the Pearson's correlation or MSEP metric.

```
par(mai = c(2, 1, 1, 1))
plot(pm1, select = 'Pearson')
```

Note that the code of the par() function has served as an auxiliary to adjust the margins of the plot, since the names on the *x* axis cannot be longer than what the plot's default parameters can support. The parameter mai allows us to modify the distance between the margins of the plot and the edge of the image. In other words, the first value corresponds to the margin with respect to the lower part of the graph, the second value with respect to the left part of the graph, the third value with respect to the upper part of the graph, and the last value with respect to the right part of the graph.

Figure 1a shows Pearson's correlation along with its corresponding confidence interval for each trait–environment combination. The results are sorted: on the left side are the trait–environment combinations with the lowest prediction abilities; on the right side are the trait–environment combinations with the best prediction abilities.

It is also possible to plot the results obtained with the MSEP metric; it is enough to change MSEP in the

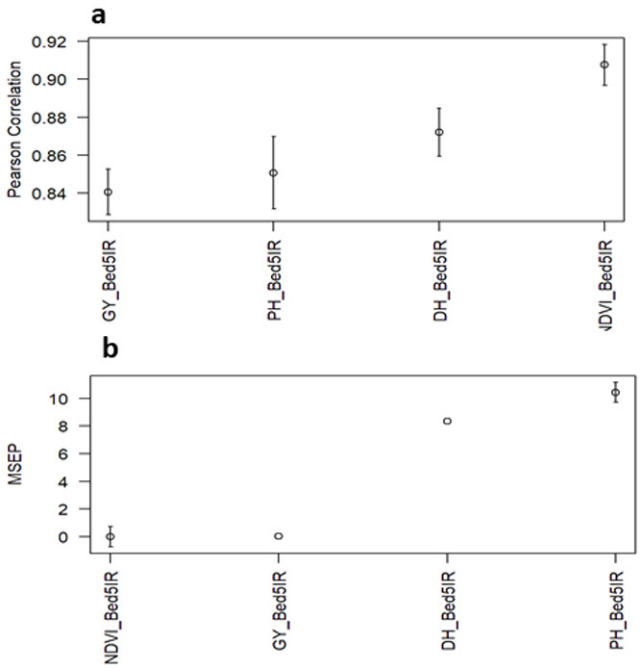

Fig. 1. Prediction accuracy in terms of (a) Pearson's correlation and (b) mean square error of prediction (MSEP) of the item-based collaborative filtering (IBCF) model for each trait–environment combination. The data are only from an environment called 'Bed5IR' (bed planting system with five irrigations).

"select" parameter. Figure 1b was generated with the code shown below.

```
plot(pm1, select = 'MSEP')
```

Example 3: Predictions with Many Environments and Traits

This example shows how to study the prediction ability of a dataset containing three environments and four traits. The whole Wheat_IBCF dataset included in the package is also used for this purpose. The way the data are loaded is the same as in the previous example, so this step was omitted. Therefore, the partitions are first generated for cross-validation and then the model is adjusted with the IBCF() function, as shown below.

```
CrossV <- CV.RandomPart(Wheat_IBCF, NPartitions = 10,
PTesting = 0.25, Set_seed = 5)
pm3 <- IBCF(CrossV)
```

Once the model has been adjusted, the results can be consulted through the summary() function, as shown in the following code.

```
summary(pm3)
##      Trait_Env Pearson SE_Cor    MSEP SE_MSEP
## 1    DH_Bed2IR  0.8201 0.0208 15.5682  1.7460
## 2    DH_Bed5IR  0.8920 0.0160  8.0849  1.3425
## 3      DH_Drip  0.9235 0.0068  1.9281  0.0988
## 4    GY_Bed2IR  0.0978 0.0483  0.3121  0.0194
## 5    GY_Bed5IR  0.5768 0.0225  0.0605  0.0040
## 6      GY_Drip  0.5591 0.0212  0.1388  0.0093
## 7  NDVI_Bed2IR  0.7356 0.0192  0.0001  0.0000
## 8  NDVI_Bed5IR  0.8321 0.0090  0.0001  0.0000
## 9    NDVI_Drip  0.7160 0.0292  0.0002  0.0000
## 10   PH_Bed2IR  0.3281 0.0374 12.6455  0.7628
## 11   PH_Bed5IR  0.5436 0.0194 28.6222  0.9848
## 12     PH_Drip  0.5356 0.0194 24.9261  1.0128
```

The only difference between the output of the summary and that of the two previous examples is that now the first column contains predictions for all combinations of four traits and three environments (i.e., 12 combinations in total).

Likewise, to obtain the summary of prediction ability with MSEP (or Pearson's correlation) for each trait-environment combination, the following code can be used:

```
par(mai = c(2, 1, 1, 1))
plot(pm3, select = 'MSEP')
```

Figure 2 shows that the best predictions in terms of MSEP were observed for the Normalized Difference Vegetation Index trait in all environments under study, whereas the worst predictions were observed for the plant height trait in the 'bed planting system with five irrigations" (Bed5IR) and drip irrigation (Drip) environments. If we want to plot the prediction accuracies in terms of Pearson's correlation, then change the select parameter to 'Pearson'.

Example 4: Predictions for Multiple Environments and Multiple Traits with Only Two Traits as Testing

This example shows how to study the predictive capability of a dataset with multiple environments and multiple traits with only two traits as testing. The dataset used is the same as the one in the previous examples, so we omitted the process of loading the dataset. With the CV.RandomPart() function, we then obtained the partitions required for implementing cross-validation.

```
CrossV <- CV.RandomPart(Wheat_IBCF, Traits.testing =
c('DH', 'GY'), NPartitions = 10,
PTesting = 0.25, Set_seed = 123)
```

We also generated 10 random partitions; in each partition, 25% of the data should be assigned to the TST dataset and the remaining 75% to the TRN dataset. The only traits that should be predicted by the model are DH and grain yield (GY). With this object, we can perform a cross-validation study with the following code:

```
pm2 <- IBCF(CrossV)
```

To view the output, simply use the summary() function with the object where the results were stored, as shown below.

```
summary(pm2)
##   Trait_Env Pearson SE_Cor    MSEP SE_MSEP
## 1 DH_Bed2IR  0.8769 0.0121 10.6964  0.8170
```

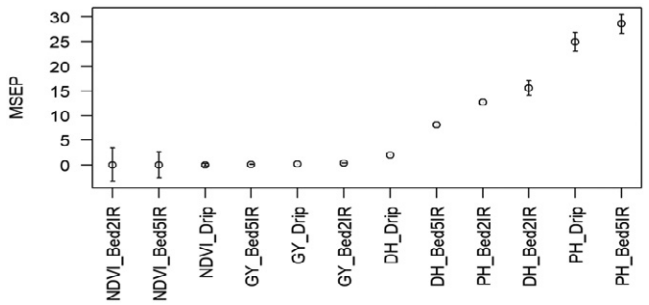

Fig. 2. Prediction abilities, in terms of mean square error of prediction (MSEP), of the model for each trait–environment combination. The data correspond to the Wheat_IBCF dataset with four traits and three environments.

```
## 2 DH_Bed5IR  0.8749 0.0185  9.6067  1.5266
## 3  DH_Drip   0.9208 0.0064  2.0380  0.1339
## 4 GY_Bed2IR  0.1819 0.0384  0.2607  0.0136
## 5 GY_Bed5IR  0.6432 0.0213  0.0544  0.0032
## 6   GY_Drip  0.6593 0.0120  0.1148  0.0068
```

It is important to point out that now the summary of the prediction ability only contains information for all the environments evaluated and for the traits specified in the traits TST set. To obtain the plot of the summary of prediction abilities in terms of Pearson's correlation for each trait–environment combination, it is not necessary to specify the "select" parameter, as shown below.

```
par(mai = c(2, 1, 1, 1))
plot(pm2)
```

Figure 3a shows the results for only two traits (DH and GY) that were specified in the Traits.testing argument and for the three different environments, with their respective confidence intervals. Similarly, the following code can be used to observe predictive capability in MSEP terms (Fig. 3b):

```
par(mai = c(2, 1, 1, 1))
plot(pm2, select = 'MSEP')
```

Example 5: Predictions for 2015 and 2016 with 2014 as Training

This example shows how to make predictions with the IBCF package when you have several traits under study and you want to predict some of them for the following years, but with information on the remaining traits in all the years that conform the training and testing data sets. This is done with the Year_IBCF dataset included in the package. The following code is used to load this dataset:

```
rm(list = ls())
library(IBCF.MTME)
data("Year_IBCF")
```

This loads the Year_IBCF object into the R environment. However, the IBCF.Years command requires the data in matrix format and because now the dataset is in Tidy Data format, we will need to transform it into the

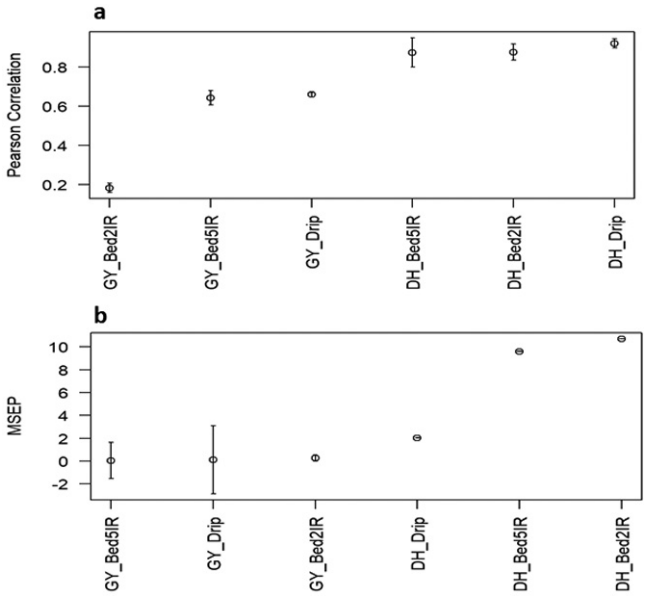

Fig. 3. Prediction accuracy in terms of (a) Pearson's correlation and (b) mean square error of prediction (MSEP) of the item-based collaborative filtering (IBCF) model for each trait in three environments. The data correspond to the Wheat_IBCF dataset.

matrix format so that can be used for the IBCF.Years command. This is achieved automatically with the getMatrixForm() function, as shown below.

```
Dataset <- getMatrixForm(Year_IBCF, onlyTrait = T)
```

Notice that onlyTrait = T has been specified to take the years as the first column, respecting the formatting set for the package. The dataset has been transformed from a Tidy Data format to the matrix format required by the IBCF.MTME package by consulting the first six observations through the head() command.

```
head(Dataset)
##   Years Gids    T1     T10      T11     T12
T2    T3   …  T9
## 1  2014    1 5.144009 8.514278 7.089700 9.167756
6.214348 7.538577 ... 8.361184
## 2  2014    2 5.678792 8.215685 7.896449 9.944295
5.806136 7.899465 ... 8.672661
## 3  2014    3 4.854895 7.725762 5.781978 7.530579
4.061641 6.119972 ... 7.056119
## 4  2014    4 3.570019 8.570091 5.733429 7.499954
4.583116 5.224950 ... 6.488097
## 5  2014    5 5.018380 8.573483 6.974145 9.030811
4.981920 5.651253 ... 6.540473
## 6  2014    6 3.196160 6.835883 5.123118 7.398668
5.812636 4.805169 ... 5.847054
```

To fit the model, we will use a modified version of the main function, which we denote as IBCF.Years(), which receives the dataset in matrix format as a parameter and the years and traits to be used as the testing set.

```
pm4 <- IBCF.Years(Dataset, Years.testing = c('2015',
'2016'), Traits.testing = c('T5', 'T6'))
```

Once the model is fitted, a summary of the predictions can be retrieved through the summary() command, as detailed below.

```
summary(pm4)
##         Year_Trait Pearson              MSEP
## 2015_T5    2015_T5  0.8582            0.3525
## 2015_T6    2015_T6  0.8475            0.3245
## 2016_T5    2016_T5  0.7953            0.3525
## 2016_T6    2016_T6  0.8539            0.3245
```

Note that the function returns prediction accuracies for year–trait combinations under Pearson's correlation and MSEP. Other components that this object contains can be observed by using the str(pm4) command. This object of the IBCFY class contains a list with six values. In this case, it contains the 2 yr used for the testing set (2015 and 2016), as well as the traits in TST: T5 and T6. It also contains a data.frame object that contains the summary of the predictions, which is shown through the summary() function. The object also provides the $observed and $predicted values. Finally, we have another data.frame object that can be retrieved as $Data. Obs_Pred, which contains the original array, along with duplicate features that correspond to the predicted values. Columns that have been added with the suffix ".1" represent the predictions made by the model for each of the traits in TST.

Next, we show the first six observations for the testing of year 2015 with the observed predicted values that return the algorithm.

```
head(pm4$Data_Obs_Pred)
##    Years Gids       T5       T6     T5.1     T6.1
## 21  2015   21 8.181751 6.912622 7.529652 7.007596
## 22  2015   22 6.865868 6.388473 7.163454 6.574511
## 23  2015   23 7.072818 6.282973 6.575581 6.110108
## 24  2015   24 9.529347 8.118168 8.462467 7.813604
## 25  2015   25 6.741211 6.101720 6.891962 6.325226
## 26  2015   26 7.693102 7.158682 7.329256 6.704499
```

Note that Columns T5 and T6 are the observed values of the original dataset for these traits, whereas Columns T5.1 and T6.1 are the predicted values generated by the model for these traits. We can also make a plot with the summary of the predictions in terms of Pearson's correlation (or MSEP) for the year–trait combinations, as shown in Fig. 4a. However, there are now no estimates of the SE because there was no resampling to form the TRN and TST dataset. This plot is obtained with the barplot() function:

```
barplot(pm4, select = 'Pearson')
```

Figure 4a shows that the worst predictions in terms of Pearson's correlation are observed in Trait T5 for 2016, whereas similar predictions are obtained in the other year–trait combinations.

The "select" parameter can also be modified to obtain the predictions in terms of MSEP, as shown in Fig. 4b, using the following code: barplot(pm4, select = 'MSEP'). Figure 4b shows that the best predictions in terms of MSEP are observed for Trait T6 in all the years under study, whereas the worst predictions are for Trait T5.

Example 6: Predictions for 2016 with 2014 and 2015 as Training

This example shows how to make predictions with the IBCF package when you have several traits under study and you want to predict Traits T5 and T6 in 2016 with 2 yr (2014 and 2015) as training but with information on the remaining traits in all the years that form the TRN and TST sets. To do this, we used the same dataset that was used in the previous example. To adjust the model, we will use the IBCF.Years() function.

```
pm5 <- IBCF.Years(Dataset , Years.testing = c('2016'),
Traits.testing = c('T5','T6'))
```

Once the model is adjusted, a summary of the predictions can be obtained through the summary() command, as detailed below.

```
summary(pm5)
##         Year_Trait Pearson   MSEP
## 2016_T5    2016_T5  0.7547 0.3577
## 2016_T6    2016_T6  0.8551 0.2623
```

Finally, we obtain Fig. 5b through the command: barplot(pm5, select = 'Pearson'). Figure 5a shows that the best predictions in terms of Pearson's correlation are observed for Trait T6 in 2016 under study, whereas the worst predictions are observed for Trait T5 also for 2015.

Example 7: Predictions of Drip Environment with Three Environments as Training

As in the previous examples, we used the first dataset for the IBCF() function (Wheat_IBCF). However, we are now interested in predicting the information of the whole environment with the remaining three environments used as TRN. To adjust the model, we will use the IBCF.Years() function again.

```
pm6 <- IBCF.Years(Dataset, colYears = "Env", Years.
testing = 'Drip', Traits.testing = c('DH','GY'))
```

The vector Years.testing specifies the environment to be used in the validation sample. In this dataset, we have information from three environments and, given that the Drip environment is being used for the validation sample, this implies that the information on the remaining environments will be used to train the model. However, in the environment that makes up TST, only the traits specified in the vector traits are missing, which, in this case, are the traits DH and GY only. Therefore, the predictive ability will be studied only for these two traits.

```
summary(pm6)
##         Year_Trait Pearson    MSEP
## Drip_DH    Drip_DH  0.6576 25.0166
## Drip_GY    Drip_GY  0.4617  0.3211
```

We can use the barplot() function to obtain a bar plot of the summary of the predictions, as shown in

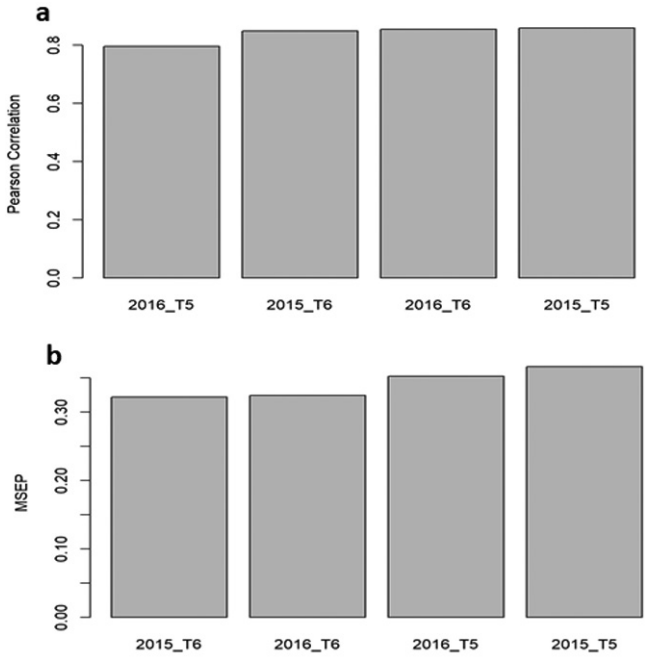

Fig. 4. Prediction accuracy in terms of (a) Pearson's correlation and (b) mean square error of prediction (MSEP) of the item-based collaborative filtering (IBCF) model for each year–trait combination. The data correspond to the dataset "Year_dataset".

the following code: barplot(pm4, select = 'Pearson'). Figure 5b shows that the best predictions in terms of Pearson´s correlation were observed for DH traits for the Drip environment, whereas the worst predictions were observed for GY traits in the Drip environment.

Example 8: Predictions of Breeding Values Taking Marker or Genomic Information into Account

This example was done with the purpose of showing how to use the IBCF.MTME package for making predictions of breeding values taking marker information or pedigree information into account. Since the IBCF.MTME package does not allow the direct incorporation of genomic (marker data) or pedigree information, to take this information into account, we need to do this in two steps. In the first step, we need to adjust the phenotypic data for the markers via a simple regression model (phenotype $\sim$ *Markers* + *error*). For illustration purposes, we used a wheat dataset containing 599 lines that belong to the Bayesian Generalized Linear Regression package (de los Campos and Pérez-Rodríguez, 2014). This dataset has phenotypic information for four traits (T1, …, T4) and information on 1279 markers (in binary format). To load this dataset, we used the following code:

```
library(BGLR)
data(wheat)
```

We then extracted the marker matrix and the responses of the traits, with the following code:

```
Markers <- wheat.X
Wheat <- wheat.Y
```

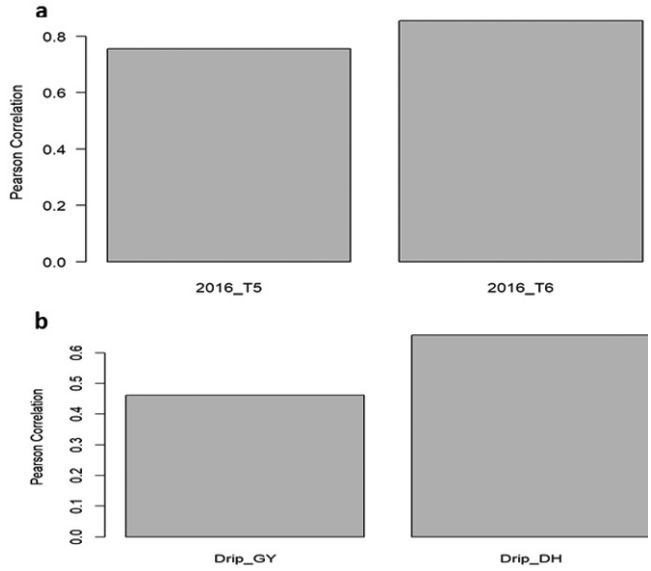

Fig. 5. Prediction accuracy in terms of Pearson's correlation of the item-based collaborative filtering (IBCF) model for (a) each year–trait combination for the data used that corresponds to the dataset for the year and (b) for each trait–environment combination in the testing set with the Wheat_dataset.

After this, we transformed the wheat dataset from matrix form to Tidy Data form with the IBCF.MTME package

```
Wheat <- data.frame(rownames(Wheat), Wheat)
colnames(Wheat) <- c('GID', 'T1_', 'T2_', 'T3_', 'T4_')
Data_Wheat <- getTidyForm(Wheat)
```

To adjust for the markers each of the four traits with the model above (phenotype $\sim$ *Markers* + *error*), we used the BGLR function of the BGLR package as follows:

```
ETA <- list(Marker = list(X = Markers, model = 'BRR'))

FM_T1 <- BGLR(Data_Wheat$Response[Data_Wheat$Trait
== 'T1'], ETA = ETA, nIter = 20000, burnIn = 15000,
verbose = FALSE)

FM_T2 <- BGLR(Data_Wheat$Response[Data_Wheat$Trait
== 'T2'], ETA = ETA, nIter = 20000, burnIn = 15000,
verbose = FALSE)

FM_T3 <- BGLR(Data_Wheat$Response[Data_Wheat$Trait
== 'T3'], ETA = ETA, nIter = 20000, burnIn = 15000,
verbose = FALSE)

FM_T4 <- BGLR(Data_Wheat$Response[Data_Wheat$Trait
== 'T4'], ETA = ETA, nIter = 20000, burnIn = 15000,
verbose = FALSE)
```

Four models were adjusted, one for each trait. With the fitted values, we generated a new dataset with the genomic breeding values, which we will call "breeding values" for short.

```
BreedingValue <- data.frame(GID = rownames(Wheat),
T1_ = FM_T1$yHat, T2_ = FM_T2$yHat, T3_ = FM_T3$yHat,
T4_ = FM_T4$yHat)
```

Next, we transformed these breeding values into Tidy Data form:

```
library(IBCF.MTME)
Breeding_DS <- getTidyForm(BreedingValue)
```

Finally, we implemented a cross-validation for 10 random partitions with 20% of the 599 lines as TST and 80% as TRN; with these partitions, we used the IBCF.MTME function to predict the breeding values of the missing lines as follows:

```
Breeding_DS_Missing <- CV.RandomPart(Breeding_DS,
NPartitions = 10, PTesting = 0.20)

PM_Breeding <- IBCF(Breeding_DS_Missing)
summary(PM_Breeding)
##   Trait_Env Pearson SE_Cor   MSEP SE_MSEP
## 1       T1_ -0.2440 0.0259 0.6388  0.0183
## 2       T2_  0.8222 0.0097 0.1015  0.0033
## 3       T3_  0.8570 0.0080 0.0608  0.0027
## 4       T4_  0.5278 0.0127 0.2257  0.0089
```

To compare the prediction accuracies without markers (working directly with the phenotypes), we obtained the predictions with the data without the markers by using the phenotypic information directly.

```
Response_DS_Missing <- CV.RandomPart(Data_Wheat,
NPartitions = 10, PTesting = 0.20)

PM_Response <- IBCF(Response_DS_Missing)
summary(PM_Response)
##   Trait_Env Pearson SE_Cor   MSEP SE_MSEP
## 1       T1_ -0.0823 0.0256 1.7856  0.0674
## 2       T2_  0.6683 0.0110 0.5414  0.0171
## 3       T3_  0.6378 0.0195 0.6218  0.0342
## 4       T4_  0.3473 0.0193 0.9490  0.0309
```

The best predictions were observed when the breeding values were used instead of the phenotypic data.

DISCUSSION

The package we developed, IBCF.MTME, enables the study of the prediction accuracies of datasets with continuous phenotypes in the context of multitrait and multienvironment data. The package offers many useful tools that help breeders perform studies of prediction accuracy more easily. Some of these tools provide: (i) an easy way to transform a dataset from the matrix format to the Tidy Data format, (ii) a way to create training and testing datasets with the function CV.RandomPart(), (iii) a summary of the prediction accuracies for the trait–environment combinations or the year–trait combinations that are given with numerical values or with plots for both metrics implemented (Pearson's correlation and the MSEP), (iv) a way to extract the observed and predicted values for the information in the testing set for each partition (or a summary of all partitions) implemented for further processing, and (v) a way to study the prediction accuracy for the information of some traits (or environments) missing in all the lines in some years (or environments) but available in other years (or environments). For these reasons, we believe that this package can help breeders become efficient in the decision-making process.

We provide some examples for transforming the whole dataset at hand from the matrix format to the Tidy Data format and vice versa. We believe this will simplify work for breeders, since they frequently need to transform their datasets from one format to the other. In six out of seven examples, we tried to cover scenarios that may be of interest to breeders. The first scenario involved prediction accuracies when breeders are interested in making predictions and the data set has only a single environment and many traits. The second scenario, on the other hand, studied prediction accuracies for datasets with multiple environments and traits, while the third scenario examined prediction accuracies with datasets with multiple traits and environments but with only two traits as TST. The fourth scenario includes datasets with information from several years when the breeder is interested in making predictions for certain traits of some the lines in 2 yr using information from only 1 yr as the TRN. The fifth scenario is similar to the previous one, but now the predictions are only for 1 yr, using 2 yr as TRN. Finally, the sixth scenario can be useful when breeders are interested in predicting the information of a whole environment with the information of three other environments as TRN.

The main advantage of the IBCF algorithm over the existing statistical models for studying prediction accuracies in multitrait and multienvironment data is that in addition to providing competitive predictions, IBCF can be implemented with reasonably large datasets, as the time required for its implementation is minimal for small and moderately large datasets. As such, we believe that this algorithm can be very useful in the context of plant breeding and that with the help of the proposed package, breeders can implement it more easily. However, a disadvantage of the proposed method is that we cannot directly incorporate pedigree or marker information in the model. For this reason, as shown in most of the examples (except Example 8), it can only be used for phenotypic selection directly. Although this often improves prediction accuracy in genomic mixed models, when the correlation between the trait–environment combination is reasonably high, the predictive power of this method is good.

However, as shown in Example 8, it is possible to include genomic information (markers) or pedigree information. However, to be able to take this information into account, a two-step process is required: in the first step, the breeding values are obtained by adjusting the phenotypes for markers; in the second step, the breeding values for the missed values of some lines are predicted. It is important to point out that at CIMMYT in some wheat datasets, the IBCF.MTME algorithm has been implemented for predicting breeding values from this two-step process. The IBCF.

MTME algorithm has also been implemented for predicting GY in maize from correlated covariates resulting from high-throughput phenotyping datasets collected by high-resolution imaging and environmental sensors. With these real applications, we found that the IBCF.MTME algorithm competes well with the conventional genomic mixed models, with the advantage that its implementation is not computationally expensive. However, more empirical evidence is required to get a clear picture of the performance of the IBCF.MTME approach.

It is important to point out that the IBCF.MTME method is not a model-based approach and therefore cannot estimate genetic variances or variance components in general; however, it can be used as a new alternative for making predictions of phenotypic or genetic information in the context of GS, as shown here with the seven examples. We believe that it can also be used for imputation of phenotypic or genetic information when there is correlated information available. All these tasks can be implemented in the proposed IBCF.MTME R package.

Finally, we believe that there are opportunities to improve this algorithm (or similar algorithms like the matrix factorization algorithm) to directly take genetic and pedigree information into account, which would be really helpful, since it would allow us to take advantage of all the information that breeders are generating. In addition, we believe that we need to be willing to test other prediction techniques that are being developed in other fields and that could be useful for genomic prediction. For this reason, we encourage people to do research on this topic because they could make a significant contribution to improving the prediction accuracy in breeding programs.

CONCLUSIONS

The package we developed provides the user with a framework for studying the prediction accuracies of multitrait and multienvironment data, which are very common in plant breeding. The developed examples cover different scenarios that could be of interest to plant breeders: (i) assessing the prediction accuracy of multitrait and multienvironment data and (ii) the prediction accuracy of year (or environment) from information from one or more years (or environments) as training. Additionally, the package provides a very user-friendly framework for transforming a dataset to a matrix format, constructing random cross-validation scenarios that could be of interest to plant breeders, and studying the prediction abilities of trait–environment combinations or year–trait combinations of interest to plant breeders. In addition, the package can plot a summary of prediction accuracies (Pearson's correlation and MSEP) for the trait–environment or year–trait combinations that would facilitate the decision-making processes of plant breeders.

Conflict of Interest Disclosure

The authors declare that there is no conflict of interest.

APPENDIX A: DATA SIMULATION

The code used for simulating the dataset Year_IBCF is shown below.

```
library(IBCF.MTME)

library(mvtnorm)

set.seed(2)
A <- matrix(0.65,ncol=12,nrow=12)
diag(A) <- 1
Sdv <- diag
(c(0.9^0.5,0.8^0.5,0.9^0.5,0.8^0.5,0.86^0.5,0.7^0.5,0.9
^0.5,0.8^0.5,0.9^0.5,0.7^0.5,0.7^0.5,0.85^0.5))

Sigma <- Sdv%*%A%*%Sdv
No.Lines <- 60
Z <- rmvnorm(No.Lines,mean=c(5,5.5,6,5.5,7,6.5,6.0,7,
6.6,8,6.3,8),sigma=Sigma)
Years <- c(rep(2014,20),rep(2015,20),rep(2016,20))
Gids <- c(1:No.Lines)

Data.Final <- data.frame(cbind(Years,Gids,Z))
colnames(Data.Final) <- c("Years","Gids","T1","T2",
"T3","T4","T5","T6","T7","T8","T9","T10","T11","T12")
head(Data.Final)
Year_IBCF <- getTidyForm(Data.Final, onlyTrait = T)
```

REFERENCES

Asoro, F.G., M.A. Newell, W.D. Beavis, M.P. Scott, N.A. Tinker, and J.-L. Jannink. 2013. Genomic, marker-assisted, and pedigree–BLUP selection methods for β-glucan concentration in elite oat. Crop Sci. 53(5):1894–1906. doi:10.2135/cropsci2012.09.0526

Bernardo, R., and J.M. Yu. 2007. Prospects for genome-wide selection for quantitative traits in maize. Crop Sci. 47:1082–1090. doi:10.2135/cropsci2006.11.0690

Beyene, Y., K. Semagn, S.M. Mugo, A. Tarekegne, R. Babu, B. Meisel, et al. 2015. Genetic gains in grain yield through genomic selection in eight biparental maize populations under drought stress. Crop Sci. 55:154–163. doi:10.2135/cropsci2014.07.0460.

Crossa, J., P. Pérez-Rodríguez, J. Cuevas, O.A. Montesinos-López, D. Jarquín, G. de Los Campos, et al. 2017. Genomic selection in plant breeding: Methods, models, and perspectives. Trends Plant Sci. 22(11):961–975. doi:10.1016/j.tplants.2017.08.011

Combs, E., and R. Bernardo. 2013. Genome-wide selection to introgress semidwarf corn germplasm into U.S. Corn Belt inbreds. Crop Sci. 53:1427–1436. doi:10.2135/cropsci2012.11.0666

de los Campos, G., and P. Pérez-Rodríguez. 2014. Bayesian generalized linear regression. R package version 1.0.4. The R Foundation. http://CRAN.R-project.org/package=BGLR (accessed 16 July 2018).

Food and Agriculture Organization of the United Nations. 2009. FAO's Director-General on how to feed the world in 2050. Popul. Dev. Rev. 35:837–839. doi:10.1111/j.1728-4457.2009.00312.x

Godfray, H.C.J., J.R. Beddington, I.R. Crute, L. Haddad, D. Lawrence, J.F. Muir, et al. 2010. Food security: The challenge of feeding 9 billion people. Science 327:812–818. doi:10.1126/science.1185383

Heffner, E.L., M.E. Sorrells, and J.-L. Jannink. 2009. Genomic selection for crop improvement. Crop Sci. 49(1):1–12. doi:10.2135/cropsci2008.08.0512

Heffner, E.L., A.J. Lorenz, J.-L. Jannink, and M.E. Sorrells. 2010. Plant breeding with genomic selection: Gain per unit time and cost. Crop Sci. 50(5):1681–1690. doi:10.2135/cropsci2009.11.0662

Linden, G., B. Smith, and J. York. (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput.. 7(1):76–80. doi:10.1109/MIC.2003.1167344.

López-Cruz, M., J. Crossa, D. Bonnett, S. Dreisigacker, J. Poland, J.-L. Jannink, et al. (2015) Increased prediction accuracy in wheat breeding trials using a marker × environment interaction genomic selection model. G3 (Bethesda) 5(4):569–582. doi:10.1534/g3.114.016097.

Lorenzana, R.E., and R. Bernardo. 2009. Accuracy of genotypic value predictions for marker-based selection in biparental plant populations. Theor. Appl. Genet. 120:151–161. doi:10.1007/s00122-009-1166-3

Lourenco, D.A.L., S. Tsuruta, B.O. Fragomeni, Y. Masuda, I. Aguilar, A. Legarra, et al 2015. Genetic evaluation using single-step genomic best linear unbiased predictor in American Angus. J. Anim. Sci. 93(6):2653–2662. doi:10.2527/jas.2014-8836

Massman, J.M., H.-J.G. Jung, and R. Bernardo. 2013. Genome-wide selection versus marker-assisted recurrent selection to improve grain yield and stover-quality traits for cellulosic ethanol in maize. Crop Sci. 53:58–66. doi:10.2135/cropsci2012.02.0112

Meuwissen, T.H., B.J. Hayes, M.E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. Genetics 157:1819–1829.

Montesinos-López, O.A., A.Montesinos-López, J. Crossa, F. Toledo, O.Pérez-Hernández, K.M. Eskridge, et al. 2016. A genomic Bayesian multi-trait and multi-environment model. G3 (Bethesda): 6(9):2725–2744.

Montesinos-López, O. A., A. Montesinos-López, J. Crossa, F.H. Toledo, J.C. Montesinos-López, P. Singh, et al. 2017. A Bayesian Poisson-lognormal model for count data for multiple-trait multiple-environment genomic-enabled prediction. G3 (Bethesda) 7(5):1595–1606. doi:10.1534/g3.117.039974.

Montesinos-López, O.A., A. Montesinos-López, J. Crossa, J.C. Montesinos-López, D. Mota-Sanchez, F. Estrada-González, et al. 2018. Prediction of multiple-trait and multiple-environment genomic data using recommender systems. G3 (Bethedsa) 8(1):131–147.

R Core Team. (2018) R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://www.R-project.org/ (accessed 17 July 2018).

Rutkoski, J., R.P. Singh, J. Huerta-Espino, S. Bhavani, J. Poland, J.L. Jannink, et al. 2015. Genetic gain from phenotypic and genomic selection for quantitative resistance to stem rust of wheat. Plant Genome 8(2):1–10. doi:10.3835/plantgenome2014.10.0074

Sarwar, B., G. Karypis, J. Konstan, and J. Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In: V.Y. Shen, N. Saito, M.R. Lyu, M.E. Zurko, editors, Proceedings of the 10th International Conference on the World Wide Web, Hong Kong. 1–5 May 2001. ACM, New York. p. 285–295.

Wickham, H. 2014. Tidy data. J. Stat. Softw. 59(10):1–22. doi:10.18637/jss.v059.i10