CBCS
HTTP://AGROBIOL.SGGW.WAW.PL/CBCS

COMMUNICATIONS IN BIOMETRY AND CROP SCIENCE
VOL. 8, NO. 2, 2013, PP. 60–69

INTERNATIONAL JOURNAL OF THE FACULTY OF AGRICULTURE AND BIOLOGY,
WARSAW UNIVERSITY OF LIFE SCIENCES – SGGW, POLAND

SOFTWARE TRICKS AND TIPS

An R function for imputation of missing cells in two-way data sets by EM-AMMI algorithm

Jakub Paderewski

Department of Experimental Design and Bioinformatics, Warsaw University of Life Sciences – SGGW, Nowoursynowska 159, 02-776 Warsaw, Poland.
Jakub Paderewski; E-mail: j.paderewski@omega.sggw.waw.pl

CITATION: Paderewski, J. (2013). An R function for imputation of missing cells in two-way data sets by EM-AMMI algorithm. *Communications in Biometry and Crop Science* 8 (2), 60–69.

Received: 23 April 2014, Accepted: 26 May 2014, Published online: 11 June 2014

ABSTRACT

Various statistical methods for two-way classification data sets (including AMMI or GGE analyses, used in crop science for interpreting genotype-by-environment interaction) require the data to be complete, that is, not to have missing cells. If there are such, however, one might impute the missing cells. The paper offers R code for imputing missing values by the EM-AMMI algorithm. In addition, a function to check the repeatability of this algorithm is proposed. This function could be used to evaluate if the missing data were imputed reliably (unambiguously), which is important especially for small data sets.

Key Words: *agricultural trials; EM-AMMI; missing cells; multivariate statistics; Singular Value Decomposition; software.*

INTRODUCTION

Multivariate data are very common in crop science. One of the most common scenarios is a two-way classification, an example being yield of genotypes (G) grown in a number of environments (E). In such trials, genotype-by-environment (GE) interaction is often of interest. The GE classification data can be analyzed by AMMI analysis (Gauch, 1992), which is based upon the following model:

$$\bar{y}_{ij} = m + g_i + e_j + \sum_{t=1}^{T} \lambda_t u_{ti} v_{tj} + \bar{\varepsilon}_{ij}$$

where $\bar{y}_{ij}$ is the mean value of a trait from n replications for the combination of the i^{th} (i=1,..,I) genotype with the j^{th} (j=1,..,J) environment, m is the grand mean, g_i is the main effect of the i^{th} genotype, e_j is the main effect of the j^{th} environment and $\bar{\varepsilon}_{ij}$ is the mean experimental error for the mean $\bar{y}_{ij}$. Coefficients λ_t, u_{ti}, v_{tj} are determined by singular value decomposition (SVD) of the matrix of interactions effects. The λ_t is the singular value

for the t^{th} principal component in non-ascending order ($\lambda_1 \geq \ldots \geq \lambda_T$). The eigenvector $\mathbf{u_t}$ (u_{t1}, ... , u_{tI}) corresponding to eigenvalue λ_t^2 contains the genotypic parameters of the t^{th} (t=1,..,T) interaction principal component, while the eigenvector $\mathbf{v_t}$ (v_{t1}, ... , v_{tJ}) corresponding to eigenvalue λ_t^2 contains the environmental parameters of the t^{th} (t=1,..,T) interaction principal component.

The AMMI model is considered when the respective values in rows (in the GE classification, values in rows commonly correspond to genotypes yields) of the data set are associated (Jackson, 1993) with some other rows. An example is when there are genotypes with a similar response to environmental conditions. Symmetrically, the values in columns (representing environments) should be linked with respective values in some other columns. In the GE classification this scenario represents a situation in which the respective environments influence yield in a similar way. The appropriate number of principal components can be found by (i) F_R test (Cornelius, 1993); (ii) the minimum of the Root Mean Square Predictive Difference (RMSPD, Gauch and Zobel, 1990; Dias and Krzanowski, 2003); or (iii) some heuristics methods (Jackson, 1993) developed in the `nScree` function from `nFactors` package (Raiche, 2010)

As a graphical data analysis tool, biplot analysis (based on the SVD procedure) is useful to make interpretation of interaction effects, for example genotype-by-environment interaction of crop yield. The limitation of some methods that are used to analyze agronomy trials, for example PCA, GGE and AMMI (Gauch, 1992; Yan and Kang, 2003; Kaya et al., 2006; Paderewski et al., 2011) analysis (all of which base on the SVD procedure), is that they require a complete two-way table. This not being the case, a possible solution is extracting from the whole data set a balanced subset by deleting the columns (environments) with missing values for some of the genotypes and/or rows (genotypes) with missing values for some of the environments. An obvious disadvantage of this solution is that it removes all the information about the removed items (genotypes or environments) from the analysis and subsequent interpretation. Alternatively, one can fill in (estimate) the missing values in the data set and conduct the analysis for such an imputed data set. This paper offers an R code that will help users to impute the missing cells for two-way data by the Expectation-Maximization AMMI (EM-AMMI) algorithm.

Imputation of missing data by EM-AMMI can also be helpful to rank genotypes based on their yielding in a particular environment (location) or in a subset of environments (in a so-called mega-environment). Similarly, genotypes can be ranked in multi-year trials even if the genotypes did not occur in all years of the trial, thereby making the trial unbalanced. Such a scenario is actually often the case because inferior genotypes are dropped and new ones are added each year.

In replicated trials it is possible that all values for some GE combinations are missing. The EM-AMMI algorithm (Gauch and Zobel, 1990; Gauch, 1992, 2007) can be used to impute these values. This paper presents the EM.AMMI function in R language (R Development Core Team, 2013) to fill in the missing values. The function can be easily used, especially before the AMMI or GGE (Genotype and Genotype by Environment interaction method, Yan and Kang, 2003) analysis will be used. R enables one to conduct the AMMI analysis (Onofri and Ciriciofolo, 2007; `agricolae` package, de Mendiburu 2012). The author of this paper is not aware of any R functions or add-on packages with the EM-AMMI procedure.

Thus, the aim of this paper is to present an R function that enables one to use the EM-AMMI procedure to impute missing cells in data from genotype-by-environment trials.

EM-AMMI

The EM-AMMI method completes a data set with missing values according to both main and interaction effects. The algorithm works as follows (Gauch and Zobel, 1990; Gauch, 1992):

1. The user can provide initial values for the missing cells (as the argument of the `EM-AMMI` function). Otherwise, the initial values are calculated as the grand mean increased by main effects of rows and main effects of columns. That way, the matrix of observations is pre-filled in.
2. The parameters of the AMMI model are estimated.
3. The adjusted means are calculated based on the AMMI model with *n* principal components.
4. The missing cells are filled with the adjusted means.
5. If the maximum change in these values (the Chebyshev distance between the missing value estimations in the two successive iteration steps) is greater than the assumed precision, the steps 2 through 5 are repeated. Otherwise, the algorithm stops.

`EM.AMMI()` function

The `EM.AMMI()` function fills in missing cells in a two-way table according to the EM-AMMI method. Its results are reliable when there are relationships between genotype yields and between environments. The user should decide on the number of principal components (*T* in the equation of the AMMI model [1]) used to estimate the missing values.

Usage:

```
EM.AMMI(X, PC.nb=NA, initial.values=NA, precision=0.01, max.iter=1000,
     change.factor=1, simplified.model=FALSE)
```

Arguments:

`X` – the matrix of observed values with NA value in missing cells; the data set could represent the averages for the two-way genotype-by-environment classification, that is, the two-way table of genotype means in each environment;

`PC.nb` – (optional) the number of principal components in the AMMI model that will be used; the default value is 1. For `PC.nb=0` only main effects are used to estimate cells in the data table (the interaction is ignored). The number of principal components must not be greater than min(number of rows in the X table, number of columns in the X table)–2. The number of principal components can be found according to the minimum of the Root Mean Square Predictive Difference (RMSPD, Gauch and Zobel, 1990; Dias and Krzanowski, 2003) or some heuristic indexes (Jackson, 1993).

`initial.values` – (optional) the initial values of the missing cells. It can be a single value, which then will be used for all empty cells, or a vector of length equal to the number of missing cells (starting from the missing values in the first column). If omitted, the initial values will be obtained by the main effects from the corresponding model, that is, by the grand mean of the observed data increased (or decreased) by row and column main effects.

`precision` – (optional) the algorithm converges if the maximal change in the values of the missing cells in two subsequent steps is not greater than this value (the default is 0.01);

`max.iter` – (optional) a maximum permissible number of iterations (that is, number of repeats of the algorithm's steps 2 through 5); the default value is 1000;

`change.factor` – (optional) introduced by analogy to step size in gradient descent method, this parameter that can shorten the time of executing the algorithm by decreasing the number of iterations. The `change.factor=1` (default) defines that the previous approximation is changed with the new values of missing cells (standard EM-AMMI algorithm). However, when `change.factor<1`, then the new approximations are computed and the values of missing cells are changed in the direction of this new approximation but

the change is smaller. It could be useful if the changes are cyclic and thus convergence could not be reached. Usually, this argument should not affect the final outcome (that is, the imputed values) as compared to the default value of `change.factor=1`.
`simplified.model` – the AMMI model contains the general mean, effects of rows, columns and interaction terms. So the EM-AMMI algorithm in step 2 calculates the current effects of rows and columns; these effects change from iteration to iteration because the empty (at the outset) cells in each iteration are filled with different values. In step 3 EM-AMMI uses those effects to re-estimate cells marked as missed (as default, `simplified.model=FALSE`). It is, however, possible that this procedure will not converge. Thus the user is offered a simplified EM-AMMI procedure that calculates the general mean and effects of rows and columns only in the first iteration and in next iterations uses these values (`simplified.model=TRUE`). In this simplified procedure the initial values affect the outcome (whilst EM-AMMI results usually do not depend on initial values). For the simplified procedure the number of iterations to convergence is usually smaller and, furthermore, convergence will be reached even in some cases where the regular procedure fails. If the regular procedure does not converge for the standard initial values (see the description of the argument `initial.values`), the simplified model can be used to determine a better set of initial values.

Value: The outcome of the function, provided as a list:
1) `X`: the imputed matrix (filled in with the missing values estimated by the EM-AMMI procedure);
2) `PC.SS`: the sum of squares representing variation explained by the principal components (the squares of eigenvalues of singular value decomposition);
3) `iteration`: the final number of iterations;
4) `precision.final`: the maximum change of the estimated values for missing cells in the last step of iteration (the precision of convergence). If the algorithm converged, this value is slightly smaller than the argument `precision`;
5) `PC.nb.final`: a number of principal components that were eventually used by the `EM.AMMI()` function. The function checks if there are too many missing cells to unambiguously compute the parameters by the SVD decomposition (Gauch and Zobel, 1990). In that case the final number of principal components used is smaller than that which was passed on to the function through the argument `PC.nb`;
6) `convergence`: the value TRUE means that the algorithm converged in the last iteration.

Description of **`repeat.EM.AMMI()`**
Aiming to check if the outcome of the `EM.AMMI()` function is reliable, the function `repeat.EM.AMMI()` repeats the EM-AMMI procedure `Nb` times, each time with a different initial values. The reliability of the imputed values tends to be higher for larger data sets (Leek, 2011), stronger relationships between rows, stronger relationships between columns, and fewer missing values to be imputed.

Usage:
```
repeat.EM.AMMI(X, Nb, PC.nb=NA, precision=0.01, max.iter=1000,
     simplified.model=FALSE)
```

Arguments:
`X`, `max.iter`, `simplified.model`, `precision` – the same as in the `EM.AMMI()` function;
`PC.nb` – a vector of number of principal components that will be used to compute the EM-AMMI algorithm;
`Nb` – a number of repetitions of the EM-AMMI algorithm with the same number of principal components.

Value: The outcome of the function `repeat.EM.AMMI()` is a list with two 3-way arrays. The first 3-way array (named '`estimated`') contains the imputed values of the missing cells. The second array (named '`parameters`') contains the parameters that describe the run of EM.AMMI procedure, such as sum of squares for the principal components ('`PC SS`'), number of iterations used by the `EM.AMMI` procedure ('`Iterations nb`'), and the final precision (in the last step of iteration) ('`precision.final`'). The **first dimension** of both 3-way arrays is associated with the index of repetition (the indices in this dimension are from 1 to `Nb`). The first repetition is initialized by the main effects and the subsequent ones are initialized by random variables obtained by random sampling from a normal distribution with mean and standard deviation equal to the mean and standard deviation of all the observed values in the original data set X. The **second dimension** of the array '`estimated`' is related to individual missing cells. The array contains the set of the imputed values so the maximum index in this dimension is the number of missing values. The second dimension of the array '`parameters`' contains: the sums of squares of the principal components, the number of iterations, and the obtained precision. The **third dimension** is associated with the number of principal components. The maximum index is the length of vector `PC.nb`, that is, the number of the different EM-AMMI models fitted (each model being fit with a different number of principal components).

Description of **`CV.LOO()`**

To select the appropriate number of principal component, Root Mean Square Predictive Difference (RMSPD) can be used (Gauch and Zobel, 1990; Dias and Krzanowski, 2003). The optimal number of principal components is that which has the smallest RMSPD value. To decide on this optimal number, the leave-one-out cross-validation procedure can be employed. Shortly, from the original data set a single observation is hidden before running the EM-AMMI; it will be used as the validation data set. The imputation is done by the EM-AMMI procedure based on the training data set, which is the data set without this single observation and without all the originally missing values. Such a procedure is repeated for each observation in the sample (so the procedure is repeated as many times as there are non-empty cells in the data set). The differences between the hidden value and that which is imputed by EM-AMMI (the 'predictive differences') are squared, averaged, and square-rooted.

Usage:

```
CV.LOO<-function(X,...,PC.nb=0:2,MNO=3)
```

Arguments:

`X`, … – the arguments passed to the `EM.AMMI()` function;
`PC.nb` – a vector of numbers of principal components that will be used to run the EM-AMMI procedure;
`MNO` – a permissible minimum number of observed values in each row and each column of X matrix. If the training data set has fewer values in any of the rows or column than `MNO`, then such a training data set and the corresponding validation set will not be used for validation purposes.
Value: The outcome of the function `CV.LOO()` is a list with (i) RMSPD values (`RMSPD`) and (ii) the predictive differences table (`differences`).

Functions in R

```
EM.AMMI<-function(X, PC.nb=1, initial.values=NA, precision=0.01,
max.iter=1000, change.factor=1, simplified.model=FALSE)
{
```

```
  X<-as.matrix(X)
  X.missing<-matrix(1,nrow(X),ncol(X))
  X.missing[is.na(X)]<-0
  max.IPC=min(c(rowSums(X.missing),colSums(X.missing)))-1
  if (max.IPC<PC.nb)
    {PC.nb.used<-max.IPC} else {PC.nb.used<-PC.nb}
  X.ini<-X
  if (length(initial.values)==1)
  {
    if (!is.na(initial.values))
    {
      initial.values<-matrix(initial.values,nrow(X),ncol(X))
      X.ini[is.na(X.ini)]<-initial.values[is.na(X)]
    } else {
      X.mean<-mean(c(X),na.rm = TRUE)
      row.m<-matrix(rowMeans(X,na.rm = TRUE),nrow(X),ncol(X))
      col.m<-t(matrix(colMeans(X,na.rm = TRUE),ncol(X),nrow(X)))
      estimated<-(-X.mean)+row.m+col.m
      X.ini[is.na(X.ini)]<-estimated[is.na(X)]
    }
  } else {
    X.ini[is.na(X.ini)]<-initial.values[is.na(X)]
  }
  iteration<-1
  X.new<-X.ini
  change<-precision+1
  while ((change>precision)&(iteration<max.iter))
  {
    if (iteration==1 | !simplified.model)
    {
      x.mean<-mean(X.new)
      X.new.Ie<-X.new-x.mean
      X.new.Ie<-scale(X.new.Ie,center = TRUE, scale = FALSE)
      x.col.center<-attr(X.new.Ie,"scaled:center")
      X.new.Ie<-t(scale(t(X.new.Ie),center = TRUE, scale = FALSE))
      x.row.center<-attr(X.new.Ie,"scaled:center")
    } else { X.new.Ie<-X.new-x.mean-row.eff-col.eff }
    if (PC.nb.used>=1)
    {
      SVD <- La.svd(X.new.Ie)
      SVD$d<-SVD$d[1:PC.nb.used]
      SVD$u<-SVD$u[,1:PC.nb.used]
      SVD$v<-SVD$v[1:PC.nb.used,]
      diag.l<-diag(SVD$d,nrow=PC.nb.used)
      interaction.adj<-SVD$u%*%diag.l%*%SVD$v
    } else interaction.adj<-0
    if (iteration==1 | !simplified.model)
    {
      row.eff<-matrix(x.row.center,nrow(X),ncol(X))
      col.eff<-t(matrix(x.col.center,ncol(X),nrow(X)))
    }
    X.next<-X.new
    X.next[is.na(X)]<-(x.mean+row.eff+col.eff+interaction.adj)[is.na(X)]
    change<-max(abs(c( (X.next-X.new)[is.na(X)] )))
    iteration<-iteration+1
    X.new<-change.factor*X.next+(1-change.factor)*X.new
  }
  if (change<=precision) {state=TRUE} else {state=FALSE}
  if (PC.nb.used<PC.nb) {state=FALSE}
  if (PC.nb.used==0) {SVD<-list(d=0)}
return(list(X=X.new,PC.SS=SVD$d^2,iteration=iteration,precision.final=chang
e,PC.nb.final=PC.nb.used, convergence=state))
}
```

```
repeat.EM.AMMI<-function(X, Nb, PC.nb=NA, precision=0.01, max.iter=1000,
simplified.model=FALSE)
{
  Y<-array(NA,c(Nb,sum(is.na(X)),length(PC.nb)))
  Y2<-array(NA,c(Nb,3,length(PC.nb)))
  dimnames(Y)[[3]]<-c(PC.nb,rep(NA,dim(Y)[3]-length(PC.nb)))
  dimnames(Y2)[[3]]<-c(PC.nb,rep(NA,dim(Y)[3]-length(PC.nb)))
  dimnames(Y2)[[2]]<-c("PC SS","Iterations nb","precision.final")
  PCn<-1
  while (PCn<=length(PC.nb))
  {
    temp<-EM.AMMI(X, PC.nb[PCn], initial.values=NA, precision,
max.iter=max.iter, simplified.model= simplified.model)
    if (temp$PC.nb.final==PC.nb[PCn])
    {
      Y[1,,PCn]<-temp$X[is.na(X)]
      Y2[1,,PCn]<-c(sum(temp$PC.SS ,na.rm=TRUE),
temp$iteration,temp$precision.final)
    } else {
      PC.nb<-PC.nb[-PCn]
      PCn<-PCn-1
    }
    PCn<-PCn+1
  }
  for (i in 2:Nb)
  {
    random.v<-rnorm(nrow(X)*ncol(X), mean(c(X), na.rm=TRUE), sd(c(X),
na.rm=TRUE))
    for (PCn in 1:length(PC.nb))
    {
      temp<-EM.AMMI(X, PC.nb[PCn], initial.values=random.v, precision,
max.iter=max.iter, simplified.model= simplified.model)
      Y[i,,PCn]<-temp$X[is.na(X)]
      Y2[i,,PCn]<-c(sum(temp$PC.SS ,na.rm=TRUE), temp$iteration,
temp$precision.final)
    }
  }
  Y[,,]<-round(Y[,,]/ precision)*precision
  return(list(estimated=Y,parameters=Y2))
}
CV.LOO<-function(X,...,PC.nb=0:2,MNO=3)
{
   PD<-matrix(NA,sum(!is.na(X)),length(PC.nb))
   dimnames(PD)[[2]]<-PC.nb
   PD.nb<-0
   for (j in 1:ncol(X))
   { for (i in 1:nrow(X))
   { if (!is.na(X[i,j]))
   {
       PD.nb<-PD.nb+1
       X.ij<-X[i,j]
       X[i,j]<-NA
       X.i<-sum(!is.na(X[i,]))
       X.j<-sum(!is.na(X[,j]))
       if ((X.i>=MNO)&(X.j>=MNO))
       {
         for (PC in 1:length(PC.nb))
         {
           temp<-EM.AMMI(X,PC.nb[PC],...)
           if ((temp$convergence)&(temp$PC.nb.final==PC.nb[PC]))
           { PD[PD.nb,PC]<-(temp$X[i,j]-X.ij) }
         }
       }
```

```
        X[i,j]<-X.ij
    }}}
    return(list(RMSPD=colSums(PD^2,na.rm=TRUE)^0.5,differences=PD))
}
```

Examples of use of the `EM.AMMI()` and `repeat.EM.AMMI()` functions.

Example 1.
Create a multiplication table:

```
tab.M<-(1:20)%*%t(1:10)
```

The positions of missing cells are:

```
missing<-cbind(c(2,4,6,8,10,2,4,6,8,10,19,20),
c(1,2,2,3,5,5,6,7,9,9,10,10))
tab.M[missing]<-NA
```

The multiplication table with the missing values is displayed by typing:

```
tab.M
```

The missing cells are imputed

```
result<-EM.AMMI(tab.M, 1, precision=0.001)
```

and rounded

```
round(result$X, digits=2)
```

Example 2.
Create a data set as follows and run EM-AMMI with one principal component:

```
x<-cbind( c(2,3,NA,5,6,7,8), c(2,4,6,8,10,NA,14), c(NA,2,3,4,NA,6,7),
c(3,NA,4,4,5,5,6))
EM.AMMI(x,1)
```

The result is the list that contains (i) the complemented matrix X; (ii) the sums of squares for the principal components used: `PC.SS = 36.909`; (iii) the final number of iterations needed: `iteration=49`; (iv) the convergence has been achieved, which is why the value of `precision.final=0.00942` is slightly smaller than the argument `precision` that was passed on to `EM.AMMI` procedure (it was not specified, so the default value of 0.01 was used); (v) the number of principal components used: `PC.nb=1`; (vi) `convergence=TRUE` because the convergence was obtained.

As mentioned above, the `change.factor` argument can be used to shorten the computation time. In this example, the idea is to compute the first 20 iterations with `change.factor=2`; the outcome obtained that way (`ini1`) will then be used to run the `EM-AMMI` function with `change.factor=1` to obtain the final result.

```
ini1<-EM.AMMI(x, 1, max.iter=20, change.factor=2)
result<-EM.AMMI(x, 1, max.iter=10, initial.values=ini1$X, change.factor=1)
result
```

Since the second run of the function needed only 9 iterations, the final number of iterations was 29 instead 49.

With the `repeat.EM.AMMI` function, the reliability of the AMMI models used in the EM-AMMI procedure with different numbers of principal components can be checked as follows:

```
print(result<-repeat.EM.AMMI(x,10,0:2))
```

The EM-AMMI based on 0 or 1 principal component gave stable outcomes, with no local minima. The values imputed with the model without principal components (`PC.nb` was 0, so the results can be approached by typing `result$estimated[ , ,"0"]` or shorter `result$e[ , ,"0"]`) were the same for each repetition. The values imputed with the model

with one principal component (`result$estimated[ , ,"1"]`) were the same for all the repetitions, but slightly different from those obtained with `PC.nb=0`. For two principal components (`result$estimated[ , ,"2"]`), however, there were too many empty cells to reach reliable imputation: the imputed values obtained in the repetitions were much different.

The optimal number of principal components (from 0 to 2) to be used can be determined by the RMSPD values:

```
CV.LOO(x,MNO=1)$RMSPD
```

The result is the vector of RMSPD values (9.0, 3.1, 6.8) with the smallest value obtained for one principal component. The imputed values of the five missing cells (`result$estimated[ , ,"1"]`) according to EM-AMMI with one principal component were exactly same, that is, all repetitions gave the same set of imputed values: 4.04, 12.75, 1.08, 5.06, 3.37. Thus we can take the set of estimated values from any of the 10 repetitions. For the first repetition these values can be reached wit the command `result$estimated[ , ,"1"][1, ]` or `result$estimated[1, ,"1"])`. We can now fill in the missing values in the matrix x with the imputed ones:

```
x[is.na(x)]<-result[,,2][1,]
x
     [,1]  [,2] [,3] [,4]
[1,] 2.00  2.00 1.08 3.00
[2,] 3.00  4.00 2.00 3.37
[3,] 4.04  6.00 3.00 4.00
[4,] 5.00  8.00 4.00 4.00
[5,] 6.00 10.00 5.06 5.00
[6,] 7.00 12.75 6.00 5.00
[7,] 8.00 14.00 7.00 6.00
```

CONCLUSIONS

As follows from the author's research (not published), the meaning of an allowable number of principal components is a bit overstated (cf. Gauch and Zobel, 1990). Especially if the chosen number of principal components is too large for the number of missing values, EM-AMMI estimations depend on the initial values of missing cells (but this dependency can occur even for large data sets with a moderate number of missing cells). The dependence on the initial values was presented in Example 2. According to Gauch and Zobel (1990), two principal components are permissible for this data set. But the imputed values obtained by `repeat.EM.AMMI` procedure with two principal components were diverse, suggesting that two principal components are not a good choice. Thus the function `repeat.EM.AMMI()` proposed in this paper can be employed to overcome this problem and to choose the best number of principal components. It aims to check whether the results depend on the initial values of missing cells; if not, the outcome can be treated as reliable.

REFERENCES

Cornelius, P.L. (1993). Statistical tests and retention of terms in the additive main effects and multiplicative interaction model for cultivar trials. *Crop Science* 33, 1186–1193

Dias, C., Krzanowski, W.J. (2003). Model selection and cross validation in additive main effect and multiplicative interaction models. *Crop Science* 43, 865–873.

Gauch, H.G. (1992). *Statistical analysis of regional yield trials. AMMI analysis of factorial designs.* Elsevier Science, New York

Gauch, H.G. (2007). *MATMODEL version 3.0: Open source software for AMMI and related analyses*. Crop and Soil Sciences, Cornell University, Ithaca, NY.

Gauch, H.G., Zobel, R.W. (1990). Imputing missing yield trial data. *Theoretical Applied Genetics* 79, 753–761

Jackson, D.A. (1993). Stopping rules in principal components analysis: A comparison of heuristical and statistical aproaches. *Ecology* 74, 2204–2214

Kaya, Y., Akcura, M., Ayaranci, R., Taner, S. (2006). Pattern analysis of multi-environment trials in bread wheat. *Communications in Biometry and Crop Science* 1 (1), 63–71.

Leek, T.J. (2011). Asymptotic conditional singular value decomposition for high-dimensional genomic data. *Biometrics* 67, 344–352

de Mendiburu, F. (2012). *agricolae: Statistical Procedures for Agricultural Research*. R package version 1.1-3. http://CRAN.R-project.org/package=agricolae

Onofri, A., Ciriciofolo, E. (2007). Using R to Perform the AMMI Analysis on Agriculture Variety Trials. *R News*. 7(1), 14–19.

Paderewski, J., Gauch, H.G. Jr., Mądry, W., Drzazga, T., Rodrigues, P.C. (2011). Yield response of winter wheat to agro-ecological conditions using additive main effects and multiplicative interaction and cluster analysis. *Crop Science* 51, 969–980

R Development Core Team. (2013). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Raiche, G. (2010). *nFactors: an R package for parallel analysis and non graphical solutions to the Cattell scree test*.R package version 2.3.3.

Yan, W. (2013). Biplot analysis of incomplete two-way data. Crop Science 53, 48–57

Yan, W., Kang, M.S. (2003). *GGE Biplot Analysis: A Graphical Tool for Breeders, Geneticists, and Agronomists*. CRC Press. Boca Raton, FL.