R Package mvngGrAd: Moving Grid Adjustment In Plant Breeding Field Trials

Frank Technow*

package version 0.1.3

1 Moving grid adjustment

Moving grid adjustment is a spatial method to adjust for environmental variation in field trials. It is most common in unreplicated plant breeding field trials. The most extreme form of unreplicated trials is single plant evaluation, for example for recurrent mass selection in populations.

The trial is arranged in a spatial row-column like design. Each entry is then associated with a cell, respectively with a row and column number. A grid is constructed around each cell (= entry) and the mean of the cells included in the grid is calculated. The moving mean of the ith entry, denoted as x_i, is calculated as:

$$x_i = \frac{\sum_j p_{j,obs} \cdot I(p_{j,obs} \in G_i)}{\sum_j I(p_{j,obs} \in G_i)} \tag{1}$$

The grid of entry i is denoted by G_i and $I(\cdot)$ is a indicator function that takes the value 1 if the condition is satisfied and 0 if not. The observed phenotypic values of all entries which could potentially be included in G_i are denoted by $p_{j,obs}$. The value x_i is taken as a measure of the growing conditions for the entry i and is used as a covariate to calculate an adjusted phenotypic value ($p_{i,adj}$) according to the following formula:

$$p_{i,adj} = p_{i,obs} - b(x_i - \bar{x}) \tag{2}$$

*University of Hohenheim, Institute of Plant Breeding, Seed Science and Population Genetics, Stuttgart, Germany

Where $p_{i,obs}$ denotes the observed phenotypic value of the ith entry, $\bar{x}$ is the mean of all x_i and b the regression coefficient in the general linear model:

$$p_{i,obs} = a + bx_i \tag{3}$$

with intercept a.

The phenotypic value is a function of the genotypic value g_i and the environmental error e_i. Before the adjustment this is

$$p_{i,obs} = g_i + e_i, \tag{4}$$

and after

$$p_{i,adj} = g_i + e_i' \tag{5}$$

with e_i' being the e_i after adjustment.

When the adjustment was successful,

$$var(e_i') < var(e_i) \tag{6}$$

and $p_{i,adj}$ is a better estimator of g_i than $p_{i,obs}$. A direct consequence of this is that the heritability (h^2) will improve. The h^2 gives the proportion of variance observed that was attributable to genetic effects. The closer to one the value is, the better, because only genetic variance can be exploited by the breeder. The heritability before adjustment is calculated as

$$h^2_{obs} = \frac{var(g)}{var(p_{obs})} \tag{7}$$

where $var(g)$ is the *exploitable* genetic variance and $var(p_{obs})$ is the variance of p_{obs} (i.e. the phenotypic variance). The heritability after adjustment can then be calculated as

$$h^2_{adj} = \frac{var(g)}{var(p_{adj})} \tag{8}$$

and because of (6)

$$h^2_{adj} > h^2_{obs} \tag{9}$$

The adjustment procedure can only be successful, and in fact valid, if two important conditions are met:

1. the entries must not influence the values of their covariates (x_i) and
2. the entries must have been randomly assigned to positions in the field.

If either one or both are violated, the adjustment can lead to erroneous results. If for example $p_{i,obs}$ is included in the calculation of x_i, the first condition is violated. The result is that the p_{adj} of superior genotypes are downwardly biased and the p_{adj} of inferior ones upwardly. When, as is the case many times, entries are not randomized but grouped according to family, the p_{adj} of superior families are downwardly biased and the p_{adj} of inferior ones upwardly. In the best case this leads to a reduction in $var(g)$[†] which might reverse (9) or at least reduces the superiority of h^2_{adj}. In the worst case, selection might be directed to the opposite of the intended direction!

Please refer to [a] for details on quantitative genetics, to [b] for details on experimental design and covariate adjustment and to [c] for plant breeding designs and moving grids in particular.

References

[a] D. S. Falconer: *Introduction to Quantitative Genetics*. Oliver and Boyd., London, 1967.

[b] W. G. Cochran and G. M. Cox: *Experimental Designs*. John Wiley & Sons, Inc., New York, 1964.

[c] I. Bos and P. Caligari: *Selection Methods in Plant Breeding*. Springer New York, 2008.

2 The package

2.1 Overview

The function performing the adjustment is called `movingGrid( )`. The function `sketchGrid( )` helps with designing the grid by plotting its shape. The functions `fitted( )`, `movingMean( )` and `entryData( )` are convenience functions to extract the most relevant information from the object created by `movingGrid( )`. The package defines a new class, `movG`, and provides methods for the functions `entryData( )`, `fitted( )`, `movingMean( )`, `summary( )`, `show( )` and `residuals( )`. What follows is a detailed tutorial on the usage of `movingGrid( )`.

[†]The adjustment is of course something that is done to the data, and as such can not alter the population variance. To distinguish $var(g)$ from the true genetic variance in the population, I termed it *exploitable genetic variance*.

As a first step, the package needs to be loaded.

```
R> library(mvngGrAd)
```

2.2 Setting up the grid

The shape of the grid must be designed by the user. This is done via three arguments to `movingGrid( )`, `shapeCross`, `layers` and `excludeCenter`. With the help of `shapeCross` one specifies the cells that are to be included in a grid that extends from the center in 0, 90, 180 and 270 degree direction, with `layers` the cells that extend the grid in all other directions. The two can be used together or alone. With the argument `excludeCenter` one can decide whether or not to include $p_{i,obs}$ in the calculation of x_i. The usage is best described with examples.

For designing the grid, function `sketchGrid( )` takes the same arguments as `movingGrid( )` and plots a visualization of the grid. This function should always be used to verify that the actual arguments to `shapeCross` and `layers` do create the intended grid.

The argument to `shapeCross` is given in form of a list with four components, each representing one of the directions.

1. DOWN (180°)
2. UP (0°)
3. LEFT (270°)
4. RIGHT (90°).

Other arguments to `sketchGrid( )` are `i`, the row of the central cell; `j` the column of the central cell; `rowLimit`, the number of rows in the field layout; `colLimit`, the number of columns; and `excludeCenter`. If only one of `shapeCross` or `layers` is to be used, the other must be `NULL`.

A grid that includes one cell above and below the center and four to its right and left and excludes the central cell can be created as follows[‡] [§](Figure 1 [a]):

[‡]The calls to `sketchGrid( )` produce always one plot. To save space, these plots are not included in this document, but instead figures are shown that combine several of them. These figures are produced from code that is included in the `sweave` file but is not echoed. However, the calls to `sketchGrid( )` are identical.

[§]This is the grid used by a stand alone software called "plabstat" (https://www.uni-hohenheim.de/plantbreeding/software/) that is around since the 70's. With these settings, the results of "plabstat" can be reproduced.

```
R> sketchGrid(i = 10,
+             j = 10,
+             rowLimit = 20,
+             colLimit = 20,
+             shapeCross =
+             list(1, ## down
+                  1, ## up
+                  1:4, ## left
+                  1:4),## right
+             layers = NULL,
+             excludeCenter = TRUE)
```

Using `shapeCross` as follows, creates a grid that also includes one cell above and below the center and four to its right and left but excludes the cells right next to the center (Figure 1 [b]).

```
R> sketchGrid(i = 10, j = 10, rowLimit = 20, colLimit = 20,
+     shapeCross = list(2, 2, 2:5, 2:5), layers = NULL,
+     excludeCenter = TRUE)
```

Using `shapeCross` on its own, without `layers`, can often make sense, using `layers` on its own usually does not. However, it is introduced independently from `shapeCross` to make its usage clear. The actual argument to `layers` is an integer vector. Each integer represents a "layer" of cells included in the grid. Integer `1` would include the cells that are right next to the center, apart from the ones in 0, 90, 180 and 270 degree direction (Figure 1 [c]):

```
R> sketchGrid(i = 10, j = 10, rowLimit = 20, colLimit = 20,
+     shapeCross = list(NULL, NULL, NULL, NULL),
+     layers = 1, excludeCenter = TRUE)
```

and integer `2` the ones on top of that (Figure 1 [d]):

```
R> sketchGrid(i = 10, j = 10, rowLimit = 20, colLimit = 20,
+     shapeCross = list(NULL, NULL, NULL, NULL),
+     layers = 1:2, excludeCenter = TRUE)
```

and so on.

By using `shapeCross` and `layers` jointly, more complex grids can be created. For example a honeycomb shape, which might be most suited for single plant evaluation (Figure 2 [e]):

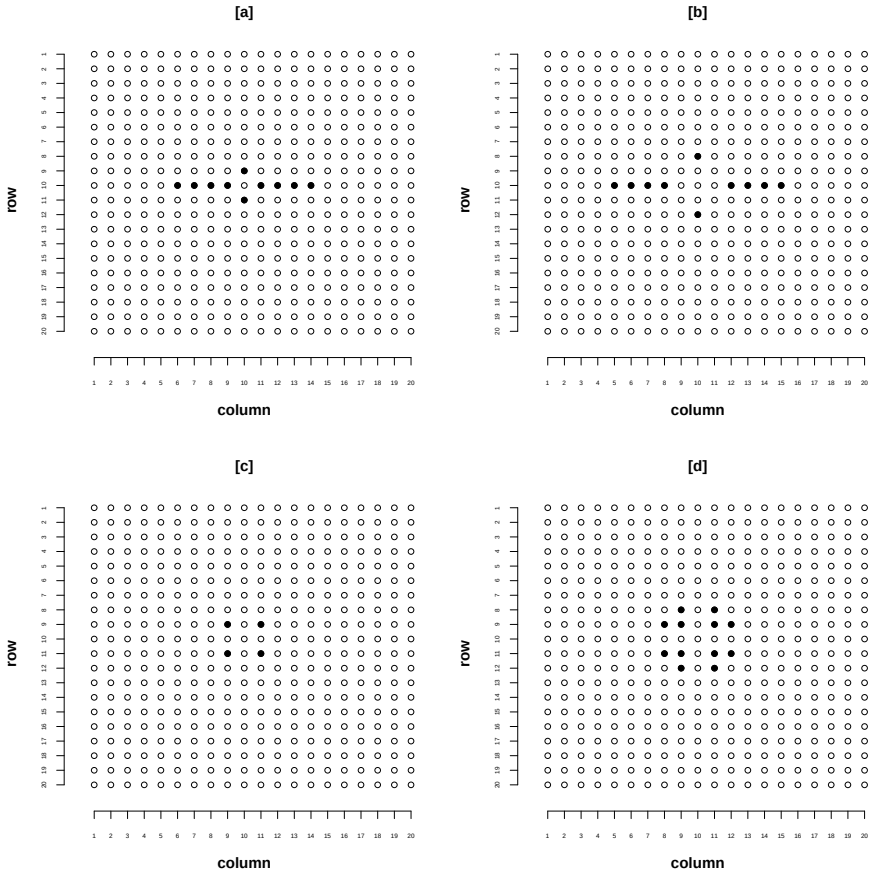

Figure 1: Grids for different settings of `shapeCross` ,`layers`, `excludeCenter` and `i, j`

```
R> sketchGrid(i = 10, j = 10, rowLimit = 20, colLimit = 20,
+     shapeCross = list(1:4, 1:4, 1:4, 1:4), layers = c(1:4),
+     excludeCenter = TRUE)
```

However, one should keep in mind that `sketchGrid( )` can not be made to display any scale differences between row and column widths. So for unequal row and column width, the grid will look different on the field than the output of `sketchGrid( )`. In this case `sketchGrid( )` will only help to see which cells are included.

Setting the argument `excludeCenter` to `FALSE` will include the central cell, the one with the entry whose p_{obs} is to be adjusted, in the calculation of x_i (Figure 2 [f]):

```
R> sketchGrid(i = 10, j = 10, rowLimit = 20, colLimit = 20,
+     shapeCross = list(1:4, 1:4, 1:4, 1:4), layers = c(1:4),
+     excludeCenter = FALSE)
```

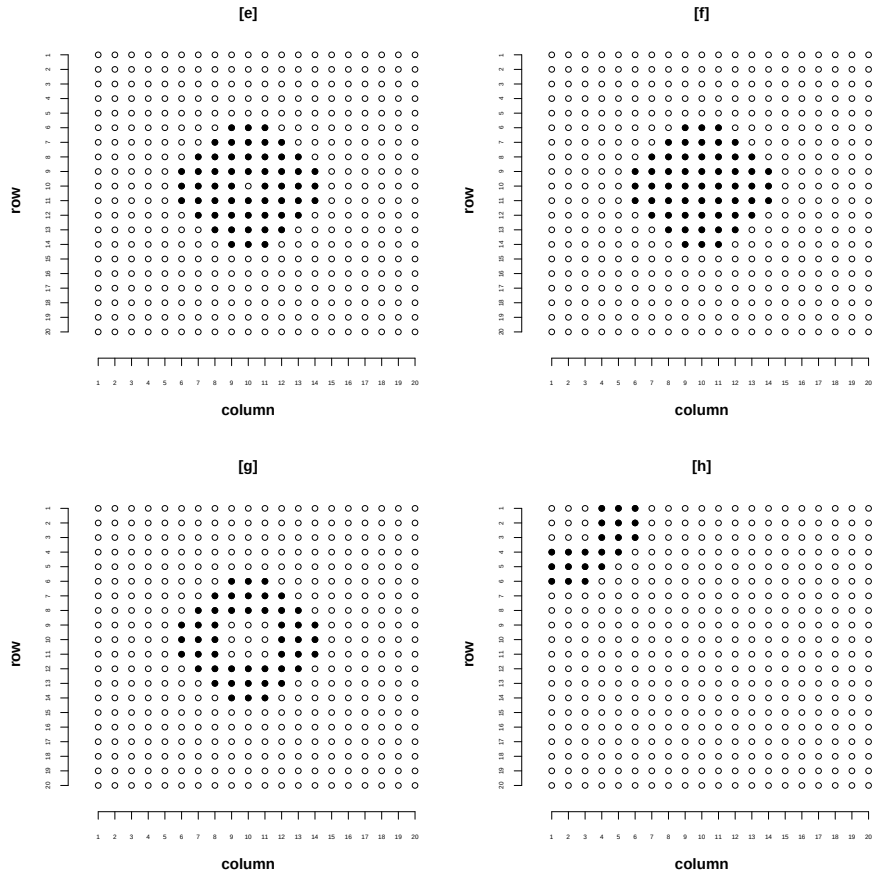

Figure 2: Grids for different settings of `shapeCross` ,`layers`, `excludeCenter` and `i, j`

This is of course in clear violation of the first condition for a valid adjustment (see Section 1 on page˜2). In fact one may consider to exclude the cells right next to the center as well. This way the effects of possible interactions between the entry in the central cell and its neighbors, do not influence x_i either (Figure 2 [g]):

```
R> sketchGrid(i = 10, j = 10, rowLimit = 20, colLimit = 20,
```

```
+     shapeCross = list(2:4, 2:4, 2:4, 2:4), layers = c(2:4),
+     excludeCenter = TRUE)
```

For cells close to the border of the experimental area, only incomplete grids can be constructed of course. This is correctly displayed by `sketchGrid( )` (Figure 2 [h]):

```
R> sketchGrid(i = 2, j = 2, rowLimit = 20, colLimit = 20,
+     shapeCross = list(2:4, 2:4, 2:4, 2:4), layers = c(2:4),
+     excludeCenter = TRUE)
```

2.3 Using `movingGrid( )`

To demonstrate the function `movingGrid ( )` itself, data needs to be generated first.

The experimental area of the simulated field trial consists of 50 rows and 50 columns. The row and column subscripts of each cell are stored because they are part of the input to `movingGrid( )`. The two vectors must of course correspond to each other.

```
R> rows <- rep(1:50, each = 50)
R> cols <- rep(1:50, 50)
```

The population parameters are as follows:

- population mean $\mu = 30$
- genotypic variation $var(g) = 25$
- environmental (error) variation $var(e) = 45$

The environmental errors e are simulated in such a way that there is a strong systematic horizontal gradient in growing conditions present in the experimental area, together with some unsystematic small scale noise.

```
R> set.seed(13)
R> envError <- rep(c(seq(from = -12.5, to = -0.5,
+     by = 0.5), seq(from = 0.5, to = 12.5, by = 0.5)),
+     each = 50) + rnorm(2500)
```

The following code will scale this vector to have a variance of 45.

```
R> scaleFactE <- (sd(envError)/sqrt(45))
R> envError <- scale(envError, center = FALSE, scale = scaleFactE)
R> envError <- as.vector(envError)
```

The genotypic effects of the 2500 entries are simulated with zero mean and variance of 25 (because of some sampling variation the vector is additionally scaled to assure a variance of 25).

```
R> gEffects <- rnorm(2500, mean = 0, sd = 5)
R> scaleFactG <- (sd(gEffects)/sqrt(25))
R> gEffects <- scale(gEffects, center = FALSE, scale = scaleFactG)
R> gEffects <- as.vector(gEffects)
```

To arrive at the genotypic values g, μ is added to `gEffects`

```
R> mu <- 30
R> gValues <- mu + gEffects
```

The observed phenotypic values (p_{obs}) are then obtained by adding `gValues` to `envError` according to (4).

```
R> obsP <- gValues + envError
```

To observe the handling of `NA` values, the third element of `obsP` is set to `NA`.

```
R> obsP[3] <- NA
```

Finally the phenotypic and genetic variance are stored for later use. Note that $var(p_{obs})$ will not be exactly 70 as it should be, but slightly different due to minimal, "random" covariance between `gEffects` and `envError`.

```
R> varP <- var(obsP, na.rm = TRUE)
R> varG <- var(gEffects)
```

The row and column vectors are given to the arguments `rows` and `columns` and the p_{obs} to `obsPhe` of `movingGrid( )` The grid is designed by `shapeCross` and `layers` as demonstrated above. The argument `excludeCenter` is `TRUE` by default.

```
R> ## creates object of class movG
R> resMG <- movingGrid(rows = rows,
+                      columns = cols,
+                      obsPhe = obsP,
+                      shapeCross =
+                      list(1:2,
+                           1:2,
+                           1:2),
+                      layers = 1)
```

A summary of the adjustment process can be obtained by the function `summary( )`.

```
R> summary(resMG)

Adjustment by function: movingGrid

Maximum possible number of values in the grid: 12

Mean number of values in grid: 11.6

Number of observation: 2500 , number of NA-observations: 1

Coefficient of correlation between moving means
and observed phe. values: 0.77

Regression coefficient of moving means regressed
on observed phe. values : 0.96

Experimental layout:
         rows: 50
      columns: 50
```

The first line names the function used. In the second line the number of values (cells) in a complete grid is given. The mean number of values in the third line is always smaller, because for entries close to the edge of the experimental area only partial grids can be constructed. NA values also have this effect. The fourth line gives the number of observations[§] and how many of these were NA.

[§] An "observation" is everything mentioned in the vectors given to `rows` and `columns`, irrespective of whether the corresponding data entry in the vector to `obsPhe` is `NA` or not.

The Pearson coefficient of correlation between x_i and $p_{i,obs}$ gives information on the strength of the influence of the covariate x_i. If this value is too low, an adjustment will not yield better estimates of g than by using p_{obs} directly. In rare cases, for example under strong inter plant competition, the correlation coefficient can be below zero. In such a case one should not perform an adjustment. The next line gives b, the regression coefficient used for the adjustment [see (2)]. Finally, the dimensions of the experimental area are given.[¶]

The function `entryData( )` extracts all information on an entry available, its row, column, $p_{i,adj}$, $p_{i,obs}$, x_i and the number of values used for calculating its x_i.

```
R> head(entryData(resMG))
```

```
  row column adjustedPhe observedPhe movingMean nValues
1   1      1      28.292      18.581     19.836       4
2   1      2      28.211      18.393     19.724       6
3   1      3          NA          NA     18.492       8
4   1      4      32.779      23.825     20.627       7
5   1      5      27.936      20.621     22.339       7
6   1      6      31.962      23.694     21.344       8
```

For the third element, which is `NA`, no p_{adj} was calculated of course. The moving mean x_i is, however, available.

The p_{adj} can be obtained by using the function `fitted( )`.

```
R> fitted(resMG)[1:10]
```

```
 [1] 28.292 28.211     NA 32.779 27.936 31.962 28.684 35.035
 [9] 30.983 27.012
```

The x are extracted by `movingMean( )`.

```
R> movingMean(resMG)[1:10]
```

```
 [1] 19.83616 19.72393 18.49230 20.62685 22.33869 21.34358
 [7] 19.83770 18.34463 17.82967 18.96546
```

[¶]The function `movingGrid( )` automatically assumes a rectangular experimental area with the given dimensions. Therefore, the number of cells in the "virtual" experimental. area is *rows* × *columns*. If the number of observations is smaller than this, the remaining cells are filled with `NA` values. These are not "observations" and are not included in the number of NA values given in the summary. They also do not influence the results of `movingGrid( )` and are ignored by the various extractor functions.

The residuals for the model in (3) are returned by the function `residuals( )`.

```
R> residuals(resMG)[1:10]
```

```
        1         2         3         4         5         6
-1.695044 -1.775721        NA  2.791641 -2.051174  1.974373
        7         8         9        10
-1.302638  5.048233  0.995523 -2.974943
```

The h^2_{obs} is 0.362

```
R> varG/varP
```

```
[1] 0.3622204
```

while the h^2_{adj} is with 0.898 considerably higher.

```
R> varPadj <- var(fitted(resMG), na.rm = TRUE)
R> varG/varPadj
```

```
[1] 0.898043
```

This demonstrates a tremendous improvement due to the adjustment procedure. In reality the improvement will usually not be this much because the contribution of random noise, which can not be picked up by moving grid adjustment, is much larger than in this example.