But… what is a ‘diallel’?

If you are not a plant breeder or a geneticist in general, you may be asking this question. From the ancient Greek language, the ‘diallel’ word means ‘reciprocating’ and a diallel cross is a set of several possible crosses and selfs between some parental lines. For example, if we take the male lines A, B and C together with the same female lines A, B and C and we imagine to cross those lines with one another, we obtain the selfs A\(\times\)A, B\(\times\)B and C\(\times\)C, the crosses A\(\times\)B, A\(\times\)C and B\(\times\)C and, in some instances, the reciprocals B\(\times\)A, C\(\times\)A and C\(\times\)B (where the father and mother are swapped). The performances of crosses and/or selfs and/or reciprocals can be compared by planning field experiments, usually known as **diallel experiments** and designed as randomised complete blocks with 3-4 replicates.

The example

Depending on how the experiment is planned, we can have four experimental methods:

1. Crosses + reciprocals + selfs (complete diallel)
2. Crosses and reciprocals (no selfs)
3. Crosses and selfs (no reciprocals)
4. Only crosses (no selfs, no reciprocals)

In this post we will concentrate on the first design (complete diallel) and we will use a simple example with three parental lines (A, B and C). The csv file (‘diallel1.csv’) is available in an external repository; in the box below we load the data and we use the `group_by()` function in the ‘dplyr’ package to obtain the means for all crosses and selfs.

```
library(tidyverse)
rm(list = ls())
df <- read_csv("https://www.casaonofri.it/_datasets/diallel1.csv")
df$Block <- factor(df$Block)
dfM <- df %>%
  group_by(Par1, Par2) %>%
  summarise(YieldM = mean(Yield), SEs = sd(Yield/sqrt(4)))
dfM
## # A tibble: 9 x 4
## # Groups:   Par1 [3]
##   Par1  Par2  YieldM   SEs
##
## 1 A     A         12 0.740
## 2 A     B         13 0.600
## 3 A     C         14 0.498
## 4 B     A         11 1.00
## 5 B     B         15 0.332
## 6 B     C         21 0.273
## 7 C     A         17 0.295
## 8 C     B         16 0.166
## 9 C     C         19 1.90
```

What model do we use?

In order to describe the above dataset, we might think of a two-way ANOVA model, where the ‘father’ and ‘mother’ lines (the ‘Par1’ and ‘Par2’ variables, respectively) are used as the explanatory factors.

This is a very tempting solution, but we should resist: a two way ANOVA model regards the ‘father’ and ‘mother’ effects as two completely different series of treatments, neglecting the fact that they are, indeed, the same genotypes in different combinations. That is exactly why we need specific **diallel models** to describe the results of diallel experiments!

The “Hayman” model (type 1)

The first diallel model was proposed by Hayman (1954) and it was devised for complete diallel experiments, where reciprocals are available. Neglecting the design effects (blocks and/or environments), the Hayman’s model is defined as:

$$y_{ijk} = \mu + \textrm{g}_i + \textrm{g}_j + \textrm{ts}_{ij} + \textrm{rg}^a_{i} + \textrm{rg}^b_{j} + rs_{ij} + \varepsilon_{ijk} \quad\quad\quad (Eq.\ 1)$$

where μ is expected value (the overall mean, in the balanced case) and ε_{ijk} is the residual random error terms for the observation in the k^{th} block and with the parentals i and j. All the other terms correspond to genetic effects, namely:

1. the $\textrm{g}_i$ and $\textrm{g}_j$ terms are the **general combining abilities** (GCAs) of the i^{th} and j^{th} parents. Each term relates to the average performances of a parental line in all its hybrid combination, under the sum-to-zero constraint (i.e. the sum of g values for all parentals must be zero). For example, with our balanced experiment, the overall mean is $\mu = 15.33$, while the mean for the A parent when used as the ‘father’ is $\mu_{A.} = 13$ and the mean for the same parent A when used as the ‘mother’ is $\mu_{.A} = 13.33$. Consequently:
   $$g_A = \left(13 + 13.33 \right)/2 – 15.33 = -2.167$$ Analogously, it is $g_B = -0.167$.
2. The rg^a_i and rg^b_j terms are the **reciprocal general combining abilities** (RGCAs) for the i^{th} and j^{th} parents. Each term relates to the discrepancy between the effect of a parent when it is used as father/mother and its average effect in all its combinations. For example, considering the parent A, the term rg^a_A is:
   $$rg^a_A = \mu_{A.} – \frac{\mu_{A.} + \mu_{.A}}{2} = 13 – 13.167 = -0.167$$ Obviously, it must be $rg^a_A = – rg^b_B$ and it must also be that the sum of all rg^a terms is zero (sum-to-zero constraint).
3. The $\textrm{ts}_{ij}$ term is the total **specific combining ability** (tSCA) for the combination between the i^{th} and j^{th} parents. It relates to the discrepancy from additivity for a specific combination of two parentals. For example, considering the ‘A $\times$ B’ cross, the expected yield under additivity would be: $$\mu_{A:B} = \mu + \textrm{g}_A + \textrm{g}_B +\textrm{rg}^a_{A} + \textrm{rg}^b_{B} =$$ $$= 15.33 – 2.167 – 0.167 – 0.167 – 0.5 = 12.333$$ while the observed yield is 13, with a with a difference of -0.667. On the other hand, considering the ‘B $\times$ A’ reciprocal cross, the expected yield under additivity would be: $$\mu_{A:B} = \mu + \textrm{g}_A + \textrm{g}_B +\textrm{rg}^a_{B} + \textrm{rg}^b_{A} =$$ $$= 15.33 – 2.167 – 0.167 + 0.167 + 0.5 = 13.667$$ while the observed yield is 11, with a difference of 2.667. The tSCA for the cross between A and B (regardless of the reciprocal) is the average difference, that is $\textrm{ts}_{AB} = (-0.667 + 2.667)/2 = 1$.
4. The rs_{ij} term is the **reciprocal specific combining ability** (RSCA) for a specific ij combination, that is the discrepancy between the performances of the two reciprocals (e.g, A $\times$ B vs. B $\times$ A). For example, the $\textrm{rs}_{AB}$ term is equal to $-0.667 – 1 = -1.667$, that is the opposite of $\textrm{rs}_{BA}$.

Model fitting with R

Hands-calculations based on means may be useful to understand the meaning of genetical effects, although they are biased with unbalanced designs and, above all, they are totally uninteresting from a practical point of view: we'd rather fit the model by using a statistical software.

Let's assume that all effects are fixed, apart from the residual standard error. This is a reasonable assumption, as we have a very low number of parentals, which would make the estimation of variance components totally unreliable. We clearly see that the Hayman's model above is a specific parameterisation of a general linear model and we should be able to fit it by the usual `lm()` function and related methods. We can, indeed, do so by using our 'lmDiallel' extension package, that provides the facilities to generate the correct design matrices for the Hayman's model (and for other diallel models, as we will show in future posts).

At the beginning, we have to install (if necessary) and load the 'lmDiallel' package (see box below). Model fitting can be performed by using the `GCA()`, `tSCA()`, `RGCA()` and `RSCA()` functions as shown in the box below: the resulting `lm` object can be explored by the usual R methods, such as `summary()` and `anova()`.

```
# library(devtools) # Install if necessary
# install_github("OnofriAndreaPG/lmDiallel")
library(lmDiallel)
dMod <- lm(Yield ~ Block + GCA(Par1, Par2) + tSCA(Par1, Par2) +
             RGCA(Par1, Par2) + RSCA(Par1, Par2), data = df)
summary(dMod)
##
## Call:
## lm(formula = Yield ~ Block + GCA(Par1, Par2) + tSCA(Par1, Par2) +
##     RGCA(Par1, Par2) + RSCA(Par1, Par2), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3500 -0.5644  0.0606  0.4722  2.7911
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1.558e+01  5.780e-01  26.962  < 2e-16 ***
## Block2                 -3.772e-01  8.174e-01  -0.461   0.6486
## Block3                 -3.011e-01  8.174e-01  -0.368   0.7158
## Block4                 -3.261e-01  8.174e-01  -0.399   0.6935
## GCA(Par1, Par2)g_A     -2.167e+00  2.890e-01  -7.497 9.77e-08 ***
## GCA(Par1, Par2)g_B     -1.667e-01  2.890e-01  -0.577   0.5695
## tSCA(Par1, Par2)ts_A:A  1.000e+00  5.780e-01   1.730   0.0965 .
## tSCA(Par1, Par2)ts_A:B -1.000e+00  4.570e-01  -2.188   0.0386 *
## tSCA(Par1, Par2)ts_B:B  1.230e-16  5.780e-01   0.000   1.0000
## RGCA(Par1, Par2)rg_A   -1.667e-01  2.890e-01  -0.577   0.5695
## RGCA(Par1, Par2)rg_B    5.000e-01  2.890e-01   1.730   0.0965 .
## RSCA(Par1, Par2)        1.667e+00  3.540e-01   4.709 8.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.734 on 24 degrees of freedom
## Multiple R-squared:  0.8269, Adjusted R-squared:  0.7476
## F-statistic: 10.42 on 11 and 24 DF,  p-value: 1.129e-06
anova(dMod)
## Analysis of Variance Table
##
## Response: Yield
##                   Df  Sum Sq Mean Sq F value    Pr(>F)
## Block              3   0.784   0.261  0.0869    0.9665
## GCA(Par1, Par2)    2 244.000 122.000 40.5743 1.999e-08 ***
## tSCA(Par1, Par2)   3  24.000   8.000  2.6606    0.0710 .
## RGCA(Par1, Par2)   2   9.333   4.667  1.5520    0.2323
## RSCA(Par1, Par2)   1  66.667  66.667 22.1717 8.710e-05 ***
## Residuals         24  72.164   3.007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the sake of simplicity, we also built a wrapper function named `lm.diallel()`, which can be used in the very same fashion as `lm()`. The syntax is:

```
lm.diallel(formula, Block, Env, data, fct)
```

where 'formula' specifies the response variable and the two variables for parentals (e.g., Yield ~ Par1 + Par2) and the two arguments 'Block' and 'Env' are used to specify optional variables, coding for blocks and environments, respectively. The argument 'data' is a 'dataframe' where to look for the explanatory variables and, finally, 'fct' is a string variable coding for the selected model ("HAYMAN1", for this example; see below).

```
dMod2 <- lm.diallel(Yield ~ Par1 + Par2, Block = Block,
                    data = df, fct = "HAYMAN1")
summary(dMod2)
##
## Call:
## lm.diallel(formula = Yield ~ Par1 + Par2, Block = Block, fct =
"HAYMAN1",
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3500 -0.5644  0.0606  0.4722  2.7911
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## Intercept  1.533e+01  2.890e-01  53.056  < 2e-16 ***
## Block1     2.511e-01  5.006e-01   0.502   0.6205
## Block2    -1.261e-01  5.006e-01  -0.252   0.8032
## Block3    -5.000e-02  5.006e-01  -0.100   0.9213
## g_A       -2.167e+00  2.890e-01  -7.497 9.77e-08 ***
## g_B       -1.667e-01  2.890e-01  -0.577   0.5695
## ts_A:A     1.000e+00  5.780e-01   1.730   0.0965 .
## ts_A:B    -1.000e+00  4.570e-01  -2.188   0.0386 *
## ts_B:B     6.152e-16  5.780e-01   0.000   1.0000
## rg_A      -1.667e-01  2.890e-01  -0.577   0.5695
```

```
## rg_B       5.000e-01  2.890e-01   1.730   0.0965 .
## rs_A:B     1.667e+00  3.540e-01   4.709 8.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.734 on 24 degrees of freedom
## Multiple R-squared:  0.8269, Adjusted R-squared:  0.7476
## F-statistic: 10.42 on 11 and 24 DF,  p-value: 1.129e-06
anova(dMod2)
## Analysis of Variance Table
##
## Response: Yield
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## Block      3   0.784   0.261  0.0869    0.9665
## GCA        2 244.000 122.000 40.5743 1.999e-08 ***
## tSCA       3  24.000   8.000  2.6606    0.0710 .
## RGCA       2   9.333   4.667  1.5520    0.2323
## RSCA       1  66.667  66.667 22.1717 8.710e-05 ***
## Residuals 24  72.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above function works very much like the `lm()` function and makes use of the general purpose linear model solver `lm.fit()`. Apart from simplicity, another advantage is that the call to `lm.diallel()` returns an object of both 'lm' and 'diallel' classes. For this latter class, we built several specific S3 methods, such as the usual `anova()`, `summary()` and `model.matrix()` methods, partly shown in the box above.

Considering that diallel models are usually fitted to determine genetical parameters, we also built the `glht.diallelMod()` method and the `diallel.eff()` function, which can be used with the 'multcomp' package, to retrieve the complete list of genetical parameters, as shown in the box below.

```
library(multcomp)
gh <- glht(linfct = diallel.eff(dMod2))
summary(gh, test = adjusted(type = "none"))
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Linear Hypotheses:
##                   Estimate Std. Error t value Pr(>|t|)
## Intercept == 0  1.533e+01  2.890e-01  53.056  < 2e-16 ***
## g_A == 0       -2.167e+00  2.890e-01  -7.497 5.85e-08 ***
## g_B == 0       -1.667e-01  2.890e-01  -0.577   0.5691
## g_C == 0        2.333e+00  2.890e-01   8.074 1.49e-08 ***
## ts_A:A == 0     1.000e+00  5.780e-01   1.730   0.0955 .
## ts_A:B == 0    -1.000e+00  4.570e-01  -2.188   0.0378 *
## ts_A:C == 0     1.443e-15  4.570e-01   0.000   1.0000
## ts_B:A == 0    -1.000e+00  4.570e-01  -2.188   0.0378 *
## ts_B:B == 0     6.152e-16  5.780e-01   0.000   1.0000
## ts_B:C == 0     1.000e+00  4.570e-01   2.188   0.0378 *
## ts_C:A == 0     1.443e-15  4.570e-01   0.000   1.0000
```

```
## ts_C:B == 0     1.000e+00  4.570e-01   2.188   0.0378 *
## ts_C:C == 0    -1.000e+00  5.780e-01  -1.730   0.0955 .
## j_A == 0       -1.667e-01  2.890e-01  -0.577   0.5691
## j_B == 0        5.000e-01  2.890e-01   1.730   0.0955 .
## j_C == 0       -3.333e-01  2.890e-01  -1.153   0.2592
## rs_A:B == 0     1.667e+00  3.540e-01   4.709 7.25e-05 ***
## rs_A:C == 0    -1.667e+00  3.540e-01  -4.709 7.25e-05 ***
## rs_B:A == 0    -1.667e+00  3.540e-01  -4.709 7.25e-05 ***
## rs_B:C == 0     1.667e+00  3.540e-01   4.709 7.25e-05 ***
## rs_C:A == 0     1.667e+00  3.540e-01   4.709 7.25e-05 ***
## rs_C:B == 0    -1.667e+00  3.540e-01  -4.709 7.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- none method)
```

Model fitting in two steps

In some cases, the analysis is performed in two steps and a diallel model is fitted to the means of selfs and crosses, which are calculated in the first step. Under the assumption of variance homogeneity and equal number of replicates, we can fit the Hayman's model by using the `lm.diallel()` function without the 'Block' argument.

```
dMod3 <- lm.diallel(YieldM ~ Par1 + Par2,
                    data = dfM, fct = "HAYMAN1")
```

In this case, we have no reliable estimate of residual error, but the `summary()` and `anova()` methods have been enhanced to give us the possibility of passing some information from the first step, i.e. an appropriate estimate of the residual mean square and degrees of freedom; the residual mean square from the first step needs to be appropriately weighted for the number of replicates (i.e., for this example, MSE = 3.007/4 with 24 degrees of freedom).

```
anova(dMod3, MSE = 3.007/4, dfr = 24)
## Analysis of Variance Table
##
## Response: YieldM
##           Df Sum Sq Mean Sq F value    Pr(>F)
## GCA        2 61.000 30.5000 40.5720 2.000e-08 ***
## tSCA       3  6.000  2.0000  2.6605   0.07101 .
## RGCA       2  2.333  1.1667  1.5519   0.23236
## RSCA       1 16.667 16.6667 22.1705 8.713e-05 ***
## Residuals 24         0.7518
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(dMod3, MSE = 3.007/4, dfr = 24)
##                Estimate        SE       t value     Pr(>|t|)
## Intercept  1.533333e+01 0.2890117  5.305436e+01 2.157713e-26
## g_A       -2.166667e+00 0.2890117 -7.496812e+00 9.771159e-08
## g_B       -1.666667e-01 0.2890117 -5.766779e-01 5.695269e-01
## ts_A:A     1.000000e+00 0.5780235  1.730034e+00 9.646589e-02
## ts_A:B    -1.000000e+00 0.4569677 -2.188339e+00 3.861373e-02
## ts_B:B     2.417819e-15 0.5780235  4.182908e-15 1.000000e+00
## rg_A      -1.666667e-01 0.2890117 -5.766779e-01 5.695269e-01
```

```
## rg_B       5.000000e-01 0.2890117  1.730034e+00 9.646589e-02
## rs_A:B     1.666667e+00 0.3539656  4.708555e+00 8.712864e-05
```

The genetical parameters can be obtained by using the `glht()` function and passing the information from the first step within the call to the `diallel.eff()` function.

```
gh2 <- glht(linfct = diallel.eff(dMod3, MSE = 3.007/4, dfr = 24))
summary(gh2, test = adjusted(type = "none"))
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Linear Hypotheses:
##                   Estimate Std. Error t value Pr(>|t|)
## Intercept == 0  1.533e+01  2.890e-01  53.054  < 2e-16 ***
## g_A == 0       -2.167e+00  2.890e-01  -7.497 5.85e-08 ***
## g_B == 0       -1.667e-01  2.890e-01  -0.577   0.5691
## g_C == 0        2.333e+00  2.890e-01   8.073 1.49e-08 ***
## ts_A:A == 0     1.000e+00  5.780e-01   1.730   0.0955 .
## ts_A:B == 0    -1.000e+00  4.570e-01  -2.188   0.0378 *
## ts_A:C == 0    -8.882e-16  4.570e-01   0.000   1.0000
## ts_B:A == 0    -1.000e+00  4.570e-01  -2.188   0.0378 *
## ts_B:B == 0     2.418e-15  5.780e-01   0.000   1.0000
## ts_B:C == 0     1.000e+00  4.570e-01   2.188   0.0378 *
## ts_C:A == 0    -8.882e-16  4.570e-01   0.000   1.0000
## ts_C:B == 0     1.000e+00  4.570e-01   2.188   0.0378 *
## ts_C:C == 0    -1.000e+00  5.780e-01  -1.730   0.0955 .
## j_A == 0       -1.667e-01  2.890e-01  -0.577   0.5691
## j_B == 0        5.000e-01  2.890e-01   1.730   0.0955 .
## j_C == 0       -3.333e-01  2.890e-01  -1.153   0.2593
## rs_A:B == 0     1.667e+00  3.540e-01   4.709 7.25e-05 ***
## rs_A:C == 0    -1.667e+00  3.540e-01  -4.709 7.25e-05 ***
## rs_B:A == 0    -1.667e+00  3.540e-01  -4.709 7.25e-05 ***
## rs_B:C == 0     1.667e+00  3.540e-01   4.709 7.25e-05 ***
## rs_C:A == 0     1.667e+00  3.540e-01   4.709 7.25e-05 ***
## rs_C:B == 0    -1.667e+00  3.540e-01  -4.709 7.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- none method)
```

Estimation of variance components (random genetic effects)

In some cases, genetic effects are regarded as random and the aim is to estimate variance components. For this, we can use the `mmer()` function in the 'sommer' package (Covarrubias-Pazaran, 2019), although we need to code two dummy variables:

1. 'dr': assuming a value of zero for selfs, a value of -1 for crosses and 1 for the respective reciprocals;
2. 'combination': which gives the reciprocals the same label (i.e. BA is given the same label as AB), so that any difference is erased.

By using the `overlay()` function in the 'sommer' package and the above dummy variables, we can code the following effects:

1. GCA effect: `~overlay(Par1, Par2)`
2. RGCA effect: `~overlay(Par1, Par2):dr` (please, note that the interaction with 'dr' excludes the selfs, which are coded as 0s, and distinguish the reciprocals, as they are coded with opposite signs)
3. SCA effect : `~combination` (reciprocals are not considered)
4. RSCA effect: `~combination:dr` (same note as above for RGCA)

It would make no sense to estimate the variance components for genetic effects with a diallel experiment based on three parentals. Therefore, we give an example based on the 'hayman54' dataset, as available in the 'lmDiallel' package and relating to a complete diallel experiment with eight parentals (Hayman, 1954).

```
rm(list=ls())
library(sommer)
library(lmDiallel)
data(hayman54)

# Dummy variables
hayman54$dr <- ifelse(as.character(hayman54$Par1) <
as.character(hayman54$Par2), -1,
                ifelse(as.character(hayman54$Par1) ==
as.character(hayman54$Par2), 0, 1))
hayman54$combination <- factor( ifelse(as.character(hayman54$Par1) <=
                                         as.character(hayman54$Par2),
                                paste(hayman54$Par1, hayman54$Par2,
sep =""),
                                paste(hayman54$Par2, hayman54$Par1,
sep =="")) )
mod1h <- mmer(Ftime ~ Block, data = hayman54,
              random = ~ overlay(Par1, Par2)
              + overlay(Par1, Par2):dr
              + combination
              + combination:dr, verbose = F)
summary(mod1h)$varcomp
##                                        VarComp VarCompSE    Zratio
Constraint
## overlay(Par1, Par2).Ftime-Ftime    1276.49877 750.02253 1.7019472
Positive
## overlay(Par1, Par2):dr.Ftime-Ftime   17.97648  19.91001 0.9028864
Positive
## combination.Ftime-Ftime            1108.86771 329.97440 3.3604659
Positive
## combination:dr.Ftime-Ftime           29.41085  46.54826 0.6318356
Positive
## units.Ftime-Ftime                   422.99303  75.36714 5.6124330
Positive
```

We do hope that you enjoyed this post; if you are interested in diallel models, please, stay tuned: we have other examples on the way.