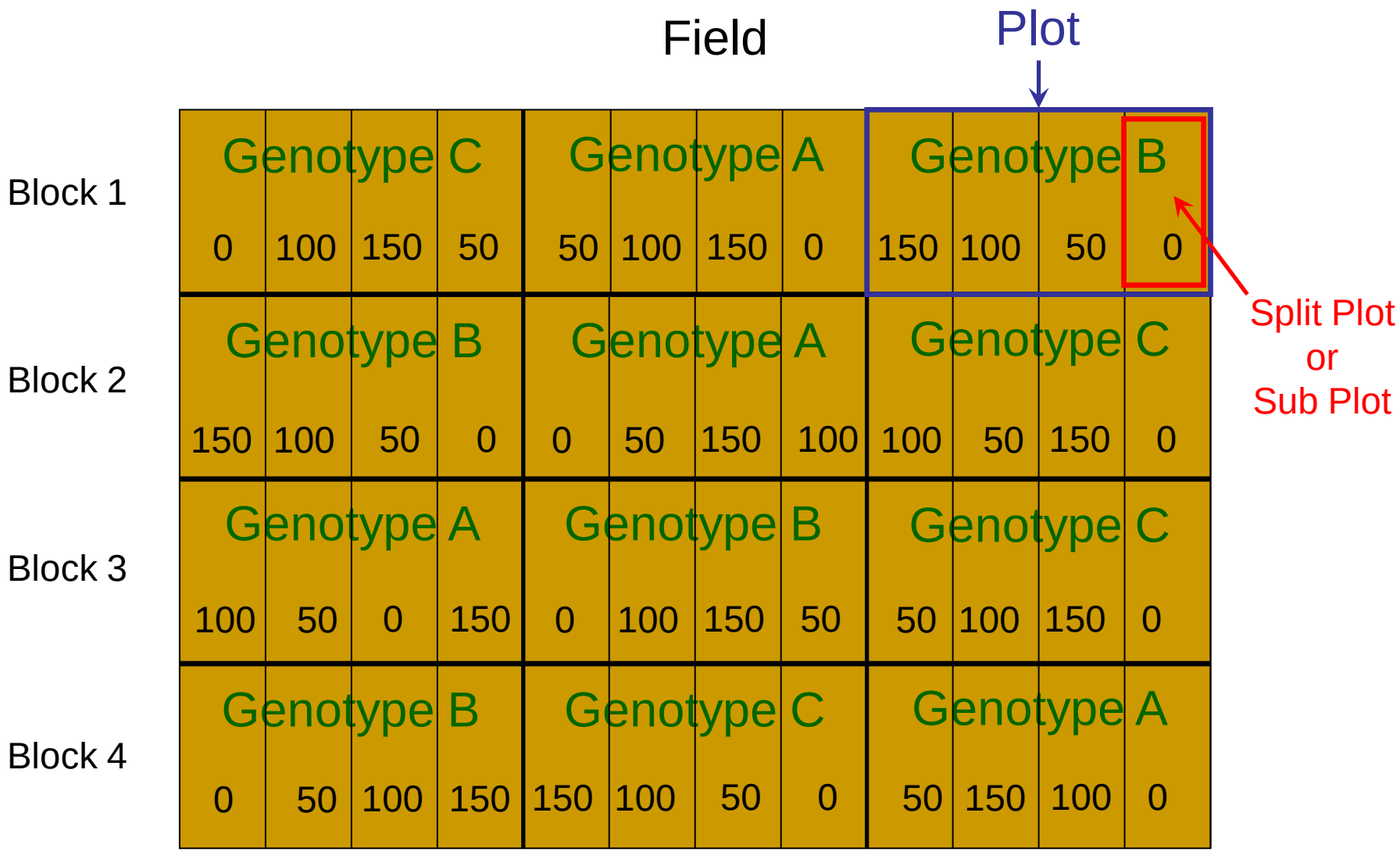
Simulation and Analysis of Data from a Classic Split Plot Experimental Design

Split-Plot Experimental Designs



```
#Example code for simulating data from our
#classic split plot example.

block = factor(rep(1:4, each = 12))
geno = factor(rep(rep(1:3, each = 4), 4))
fert = rep(seq(0, 150, by = 50), 12)

X = model.matrix(~geno + fert + I(fert^2) + geno:fert)
beta = c(125, 15, -10, .4, -0.0015, 0, .2)

Z1 = model.matrix(~ 0 + block)
Z2 = model.matrix(~ 0 + geno:block)
Z = cbind(Z1, Z2)
```

```
#The code below generates the random effects
#and random errors and assembles the response
#vector. The function set.seed is used to
#control the random number generator so that
#the same random effects and errors will be
#generated each time this code is called.

set.seed(532)
u = c(rnorm(4, 0, 6), rnorm(12, 0, 7))
e = rnorm(48, 0, 6)
y = X %*% beta + Z %*% u + e
y = round(y, 1)
d = data.frame(block, geno, fert, y)
d
   block geno fert     y
1      1    1    0 148.7
2      1    1   50 150.4
3      1    1  100 166.7
4      1    1  150 156.5
5      1    2    0 162.5
6      1    2   50 168.6
```

7	1	2	100	180.2
8	1	2	150	181.1
9	1	3	0	144.5
10	1	3	50	177.3
11	1	3	100	188.1
12	1	3	150	199.1
13	2	1	0	114.2
14	2	1	50	131.5
15	2	1	100	150.8
16	2	1	150	139.8
17	2	2	0	141.6
18	2	2	50	150.9
19	2	2	100	171.8
20	2	2	150	187.4
21	2	3	0	107.9
22	2	3	50	138.0
23	2	3	100	161.8
24	2	3	150	163.5
25	3	1	0	126.5
26	3	1	50	138.8
27	3	1	100	134.5

28	3	1	150	140.6
29	3	2	0	129.8
30	3	2	50	155.8
31	3	2	100	168.0
32	3	2	150	164.8
33	3	3	0	100.5
34	3	3	50	139.3
35	3	3	100	150.7
36	3	3	150	158.8
37	4	1	0	114.7
38	4	1	50	138.4
39	4	1	100	141.8
40	4	1	150	143.3
41	4	2	0	160.2
42	4	2	50	162.5
43	4	2	100	178.8
44	4	2	150	171.3
45	4	3	0	102.1
46	4	3	50	126.9
47	4	3	100	142.2
48	4	3	150	152.9

```
#ANOVA-based analysis

o=lm(y~block+geno+block:geno+factor(fert)+geno:factor(fert))

anova(o)
Analysis of Variance Table

Response: y
                   Df Sum Sq Mean Sq F value    Pr(>F)
block               3 5349.5 1783.16 44.9089 1.252e-10
geno                2 5237.2 2618.62 65.9500 4.057e-11
factor(fert)        3 8737.7 2912.57 73.3531 4.233e-13
block:geno          6 1853.4  308.90  7.7796 6.355e-05
geno:factor(fert)   6 1557.3  259.56  6.5370 0.0002381
Residuals          27 1072.1   39.71
---

a = as.matrix(anova(o))
```

```
#ANOVA estimates of variance components:

#Estimate of sigma^2_e

MSE = a[6, 3]
MSE
[1] 39.70613

#Estimate of sigma^2_w

MSBlockGeno = a[4, 3]

(MSBlockGeno - MSE) / 4
[1] 67.2981

#Save the square roots of these estimates
#for comparison with REML estimates computed
#later.

sige = sqrt(MSE)
sigw = sqrt((MSBlockGeno - MSE) / 4)
```

```
#F test for genotype main effects

MSGeno = a[2, 3]
Fstat = MSGeno / MSBlockGeno
Fstat
[1] 8.47728

pval = 1 - pf(Fstat, a[2, 1], a[4, 1])
pval
[1] 0.01785858

#95% confidence interval for geno 2 - geno 1

gmeans = tapply(y, geno, mean)
gmeans
       1        2        3
139.8250 164.7063 147.1000

est = gmeans[2] - gmeans[1]
names(est) = NULL
```

```
#We showed previously that the variance of
#the difference between genotype means
#is 2 * E(MS_block*geno) / (nblocks * nferts)

#Thus, we compute a standard error as

se = sqrt(2 * MSBlockGeno / (4 * 4))

lower = est - qt(.975, a[4, 1]) * se
upper = est + qt(.975, a[4, 1]) * se

c(estimate = est, se = se, lower = lower, upper=upper)
 estimate        se     lower     upper
24.881250  6.213881  9.676431 40.086069
```

```
#REML analysis via lme

library(nlme)

#Below I create f and g factors to shorten
#code and the names that R assigns to the
#elements of beta hat.

f = factor((fert + 50) / 50)
f
 [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2
[39] 3 4 1 2 3 4 1 2 3 4
Levels: 1 2 3 4

g = geno
```

```
o = lme(y ~ g * f, random = ~ 1 | block / g)
o
Linear mixed-effects model fit by REML
  Data: NULL
  Log-restricted-likelihood: -137.5281
  Fixed: y ~ g * f
(Intercept)          g2          g3          f2          f3          f4
    126.025      22.500     -12.275      13.750      22.425      19.025

      g2:f2       g3:f2       g2:f3       g3:f3       g2:f4       g3:f4
     -2.825      17.875       3.750      24.525       8.600      35.800
```

```
Random effects:
 Formula: ~1 | block
        (Intercept)
StdDev:    11.08399

 Formula: ~1 | g %in% block
        (Intercept) Residual
StdDev:    8.203544  6.30128

Number of Observations: 48
Number of Groups:
       block g %in% block
           4           12

#Note that the REML estimates of standard deviation
#match the ANOVA estimates computed
#from lm output.
sigw
[1] 8.203542
sige
[1] 6.30128
```

```
#The ANOVA table computed from lme output
#automatically gives the correct tests for
#genotype, fertilizer, and
#genotype by fertilizer interaction for
#the balanced data case.

anova(o)
            numDF denDF  F-value p-value
(Intercept)     1    27 610.0661  <.0001
g               2     6   8.4773  0.0179
f               3    27  73.3531  <.0001
g:f             6    27   6.5370  0.0002
```

```
#The GLS estimate of the fixed effect
#parameter beta is obtained as follows.

fixed.effects(o)
(Intercept)          g2          g3          f2          f3          f4
    126.025      22.500     -12.275      13.750      22.425      19.025

      g2:f2       g3:f2       g2:f3       g3:f3       g2:f4       g3:f4
     -2.825      17.875       3.750      24.525       8.600      35.800

#The estimated variance covariance matrix of
#the GLS estimator is obtained as follows.

vcov(o)
            (Intercept)          g2          g3          f2          f3          f4
(Intercept)   57.464798 -26.751067 -26.751067  -9.926532  -9.926532  -9.926532
g2           -26.751067  53.502135  26.751067   9.926532   9.926532   9.926532
g3           -26.751067  26.751067  53.502135   9.926532   9.926532   9.926532
f2            -9.926532   9.926532   9.926532  19.853064   9.926532   9.926532
f3            -9.926532   9.926532   9.926532   9.926532  19.853064   9.926532
f4            -9.926532   9.926532   9.926532   9.926532   9.926532  19.853064
g2:f2          9.926532 -19.853064  -9.926532 -19.853064  -9.926532  -9.926532
g3:f2          9.926532  -9.926532 -19.853064 -19.853064  -9.926532  -9.926532
g2:f3          9.926532 -19.853064  -9.926532  -9.926532 -19.853064  -9.926532
g3:f3          9.926532  -9.926532 -19.853064  -9.926532 -19.853064  -9.926532
g2:f4          9.926532 -19.853064  -9.926532  -9.926532  -9.926532 -19.853064
g3:f4          9.926532  -9.926532 -19.853064  -9.926532  -9.926532 -19.853064
```

	g2:f2	g3:f2	g2:f3	g3:f3	g2:f4	g3:f4
(Intercept)	9.926532	9.926532	9.926532	9.926532	9.926532	9.926532
g2	-19.853064	-9.926532	-19.853064	-9.926532	-19.853064	-9.926532
g3	-9.926532	-19.853064	-9.926532	-19.853064	-9.926532	-19.853064
f2	-19.853064	-19.853064	-9.926532	-9.926532	-9.926532	-9.926532
f3	-9.926532	-9.926532	-19.853064	-19.853064	-9.926532	-9.926532
f4	-9.926532	-9.926532	-9.926532	-9.926532	-19.853064	-19.853064
g2:f2	39.706128	19.853064	19.853064	9.926532	19.853064	9.926532
g3:f2	19.853064	39.706128	9.926532	19.853064	9.926532	19.853064
g2:f3	19.853064	9.926532	39.706128	19.853064	19.853064	9.926532
g3:f3	9.926532	19.853064	19.853064	39.706128	9.926532	19.853064
g2:f4	19.853064	9.926532	19.853064	9.926532	39.706128	19.853064
g3:f4	9.926532	19.853064	9.926532	19.853064	19.853064	39.706128

```
#We can use the estimate of beta and it's
#variance covariance matrix to construct
#test statistics and confidence intervals
#for testable and estimable quantities.
#This will work in the unbalanced case
#as well. However, care must be taken to
#assign the appropriate degrees of freedom
#and inferences will be only approximate
#for the unbalanced case and whenever
#variance estimates depend on more than
#one mean square.
```

```
#For example, here is a revised version of the
#confidence interval function that we used for the
#normal theory Gauss-Markov linear model.  The test
#function we previously used could be modified in a
#similar way.
ci = function(lmeout, C, df, a = 0.05)
{
  b = fixed.effects(lmeout)
  V = vcov(lmeout)
  Cb = C %*% b
  se = sqrt(diag(C %*% V %*% t(C)))
  tval = qt(1 – a / 2, df)
  low = Cb – tval * se
  up = Cb + tval * se
  m = cbind(C, Cb, se, low, up)
  dimnames(m)[[2]] = c(paste("c", 1:ncol(C), sep=""),
              "estimate","se",
              paste(100 * (1 - a), "% Conf.", sep=""),
              "limits")
  m
}
```

```
#Suppose would like a confidence interval
#for the genotype 2 mean minus the
#genotype 1 mean while averaging over the
#levels of fertilizer.

#The following table shows the cell means
#in terms of the R parameterization.

#                              f
#############################################################
#     1              2                  3                  4
#############################################################
# g
#
# 1  mu        mu   +f2          mu   +f3          mu   +f4
#
# 2  mu+g2    mu+g2+f2+g2f2    mu+g2+f3+g2f3    mu+g2+f4+g2f4
#
# 3  mu+g3    mu+g3+f2+g3f2    mu+g3+f3+g3f3    mu+g3+f4+g3f4
#
#############################################################
```

```
#The average of row 2 minus the average of row 1 is
#
#  g2 + g2f2/4 + g2f3/4 + g2f4/4
#

C = matrix(c(0,1,0,0,0,0,.25,0,.25,0,.25,0), nrow = 1)

#Note that interval produced below matches
#the interval computed from the lm output.

ci(o, C, 6)

 estimate       se 95% Conf.    limits
 24.88125 6.213883  9.676427  40.08607
```

```
#We can also come up with the coefficients in
#the balanced case using the following code.

X = model.matrix(o)

apply(X[g == 2, ],2,mean) - apply(X[g == 1, ],2,mean)
```

```
(Intercept)          g2          g3          f2          f3          f4
       0.00        1.00        0.00        0.00        0.00        0.00

      g2:f2       g3:f2       g2:f3       g3:f3       g2:f4       g3:f4
       0.25        0.00        0.25        0.00        0.25        0.00
```

```
#We can obtain the best linear unbiased predictions
#(BLUPs) for the random effects as follows.

random.effects(o)
```

```
Level: block
  (Intercept)
1   14.962791
2   -3.260569
3   -6.781226
4   -4.920996
```

```
Level: g %in% block
    (Intercept)
1/1   0.6860200
1/2  -5.7246507
1/3  13.2350305
2/1  -2.1694371
2/2   1.2891661
2/3  -0.9058214
3/1   1.7919168
3/2  -2.8976223
3/3  -2.6089515
4/1  -0.3084997
4/2   7.3331070
4/3  -9.7202576
```

```
#Because we have simulated the data, we can compare
#the predictions with the true values of the random
#effects.
cbind(u, unlist(random.effects(o)))
                                 u
block.(Intercept)1  18.6303551 14.9627915
block.(Intercept)2  -7.9765912 -3.2605692
block.(Intercept)3  -8.7968392 -6.7812260
block.(Intercept)4  -2.0717338 -4.9209963
g.(Intercept)1      -4.5489233  0.6860200
g.(Intercept)2      -1.9147617 -5.7246507
g.(Intercept)3      11.1019481 13.2350305
g.(Intercept)4      -2.3538300 -2.1694371
g.(Intercept)5      14.4051819  1.2891661
g.(Intercept)6       4.7035930 -0.9058214
g.(Intercept)7       3.0466152  1.7919168
g.(Intercept)8       3.8042996 -2.8976223
g.(Intercept)9      -1.0352073 -2.6089515
g.(Intercept)10     -0.8256385 -0.3084997
g.(Intercept)11     10.6835477  7.3331070
g.(Intercept)12    -12.3977860 -9.7202576
```

```
#The same sorts of analyses could be carried out
#using lmer.

library(lme4)
o = lmer(y ~ g * f + (1 | block) + (1 | block:g))
o
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ g * f + (1 | block) + (1 | block:g)
REML criterion at convergence: 275.0563
Random effects:
 Groups   Name        Std.Dev.
 block:g  (Intercept)  8.204
 block    (Intercept) 11.084
 Residual              6.301
Number of obs: 48, groups: block:g, 12; block, 4
Fixed Effects:
(Intercept)          g2          g3          f2          f3          f4
    126.025      22.500     -12.275      13.750      22.425      19.025

      g2:f2       g3:f2       g2:f3       g3:f3       g2:f4       g3:f4
     -2.825      17.875       3.750      24.525       8.600      35.800
```