Chapter 5: More restrictive designs with R

Jixiang Wu

Associate Professor of Quantitative Genetics/Biostatistics

Agronomy, Horticulture, and Plant Science Department

South Dakota State University, Brookings, SD 57007

Phone: 605-688-5947

Email: jixiang.wu@sdstate.edu

Introduction

In this chapter, I will emphasize the use of R to run some data analysis for more restrictive experimental designs: Latin square design, incomplete block design, and rectangular/row-column design.

Latin Design

Generate a latin design

It is very easy to generate a Latin design with R. The following R codes are actually an R function to generate a complete Latin design

```
get_latin_design=function(t){
  L=matrix(0,t,t)
  for(i in 1:t){
     if(i==1)v=1:t
     else v=c(i:t,1:(i-1))
     L[i,]=v
  }
  id=sample(1:t)
  L1=L[id,]
  id=sample(1:t)
  L1=L1[,id]
  rownames(L1)=paste("R",1:t,sep="")
  colnames(L1)=paste("C",1:t,sep="")
  return(L1)

}
```

Once you run this r function, you can use this r function to generate a Latin square designs.For example, the following R code shows you how to use the above function to generate a 5*5 Latin square design.

```
t=5
LT=get_latin_design(t)
data.frame(LT)
```

	C1	C2	C3	C4	C5
R1	1	3	4	5	2
R2	3	5	1	2	4
R3	5	2	3	4	1
R4	4	1	2	3	5
R5	2	4	5	1	3

You can use use this r function to generate another Latin square design.

```
t=5
LT=get_latin_design(t)
data.frame(LT)
```

	C1	C2	C3	C4	C5
R1	5	3	1	2	4
R2	1	4	2	3	5
R3	2	5	3	4	1
R4	4	2	5	1	3
R5	3	1	4	5	2

In factor you can use this R function to generate many Latin designs.

Data analysis for Latin Design

Load a data set

A data set mentioned in our class is avialble in the R package coursedata. You can easily load this data set from this package for our data analysis.

```
require(coursedata)

## Loading required package: coursedata

data(latin)
latin
```

Row	Column	Treat	Yield

1	1	E	59.45
2	1	C	55.16
3	1	B	44.41
4	1	A	42.26
5	1	D	60.89
1	2	A	47.28
2	2	D	60.89
3	2	C	53.72
4	2	B	50.14
5	2	E	59.45
1	3	C	54.44
2	3	B	56.59
3	3	D	55.87
4	3	E	55.87
5	3	A	49.43
1	4	B	50.14
2	4	E	60.17
3	4	A	47.99
4	4	D	58.74
5	4	C	59.45
1	5	D	59.45
2	5	A	48.71
3	5	E	59.45
4	5	C	55.87
5	5	B	57.31

The above data set includes five treatments, five rows and columns. Therefore there are 25 observations in this data set.

```
require(coursedata)
data(latin)
str(latin)
```

```
## 'data.frame':    25 obs. of  4 variables:
##  $ Row   : int  1 2 3 4 5 1 2 3 4 5 ...
##  $ Column: int  1 1 1 1 1 2 2 2 2 2 ...
##  $ Treat : Factor w/ 5 levels "A","B","C","D",..: 5 3 2 1 4 1 4 3 2 5 ...
##  $ Yield : num  59.5 55.2 44.4 42.3 60.9 ...
```

Both variables Column and Row are integers or numerical. Thus these two variables must be converted to categorical variables before you conduct any ANOVA analysis. By doing so, you can use tranform function to convert the data as shown below.

```
latin=transform(latin,Row=factor(Row),Column=factor(Column))
str(latin)
```

```
## 'data.frame':    25 obs. of  4 variables:
##  $ Row   : Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
##  $ Column: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ Treat : Factor w/ 5 levels "A","B","C","D",..: 5 3 2 1 4 1 4 3 2 5 ...
##  $ Yield : num  59.5 55.2 44.4 42.3 60.9 ...
```

Visualize data

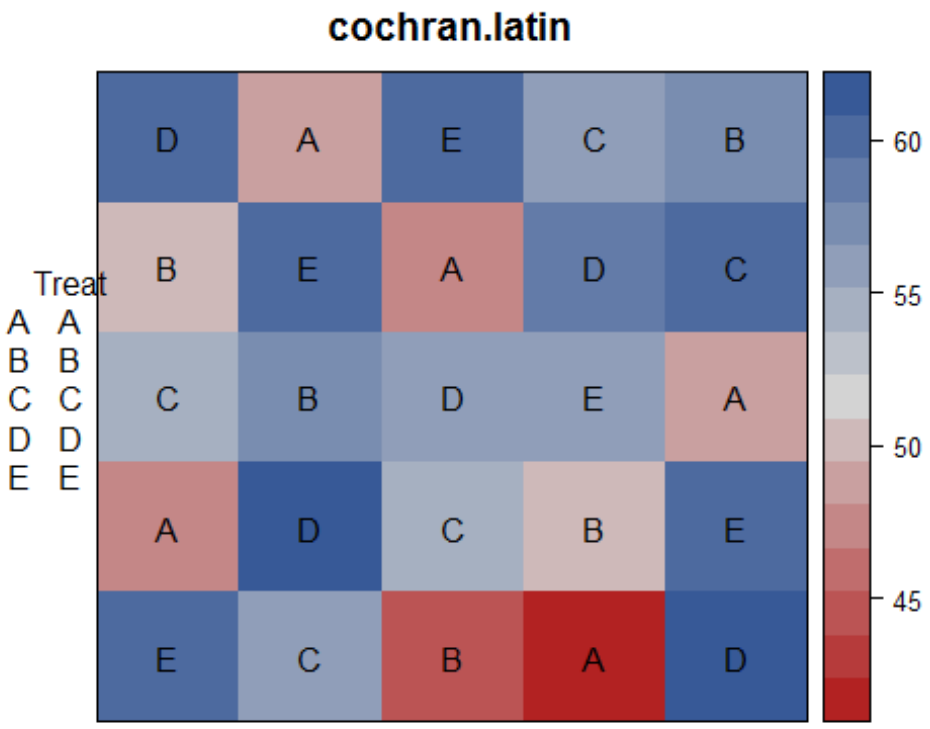
Using the desplot function we can visualize the field pattern. The desplot function is available in the package agridat

```
require(agridat)
```

```
## Loading required package: agridat
```

```
desplot(Yield~Row+Column, latin, text=Treat, cex=1, main="cochran.latin")
```

```
## Warning in Ops.factor(Row, Column): '+' not meaningful for factors
```



Various ANOVA analyses

Full model

With this model, all effects, row effects, column effects, and treatment effects are included.

```
mod0=aov(Yield~Row+Column+Treat, data=latin)
a=summary(mod0)
table0=a[[1]]
data.frame(table0)
```

	Df	Sum.Sq	Mean.Sq	F.value	Pr..F.
Row	4	99.20	24.801	5.255	0.0111
Column	4	38.48	9.620	2.038	0.1527
Treat	4	522.30	130.574	27.669	0.0000
Residuals	12	56.63	4.719	NA	NA

Model without Row effect

With this model, all effects, row effects, column effects, and treatment effects are included.

```
mod1=aov(Yield~Column+Treat, data=latin)
a=summary(mod1)
table1=a[[1]]
data.frame(table1)
```

	Df	Sum.Sq	Mean.Sq	F.value	Pr..F.
Column	4	38.48	9.62	0.9877	0.4422
Treat	4	522.30	130.57	13.4065	0.0001
Residuals	16	155.83	9.74	NA	NA

Model without Column effect

With this model, all effects, row effects, column effects, and treatment effects are included.

```
mod1=aov(Yield~Row+Treat, data=latin)
a=summary(mod1)
table2=a[[1]]
data.frame(table2)
```

	Df	Sum.Sq	Mean.Sq	F.value	Pr..F.
Row	4	99.20	24.801	4.172	0.0168
Treat	4	522.30	130.574	21.966	0.0000
Residuals	16	95.11	5.944	NA	NA

Model without Row effect

With this model, all effects, row effects, column effects, and treatment effects are included.

Model without Row or Column effects

```
mod1=aov(Yield~Treat, data=latin)
a=summary(mod1)
table3=a[[1]]
data.frame(table3)
```

	Df	Sum.Sq	Mean.Sq	F.value	Pr..F.
Treat	4	522.3	130.574	13.44	0
Residuals	20	194.3	9.716	NA	NA

Relative efficiency

It is very easy to calculate an RE using R. Since the full model include row and column effects, there could be several reduced models to be applied, indicating several REs can be calculated. We will still focus on using mean squae error from a full model and a reduced model to calculate an RE.The MSE calculated from the first model is from a full model and the MSE from the second model is a reduced model (Row effect model). We can extract MSE from the results obtained by aov analysis for these two models. Please pay a little attention on how I extracted a MSE from an AOV analyis.

Full model compared with a reduced model without column effect

```
require(coursedata)
data(latin)
latin=transform(latin,Row=factor(Row),Column=factor(Column))
# A full model
mod0=aov(Yield~Treat+Row+Column,data=latin)
a=summary(mod0)

dfe0=mod0$df.residual
rn0=nrow(a[[1]])
mse0=a[[1]][rn0,3]

# A reduced model
mod1=aov(Yield~Treat+Row,data=latin)
a=summary(mod1)
dfe1=mod1$df.residual
rn1=nrow(a[[1]])
mse1=a[[1]][rn1,3]

# Calculate relative efficiency (RE)
RE=mse1/mse0
RE

## [1] 1.26
```

The RE calculated from above is 1.26, showing the Latin design for the wheat experiment is estimated to 25% more efficient compared to an RCB design with row effects.

Full model compared with a reduced model without row effect

```
require(coursedata)
data(latin)
latin=transform(latin,Row=factor(Row),Column=factor(Column))
# A full model
mod0=aov(Yield~Treat+Row+Column,data=latin)
a=summary(mod0)

dfe0=mod0$df.residual
rn0=nrow(a[[1]])
mse0=a[[1]][rn0,3]

# A reduced model
mod1=aov(Yield~Treat+Column,data=latin)
a=summary(mod1)
dfe1=mod1$df.residual
rn1=nrow(a[[1]])
mse1=a[[1]][rn1,3]

# Calculate relative efficiency (RE)
RE=mse1/mse0
RE

## [1] 2.064
```

The RE calculated from above is 2.06, showing the the latin design for the wheat experiment is estimated to be slightly more than twice as efficient as a design considering columns as blocks.Or we can conlude that the design using column as block will require twice replications for each treatment as compared with a latin design. We can also conclude that the row effects should be included to reduce experimental errors effectively.

Full model compared with a reduced model without row or column effect

```
require(coursedata)
data(latin)
latin=transform(latin,Row=factor(Row),Column=factor(Column))
# A full model
mod0=aov(Yield~Treat+Row+Column,data=latin)
a=summary(mod0)

dfe0=mod0$df.residual
rn0=nrow(a[[1]])
mse0=a[[1]][rn0,3]

# A reduced model
mod1=aov(Yield~Treat,data=latin)
a=summary(mod1)
```

```
dfe1=mod1$df.residual
rn1=nrow(a[[1]])
mse1=a[[1]][rn1,3]

# Calculate relative efficiency (RE)
RE=mse1/mse0
RE

## [1] 2.059
```

The RE calculated from above is 2.06, showing the the latin design for the wheat experiment is estimated to be slightly more than twice as efficient as a CR design.Or we can conlude that the CR design will require twice replications for each treatment as compared with a latin design. We can also conclude that the row effects should be included to reduce experimental errors effectively.

Pair-wise Mean Comparison

Once a F-test for treatment effects is significant, it is very reasonable to conduct pair-wise mean comparisons between treatments. Since the model for an RCB design is different from that for a CR design, the procedures in a CR design may not all work for RCB design or other complicated designs because more factors are used for modelling. Now let us try LSD test among these treatments. Again, we need load the package agricolae before we can use the function LSD.test. Again, it is very important to provide the right MSE and its degrees of freedom.

LSD test

```
require(agricolae)

## Loading required package: agricolae

mod=aov(Yield~Treat+Row+Column,data=latin)
a=summary(mod)
dfe=mod$df.residual
rn=nrow(a[[1]])
mse=a[[1]][rn,3]
res=LSD.test(latin$Yield, latin$Treat, DFerror=dfe, MSerror=mse)
res

## $statistics
##    Mean    CV MSerror   LSD
##   54.53 3.984   4.719 2.994
##
## $parameters
##   Df ntr t.value alpha       test      name.t
##   12   5   2.179  0.05 Fisher-LSD latin$Treat
##
## $means
##   latin$Yield   std r   LCL   UCL   Min   Max
## A       47.13 2.840 5 45.02 49.25 42.26 49.43
```

```
## B       51.72 5.324 5 49.60 53.83 44.41 57.31
## C       55.73 2.230 5 53.61 57.84 53.72 59.45
## D       59.17 2.066 5 57.05 61.28 55.87 60.89
## E       58.88 1.710 5 56.76 60.99 55.87 60.17
##
## $comparison
## NULL
##
## $groups
##   trt means M
## 1   D 59.17 a
## 2   E 58.88 a
## 3   C 55.73 b
## 4   B 51.72 c
## 5   A 47.13 d
```

Duncan Test

You can try Duncan test for pair-wise comparisons among treatments.Again, you need to define which factor to be used for pair-wise comparisons.

```
require(agricolae)
mod=aov(Yield~Treat+Row+Column,data=latin)
res=duncan.test(mod, "Treat",main="Yield with different treatment")
res
```

```
## $statistics
##    Mean    CV MSerror
##   54.53 3.984   4.719
##
## $parameters
##   Df ntr alpha   test name.t
##   12   5  0.05 Duncan  Treat
##
## $Duncan
##   Table CriticalRange
## 2 3.081         2.994
## 3 3.225         3.133
## 4 3.312         3.218
## 5 3.370         3.274
##
## $means
##   Yield   std r   Min   Max
## A 47.13 2.840 5 42.26 49.43
## B 51.72 5.324 5 44.41 57.31
## C 55.73 2.230 5 53.72 59.45
## D 59.17 2.066 5 55.87 60.89
## E 58.88 1.710 5 55.87 60.17
##
## $comparison
```

```
## NULL
##
## $groups
##   trt means M
## 1   D 59.17 a
## 2   E 58.88 a
## 3   C 55.73 b
## 4   B 51.72 c
## 5   A 47.13 d
```

Tukey Test

You can try Tukey test for pair-wise comparisons among treatments as well. The procedure is very similar to the Duncan test.

```
require(agricolae)
mod=aov(Yield~Treat+Row+Column,data=latin)
res=HSD.test(mod, "Treat",main="Yield with different treatment")
res

## $statistics
##    Mean    CV MSerror   HSD
##   54.53 3.984   4.719 4.379
##
## $parameters
##   Df ntr StudentizedRange alpha  test name.t
##   12   5            4.508  0.05 Tukey  Treat
##
## $means
##   Yield   std r   Min   Max
## A 47.13 2.840 5 42.26 49.43
## B 51.72 5.324 5 44.41 57.31
## C 55.73 2.230 5 53.72 59.45
## D 59.17 2.066 5 55.87 60.89
## E 58.88 1.710 5 55.87 60.17
##
## $comparison
## NULL
##
## $groups
##   trt means  M
## 1   D 59.17  a
## 2   E 58.88  a
## 3   C 55.73 ab
## 4   B 51.72  b
## 5   A 47.13  c
```

Linear Mixed Model Analyses

You may also try to use linear mixed model approaches to analyze this data set.

Model 1: All effects are fixed

We can run the data analysis without jackknife procedure

```
require(minque)
```

```
## Loading required package: minque
```

```
## Loading required package: klaR
```

```
## Loading required package: MASS
```

```
## Loading required package: Matrix
```

```
res=lmm(Yield~Treat+Row+Column,data=latin)[[1]]
res$Var
```

```
## $Yield
##        Est    SE Chi_sq  P_value
## V(e) 4.719 1.927      6 0.007153
```

```
res$FixedEffect
```

```
## $Yield
##                Est     SE  z_value   P_value
## mu         54.5252 0.5557 98.12506 0.000e+00
## Treat(E)    4.3528 0.8917  4.88162 1.052e-06
## Treat(C)    1.2028 0.8917  1.34893 1.774e-01
## Treat(B)   -2.8072 0.8917 -3.14824 1.643e-03
## Treat(A)   -7.3912 0.8917 -8.28915 1.110e-16
## Treat(D)    4.6428 0.8917  5.20685 1.921e-07
## Row(1)     -0.3732 0.8917 -0.41854 6.756e-01
## Row(2)      1.7788 0.8917  1.99490 4.605e-02
## Row(3)     -2.2372 0.8917 -2.50899 1.211e-02
## Row(4)     -1.9492 0.8917 -2.18601 2.882e-02
## Row(5)      2.7808 0.8917  3.11864 1.817e-03
## Column(1) -2.0912 0.8917 -2.34526 1.901e-02
## Column(2) -0.2292 0.8917 -0.25705 7.971e-01
## Column(3) -0.0852 0.8917 -0.09555 9.239e-01
## Column(4)  0.7728 0.8917  0.86669 3.861e-01
## Column(5)  1.6328 0.8917  1.83117 6.708e-02
```

We can run the data analysis with jackknife procedure.

```
require(minque)
res=lmm.jack(Yield~Treat+Row+Column,data=latin)[[1]]
res$Var
```

```
## $Yield
##      Estimate    SE   PValue 2.5%LL 97.5%UL
## V(e)    4.846 1.047 0.004941  1.148   8.544
```

```
res$PVar
```

```
## $Yield
##         Estimate SE PValue 2.5%LL 97.5%UL
## V(e)/VP        1  0      0      1       1

res$FixedEffect

## $Yield
##           Estimate     SE    PValue  2.5%LL 97.5%UL
## mu        54.52299 0.1782 0.000e+00 53.8937 55.1523
## Treat(E)   4.37295 0.2986 5.548e-07  3.3184  5.4275
## Treat(C)   1.15823 0.1510 1.236e-04  0.6249  1.6915
## Treat(B)  -2.78813 0.3133 3.744e-05 -3.8948 -1.6815
## Treat(A)  -7.38206 0.3957 6.703e-08 -8.7798 -5.9843
## Treat(D)   4.63901 0.1855 5.027e-09  3.9837  5.2943
## Row(1)    -0.42178 0.4662 8.229e-01 -2.0684  1.2249
## Row(2)     1.85384 0.5892 4.627e-02 -0.2274  3.9351
## Row(3)    -2.30553 0.5614 1.055e-02 -4.2886 -0.3224
## Row(4)    -1.91437 0.2711 2.365e-04 -2.8720 -0.9567
## Row(5)     2.78784 0.3529 9.794e-05  1.5412  4.0344
## Column(1) -2.09124 0.5314 1.365e-02 -3.9684 -0.2140
## Column(2) -0.26631 0.3073 8.394e-01 -1.3518  0.8192
## Column(3) -0.09612 0.4385 9.864e-01 -1.6449  1.4527
## Column(4)  0.81937 0.3992 2.490e-01 -0.5907  2.2294
## Column(5)  1.63429 0.2915 1.324e-03  0.6047  2.6639
```

Model 2: Both row and colum effects are random

We can run the data analysis without jackknife procedure

```
require(minque)
res=lmm(Yield~Treat|Row+Column,data=latin)[[1]]
res$Var

## $Yield
##              Est    SE Chi_sq  P_value
## V(Row)    4.0163 3.528 1.2956 0.127506
## V(Column) 0.9802 1.414 0.4805 0.244093
## V(e)      4.7192 1.927 6.0000 0.007153

res$FixedEffect

## $Yield
##             Est     SE z_value   P_value
## mu       54.525 1.1082  49.202 0.000e+00
## Treat(E)  4.353 0.8917   4.882 1.052e-06
## Treat(C)  1.203 0.8917   1.349 1.774e-01
## Treat(B) -2.807 0.8917  -3.148 1.643e-03
## Treat(A) -7.391 0.8917  -8.289 1.110e-16
## Treat(D)  4.643 0.8917   5.207 1.921e-07

res$RandomEffect
```

```
## $Yield
##                Pre    SE  z_value P_value
## Row(1)    -0.33582 1.613 -0.20820  0.8351
## Row(2)     1.60064 1.613  0.99236  0.3210
## Row(3)    -2.01313 1.613 -1.24809  0.2120
## Row(4)    -1.75397 1.613 -1.08742  0.2769
## Row(5)     2.50228 1.613  1.55135  0.1208
## Column(1) -1.49260 0.632 -2.36154  0.0182
## Column(2) -0.16359 0.632 -0.25883  0.7958
## Column(3) -0.06081 0.632 -0.09621  0.9234
## Column(4)  0.55159 0.632  0.87270  0.3828
## Column(5)  1.16542 0.632  1.84388  0.0652
```

We can run the data analysis with jackknife procedure.

```
require(minque)
res=lmm.jack(Yield~Treat|Row+Column,data=latin)[[1]]
res$Var

## $Yield
##           Estimate     SE    PValue 2.5%LL 97.5%UL
## V(Row)      3.8961 0.5589 0.0002611  1.922   5.870
## V(Column)   0.9171 0.6092 0.5010046 -1.235   3.069
## V(e)        4.8335 0.8873 0.0016271  1.699   7.968

res$PVar

## $Yield
##              Estimate      SE    PValue  2.5%LL 97.5%UL
## V(Row)/VP     0.40581 0.05057 8.637e-05  0.2272  0.5844
## V(Column)/VP  0.09291 0.06407 5.317e-01 -0.1334  0.3192
## V(e)/VP       0.50127 0.07238 2.749e-04  0.2456  0.7570

res$FixedEffect

## $Yield
##          Estimate     SE    PValue    2.5%LL 97.5%UL
## mu         54.514 0.2251 0.000e+00 53.718789 55.3090
## Treat(E)    4.333 0.3493 2.320e-06  3.098978  5.5664
## Treat(C)    1.208 0.3402 2.461e-02  0.005892  2.4093
## Treat(B)   -2.768 0.6288 6.845e-03 -4.988683 -0.5466
## Treat(A)   -7.422 0.3405 1.699e-08 -8.624543 -6.2190
## Treat(D)    4.649 0.2551 8.244e-08  3.747902  5.5504

res$RandomEffect

## $Yield
##                Pre     SE    PValue  2.5%LL 97.5%UL
## Row(1)    -0.34111 0.3925 8.384e-01 -1.7276  1.0454
## Row(2)     1.52971 0.3061 2.964e-03  0.4484  2.6110
## Row(3)    -1.91806 0.4654 1.032e-02 -3.5620 -0.2741
## Row(4)    -1.71296 0.2359 1.902e-04 -2.5462 -0.8797
```

```
## Row(5)     2.44242 0.3004 7.785e-05  1.3812  3.5036
## Column(1) -1.31182 0.6108 2.171e-01 -3.4695  0.8458
## Column(2) -0.08232 0.1412 9.333e-01 -0.5810  0.4164
## Column(3) -0.06196 0.3546 9.893e-01 -1.3146  1.1907
## Column(4)  0.48300 0.3144 4.842e-01 -0.6275  1.5935
## Column(5)  0.97310 0.4723 2.462e-01 -0.6952  2.6414
```

Rectangular Design

With increase in number of treatments, it is very difficult to make a Latin design; however, many experimental designs are in rectangular shape. It is that sometimes both row and column effects may exist. With the application of linear mixed model approaches, such a data set can be analyzed with both row and column effects included. Please refer to the following example as a demonstration.

Case 1: Lattice experimental design

The data set that is built in agridat was collected from a RCB design with six treatments and four blocks. The field layout was actually an rectangular design. Let's look at this data set with field visualization.

```
require(agridat)
gnut = ryder.groundnut
gnut
```

block	row	col	gen	wet	dry
B1	4	1	F	3.8	2.3
B1	4	2	A	5.2	3.3
B1	4	3	D	3.0	1.8
B1	4	4	B	2.6	1.8
B1	4	5	E	4.2	2.8
B1	4	6	C	2.4	1.4
B2	3	1	C	1.7	0.9
B2	3	2	F	4.3	2.6
B2	3	3	B	1.9	1.1
B2	3	4	E	3.8	2.4
B2	3	5	D	3.1	2.0
B2	3	6	A	4.8	3.0
B3	2	1	F	2.3	1.2
B3	2	2	D	2.5	1.5
B3	2	3	A	2.4	1.4
B3	2	4	B	4.8	2.8

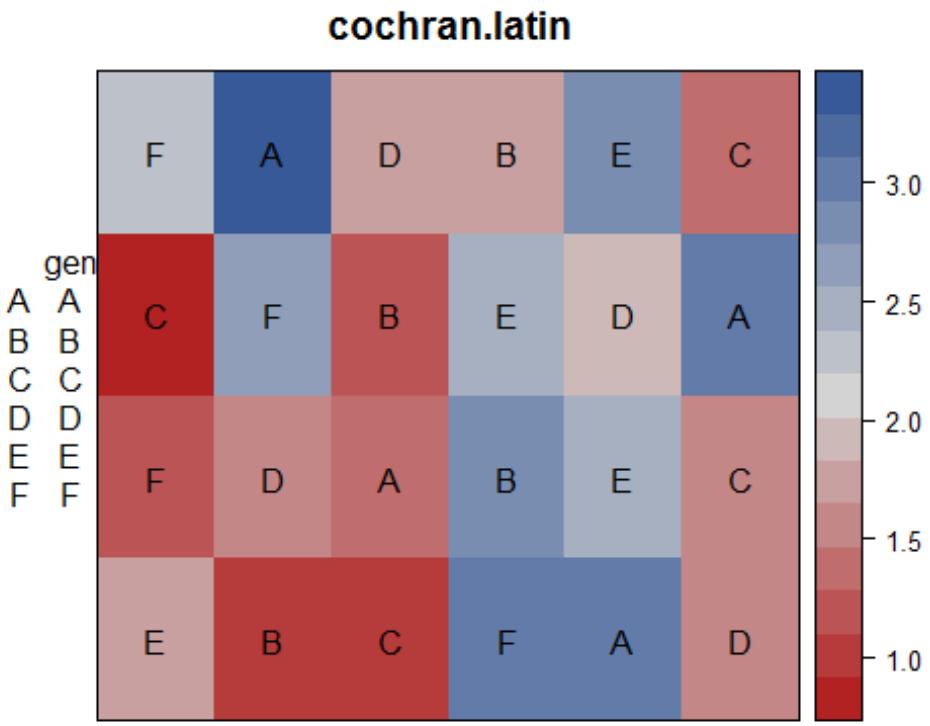
B3	2	5	E	3.8	2.4
B3	2	6	C	2.5	1.5
B4	1	1	E	2.8	1.7
B4	1	2	B	1.8	1.0
B4	1	3	C	2.0	1.0
B4	1	4	F	5.0	3.0
B4	1	5	A	4.8	3.1
B4	1	6	D	2.5	1.5

```
gnut=transform(gnut,row=factor(row),col=factor(col))
str(gnut)
```

```
## 'data.frame':    24 obs. of  6 variables:
##  $ block: Factor w/ 4 levels "B1","B2","B3",..: 1 1 1 1 1 1 2 2 2 2 ...
##  $ row  : Factor w/ 4 levels "1","2","3","4": 4 4 4 4 4 4 3 3 3 3 ...
##  $ col  : Factor w/ 6 levels "1","2","3","4",..: 1 2 3 4 5 6 1 2 3 4 ...
##  $ gen  : Factor w/ 6 levels "A","B","C","D",..: 6 1 4 2 5 3 3 6 2 5 ...
##  $ wet  : num  3.8 5.2 3 2.6 4.2 2.4 1.7 4.3 1.9 3.8 ...
##  $ dry  : num  2.3 3.3 1.8 1.8 2.8 1.4 0.9 2.6 1.1 2.4 ...
```

```
desplot(dry~col+row, gnut, text=gen, cex=1, main="cochran.latin")
```

```
## Warning in Ops.factor(col, row): '+' not meaningful for factors
```



RCB design analysis

First we may only consider this is an RCB design. The following R scripts show ananlysis of variance and three pair-wise mean comparisons among genotypes.Three types of pair-wise comparisons are provided here.

```
mod=aov(dry~block+gen,data=gnut)
a=summary(mod)
dfe=mod$df.residual
rn=nrow(a[[1]])
table=a[[1]]
data.frame(table)
```

	Df	Sum.Sq	Mean.Sq	F.value	Pr..F.
block	3	0.6379	0.2126	0.4844	0.6981
gen	5	6.0171	1.2034	2.7414	0.0594
Residuals	15	6.5846	0.4390	NA	NA

```
mse=a[[1]][rn,3]
res=LSD.test(gnut$dry, gnut$gen, DFerror=dfe, MSerror=mse)
res
```

```
## $statistics
##    Mean    CV MSerror    LSD
##   1.979 33.48   0.439 0.9986
##
## $parameters
##   Df ntr t.value alpha       test   name.t
##   15   6   2.131  0.05 Fisher-LSD gnut$gen
##
## $means
##   gnut$dry    std r    LCL   UCL Min Max
## A    2.700 0.8756 4 1.9939 3.406 1.4 3.3
## B    1.675 0.8302 4 0.9689 2.381 1.0 2.8
## C    1.200 0.2944 4 0.4939 1.906 0.9 1.5
## D    1.700 0.2449 4 0.9939 2.406 1.5 2.0
## E    2.325 0.4573 4 1.6189 3.031 1.7 2.8
## F    2.275 0.7719 4 1.5689 2.981 1.2 3.0
##
## $comparison
## NULL
##
## $groups
##   trt means  M
## 1   A 2.700  a
## 2   E 2.325 ab
## 3   F 2.275 ab
## 4   D 1.700 bc
```

```
## 5   B 1.675 bc
## 6   C 1.200  c
```

```
res=duncan.test(mod, "gen",main="Dry weight with different genotypes")
res
```

```
## $statistics
##    Mean    CV MSerror
##   1.979 33.48   0.439
##
## $parameters
##   Df ntr alpha   test name.t
##   15   6  0.05 Duncan    gen
##
## $Duncan
##   Table CriticalRange
## 2 3.014        0.9986
## 3 3.160        1.0468
## 4 3.250        1.0767
## 5 3.312        1.0971
## 6 3.356        1.1118
##
## $means
##     dry    std r Min Max
## A 2.700 0.8756 4 1.4 3.3
## B 1.675 0.8302 4 1.0 2.8
## C 1.200 0.2944 4 0.9 1.5
## D 1.700 0.2449 4 1.5 2.0
## E 2.325 0.4573 4 1.7 2.8
## F 2.275 0.7719 4 1.2 3.0
##
## $comparison
## NULL
##
## $groups
##   trt means  M
## 1   A 2.700  a
## 2   E 2.325  a
## 3   F 2.275 ab
## 4   D 1.700 ab
## 5   B 1.675 ab
## 6   C 1.200  b
```

```
res=HSD.test(mod, "gen",main="Dry weight with different genotypes")
res
```

```
## $statistics
##    Mean    CV MSerror   HSD
##   1.979 33.48   0.439 1.522
##
## $parameters
```

```
##   Df ntr StudentizedRange alpha  test name.t
##   15   6            4.595  0.05 Tukey    gen
##
## $means
##     dry    std r Min Max
## A 2.700 0.8756 4 1.4 3.3
## B 1.675 0.8302 4 1.0 2.8
## C 1.200 0.2944 4 0.9 1.5
## D 1.700 0.2449 4 1.5 2.0
## E 2.325 0.4573 4 1.7 2.8
## F 2.275 0.7719 4 1.2 3.0
##
## $comparison
## NULL
##
## $groups
##   trt means M
## 1   A 2.700 a
## 2   E 2.325 a
## 3   F 2.275 a
## 4   D 1.700 a
## 5   B 1.675 a
## 6   C 1.200 a
```

Linear Mixed Model Analysis

If we include both row and column effects in a model, we can try to use a linear mixed model approache

```
require(agridat)
require(minque)
gnut = ryder.groundnut
res = lmm.jack(dry~gen|row+col,method="reml",data=gnut)[[1]]
res$Var

## $dry
##        Estimate      SE   PValue   2.5%LL 97.5%UL
## V(row) 0.007472 0.02224 0.975904 -0.07110 0.08604
## V(col) 0.163879 0.03788 0.007637  0.03008 0.29768
## V(e)   0.260819 0.05980 0.007253  0.04960 0.47204

res$PVar

## $dry
##           Estimate      SE   PValue  2.5%LL 97.5%UL
## V(row)/VP  0.02085 0.06291 0.976396 -0.2013  0.2431
## V(col)/VP  0.37886 0.06774 0.001348  0.1396  0.6181
## V(e)/VP    0.60029 0.11158 0.001777  0.2062  0.9944

res$FixedEffect
```

```
## $dry
##        Estimate      SE    PValue   2.5%LL  97.5%UL
## mu       1.9825 0.05290 1.362e-10  1.79560  2.16935
## gen(F)   0.3778 0.09600 1.365e-02  0.03865  0.71688
## gen(A)   0.7459 0.08963 6.449e-05  0.42934  1.06254
## gen(D)  -0.2712 0.08865 5.307e-02 -0.58432  0.04192
## gen(B)  -0.4429 0.17236 1.147e-01 -1.05168  0.16596
## gen(E)   0.2133 0.08934 1.518e-01 -0.10228  0.52888
## gen(C)  -0.6229 0.07191 4.660e-05 -0.87694 -0.36895

res$RandomEffect

## $dry
##                Pre       SE    PValue   2.5%LL  97.5%UL
## row(4)  0.0402818 0.099311 0.9670408 -0.31051  0.39107
## row(3)  0.0009912 0.013147 0.9938467 -0.04545  0.04743
## row(2) -0.0391930 0.106443 0.9720684 -0.41518  0.33679
## row(1) -0.0020800 0.005117 0.9669166 -0.02016  0.01600
## col(1) -0.4517463 0.067266 0.0003478 -0.68935 -0.21414
## col(2)  0.0256044 0.081937 0.9784166 -0.26382  0.31503
## col(3) -0.4589110 0.117396 0.0141902 -0.87359 -0.04424
## col(4)  0.5060805 0.102810 0.0032825  0.14293  0.86923
## col(5)  0.3278135 0.059562 0.0015124  0.11742  0.53820
## col(6)  0.0511590 0.054189 0.8051377 -0.14025  0.24257
```

Incomplete Block Design

The data set coticb used for this section came from one of my collaborators at one USDA-ARS research unit. Due to the large number of genotypes to be evaluated, it was very difficult to find a field block which was uniform within each field block. To avoid this issue, my collaborator divided each major block into small sub-blocks, where these genotypes were assigned to each sub-block. The experiment was repeated three times. Two important agronomic traits, lint yield (LY) and 100-seed weight (SI) were privided here. Please use the following R codes to load this data set and change the names for these columns.

```
require(coursedata)
data(coticb)
names(coticb)=c("Block","SBlock","Genotype","LY","SI")
coticb=transform(coticb,Genotype=factor(Genotype),Block=factor(Block),SBlock=
factor(SBlock))
head(coticb)
```

Block	SBlock	Genotype	LY	SI
2	2	62	467	6.0
2	2	23	609	6.2
4	18	62	583	7.3
4	12	111	562	8.5

4	22	71	614	8.2
4	5	117	649	7.2

Linear mixed model analysis without jackknife

In the following analysis, both block and sub-block effects are treated as random while genotype treated as fixed effects.

```
res=lmm(LY~Genotype|Block+SBlock:Block,data=coticb)[[1]]
data.frame(res$Var)
```

	LY.Est	LY.SE	LY.Chi_sq	LY.P_value
V(Block)	59706	51231	1.358	0.1219
V(Block:SBlock)	62533	9863	40.197	0.0000
V(e)	139544	3412	1672.604	0.0000

```
data.frame(res$FixedEffect)
```

	LY.Est	LY.SE	LY.z_value	LY.P_value
mu	1838.119	119.53	15.3785	0.0000
Genotype(62)	-895.219	125.00	-7.1618	0.0000
Genotype(23)	-458.567	121.98	-3.7595	0.0002
Genotype(111)	-75.585	126.71	-0.5965	0.5508
Genotype(71)	-546.114	127.52	-4.2826	0.0000
Genotype(117)	-236.362	129.32	-1.8277	0.0676
Genotype(78)	-138.700	122.59	-1.1314	0.2579
Genotype(40)	-531.777	122.85	-4.3287	0.0000
Genotype(127)	-275.347	122.65	-2.2450	0.0248
Genotype(84)	-269.116	125.23	-2.1490	0.0316
Genotype(108)	150.775	126.27	1.1941	0.2325
Genotype(37)	53.296	124.73	0.4273	0.6692
Genotype(124)	341.763	126.29	2.7062	0.0068
Genotype(88)	56.555	125.80	0.4496	0.6530
Genotype(91)	44.065	131.94	0.3340	0.7384
Genotype(19)	330.178	124.69	2.6480	0.0081
Genotype(138)	-192.431	124.27	-1.5485	0.1215
Genotype(102)	-267.963	126.71	-2.1148	0.0344
Genotype(70)	-183.971	131.43	-1.3997	0.1616
Genotype(141)	-239.194	127.53	-1.8756	0.0607
Genotype(110)	-169.696	126.11	-1.3456	0.1784

Genotype(122)	6.937	129.26	0.0537	0.9572
Genotype(75)	-155.174	125.74	-1.2341	0.2172
Genotype(18)	260.433	125.29	2.0787	0.0376
Genotype(125)	22.101	105.91	0.2087	0.8347
Genotype(63)	100.195	122.91	0.8152	0.4149
Genotype(73)	-174.821	123.77	-1.4125	0.1578
Genotype(55)	-302.033	124.04	-2.4349	0.0149
Genotype(43)	482.690	126.98	3.8014	0.0001
Genotype(5)	112.547	124.62	0.9031	0.3665
Genotype(90)	66.694	126.66	0.5266	0.5985
Genotype(68)	451.427	127.84	3.5312	0.0004
Genotype(44)	9.594	127.32	0.0754	0.9399
Genotype(74)	-122.063	124.83	-0.9779	0.3281
Genotype(60)	-365.413	128.76	-2.8378	0.0045
Genotype(143)	-491.431	126.29	-3.8913	0.0001
Genotype(120)	215.490	125.60	1.7157	0.0862
Genotype(134)	-192.445	124.94	-1.5404	0.1235
Genotype(34)	97.703	124.92	0.7821	0.4341
Genotype(121)	-109.279	129.27	-0.8454	0.3979
Genotype(136)	-169.728	119.76	-1.4173	0.1564
Genotype(107)	-30.829	128.76	-0.2394	0.8108
Genotype(56)	19.565	128.42	0.1524	0.8789
Genotype(109)	117.442	129.72	0.9053	0.3653
Genotype(17)	-243.299	120.88	-2.0128	0.0441
Genotype(106)	-172.124	123.29	-1.3961	0.1627
Genotype(8)	451.335	123.08	3.6671	0.0002
Genotype(67)	11.174	120.98	0.0924	0.9264
Genotype(14)	176.845	127.61	1.3858	0.1658
Genotype(104)	158.217	113.05	1.3995	0.1617
Genotype(4)	292.217	120.54	2.4243	0.0153
Genotype(99)	-363.387	126.19	-2.8797	0.0040
Genotype(129)	-164.926	127.53	-1.2932	0.1959
Genotype(77)	-125.766	127.21	-0.9886	0.3228
Genotype(13)	158.153	117.52	1.3458	0.1784
Genotype(53)	112.133	117.36	0.9555	0.3393
Genotype(20)	405.706	116.17	3.4924	0.0005

Genotype(145)	113.186	122.51	0.9239	0.3555
Genotype(87)	136.815	131.25	1.0424	0.2972
Genotype(10)	-163.068	119.38	-1.3660	0.1719
Genotype(81)	306.295	126.63	2.4188	0.0156
Genotype(146)	369.530	126.33	2.9252	0.0034
Genotype(9)	9.704	123.96	0.0783	0.9376
Genotype(42)	42.537	124.86	0.3407	0.7333
Genotype(65)	257.546	127.74	2.0161	0.0438
Genotype(50)	458.233	117.73	3.8924	0.0001
Genotype(135)	385.973	135.18	2.8552	0.0043
Genotype(47)	-239.907	128.07	-1.8732	0.0610
Genotype(16)	285.428	121.44	2.3504	0.0188
Genotype(123)	-223.744	117.43	-1.9054	0.0567
Genotype(31)	614.309	122.19	5.0275	0.0000
Genotype(32)	424.947	125.89	3.3755	0.0007
Genotype(116)	179.761	128.54	1.3985	0.1620
Genotype(144)	-416.853	127.44	-3.2709	0.0011
Genotype(113)	-193.802	125.30	-1.5467	0.1219
Genotype(41)	-460.929	121.75	-3.7860	0.0002
Genotype(46)	-2.153	117.30	-0.0184	0.9854
Genotype(25)	-507.149	124.33	-4.0792	0.0000
Genotype(27)	-218.764	99.30	-2.2031	0.0276
Genotype(131)	-156.131	97.26	-1.6053	0.1084
Genotype(7)	74.498	123.59	0.6028	0.5467
Genotype(66)	-363.175	127.73	-2.8433	0.0045
Genotype(118)	-359.883	128.63	-2.7977	0.0051
Genotype(28)	-100.125	128.09	-0.7817	0.4344
Genotype(52)	-45.678	137.16	-0.3330	0.7391
Genotype(2)	234.528	128.21	1.8292	0.0674
Genotype(103)	-362.362	117.71	-3.0785	0.0021
Genotype(36)	99.929	128.49	0.7777	0.4367
Genotype(89)	116.883	114.78	1.0183	0.3085
Genotype(39)	276.481	120.54	2.2937	0.0218
Genotype(22)	-3.625	125.39	-0.0289	0.9769
Genotype(58)	115.755	127.52	0.9077	0.3640
Genotype(29)	438.639	126.79	3.4596	0.0005

Genotype(79)	-111.536	118.60	-0.9405	0.3470
Genotype(6)	254.950	120.02	2.1242	0.0337
Genotype(38)	59.054	106.70	0.5534	0.5800
Genotype(64)	9.239	117.77	0.0784	0.9375
Genotype(85)	176.233	115.43	1.5268	0.1268
Genotype(114)	-277.961	124.91	-2.2252	0.0261
Genotype(49)	116.796	118.44	0.9861	0.3241
Genotype(48)	160.615	131.99	1.2168	0.2237
Genotype(112)	-237.657	115.89	-2.0507	0.0403
Genotype(130)	162.804	125.87	1.2934	0.1959
Genotype(96)	-49.366	95.56	-0.5166	0.6054
Genotype(3)	505.905	128.03	3.9514	0.0001
Genotype(98)	403.698	127.44	3.1678	0.0015
Genotype(132)	-68.831	111.50	-0.6173	0.5370
Genotype(35)	-263.541	125.92	-2.0929	0.0364
Genotype(51)	-5.118	127.67	-0.0401	0.9680
Genotype(72)	201.500	126.29	1.5955	0.1106
Genotype(97)	-450.363	108.76	-4.1409	0.0000
Genotype(76)	-18.689	117.54	-0.1590	0.8737
Genotype(54)	-201.647	107.60	-1.8741	0.0609
Genotype(61)	-448.304	118.16	-3.7940	0.0001
Genotype(59)	-1193.526	88.30	-13.5165	0.0000
Genotype(93)	208.353	127.55	1.6335	0.1024
Genotype(142)	-1376.788	128.22	-10.7373	0.0000
Genotype(119)	91.346	129.69	0.7043	0.4812
Genotype(86)	105.213	105.45	0.9978	0.3184
Genotype(45)	258.464	130.08	1.9869	0.0469
Genotype(139)	-170.571	107.96	-1.5799	0.1141
Genotype(80)	94.155	107.71	0.8741	0.3820
Genotype(105)	-248.204	119.00	-2.0857	0.0370
Genotype(128)	-208.863	124.46	-1.6781	0.0933
Genotype(69)	-567.278	117.35	-4.8340	0.0000
Genotype(33)	1153.015	135.73	8.4952	0.0000
Genotype(101)	128.824	108.88	1.1832	0.2367
Genotype(137)	-855.880	126.20	-6.7821	0.0000
Genotype(126)	194.433	130.56	1.4892	0.1364

Genotype(57)	45.232	125.23	0.3612	0.7180
Genotype(83)	-344.822	126.89	-2.7175	0.0066
Genotype(95)	115.463	126.64	0.9118	0.3619
Genotype(82)	-67.258	128.54	-0.5233	0.6008
Genotype(94)	274.708	126.36	2.1740	0.0297
Genotype(100)	519.514	129.50	4.0116	0.0001
Genotype(140)	724.845	120.35	6.0228	0.0000
Genotype(30)	259.079	127.67	2.0293	0.0424
Genotype(133)	-52.820	128.00	-0.4127	0.6799
Genotype(1)	313.446	129.91	2.4128	0.0158
Genotype(15)	231.562	126.46	1.8310	0.0671
Genotype(12)	302.657	128.31	2.3588	0.0183
Genotype(115)	180.908	124.19	1.4567	0.1452
Genotype(92)	427.806	123.89	3.4530	0.0006
Genotype(11)	328.978	128.10	2.5682	0.0102
Genotype(24)	192.823	129.35	1.4908	0.1360
Genotype(21)	275.267	128.47	2.1426	0.0321
Genotype(26)	576.035	129.27	4.4561	0.0000

```
data.frame(res$RandomEffect)
```

	LY.Pre	LY.SE	LY.z_value	LY.P_value
Block(2)	-3.9218	206.4	-0.0190	0.9848
Block(4)	-334.7861	206.4	-1.6218	0.1048
Block(3)	238.8492	206.4	1.1571	0.2472
Block(1)	99.8586	206.4	0.4837	0.6286
Block:SBlock(2:2)	-190.3361	190.2	-1.0009	0.3169
Block:SBlock(4:18)	275.8694	182.0	1.5156	0.1296
Block:SBlock(4:12)	-93.9170	190.0	-0.4942	0.6211
Block:SBlock(4:22)	43.8791	190.0	0.2309	0.8174
Block:SBlock(4:5)	-293.8701	192.3	-1.5280	0.1265
Block:SBlock(2:16)	-258.6936	190.7	-1.3568	0.1749
Block:SBlock(4:11)	114.0365	189.6	0.6014	0.5476
Block:SBlock(3:1)	-98.4864	190.0	-0.5183	0.6043
Block:SBlock(4:14)	-58.3338	191.2	-0.3051	0.7603
Block:SBlock(4:30)	-278.5990	190.1	-1.4657	0.1427
Block:SBlock(3:30)	-852.1742	190.0	-4.4843	0.0000
Block:SBlock(2:6)	-230.9932	191.1	-1.2085	0.2268

Block:SBlock(1:25)	-402.9339	181.7	-2.2173	0.0266
Block:SBlock(2:5)	-351.2721	190.2	-1.8470	0.0648
Block:SBlock(4:17)	13.1972	189.8	0.0695	0.9446
Block:SBlock(4:7)	-219.9430	181.7	-1.2103	0.2262
Block:SBlock(2:20)	-126.7498	191.3	-0.6626	0.5076
Block:SBlock(2:17)	-51.9281	191.9	-0.2707	0.7867
Block:SBlock(2:1)	-415.4447	228.6	-1.8175	0.0691
Block:SBlock(1:21)	-327.0134	190.1	-1.7204	0.0854
Block:SBlock(1:24)	260.6749	182.0	1.4321	0.1521
Block:SBlock(1:4)	-374.8866	294.4	-1.2735	0.2028
Block:SBlock(4:27)	-177.7113	190.9	-0.9309	0.3519
Block:SBlock(4:6)	124.9801	190.2	0.6570	0.5112
Block:SBlock(4:15)	-181.9424	190.5	-0.9552	0.3395
Block:SBlock(4:20)	-170.4549	192.7	-0.8847	0.3763
Block:SBlock(4:3)	52.0959	189.8	0.2745	0.7837
Block:SBlock(1:22)	-373.8368	192.3	-1.9437	0.0519
Block:SBlock(4:10)	272.4879	190.5	1.4306	0.1526
Block:SBlock(3:8)	-299.3893	182.1	-1.6444	0.1001
Block:SBlock(4:2)	-269.6077	189.6	-1.4219	0.1551
Block:SBlock(2:25)	5.2052	191.7	0.0271	0.9783
Block:SBlock(1:23)	272.8250	189.8	1.4373	0.1506
Block:SBlock(4:1)	-70.6402	192.4	-0.3671	0.7135
Block:SBlock(2:13)	-288.6299	190.8	-1.5130	0.1303
Block:SBlock(4:23)	-120.6823	180.9	-0.6671	0.5047
Block:SBlock(2:9)	-192.2670	181.0	-1.0622	0.2882
Block:SBlock(4:13)	179.6109	190.1	0.9448	0.3447
Block:SBlock(1:29)	-350.0938	192.4	-1.8193	0.0689
Block:SBlock(4:26)	-37.4601	209.0	-0.1793	0.8577
Block:SBlock(1:17)	-387.1219	190.2	-2.0354	0.0418
Block:SBlock(3:25)	-458.7187	191.0	-2.4023	0.0163
Block:SBlock(4:21)	-107.8407	190.2	-0.5670	0.5707
Block:SBlock(2:24)	-255.9550	193.4	-1.3233	0.1857
Block:SBlock(2:8)	-294.8176	190.0	-1.5514	0.1208
Block:SBlock(4:19)	121.7773	189.4	0.6430	0.5202
Block:SBlock(1:26)	-34.3799	189.6	-0.1813	0.8561
Block:SBlock(4:16)	-232.5916	191.1	-1.2174	0.2235

Block:SBlock(2:21)	-212.5677	190.8	-1.1140	0.2653
Block:SBlock(3:12)	-249.5980	190.2	-1.3124	0.1894
Block:SBlock(3:15)	39.6601	181.0	0.2191	0.8266
Block:SBlock(4:4)	205.2770	190.3	1.0787	0.2807
Block:SBlock(2:12)	-60.4425	190.0	-0.3181	0.7504
Block:SBlock(1:15)	-90.6987	190.5	-0.4762	0.6339
Block:SBlock(1:11)	-36.3676	190.2	-0.1912	0.8484
Block:SBlock(1:30)	-95.2184	189.8	-0.5018	0.6158
Block:SBlock(1:3)	41.9833	190.9	0.2199	0.8259
Block:SBlock(2:10)	0.8324	182.1	0.0046	0.9964
Block:SBlock(3:26)	-43.2240	190.7	-0.2267	0.8207
Block:SBlock(3:17)	-116.4110	190.8	-0.6101	0.5418
Block:SBlock(1:10)	156.8701	189.9	0.8263	0.4087
Block:SBlock(1:27)	33.5690	190.0	0.1766	0.8598
Block:SBlock(1:14)	543.2001	189.4	2.8684	0.0041
Block:SBlock(3:16)	-327.8341	188.2	-1.7417	0.0816
Block:SBlock(1:20)	-42.5645	180.9	-0.2353	0.8140
Block:SBlock(1:19)	-50.3116	191.1	-0.2633	0.7923
Block:SBlock(4:8)	-301.5873	192.0	-1.5711	0.1162
Block:SBlock(2:4)	488.7208	195.0	2.5059	0.0122
Block:SBlock(1:8)	-88.6394	290.3	-0.3053	0.7601
Block:SBlock(1:6)	-93.1771	192.7	-0.4836	0.6287
Block:SBlock(2:15)	-136.8744	200.2	-0.6836	0.4942
Block:SBlock(4:9)	429.9443	189.9	2.2646	0.0235
Block:SBlock(3:29)	-304.9682	195.0	-1.5637	0.1179
Block:SBlock(1:9)	379.9388	190.5	1.9947	0.0461
Block:SBlock(1:2)	20.6956	192.0	0.1078	0.9141
Block:SBlock(4:24)	7.1429	294.4	0.0243	0.9806
Block:SBlock(4:29)	266.5576	190.4	1.4003	0.1614
Block:SBlock(4:28)	43.4246	181.3	0.2395	0.8107
Block:SBlock(3:20)	242.8965	190.2	1.2773	0.2015
Block:SBlock(1:5)	23.6081	190.1	0.1242	0.9012
Block:SBlock(3:21)	8.5457	191.3	0.0447	0.9644
Block:SBlock(1:7)	223.3683	189.6	1.1780	0.2388
Block:SBlock(2:14)	452.0417	191.0	2.3673	0.0179
Block:SBlock(1:18)	392.4571	190.4	2.0617	0.0392

Block:SBlock(1:1)	112.8462	190.0	0.5939	0.5526
Block:SBlock(1:16)	170.8759	181.3	0.9423	0.3460
Block:SBlock(2:29)	440.4906	190.0	2.3180	0.0205
Block:SBlock(1:13)	-13.6581	190.3	-0.0718	0.9428
Block:SBlock(4:25)	123.5714	290.3	0.4256	0.6704
Block:SBlock(3:28)	-4.4503	193.4	-0.0230	0.9816
Block:SBlock(1:28)	343.9507	191.2	1.7991	0.0720
Block:SBlock(2:3)	19.6343	194.3	0.1010	0.9195
Block:SBlock(3:7)	156.2588	262.3	0.5958	0.5513
Block:SBlock(3:4)	-128.8748	189.7	-0.6792	0.4970
Block:SBlock(2:28)	71.7025	200.0	0.3585	0.7199
Block:SBlock(2:30)	-140.3684	199.0	-0.7052	0.4807
Block:SBlock(3:13)	46.9076	190.0	0.2469	0.8050
Block:SBlock(1:12)	-114.1509	209.0	-0.5462	0.5849
Block:SBlock(3:11)	-90.0284	190.2	-0.4732	0.6360
Block:SBlock(2:26)	-43.1984	262.3	-0.1647	0.8692
Block:SBlock(3:27)	235.2670	191.9	1.2263	0.2201
Block:SBlock(2:11)	94.3378	190.2	0.4959	0.6200
Block:SBlock(3:6)	283.3806	194.3	1.4583	0.1448
Block:SBlock(3:24)	164.1651	190.0	0.8639	0.3876
Block:SBlock(3:19)	321.2074	191.1	1.6805	0.0929
Block:SBlock(2:7)	-169.1261	190.7	-0.8867	0.3753
Block:SBlock(2:23)	217.7394	190.0	1.1458	0.2519
Block:SBlock(3:5)	-63.5248	200.0	-0.3176	0.7508
Block:SBlock(3:2)	249.2397	190.0	1.3116	0.1897
Block:SBlock(2:22)	118.0152	189.4	0.6232	0.5331
Block:SBlock(3:3)	249.2960	191.7	1.3003	0.1935
Block:SBlock(3:22)	126.9987	199.0	0.6381	0.5234
Block:SBlock(2:19)	551.3740	188.2	2.9293	0.0034
Block:SBlock(3:9)	188.7415	190.8	0.9894	0.3225
Block:SBlock(2:27)	508.1761	190.0	2.6746	0.0075
Block:SBlock(2:18)	447.3960	189.7	2.3579	0.0184
Block:SBlock(3:23)	267.1033	228.6	1.1685	0.2426
Block:SBlock(3:10)	280.8406	200.2	1.4026	0.1607
Block:SBlock(3:14)	90.9995	190.7	0.4771	0.6333
Block:SBlock(3:18)	329.6916	189.4	1.7411	0.0817

Linear mixed model analysis with jackknife

In the following analysis, the above model is anlayzed by a linear mixed model approach with a Jackknife approach applied.

```
res=lmm.jack(LY~Genotype|Block+SBlock:Block,data=coticb)[[1]]

data.frame(res$Var)
```

	LY.Estimate	LY.SE	LY.PValue	LY.2.5.LL	LY.97.5.UL
V(Block)	60332	4383	0e+00	44849	75814
V(Block:SBlock)	62535	7383	1e-04	36455	88614
V(e)	139150	3355	0e+00	127301	150999

```
data.frame(res$PVar)
```

	LY.Estimate	LY.SE	LY.PValue	LY.2.5.LL	LY.97.5.UL
V(Block)/VP	0.2302	0.0141	0	0.1804	0.2800
V(Block:SBlock)/VP	0.2383	0.0229	0	0.1575	0.3191
V(e)/VP	0.5315	0.0210	0	0.4572	0.6058

```
data.frame(res$FixedEffect)
```

	LY.Estimate	LY.SE	LY.PValue	LY.2.5.LL	LY.97.5.UL
mu	1841.353	35.07	0.0000	1717.47	1965.2
Genotype(62)	-1036.794	382.10	0.0916	-2386.46	312.9
Genotype(23)	-693.322	314.11	0.1989	-1802.84	416.2
Genotype(111)	-55.983	119.25	0.9569	-477.22	365.3
Genotype(71)	-414.743	531.83	0.8735	-2293.29	1463.8
Genotype(117)	-263.341	171.30	0.4836	-868.41	341.7
Genotype(78)	-211.608	139.28	0.4934	-703.57	280.4
Genotype(40)	-611.469	144.35	0.0087	-1121.34	-101.6
Genotype(127)	-268.046	133.17	0.2634	-738.45	202.4
Genotype(84)	-305.240	81.01	0.0176	-591.38	-19.1
Genotype(108)	112.527	278.68	0.9673	-871.84	1096.9
Genotype(37)	-32.558	196.89	0.9898	-728.04	662.9
Genotype(124)	333.477	145.74	0.1764	-181.32	848.3
Genotype(88)	4.030	356.29	0.9958	-1254.47	1262.5
Genotype(91)	5.906	120.41	0.9947	-419.41	431.2
Genotype(19)	252.993	170.48	0.5129	-349.19	855.2
Genotype(138)	-277.599	211.33	0.6090	-1024.07	468.9

Genotype(102)	-302.447	166.05	0.3418	-888.97	284.1
Genotype(70)	-231.804	147.17	0.4633	-751.64	288.0
Genotype(141)	-74.178	305.95	0.9847	-1154.86	1006.5
Genotype(110)	58.850	359.37	0.9899	-1210.55	1328.3
Genotype(122)	35.336	180.99	0.9880	-603.98	674.6
Genotype(75)	-196.859	129.89	0.4955	-655.66	261.9
Genotype(18)	202.258	202.19	0.7783	-511.93	916.4
Genotype(125)	255.604	373.51	0.9056	-1063.73	1574.9
Genotype(63)	18.175	185.02	0.9930	-635.38	671.7
Genotype(73)	-262.285	223.99	0.6890	-1053.47	528.9
Genotype(55)	-384.973	205.29	0.3183	-1110.13	340.2
Genotype(43)	528.645	521.52	0.7717	-1313.49	2370.8
Genotype(5)	209.858	328.69	0.9189	-951.17	1370.9
Genotype(90)	193.023	494.74	0.9692	-1554.52	1940.6
Genotype(68)	512.056	314.81	0.4363	-599.95	1624.1
Genotype(44)	164.150	565.28	0.9806	-1832.59	2160.9
Genotype(74)	-228.818	409.85	0.9391	-1676.53	1218.9
Genotype(60)	-149.783	434.08	0.9749	-1683.08	1383.5
Genotype(143)	-350.965	496.17	0.8984	-2103.58	1401.6
Genotype(120)	210.593	72.92	0.0696	-46.99	468.2
Genotype(134)	-214.530	93.94	0.1775	-546.35	117.3
Genotype(34)	93.050	51.62	0.3503	-89.27	275.4
Genotype(121)	-130.133	118.48	0.7282	-548.64	288.4
Genotype(136)	-496.739	922.19	0.9434	-3754.18	2760.7
Genotype(107)	-119.404	425.64	0.9815	-1622.89	1384.1
Genotype(56)	37.585	505.43	0.9939	-1747.72	1822.9
Genotype(109)	-24.326	610.01	0.9950	-2179.06	2130.4
Genotype(17)	-181.243	404.46	0.9605	-1609.90	1247.4
Genotype(106)	-149.317	107.59	0.5668	-529.35	230.7
Genotype(8)	427.021	176.94	0.1459	-197.97	1052.0
Genotype(67)	-190.123	577.22	0.9766	-2229.01	1848.8
Genotype(14)	50.489	257.09	0.9879	-857.63	958.6
Genotype(104)	-203.219	781.42	0.9833	-2963.41	2557.0
Genotype(4)	168.386	343.37	0.9531	-1044.49	1381.3
Genotype(99)	-294.327	184.15	0.4510	-944.79	356.1
Genotype(129)	-147.703	495.67	0.9799	-1898.54	1603.1

Genotype(77)	8.772	517.80	0.9957	-1820.22	1837.8
Genotype(13)	382.442	277.64	0.5727	-598.24	1363.1
Genotype(53)	304.173	247.55	0.6569	-570.23	1178.6
Genotype(20)	524.146	346.74	0.4977	-700.64	1748.9
Genotype(145)	137.053	404.06	0.9755	-1290.20	1564.3
Genotype(87)	154.700	176.93	0.8361	-470.25	779.7
Genotype(10)	-71.470	426.07	0.9897	-1576.46	1433.5
Genotype(81)	311.954	83.60	0.0186	16.65	607.3
Genotype(146)	386.019	113.25	0.0306	-14.00	786.0
Genotype(9)	-12.192	82.26	0.9907	-302.77	278.4
Genotype(42)	367.036	516.90	0.8975	-1458.80	2192.9
Genotype(65)	286.124	121.57	0.1598	-143.31	715.6
Genotype(50)	496.969	158.03	0.0464	-61.25	1055.2
Genotype(135)	203.251	434.99	0.9573	-1333.26	1739.8
Genotype(47)	-315.317	415.03	0.8807	-1781.31	1150.7
Genotype(16)	208.263	250.43	0.8537	-676.34	1092.9
Genotype(123)	-194.189	140.18	0.5683	-689.35	301.0
Genotype(31)	725.270	393.29	0.3317	-663.93	2114.5
Genotype(32)	467.176	76.08	0.0007	198.44	735.9
Genotype(116)	259.698	248.06	0.7550	-616.52	1135.9
Genotype(144)	-467.314	415.73	0.7145	-1935.77	1001.1
Genotype(113)	-340.015	524.30	0.9161	-2191.98	1512.0
Genotype(41)	-478.153	299.09	0.4508	-1534.61	578.3
Genotype(46)	-1105.442	3365.39	0.9767	-12992.91	10782.0
Genotype(25)	-611.622	486.29	0.6406	-2329.33	1106.1
Genotype(27)	-296.498	184.87	0.4481	-949.50	356.5
Genotype(131)	-212.600	491.40	0.9630	-1948.35	1523.1
Genotype(7)	45.198	380.52	0.9921	-1298.89	1389.3
Genotype(66)	-353.945	340.99	0.7595	-1558.41	850.5
Genotype(118)	-367.762	288.48	0.6309	-1386.74	651.2
Genotype(28)	-208.947	531.16	0.9688	-2085.14	1667.2
Genotype(52)	-152.490	438.60	0.9746	-1701.73	1396.7
Genotype(2)	161.418	368.64	0.9622	-1140.73	1463.6
Genotype(103)	-386.446	522.47	0.8877	-2231.96	1459.1
Genotype(36)	97.125	268.15	0.9728	-850.04	1044.3
Genotype(89)	-33.928	568.73	0.9944	-2042.85	1975.0

Genotype(39)	138.927	547.75	0.9838	-1795.89	2073.7
Genotype(22)	-27.797	248.39	0.9924	-905.16	849.6
Genotype(58)	3.257	403.95	0.9959	-1423.60	1430.1
Genotype(29)	378.408	332.01	0.7061	-794.35	1551.2
Genotype(79)	-209.430	438.36	0.9554	-1757.82	1339.0
Genotype(6)	146.663	568.39	0.9834	-1861.05	2154.4
Genotype(38)	-14.958	368.68	0.9950	-1317.23	1287.3
Genotype(64)	-133.445	497.07	0.9826	-1889.23	1622.3
Genotype(85)	46.990	584.39	0.9937	-2017.23	2111.2
Genotype(114)	-322.840	405.46	0.8674	-1755.04	1109.4
Genotype(49)	250.245	508.95	0.9528	-1547.52	2048.0
Genotype(48)	60.425	428.79	0.9911	-1454.18	1575.0
Genotype(112)	-294.826	418.46	0.8993	-1772.94	1183.3
Genotype(130)	187.450	197.61	0.8030	-510.56	885.5
Genotype(96)	-78.747	507.25	0.9903	-1870.48	1713.0
Genotype(3)	424.511	407.64	0.7578	-1015.38	1864.4
Genotype(98)	256.661	482.87	0.9449	-1448.96	1962.3
Genotype(132)	-125.417	430.40	0.9805	-1645.71	1394.9
Genotype(35)	-290.224	216.38	0.5932	-1054.55	474.1
Genotype(51)	-52.031	394.01	0.9915	-1443.79	1339.7
Genotype(72)	65.758	507.59	0.9916	-1727.18	1858.7
Genotype(97)	493.785	3434.72	0.9909	-11638.55	12626.1
Genotype(76)	-952.388	4227.81	0.9860	-15886.13	13981.4
Genotype(54)	453.195	3465.06	0.9916	-11786.33	12692.7
Genotype(61)	874.014	4329.56	0.9876	-14419.14	16167.2
Genotype(59)	461.309	2256.98	0.9874	-7510.93	8433.6
Genotype(93)	3823.203	4514.36	0.8475	-12122.73	19769.1
Genotype(142)	734.844	7157.46	0.9928	-24547.23	26016.9
Genotype(119)	670.340	4446.98	0.9906	-15037.60	16378.3
Genotype(86)	-3166.880	5017.39	0.9209	-20889.64	14555.9
Genotype(45)	738.576	4727.25	0.9903	-15959.33	17436.5
Genotype(139)	-44.523	4291.25	0.9958	-15202.37	15113.3
Genotype(80)	-1008.823	3976.75	0.9838	-15055.77	13038.1
Genotype(105)	-3054.175	5796.72	0.9459	-23529.75	17421.4
Genotype(128)	2066.798	4131.92	0.9512	-12528.27	16661.9
Genotype(69)	-3320.435	6384.49	0.9473	-25872.17	19231.3

Genotype(33)	2011.515	4166.83	0.9545	-12706.83	16729.9
Genotype(101)	-3163.172	3620.20	0.8363	-15950.69	9624.3
Genotype(137)	404.397	4459.98	0.9933	-15349.46	16158.3
Genotype(126)	245.568	1237.24	0.9878	-4124.68	4615.8
Genotype(57)	56.826	221.13	0.9835	-724.27	837.9
Genotype(83)	-75.865	815.71	0.9932	-2957.16	2805.4
Genotype(95)	226.285	939.81	0.9848	-3093.37	3545.9
Genotype(82)	19.895	333.53	0.9944	-1158.22	1198.0
Genotype(94)	260.825	208.27	0.6436	-474.83	996.5
Genotype(100)	620.401	427.99	0.5321	-891.39	2132.2
Genotype(140)	811.291	219.55	0.0197	35.78	1586.8
Genotype(30)	289.170	316.39	0.8188	-828.40	1406.7
Genotype(133)	203.387	755.06	0.9825	-2463.69	2870.5
Genotype(1)	402.795	444.32	0.8221	-1166.68	1972.3
Genotype(15)	673.864	1287.40	0.9466	-3873.59	5221.3
Genotype(12)	360.620	298.18	0.6677	-692.62	1413.9
Genotype(115)	284.681	803.94	0.9738	-2555.06	3124.4
Genotype(92)	205.010	541.69	0.9708	-1708.39	2118.4
Genotype(11)	394.090	255.14	0.4797	-507.13	1295.3
Genotype(24)	242.062	304.44	0.8678	-833.29	1317.4
Genotype(21)	346.612	390.29	0.8302	-1032.01	1725.2
Genotype(26)	712.713	424.04	0.4086	-785.10	2210.5

```
data.frame(res$RandomEffect)
```

	LY.Pre	LY.SE	LY.PValue	LY.2.5.LL	LY.97.5.UL
Block(2)	-2.113	8.884	0.9850	-33.494	29.2674
Block(4)	-336.436	16.326	0.0000	-394.103	-278.7690
Block(3)	239.927	9.041	0.0000	207.990	271.8634
Block(1)	98.623	11.788	0.0001	56.985	140.2604
Block:SBlock(2:2)	-179.338	87.581	0.2507	-488.697	130.0222
Block:SBlock(4:18)	261.210	54.606	0.0040	68.326	454.0945
Block:SBlock(4:12)	-92.438	95.490	0.7939	-429.735	244.8583
Block:SBlock(4:22)	38.353	95.198	0.9674	-297.911	374.6179
Block:SBlock(4:5)	-280.846	54.105	0.0023	-471.958	-89.7341
Block:SBlock(2:16)	-249.864	58.618	0.0084	-456.918	-42.8099
Block:SBlock(4:11)	119.879	45.301	0.1017	-40.138	279.8964
Block:SBlock(3:1)	-84.196	72.921	0.6980	-341.772	173.3798

Block:SBlock(4:14)	-60.825	112.157	0.9426	-456.994	335.3445
Block:SBlock(4:30)	-260.252	71.420	0.0213	-512.527	-7.9777
Block:SBlock(3:30)	-813.147	145.827	0.0014	-1328.248	-298.0455
Block:SBlock(2:6)	-237.057	73.853	0.0419	-497.926	23.8123
Block:SBlock(1:25)	-379.562	73.701	0.0024	-639.893	-119.2315
Block:SBlock(2:5)	-336.129	45.409	0.0002	-496.527	-175.7308
Block:SBlock(4:17)	20.303	37.428	0.9426	-111.904	152.5107
Block:SBlock(4:7)	-206.082	54.171	0.0166	-397.429	-14.7341
Block:SBlock(2:20)	-134.478	49.697	0.0926	-310.021	41.0649
Block:SBlock(2:17)	-52.793	57.155	0.8145	-254.680	149.0942
Block:SBlock(2:1)	-398.046	62.504	0.0005	-618.825	-177.2665
Block:SBlock(1:21)	-303.807	76.703	0.0131	-574.743	-32.8707
Block:SBlock(1:24)	247.964	33.037	0.0001	131.268	364.6596
Block:SBlock(1:4)	-362.034	113.595	0.0434	-763.281	39.2130
Block:SBlock(4:27)	-178.156	39.417	0.0058	-317.388	-38.9254
Block:SBlock(4:6)	125.756	93.588	0.5918	-204.823	456.3355
Block:SBlock(4:15)	-180.754	143.115	0.6376	-686.273	324.7664
Block:SBlock(4:20)	-164.668	89.368	0.3324	-480.342	151.0046
Block:SBlock(4:3)	52.340	58.213	0.8254	-153.283	257.9635
Block:SBlock(1:22)	-360.492	40.725	0.0000	-504.343	-216.6407
Block:SBlock(4:10)	255.620	45.677	0.0013	94.275	416.9644
Block:SBlock(3:8)	-276.502	78.203	0.0251	-552.735	-0.2684
Block:SBlock(4:2)	-265.125	54.653	0.0036	-458.175	-72.0751
Block:SBlock(2:25)	1.842	76.279	0.9955	-267.595	271.2788
Block:SBlock(1:23)	262.690	107.410	0.1388	-116.710	642.0898
Block:SBlock(4:1)	-70.281	49.667	0.5514	-245.719	105.1567
Block:SBlock(2:13)	-278.542	122.266	0.1790	-710.418	153.3334
Block:SBlock(4:23)	-115.007	66.395	0.3834	-349.531	119.5177
Block:SBlock(2:9)	-179.032	70.160	0.1179	-426.858	68.7936
Block:SBlock(4:13)	174.583	43.288	0.0118	21.676	327.4896
Block:SBlock(1:29)	-340.500	58.707	0.0010	-547.870	-133.1301
Block:SBlock(4:26)	-31.460	58.760	0.9441	-239.016	176.0966
Block:SBlock(1:17)	-369.769	59.522	0.0006	-580.017	-159.5215
Block:SBlock(3:25)	-450.995	63.278	0.0002	-674.509	-227.4811
Block:SBlock(4:21)	-96.611	47.355	0.2534	-263.882	70.6589
Block:SBlock(2:24)	-242.517	95.868	0.1220	-581.150	96.1156

Block:SBlock(2:8)	-276.716	46.569	0.0009	-441.210	-112.2211
Block:SBlock(4:19)	108.452	65.118	0.4163	-121.564	338.4682
Block:SBlock(1:26)	-29.274	36.780	0.8675	-159.191	100.6419
Block:SBlock(4:16)	-224.989	52.783	0.0084	-411.434	-38.5434
Block:SBlock(2:21)	-206.764	74.317	0.0822	-469.272	55.7430
Block:SBlock(3:12)	-231.993	44.028	0.0021	-387.512	-76.4731
Block:SBlock(3:15)	38.093	80.981	0.9567	-247.953	324.1396
Block:SBlock(4:4)	189.461	62.066	0.0536	-29.773	408.6954
Block:SBlock(2:12)	-49.981	79.910	0.9225	-332.245	232.2830
Block:SBlock(1:15)	-91.309	62.209	0.5219	-311.047	128.4286
Block:SBlock(1:11)	-35.514	36.786	0.7951	-165.454	94.4252
Block:SBlock(1:30)	-84.653	44.320	0.3036	-241.203	71.8977
Block:SBlock(1:3)	43.910	48.444	0.8222	-127.206	215.0254
Block:SBlock(2:10)	7.566	74.851	0.9929	-256.827	271.9596
Block:SBlock(3:26)	-44.200	54.922	0.8642	-238.197	149.7979
Block:SBlock(3:17)	-110.504	48.412	0.1778	-281.510	60.5015
Block:SBlock(1:10)	157.974	99.230	0.4543	-192.533	508.4816
Block:SBlock(1:27)	31.860	55.290	0.9349	-163.437	227.1579
Block:SBlock(1:14)	515.003	74.297	0.0003	252.566	777.4405
Block:SBlock(3:16)	-326.606	68.841	0.0042	-569.771	-83.4421
Block:SBlock(1:20)	-37.821	43.663	0.8395	-192.049	116.4068
Block:SBlock(1:19)	-48.475	51.206	0.8039	-229.347	132.3970
Block:SBlock(4:8)	-294.346	78.107	0.0176	-570.242	-18.4507
Block:SBlock(2:4)	485.662	85.251	0.0012	184.532	786.7918
Block:SBlock(1:8)	-94.606	96.990	0.7904	-437.200	247.9870
Block:SBlock(1:6)	-88.562	75.955	0.6917	-356.855	179.7304
Block:SBlock(2:15)	-130.783	66.766	0.2840	-366.620	105.0539
Block:SBlock(4:9)	414.799	57.388	0.0002	212.090	617.5079
Block:SBlock(3:29)	-293.554	66.396	0.0067	-528.083	-59.0247
Block:SBlock(1:9)	364.274	60.968	0.0008	148.917	579.6314
Block:SBlock(1:2)	26.787	58.450	0.9588	-179.673	233.2462
Block:SBlock(4:24)	1.356	46.381	0.9953	-162.473	165.1843
Block:SBlock(4:29)	259.957	47.846	0.0017	90.952	428.9620
Block:SBlock(4:28)	34.886	59.678	0.9329	-175.914	245.6861
Block:SBlock(3:20)	242.619	67.596	0.0231	3.853	481.3844
Block:SBlock(1:5)	21.110	59.413	0.9737	-188.752	230.9715

Block:SBlock(3:21)	5.072	96.231	0.9946	-334.841	344.9855
Block:SBlock(1:7)	214.338	49.986	0.0081	37.776	390.9009
Block:SBlock(2:14)	438.724	55.431	0.0001	242.927	634.5211
Block:SBlock(1:18)	374.762	39.189	0.0000	236.337	513.1879
Block:SBlock(1:1)	103.036	43.331	0.1540	-50.020	256.0921
Block:SBlock(1:16)	155.245	137.686	0.7127	-331.100	641.5903
Block:SBlock(2:29)	427.187	53.838	0.0001	237.016	617.3582
Block:SBlock(1:13)	-14.677	42.508	0.9748	-164.826	135.4724
Block:SBlock(4:25)	119.224	54.651	0.2066	-73.819	312.2679
Block:SBlock(3:28)	3.833	47.434	0.9936	-163.718	171.3830
Block:SBlock(1:28)	337.094	106.052	0.0440	-37.511	711.6992
Block:SBlock(2:3)	13.067	64.906	0.9876	-216.197	242.3302
Block:SBlock(3:7)	138.467	162.393	0.8452	-435.150	712.0834
Block:SBlock(3:4)	-145.322	72.296	0.2644	-400.692	110.0477
Block:SBlock(2:28)	71.560	48.110	0.5110	-98.377	241.4983
Block:SBlock(2:30)	-149.067	51.531	0.0690	-331.089	32.9550
Block:SBlock(3:13)	45.742	29.510	0.4767	-58.495	149.9787
Block:SBlock(1:12)	-113.742	43.382	0.1057	-266.978	39.4939
Block:SBlock(3:11)	-90.609	67.759	0.5955	-329.952	148.7349
Block:SBlock(2:26)	-39.586	60.146	0.9134	-252.038	172.8648
Block:SBlock(3:27)	235.812	35.582	0.0004	110.125	361.4983
Block:SBlock(2:11)	90.261	51.404	0.3720	-91.313	271.8358
Block:SBlock(3:6)	275.998	86.243	0.0425	-28.636	580.6305
Block:SBlock(3:24)	172.539	71.835	0.1485	-81.201	426.2780
Block:SBlock(3:19)	303.336	73.433	0.0102	43.950	562.7226
Block:SBlock(2:7)	-164.375	24.253	0.0003	-250.042	-78.7086
Block:SBlock(2:23)	209.397	70.408	0.0607	-39.303	458.0962
Block:SBlock(3:5)	-64.697	75.326	0.8426	-330.770	201.3767
Block:SBlock(3:2)	242.648	87.492	0.0833	-66.396	551.6922
Block:SBlock(2:22)	109.550	50.010	0.2039	-67.098	286.1989
Block:SBlock(3:3)	240.302	69.486	0.0284	-5.140	485.7439
Block:SBlock(3:22)	117.046	45.320	0.1123	-43.036	277.1274
Block:SBlock(2:19)	530.025	72.092	0.0002	275.375	784.6742
Block:SBlock(3:9)	176.591	65.508	0.0941	-54.802	407.9831
Block:SBlock(2:27)	487.481	77.989	0.0006	212.002	762.9605
Block:SBlock(2:18)	430.545	112.984	0.0165	31.456	829.6351

Block:SBlock(3:23)	263.675	82.784	0.0435	-28.739	556.0890
Block:SBlock(3:10)	269.312	46.236	0.0010	105.994	432.6305
Block:SBlock(3:14)	90.012	27.281	0.0364	-6.351	186.3745
Block:SBlock(3:18)	317.840	38.489	0.0001	181.885	453.7942

Conclusions

When experimental designs become more complicated, it it likely that the data analysis becomes more challenging if we use the conventional statistical methods and statistical tests. Linear mixed model approaches could be more prefered.