# HW 8

## Contents



## Instructions

You will be asked to submit this HW twice - first submit the `HW7.nb.html` file as normal on the Canvas HW7 page. Then, click the **Down arrow next to the Preview button**, and select **Knit to PDF**. Submit the pdf on the HW7b page through gradescope . You will be asked to assign each of the 8 questions to 1+ pages of the pdf . We are testing gradescope for the final . You will still be expected to do a peer-grade of this assignment through Canvas .

## Assignment

---

The following data are from a spring oats trial run in Craibstone, near Aberdeen in Scottland.

24 varieties of oats were grown each grown in 3 fields (`rep`), and the yield (tonnes / ha) was assessed at the end of the season. Each field was divided into 6 blocks.

```
oats = read.csv('Oats_trials.csv')
oats$plot = as.factor(oats$plot)

str(oats)
```

```
## 'data.frame':    72 obs. of  5 variables:
##  $ plot : Factor w/ 72 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ rep  : chr  "R1" "R1" "R1" "R1" ...
##  $ block: chr  "B1" "B1" "B1" "B1" ...
##  $ gen  : chr  "G11" "G04" "G05" "G22" ...
##  $ yield: num  4.12 4.45 5.88 4.58 4.65 ...
```

Your goal over the following set of questions is to identify a set of genotypes to keep / recommend for use.

## 1.1 Blocks are specific to each field

How many unique blocks are there? If necessary, re-name the blocks so that they are appropriately nested in the fields.

```
oats$unique_blocks <- paste(oats$rep, oats$block)
str(interaction(oats$unique_blocks, oats$gen, drop = TRUE))
```

```
##  Factor w/ 72 levels "R1 B5.G01","R2 B4.G01",..: 31 10 13 64 61 28 58 4 67 40 ...
```

There's 18 unique blocks, 6 for each field, since no two blocks in a field will share the same set of genotypes. I'll rename them as Rep Block. Probably a better way to do this but without knowing an `f` string equivalent in R that's what I've got.

## 1.2 Characterize the experimental design

Start by making a model table:

**Design**: Try to identify the design, but not required.

| Structure | Variable | Type | # levels | Block | EU |
|---|---|---|---|---|---|
| Treatment | gen | Categorical | 24 | block:rep | plot |
| Design | block | Categorical | 6 | | |
| | rep | Categorical | 3 | | |
| | rep:block | Categorical | 18 | | |
| | Plot | Categorical | 72 | | |
| Response | yield | Numeric | 72 | | |

## 1.3 This is an incomplete block design. Fill in the following:

| parameter | value |
|---|---|
| # treatments (t) | 24 |
| # blocks (b) | 18 |
| # reps per trt (n_i) | 3 |
| # trts per block (k) | 4 |

Could this design be balanced? What is lambda (the average times each pair of treatments occur in the same block)?

```
n_i = 3
b = 18
t = 24
k = 4
lam = n_i * (k - 1) / (t - 1)
lam
```

```
## [1] 0.3913043
```

This doesn't look balanced, with a lambda of below 1, meaning that some pairs occur and some do not.

## 1.4 Make an incidence matrix for the design.

Use the `crossprod` function as in Lab_8. Is the design balanced?

```
options(width=80)
inmat = is_crossed(unique_blocks~gen, oats)
```

```
## [1] TRUE
```

```
crossprod(inmat)
```

```
##      gen
## gen   G01 G02 G03 G04 G05 G06 G07 G08 G09 G10 G11 G12 G13 G14 G15 G16 G17 G18
##   G01   3   0   0   0   1   0   1   0   1   1   1   0   0   1   1   0   1   0
##   G02   0   3   0   0   0   0   0   1   1   1   0   0   1   0   1   0   0   1
##   G03   0   0   3   0   1   0   1   1   0   0   0   0   1   0   1   0   0   0
##   G04   0   0   0   3   1   1   0   1   0   0   1   0   0   1   0   0   1   1
##   G05   1   0   1   1   3   0   1   0   1   1   1   0   0   0   0   0   0   0
##   G06   0   0   0   1   0   3   1   0   1   0   0   1   0   0   0   1   1   1
##   G07   1   0   1   0   1   1   3   0   0   0   0   0   0   0   1   1   1   0
##   G08   0   1   1   1   0   0   0   3   1   0   0   0   1   1   1   0   0   0
##   G09   1   1   0   0   1   1   0   1   3   1   0   1   0   0   1   0   0   0
##   G10   1   1   0   0   1   0   0   0   1   3   0   1   1   0   0   0   0   0
##   G11   1   0   0   1   1   0   0   0   0   0   3   1   0   1   0   0   1   0
##   G12   0   0   0   0   0   1   0   0   1   1   1   3   1   0   0   0   1   0
##   G13   0   1   1   0   0   0   0   1   0   1   0   1   3   0   0   0   0   1
##   G14   1   0   0   1   0   0   0   1   0   0   1   0   0   3   0   1   0   1
##   G15   1   1   1   0   0   0   1   1   1   0   0   0   0   0   3   0   1   0
##   G16   0   0   0   0   0   1   1   0   0   0   0   0   0   1   0   3   0   1
##   G17   1   0   0   1   0   1   1   0   0   0   1   1   0   0   1   0   3   1
##   G18   0   1   0   1   0   1   0   0   0   0   0   0   1   1   0   1   1   3
##   G19   1   0   1   0   0   1   1   1   0   0   1   0   1   1   0   1   0   0
##   G20   0   1   1   1   1   0   1   1   0   1   0   0   0   1   0   0   0   0
##   G21   0   1   0   0   0   0   0   0   0   1   1   1   0   0   0   1   1   0
##   G22   0   1   0   1   1   0   0   0   0   0   1   0   1   0   0   1   0   1
##   G23   0   0   1   0   0   0   0   0   0   1   0   1   1   1   1   1   0   1
##   G24   0   0   1   0   0   1   0   0   1   0   0   1   0   0   1   1   0   0
##      gen
## gen   G19 G20 G21 G22 G23 G24
##   G01   1   0   0   0   0   0
##   G02   0   1   1   1   0   0
##   G03   1   1   0   0   1   1
##   G04   0   1   0   1   0   0
##   G05   0   1   0   1   0   0
##   G06   1   0   0   0   0   1
##   G07   1   1   0   0   0   0
##   G08   1   1   0   0   0   0
##   G09   0   0   0   0   0   1
##   G10   0   1   1   0   1   0
##   G11   1   0   1   1   0   0
##   G12   0   0   1   0   1   1
##   G13   1   0   0   1   1   0
##   G14   1   1   0   0   1   0
##   G15   0   0   0   0   1   1
##   G16   1   0   1   1   1   1
##   G17   0   0   1   0   0   0
##   G18   0   0   0   1   1   0
##   G19   3   0   0   0   0   0
```

```
##   G20   0   3   1   0   0   0
##   G21   0   1   3   1   0   1
##   G22   0   0   1   3   0   1
##   G23   0   0   0   0   3   1
##   G24   0   0   1   1   1   3
```

```
print("No, the design is not balanced.")
```

```
## [1] "No, the design is not balanced."
```

### 1.5 Is the design resolvable?

Resolvable designs are designs in which a subset of the blocks constitutes a complete block of the treatments. Resolvable designs are nice because each larger block is a complete replicate of the experiment, so each super-block can be run separately.

Identify blocks that can be resolved into a complete block. **Hint**: Check the Reps!

```
options(width=80)
is_crossed(rep~gen, oats, only_full_crossed = TRUE)
```

```
## [1] TRUE
```

> Each `rep` can be resolved into a complete block. Each contains all 24 of the `gen` treatments, and is a complete replicate of the experiment.

### 1.6 Fit a linear model to the yield data.

Use an ANOVA to assess whether there is strong evidence for differences among the genotypes.

**Note**: If you use fixed blocks, ensure that in your model and ANOVA table, blocks are included before `gen`. If you use `rep:block` to nest block in `rep`, then R will move the `rep:block` term to the end of the model. To avoid this, make a new term `unique_blocks` with a separate ID for block in each rep. Alternatively, declare blocks random (but be careful about nesting!).

```
oatmod = lm(yield ~ unique_blocks + gen, data = oats)
print(anova(oatmod))
```

```
## Analysis of Variance Table
##
## Response: yield
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## unique_blocks 17 13.7537 0.80904  9.6934 4.171e-08 ***
## gen           23 10.0619 0.43747  5.2415 1.459e-05 ***
## Residuals     31  2.5874 0.08346
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> There certainly seems to be strong evidence for difference among genotypes, when using `unique_blocks` as my blocks.

### 1.7 Make a Compact Letter Display to compare the genotypes.

Identify the best-preforming genotype, and any others that can't be statistically distinguished (alpha = 0.1) from the one that performed best.

```
oatmeans = emmeans(oatmod, specs = 'gen')
randblock = lmer(yield ~ gen + rep + (1|block), data = oats)
```

```r
randmeans = emmeans(randblock, specs = 'gen')
print(cld(oatmeans, alpha = 0.1, Letters = letters))
```

```
##  gen emmean    SE df lower.CL upper.CL .group
##  G09   3.44 0.194 31     3.04     3.84  a
##  G03   3.61 0.195 31     3.21     4.01  ab
##  G07   4.11 0.195 31     3.71     4.51  abc
##  G24   4.14 0.195 31     3.74     4.54  abc
##  G20   4.20 0.194 31     3.80     4.59  abc
##  G11   4.22 0.195 31     3.82     4.62  abc
##  G23   4.31 0.194 31     3.92     4.71  abc
##  G18   4.32 0.195 31     3.92     4.71  abc
##  G10   4.36 0.195 31     3.96     4.76  abc
##  G06   4.43 0.194 31     4.03     4.82  abc
##  G22   4.46 0.194 31     4.06     4.86  abc
##  G02   4.47 0.195 31     4.08     4.87   bc
##  G17   4.51 0.194 31     4.11     4.91   bc
##  G04   4.54 0.195 31     4.14     4.93   bc
##  G12   4.64 0.195 31     4.25     5.04   bc
##  G08   4.67 0.195 31     4.27     5.06    c
##  G16   4.72 0.195 31     4.33     5.12    c
##  G13   4.73 0.194 31     4.34     5.13    c
##  G21   4.76 0.194 31     4.36     5.16    c
##  G19   4.84 0.194 31     4.45     5.24    c
##  G14   4.90 0.194 31     4.51     5.30    c
##  G15   5.02 0.194 31     4.62     5.41    c
##  G05   5.03 0.194 31     4.64     5.43    c
##  G01   5.08 0.195 31     4.68     5.47    c
##
## Results are averaged over the levels of: unique_blocks
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 24 estimates
## significance level used: alpha = 0.1
## NOTE: If two or more means share the same grouping symbol,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

```r
print(cld(randmeans, alpha = 0.1, Letters = letters))
```

```
##  gen emmean    SE   df lower.CL upper.CL .group
##  G03   3.34 0.216 45.8     2.91     3.78  a
##  G09   3.61 0.216 45.8     3.18     4.05  ab
##  G20   3.80 0.216 45.9     3.37     4.24  abc
##  G07   4.10 0.224 43.4     3.65     4.56  abcd
##  G24   4.14 0.216 45.9     3.70     4.57  abcd
##  G23   4.14 0.216 45.9     3.70     4.58  abcd
##  G08   4.35 0.216 45.9     3.91     4.78  abcd
##  G10   4.39 0.223 43.0     3.94     4.85  abcd
##  G11   4.41 0.223 42.7     3.96     4.86  abcd
##  G04   4.42 0.223 42.7     3.97     4.87  abcd
##  G18   4.42 0.224 43.1     3.97     4.87  abcd
##  G02   4.53 0.224 43.1     4.08     4.98   bcd
##  G14   4.58 0.223 42.7     4.13     5.03   bcd
##  G22   4.64 0.224 43.6     4.18     5.09   bcd
```

```
##  G06   4.69 0.224 43.3     4.24     5.14   bcd
##  G16   4.70 0.216 45.9     4.27     5.14   bcd
##  G17   4.71 0.224 43.1     4.26     5.16   bcd
##  G21   4.81 0.216 45.9     4.38     5.25    cd
##  G13   4.82 0.224 43.5     4.36     5.27    cd
##  G19   4.87 0.216 45.9     4.43     5.30    cd
##  G12   4.90 0.216 45.9     4.46     5.33    cd
##  G15   4.91 0.224 43.1     4.46     5.36     d
##  G05   5.07 0.216 45.9     4.63     5.50     d
##  G01   5.16 0.216 45.9     4.73     5.60     d
##
## Results are averaged over the levels of: rep
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 24 estimates
## significance level used: alpha = 0.1
## NOTE: If two or more means share the same grouping symbol,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

The best performing genotype is `G01`, though it cannot be shown to be different for nearly every other genotype save `G03` and `G09`.

### 1.8. In the above analysis, if you declare the blocks to be random, you should find that G02 is declared different from G09 but G04 is not. However the estimated value of G02 is closer to G09 than G04 is.

Explain this observation. Why can we distinguish G02 and G09, but not G04 and G09 at alpha = 0.1?

Well, if blocks are random, that means that the behavior/effect of the blocks on the yield are not necessarily consistent, meaning that you could expect some blocks to simply have lower yields irrespective to genotype. If `G02` and `G09` are present in the *same* block, then its easier to determine if they are different, since any randomness in yield derived from block would wash out (and they are!). Conversely, `G04` and `G09` do not appear in a block together; it is more difficult to determine that they are different, even if the absolute numbers would suggest otherwise.