

# DSA 8020 R Session 11: Random and Mixed Effects Models and Computer Experiments

Whitney

## Contents

Random Effects Example . . . . .	1
Read the data into R . . . . .	1
Fitting a fixed effects model . . . . .	2
Fitting a random effects model . . . . .	3
RCBD: Fixed vs. Random Block . . . . .	3
Load R libraries . . . . .	3
Read the data . . . . .	4
Fixed block . . . . .	4
Random block . . . . .	4
Computer Experiments . . . . .	5
Design: Latin hypercube . . . . .	5
Analysis: Gaussian Process . . . . .	7

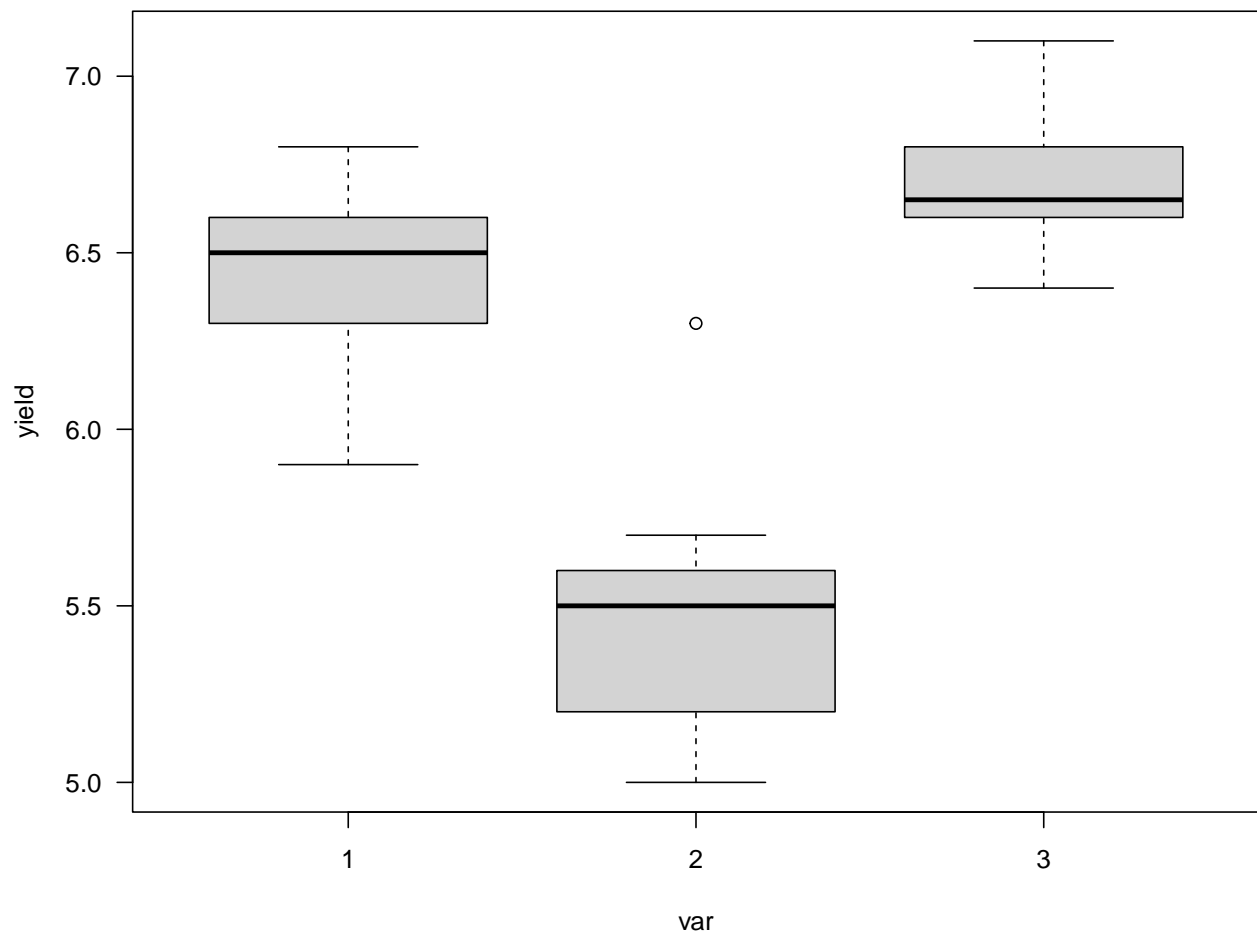
## Random Effects Example

Suppose that an agronomist is studying a large number of varieties of soybeans for yield. The agronomist randomly selects three varieties, and then randomly assigns each of those varieties to 10 of 30 available plots.

Model:  $y_{ij} = \mu + \alpha_i + \epsilon_{ij}$ ,  $\alpha_i s \stackrel{i.i.d.}{\sim} N(0, \sigma_\alpha^2)$ ,  $\epsilon_{ij} s \stackrel{i.i.d.}{\sim} N(0, \sigma^2)$ .  $\alpha_i s$  and  $\epsilon_{ij} s$  are independent to each other

### Read the data into R

```
v1 <- c(6.6, 6.4, 5.9, 6.6, 6.2, 6.7, 6.3, 6.5, 6.5, 6.8)
v2 <- c(5.6, 5.2, 5.3, 5.1, 5.7, 5.6, 5.6, 6.3, 5.0, 5.4)
v3 <- c(6.9, 7.1, 6.4, 6.7, 6.5, 6.6, 6.6, 6.6, 6.8, 6.8)
yield <- c(v1, v2, v3)
var <- factor(c(rep(1, 10), rep(2, 10), rep(3, 10)))
plot(yield ~ var, las = 1)
```



### Fitting a fixed effects model

```
fixef <- lm(yield ~ var)
anova(fixef)
```

```
## Analysis of Variance Table
##
## Response: yield
##          Df Sum Sq Mean Sq F value    Pr(>F)
## var         2  8.306   4.1530  49.593 9.114e-10 ***
## Residuals  27  2.261   0.0837
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coefficients(fixef)
```

```
## (Intercept)      var2      var3
##          6.45      -0.97       0.25
```

## Fitting a random effects model

```
library(lme4)
```

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

```
randef <- lmer(yield ~ 1 + (1|var), REML = TRUE)  
summary(randef)
```

```
## Linear mixed model fit by REML ['lmerMod']  
## Formula: yield ~ 1 + (1 | var)  
##  
## REML criterion at convergence: 21.6  
##  
## Scaled residuals:  
##      Min       1Q   Median       3Q      Max   
## -1.8839 -0.6181  0.1118  0.4962  2.7828   
##  
## Random effects:  
##   Groups    Name      Variance Std.Dev.  
##   var      (Intercept) 0.40693  0.6379  
##   Residual                0.08374  0.2894  
## Number of obs: 30, groups:  var, 3  
##  
## Fixed effects:  
##              Estimate Std. Error t value  
## (Intercept)   6.2100    0.3721   16.69
```

Let's construct CIs for  $\sigma_\alpha^2$ ,  $\sigma^2$ , and  $\mu$

```
## Compute the confidence intervals (CIs) using profile likelihood  
CIs <- confint(randef, oldNames = FALSE)
```

```
## Computing profile confidence intervals ...
```

```
CIs
```

```
##              2.5 %    97.5 %  
## sd_(Intercept)|var 0.2637525 1.5512218  
## sigma              0.2265053 0.3877781  
## (Intercept)       5.3618584 7.0581407
```

## RCBD: Fixed vs. Random Block

Load R libraries

```
library(lsmeans)  
library(lmerTest)
```

## Read the data

```
### Create the data set
x <- c(52, 47, 44, 51, 42, 60, 55, 49, 52, 43, 56, 48, 45, 44, 38)
trt <- rep(c("A", "B", "C"), each = 5)
blk <- rep(1:5, 3)
dat <- data.frame(x = x, trt = trt, blk = as.factor(blk))
```

## Fixed block

```
fixef <- lm(x ~ trt + blk, data = dat)
anova(fixef)
```

```
## Analysis of Variance Table
##
## Response: x
##           Df Sum Sq Mean Sq F value    Pr(>F)
## trt         2   89.2    44.60   7.6239 0.0140226 *
## blk         4  363.6    90.90  15.5385 0.0007684 ***
## Residuals   8   46.8     5.85
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Random block

```
randef <- lmer(x ~ trt + (1|blk), REML = TRUE, data = dat)
summary(randef)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: x ~ trt + (1 | blk)
## Data: dat
##
## REML criterion at convergence: 71.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.1417 -0.6147 -0.1494  0.5772  1.3390
##
## Random effects:
## Groups Name Variance Std.Dev.
## blk (Intercept) 28.35  5.324
## Residual 5.85  2.419
## Number of obs: 15, groups: blk, 5
##
## Fixed effects:
## Estimate Std. Error df t value Pr(>|t|)
## (Intercept) 47.200 2.615 5.054 18.047 8.76e-06 ***
```

```
## trtB          4.600      1.530  8.000   3.007   0.0169 *
## trtC          -1.000      1.530  8.000  -0.654   0.5316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) trtB
## trtB -0.292
## trtC -0.292  0.500
```

```
lsmeans(randef, list(pairwise ~ trt), adjust = "none")
```

```
## $'lsmeans of trt'
##   trt lsmean   SE    df lower.CL upper.CL
##   A     47.2 2.62  5.05     40.5     53.9
##   B     51.8 2.62  5.05     45.1     58.5
##   C     46.2 2.62  5.05     39.5     52.9
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $'pairwise differences of trt'
##   1      estimate    SE df t.ratio p.value
## A - B      -4.6 1.53   8  -3.007  0.0169
## A - C       1.0 1.53   8   0.654  0.5316
## B - C       5.6 1.53   8   3.661  0.0064
##
## Degrees-of-freedom method: kenward-roger
```

## Computer Experiments

### Design: Latin hypercube

This R code chunk uses the *lhs* package to generate and visualize a Latin Hypercube Design (LHD), comparing it to a simple random design for two variables.

```
library(lhs)
# Generate a good n x k LHD
LHD = maximinLHS(n = 30, k = 2, dup = 5)
# Display the LHD
LHD
```

```
##           [,1]      [,2]
## [1,] 0.8829643101 0.36917664
## [2,] 0.3927449422 0.60017731
## [3,] 0.5226350162 0.28774314
## [4,] 0.8447334202 0.86233355
## [5,] 0.2466468239 0.67771969
## [6,] 0.7083290799 0.18667521
## [7,] 0.1860676445 0.86884596
## [8,] 0.4369144799 0.46550041
## [9,] 0.6864831088 0.65496434
```

```
## [10,] 0.2331200676 0.13031813
## [11,] 0.4871270499 0.75868347
## [12,] 0.5792049318 0.13741158
## [13,] 0.9191522078 0.22373065
## [14,] 0.2816961439 0.56665499
## [15,] 0.7602814951 0.52895890
## [16,] 0.0666937350 0.79430524
## [17,] 0.1523812566 0.25874999
## [18,] 0.6451576579 0.30552841
## [19,] 0.3646240980 0.33568690
## [20,] 0.8295403850 0.70875852
## [21,] 0.3208128792 0.80056604
## [22,] 0.6060469890 0.94480832
## [23,] 0.4119352353 0.06770009
## [24,] 0.9376483781 0.40053349
## [25,] 0.1095764338 0.46747957
## [26,] 0.9764494380 0.92968291
## [27,] 0.7754833764 0.03820486
## [28,] 0.0001192785 0.97688445
## [29,] 0.0397955815 0.02911316
## [30,] 0.5550254974 0.57915381
```

“dup” is an integer tuning parameter that determines the number of candidate designs considered. Larger values may improve the design quality but require more computational resources.

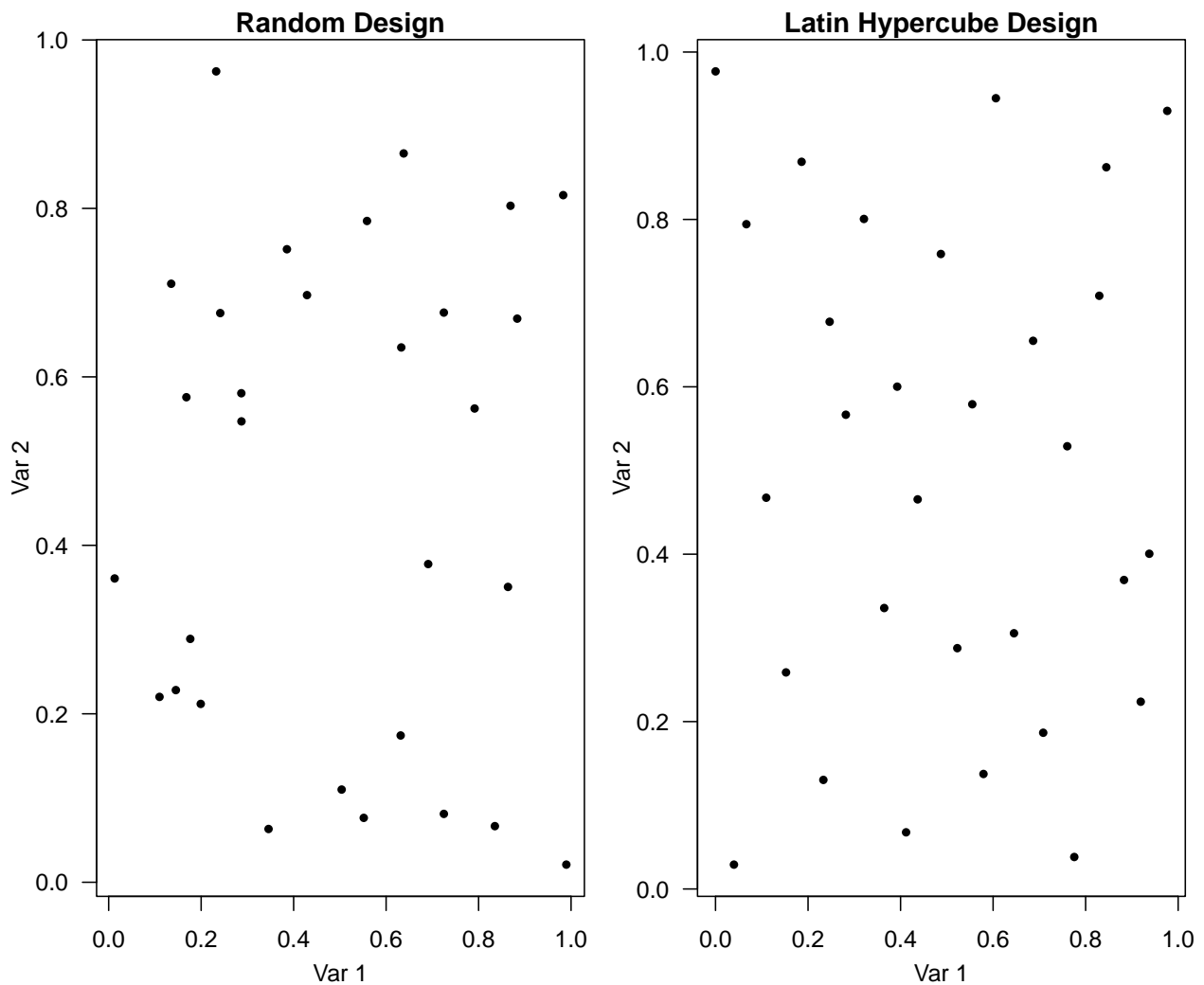
```
# Display the LHD
```

```
LHD
```

```
##           [,1]           [,2]
## [1,] 0.8829643101 0.36917664
## [2,] 0.3927449422 0.60017731
## [3,] 0.5226350162 0.28774314
## [4,] 0.8447334202 0.86233355
## [5,] 0.2466468239 0.67771969
## [6,] 0.7083290799 0.18667521
## [7,] 0.1860676445 0.86884596
## [8,] 0.4369144799 0.46550041
## [9,] 0.6864831088 0.65496434
## [10,] 0.2331200676 0.13031813
## [11,] 0.4871270499 0.75868347
## [12,] 0.5792049318 0.13741158
## [13,] 0.9191522078 0.22373065
## [14,] 0.2816961439 0.56665499
## [15,] 0.7602814951 0.52895890
## [16,] 0.0666937350 0.79430524
## [17,] 0.1523812566 0.25874999
## [18,] 0.6451576579 0.30552841
## [19,] 0.3646240980 0.33568690
## [20,] 0.8295403850 0.70875852
## [21,] 0.3208128792 0.80056604
## [22,] 0.6060469890 0.94480832
## [23,] 0.4119352353 0.06770009
## [24,] 0.9376483781 0.40053349
## [25,] 0.1095764338 0.46747957
```

```
## [26,] 0.9764494380 0.92968291
## [27,] 0.7754833764 0.03820486
## [28,] 0.0001192785 0.97688445
## [29,] 0.0397955815 0.02911316
## [30,] 0.5550254974 0.57915381
```

```
par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2.2, 1, 0), mfrow = c(1, 2))
plot(cbind(runif(n = 30), runif(n = 30)), cex = 0.8, pch = 16,
     xlab = "Var 1", ylab = "Var 2", main = "Random Design")
plot(LHD, cex = 0.8, pch = 16, xlab = "Var 1", ylab = "Var 2",
     main = "Latin Hypercube Design")
```



## Analysis: Gaussian Process

1. Load the data:

Reads a dataset from a URL and previews the first 10 rows.

```
# Load the data
neuron <- read.table("http://deanvosdraguljic.ietsandbox.net/DeanVossDraguljic/R-data/neuron.txt",
                     header = T)
head(neuron, 10)
```

```
##           gNaFsc      gKdrsc fr
## 1  0.38593729 0.2120652 33
## 2  0.04666927 0.4594742  0
## 3  1.00000000 0.4473344 46
## 4  0.95467637 0.3351407 44
## 5  0.53334929 0.7981310 41
## 6  0.59166751 0.6042714 41
## 7  0.18570301 0.3799469 31
## 8  0.49927784 0.2444170 36
## 9  0.74609113 0.3949591 42
## 10 0.07269414 1.0000000  0
```

## 2. Fit a Gaussian Process (GP):

- Uses the first two columns as input variables (g\_NaF and g\_KDR) and the third column as the response.
- Fits a GP using maximum likelihood via the *mlegp* package.
- Displays a summary of the fitted model.

```
library(mlegp)
GPfit <- mlegp(neuron[, 1:2], neuron[, 3])
```

```
## no reps detected - nugget will not be estimated
##
## ===== FITTING GP # 1 =====
## running simplex # 1...
## ...done
## ...simplex #1 complete, loglike = -104.446501 (convergence)
## running simplex # 2...
## ...done
## ...simplex #2 complete, loglike = -104.446501 (convergence)
## running simplex # 3...
## ...done
## ...simplex #3 complete, loglike = -104.446502 (convergence)
## running simplex # 4...
## ...done
## ...simplex #4 complete, loglike = -104.446501 (convergence)
## running simplex # 5...
## ...done
## ...simplex #5 complete, loglike = -104.446501 (convergence)
##
## using L-BFGS method from simplex #1...
## iteration: 1, loglike = -104.446501
## ...L-BFGS method complete
##
## Maximum likelihood estimates found, log like = -104.446501
## creating gp object.....done
```



```
summary(GPfit)
```

```
##
## Total observations = 30
## Dimensions = 2
##
## mu = 27.61156
## sig2: 251.8769
## nugget: 0
##
## Correlation parameters:
##
##      beta a
## 1  5.027973 2
## 2 50.227290 2
##
## Log likelihood = -104.4465
##
## CV RMSE: 7.312656
## CV RMaxSE: 1020.783
```

### 3. Make predictions:

- Creates a grid of input values over the range  $[0, 1]$  for both variables.
- Uses the fitted GP model to predict responses and their standard errors on the grid.

```
predictedX = expand.grid(g_NaF = seq(0, 1, 0.02), g_KDR = seq(0, 1, 0.02))
yhats = predict(GPfit, predictedX, se.fit = T)
```

### 4. Visualize results:

```
library(fields)
par(mfrow = c(1, 2))
image.plot(seq(0, 1, 0.02), seq(0, 1, 0.02), matrix(yhats$fit, 51, 51),
           xlab = "g NaF (mS/cm^2)", ylab = "g KDR (mS/cm^2)", las = 1,
           main = "Predictions")
points(neuron[, 1:2], pch = 16, cex = 0.75)
image.plot(seq(0, 1, 0.02), seq(0, 1, 0.02), matrix(yhats$se.fit, 51, 51),
           xlab = "g NaF (mS/cm^2)", ylab = "g KDR (mS/cm^2)", las = 1,
           main = "Predictions Uncertainty")
points(neuron[, 1:2], pch = 16, cex = 0.75)
```

