Random and Mixed Effects Models and Computer Experiments

Blake Pappas

December 17, 2023

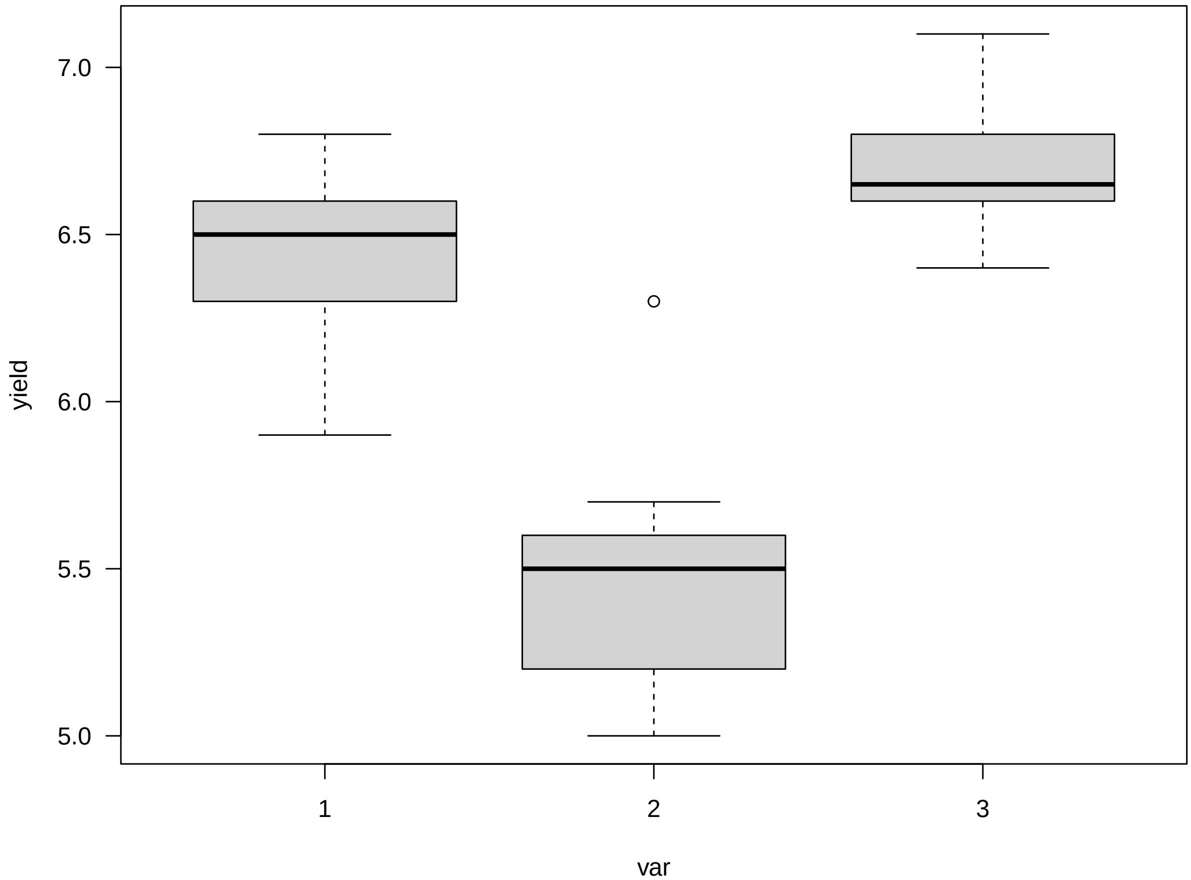
Random Effects Example

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: $y_{ij} = \mu + \alpha_i + \epsilon_{ij}$, $\alpha_i s \overset{i.i.d.}{\sim} \mathrm{N}(0, \sigma_\alpha^2)$, $\epsilon_{ij} s \overset{i.i.d.}{\sim} \mathrm{N}(0, \sigma^2)$. $\alpha_i s$ 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) # Response
var <- factor(c(rep(1, 10), rep(2, 10), rep(3, 10))) # Predictor
plot(yield ~ var, las = 1) # Creates boxplot of the three varieties
```



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

```r
# install.packages("lme4")
library(lme4)
```

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

```r
randef <- lmer(yield ~ 1 + (1 | var), REML = TRUE) #lmer() fits the linear mixed effect models
summary(randef) # 0.6379 = sigma alpha; 0.2894 = sigma
```

```
## 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 σ_α^2, σ^2, and μ.

```r
## Compute the Confidence Intervals (CIs) Using Profile Likelihood
CIs <- confint(randef, oldNames = FALSE)
```

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

```r
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

```r
# install.packages("lsmeans", "lmerTest")
library(lsmeans)
library(lmerTest)
```

Read the Data

```r
### 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

```r
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

```r
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

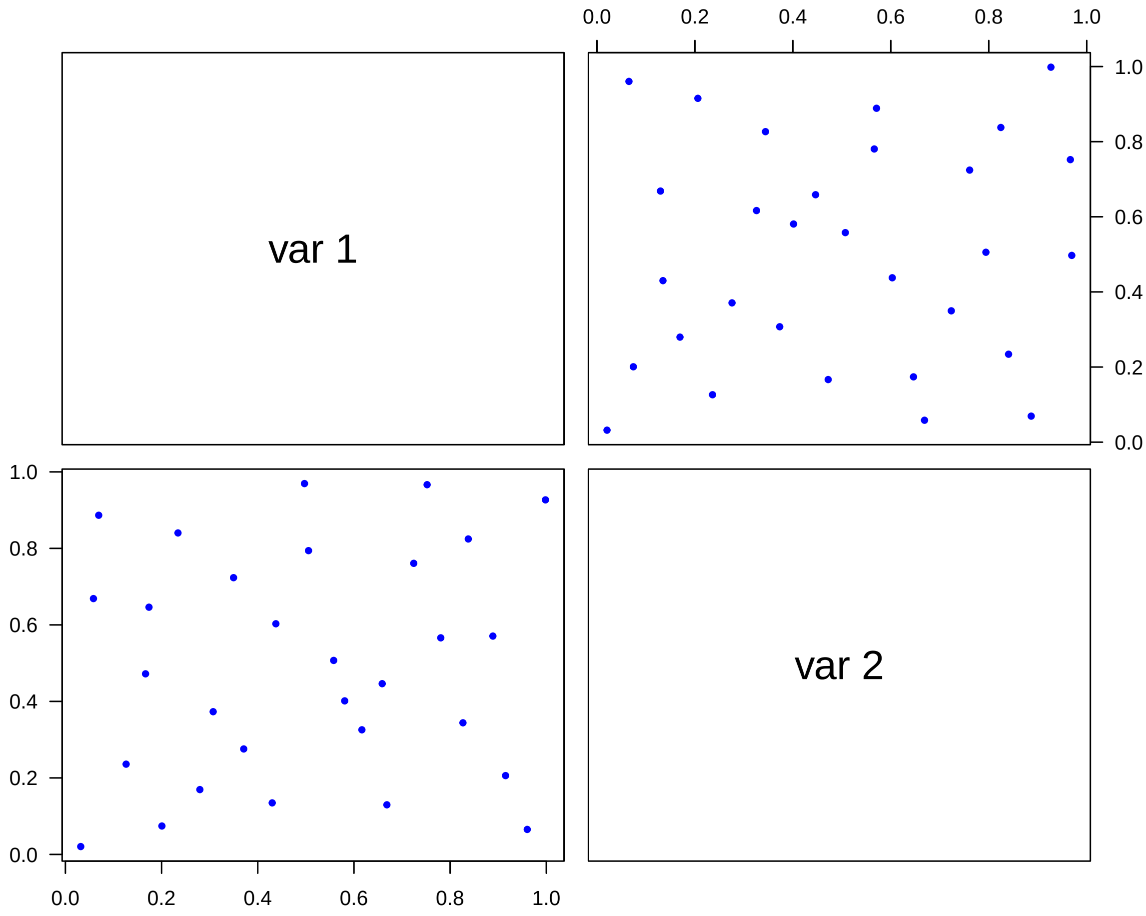
Design: Latin Hypercube

```
# install.packages("lhs") # Latin Hypercube Sample Package
library(lhs)

# Generate a Good n x k LHD
LHD = maximinLHS(n = 30, k = 2, dup = 5)
# "dup" is an integer tuning parameter that determines the number of candidate points considered. Large

# Display the LHD
LHD
```

```
##             [,1]       [,2]
##  [1,] 0.58076660 0.40140907
```

```
##  [2,] 0.78053933 0.56619079
##  [3,] 0.65872304 0.44634264
##  [4,] 0.37082394 0.27571391
##  [5,] 0.05831868 0.66878428
##  [6,] 0.83776638 0.82461598
##  [7,] 0.27960253 0.16948912
##  [8,] 0.23409251 0.84037757
##  [9,] 0.43765139 0.60296753
## [10,] 0.50555449 0.79409884
## [11,] 0.91530113 0.20602494
## [12,] 0.17381666 0.64627016
## [13,] 0.75216008 0.96653640
## [14,] 0.61653743 0.32576948
## [15,] 0.20066325 0.07429457
## [16,] 0.30724342 0.37311606
## [17,] 0.12624755 0.23593988
## [18,] 0.88890088 0.57078537
## [19,] 0.66850047 0.12970621
## [20,] 0.34965683 0.72343579
## [21,] 0.72432851 0.76091284
## [22,] 0.82666794 0.34409309
## [23,] 0.43000465 0.13475029
## [24,] 0.16662595 0.47209320
## [25,] 0.49715074 0.96939881
## [26,] 0.55784171 0.50711484
## [27,] 0.06925444 0.88659816
## [28,] 0.96037775 0.06531008
## [29,] 0.03188225 0.02053272
## [30,] 0.99827846 0.92684767
```

```r
pairs(LHD, col = "blue", cex = 0.8, pch = 16, las = 1)
```



Analysis: Gaussian Process

```
# Load the Data
neuron <- read.table("http://deanvossdraguljic.ietsandbox.net/DeanVossDraguljic/R-data/neuron.txt", head
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
```

```r
# Fit a GP
# install.packages("mlegp")
library(mlegp)
GPFit <- mlegp(neuron[, 1:2], neuron[, 3]) # neuron[, 1:2] = x-variable; neuron[, 3] = y-variable
```

```
## 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
```

```r
summary(GPFit)
```

```
##
## Total observations = 30
## Dimensions = 2
##
## mu = 27.61157
## sig2:    251.8751
## nugget:  0
##
## Correlation parameters:
##
##        beta a
## 1  5.027878 2
## 2 50.228477 2
##
## Log likelihood = -104.4465
##
## CV RMSE: 7.312618
## CV RMaxSE: 1020.777
```

```r
# Make Prediction
predictedX = expand.grid(g_NaF = seq(0, 1, 0.02), g_KDR = seq(0, 1, 0.02))
yhats = predict(GPFit, predictedX, se.fit = T)

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