

# Prediction Intervals for Multilevel Models

Bret Larget

April 30, 2008

## ABSTRACT

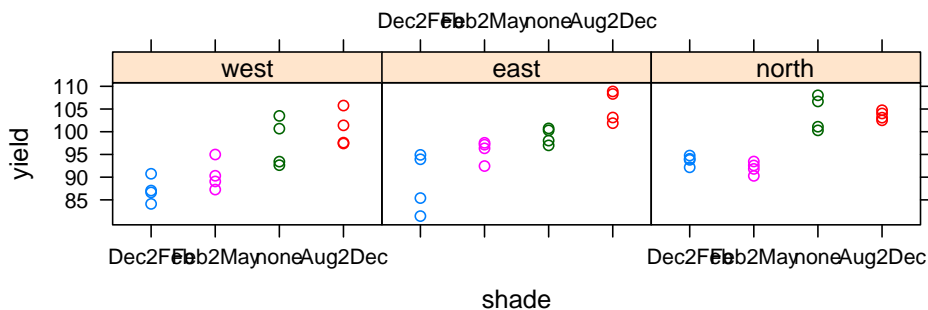
There are many methods for constructing prediction intervals for differences in observations. This document will explain several of these methods in the context of different models using data from a kiwi yield experiment as an example.

## 1. KIWI DATA

In an experiment (discussed in more detail in class), kiwi yields were measured in a designed experiment. There were three blocks (north, east, and west) at a single site. Each block had four plots. (The plots were arranged in a single column in the east and west blocks, but in a square pattern in the north block.) Due to natural and artificial barriers, each of the four plots was subject to a different shading treatment; no shade from August to December, from December to February, and from February to May. one plot in each block has each shading treatment. Within each plot there were four vines. Kiwi yield was measured at each vine.

The following plot shows the data. The third and fourth lines of R code reorder the shade and block factors in order from smallest to largest by mean kiwi yield. We observe that there is no obvious need to transform the response variable, and that there do appear to be both block and shade treatment effects. Perhaps the shade treatment acts differently in the north block than in the east and west blocks.

```
> library(lattice)
> kiwi = read.table("kiwi.txt", header = T)
> kiwi$shade = with(kiwi, reorder(shade, yield))
> kiwi$block = with(kiwi, reorder(block, yield))
> print(xyplot(yield ~ shade | block, data = kiwi, layout = c(3,
+ 1), groups = shade))
```



## 2. PREDICTION INTERVALS

There are many possible comparisons of potential interest. We will focus on two specific comparisons: (1) the difference in yield between two vines in the same plot (say the no shade treatment in the north block), and (2) the difference in average yield between the no shade and December to February shade treatments in the east block. We will examine prediction intervals in the context of several models.

### 2.1. Classical Models

A classical model might treat all of the variables as “unmodeled”, or as *fixed effects*. Here is R code for such a model. Note that block and shade are crossed and that the plot variable is essentially equal to their interaction. This model has a free parameter for each of the 12 plots. The `arm` library must be loaded to use `display()`.

```
> library(arm)
> options(digits = 7)

> fit.0 = lm(yield ~ shade + block + plot, data = kiwi)
> display(fit.0)

lm(formula = yield ~ shade + block + plot, data = kiwi)
      coef.est coef.se
(Intercept)   87.15   1.75
shadeFeb2May    3.26   2.47
shadenone      10.41   2.47
shadeAug2Dec   13.40   2.47
blockeast      5.01   2.47
blocknorth     6.47   2.47
ploteast.Dec2Feb -3.23   3.49
ploteast.Feb2May 0.47   3.49
ploteast.none  -3.54   3.49
plotnorth.Aug2Dec -3.42   3.49
plotnorth.Dec2Feb 0.08   3.49
plotnorth.Feb2May -4.84   3.49
---
n = 48, k = 12
residual sd = 3.49, R-Squared = 0.79
```

The fitted model has 12 coefficients for the means and an estimated residual standard deviation of 3.49.

**Prediction interval for differences between vines.**— The fitted value for two measurements at different vines in the same plot will be the same: all variation will be a function of the individual measurement level error. The fitted value in terms of model parameters is:

$$E[y_i] = \mu + \alpha_{j[i]} + \beta_{k[i]} + \gamma_{m[i]} \quad (1)$$

where  $\mu$  is the intercept which corresponds to the fitted mean in the west plot with the December to February shading (which is in the summer months),  $\{\alpha_j\}$  for  $j = 1, 2, 3$  are shade treatment

effects for the other treatments relative to December to February shading,  $\{\beta_k\}$  for  $k = 1, 2$  are block effects for the non-west blocks, and  $\{\gamma_m\}$  are plot effects for  $m = 1, 2, \dots, 6$  for six of the plots.

The fitted value for a vine in the no shade north block is:

$$E[y_i] = 87.15 + 6.47 + 10.41 = 104.03.$$

R code to find this directly using `predict()` is as follows.

```
> x0.new = data.frame(shade = "none", block = "north", plot = "north.none")
> predict(fit.0, x0.new)
```

```
[1] 104.025
```

Alternatively, this fitted value is the sum of coefficients 1, 3, and 6 in this model.

```
> sum(coef(fit.0)[c(1, 3, 6)])
```

```
[1] 104.025
```

To find a prediction interval for the difference in yields in two vines in this plot, we can recognize that the only difference will be the actual individual random errors as the true means should be the same. The estimated standard deviation of yield for individual vines is 3.49, so the estimated standard deviation for a difference in two yields from the same plot would be  $4.94 = \sqrt{3.49^2 + 3.49^2}$ . A rough 95% prediction interval would then be centered at zero and extend about two standard deviations in each direction, say about  $(-9.9, 9.9)$ . This interval does not account for uncertainty in estimating  $\sigma$ .

We can do a more accurate prediction by using `sim()` to account for estimation error in  $\sigma$ . We use `set.seed()` once here so that we can replicate the simulation later if needed. It is good practice to set the random seed once (and only once) for a single session of related simulations.

Inferences based on simulation are subject to simulation variability. This variability can be reduced by taking larger simulation samples. In estimating quantiles as is needed for prediction intervals, I find that I can get fairly substantial differences in the second significant digit using a sample size of 1000, but that the variation begins in the third significant digit when I use a simulation size of 10,000, which is what I have chosen to do.

```
> set.seed(235435)
> N = 10000
> sim.0 = sim(fit.0, N)
> sigma.0 = sim.0$sigma
> vine.0.1 = sim.0$beta[, 1] + sim.0$beta[, 3] + sim.0$beta[, 6] +
+   rnorm(N, 0, sigma.0)
> vine.0.2 = sim.0$beta[, 1] + sim.0$beta[, 3] + sim.0$beta[, 6] +
+   rnorm(N, 0, sigma.0)
> quantile(vine.0.1 - vine.0.2, c(0.025, 0.975))
```

```
      2.5%      97.5%
-10.11279  10.01269
```

An alternative to explicitly summing the columns is to use `apply()`. The first argument is the beta matrix with only columns 1, 3, and 6. The second argument 1 says to apply the function to rows of the matrix. The third argument says to find the sum.

```

> fitted.0 = apply(sim.0$beta[, c(1, 3, 6)], 1, sum)
> vine.0.1b = fitted.0 + rnorm(N, 0, sigma.0)
> vine.0.2b = fitted.0 + rnorm(N, 0, sigma.0)
> quantile(vine.0.1b - vine.0.2b, c(0.025, 0.975))

```

```

      2.5%      97.5%
-10.04555  10.06638

```

Notice that both answers here are very close to what we would get ignoring the uncertainty in  $\sigma$ .

```

> 2 * sqrt(3.49^2 + 3.49^2)

```

```

[1] 9.87121

```

**Prediction interval for difference in means.**— Under the same model, we can find a prediction interval for the difference in the averages between two plots with different shading treatments located within the same block; specifically, we will find a prediction interval for the difference in yield between the no shade and December to February shading treatments within the east block. In reference to parameters in equation 1 and the order of these parameters in the displayed fit of this model shown in section 2.1, the average in the east no shade plot will have the form

$$\mu + \alpha_2 + \beta_1 + \gamma_3 + (e_1 + e_2 + e_3 + e_4)/4$$

while the average for the east December to February shade plot has the form

$$\mu + \beta_1 + \gamma_1 + (e_5 + e_6 + e_7 + e_8)/4$$

where the  $e_i$  are independent errors. We can simulate averages of size four by finding the fitted values and adding to them random deviations with standard deviation  $\sigma/\sqrt{4}$ . The fitted value for the first plot is the sum of coefficients 1, 3, 5, and 9 from the model displayed in section 2.1 while the second is the sum of coefficients 1, 5, and 7. We add to these fitted values random error with standard deviation equal to the mean of four observations, take the difference, and find quantiles.

```

> fitted.0a = apply(sim.0$beta[, c(1, 3, 5, 9)], 1, sum)
> fitted.0b = apply(sim.0$beta[, c(1, 5, 7)], 1, sum)
> avg.0a = fitted.0a + rnorm(N, 0, sim.0$sigma/2)
> avg.0b = fitted.0b + rnorm(N, 0, sim.0$sigma/2)
> quantile(avg.0a - avg.0b, c(0.025, 0.975))

```

```

      2.5%      97.5%
 3.137194 17.185373

```

## 2.2. Multilevel Models

Multilevel models have multiple sources of variation, some operating on units of different size. For this kiwi example, we consider a model where we model the effects for block and plot, but not shade treatment. To make the shade treatment effects more clear, we remove the intercept from the model so that each estimated shade coefficient can be thought of as the mean yield for that treatment averaged over block and plot.

```
> fit.2 = lmer(yield ~ shade - 1 + (1 | block) + (1 | plot), data = kiwi)
> display(fit.2, digits = 4)
```

```
lmer(formula = yield ~ shade - 1 + (1 | block) + (1 | plot),
      data = kiwi)
      coef.est coef.se
shadeDec2Feb  89.9208  1.7577
shadeFeb2May  92.7742  1.7577
shadenone    100.2025  1.7577
shadeAug2Dec 103.2333  1.7577
```

Error terms:

```
Groups   Name Std.Dev.
plot          1.4784
block          2.0090
Residual      3.4911
```

---

```
number of obs: 48, groups: plot, 12; block, 3
AIC = 264, DIC = 273.6
deviance = 262.8
```

The fitted model has four “unmodeled” coefficients and three sources of random variation, each with its own standard deviation. The algebraic form of this model is as follows:

$$y_i = \alpha_{j[i]} + \beta_{k[i]} + \gamma_{m[i]} + e_i \quad (2)$$

where  $\alpha_j$ ,  $j = 1, \dots, 4$  are the shade treatment effects,  $\beta_k \sim \text{iid } N(0, \sigma_\beta^2)$ ,  $k = 1, \dots, 3$  are the block effects,  $\gamma_m \sim \text{iid } N(0, \sigma_\gamma^2)$ ,  $m = 1, \dots, 12$  are the plot effects, and  $e_i \sim \text{iid } N(0, \sigma^2)$ ,  $i = 1, \dots, 48$  are the individual vine effects.

We can use `ranef()` to extract the estimated random effects and `fixef()` to find the estimated fixed effects from a multilevel model. These functions are useful when we want to make inferences about specific blocks or plots in this example.

```
> fixef(fit.2)
```

```
shadeDec2Feb shadeFeb2May   shadenone shadeAug2Dec
      89.92083    92.77417    100.20250    103.23333
```

```
> ranef(fit.2)
```

```
An object of class "ranef.lmer"
[[1]]
```

```
      (Intercept)
east.Aug2Dec    0.7135300
east.Dec2Feb   -0.6743001
east.Feb2May    1.0410816
east.none      -0.7480950
north.Aug2Dec  -0.4184660
north.Dec2Feb   1.0080020
```

```
north.Feb2May -0.8782914
north.none    1.0271469
west.Aug2Dec  -0.2950640
west.Dec2Feb  -0.3337019
west.Feb2May  -0.1627902
west.none     -0.2790519
```

```
[[2]]
```

```
(Intercept)
west      -1.976943
east       0.613458
north     1.363485
```

**Prediction interval for differences between vines.**— To find a prediction interval from a multilevel model, we can again put on our thinking caps and consider the coefficients in the fitted model. Two vines from the same plot will have the same shading treatment, the same block effect, and the same plot effect. The difference will be governed only by the individual vine level variation. In this model, the vine level standard deviation is again 3.49, so that the previous prediction interval of about  $(-9.9, 9.9)$  would still hold.

Again, we can use the results of a simulation to better account for uncertainty in parameter estimates. However, `sim()` is only appropriate for models fit using `lm()` and `glm()`. The function `mcmcscamp()` from the R `lme4` package can be used for models fit using `lmer()`. The usage of `mcmcscamp()` is identical to the usage of `sim()`, but the output is stored differently. Here is an example of using `mcmcscamp()` and a display of the first three rows of output.

```
> sim.2 = mcmcscamp(fit.2, N)
> sim.2[1:3, ]

      shadeDec2Feb shadeFeb2May shadenone shadeAug2Dec log(sigma^2)
[1,]    90.71117    93.81232  102.2962    104.1205    2.344128
[2,]    89.28845    91.55800  100.0823    100.1359    2.714682
[3,]    89.05147    90.65567   98.6299    100.0191    2.193516
      log(plot.(In)) log(blck.(In))
[1,]   -0.5213894    0.9012542
[2,]   -0.4499040    3.0851983
[3,]   -0.5201094    1.3344206
```

Unlike `sim()` that separates the mean parameters  $\beta$  from the error parameter  $\sigma$  in a list with components `$beta` and `$sigma`, the former being a matrix with a column for each  $\beta$  coefficient and the latter being an array or single column matrix for  $\sigma$ , the output of `mcmcscamp()` is a single matrix where the coefficients for the mean and the parameters for the variance components are combined. Furthermore, the variance components are not simple standard deviations, but are logs of variances. Hence, to find the standard deviations, we need to first exponentiate and then take the square root. The information is the same, but it is packaged differently. This is an artifact of R and R packages which are written by volunteers who can make their own design decisions.

In any case, we can realize here again that whatever the shading treatment, block, and plot effects might be, their effects will cancel when taking the difference between two vines from the same plot. We can see, however, from the original model what the fitted values for the no shade treatment

in the north block would be. Note the use of `[[[]]]` to extract components of the list returned by `ranef()` with a component for each variance components. These components are actually data frames which can be treated as matrices, so we extract numbers from the first column.

```
> fe.2 = fixef(fit.2)
> re.2 = ranef(fit.2)
> re.2.plot = re.2[[1]]
> re.2.block = re.2[[2]]
> fitted.2 = fe.2[3] + re.2.block[3, 1] + re.2.plot[8, 1]
> fitted.2
```

```
shadenone
102.5931
```

To get the vine level standard deviation, we need to back-transform the 5th column of `sim.2`.

```
> sigma.2 = sqrt(exp(sim.2[, 5]))
```

With the fitted value and standard deviation in hand, we can now use the simulation output to find the prediction interval.

```
> vine1.2 = fitted.2 + rnorm(N, 0, sigma.2)
> vine2.2 = fitted.2 + rnorm(N, 0, sigma.2)
> quantile(vine1.2 - vine2.2, c(0.025, 0.975))
```

```
      2.5%      97.5%
-10.69608  10.88742
```

Notice that this interval is slightly larger than those for the same difference found by other methods.

**Prediction interval for difference in means.**— For a comparison of means for the east block between two different shading treatments, we can incorporate uncertainty from plot to plot variation, vine level uncertainty, and uncertainty in the estimated shading treatments. Each average will be the sum of shading effect and the block effect plus a random plot effect and an error with standard deviation one half that for the vine effect due to the sample of four vines. Because the question referred to the east block, I use the estimated random effect for this block rather than a random block effect. It does not matter as the effects cancel.

```
> shading.2a = sim.2[, 3]
> shading.2b = sim.2[, 1]
> sigma.2.plot = sqrt(exp(sim.2[, 6]))
> plot.2a = rnorm(N, 0, sigma.2.plot)
> plot.2b = rnorm(N, 0, sigma.2.plot)
> block.east = re.2.block[2, 1]
> avg.2a = shading.2a + plot.2a + block.east + rnorm(N, 0, sigma.2/2)
> avg.2b = shading.2b + plot.2b + block.east + rnorm(N, 0, sigma.2/2)
> quantile(avg.2a - avg.2b, c(0.025, 0.975))
```

```
      2.5%      97.5%
 2.521180 17.950940
```

This interval again is larger than the one from the classical model.