Solution Swabs exercise (R software)

Contents

NOTE: This document proposes an R syntax giving the necessary outputs to answer to the questions of the exercise. The focus here is then on the implementation in R software and not on the interpretation or the validity of the results: we refer to the SAS solution for a discussion of the two latter points.

Because of the multiplicity of the packages in R, there are often several ways to perform a given operation (e.g. fitting a mixed model or converting a dataset from the long to the wide format). Some can be more efficient (in term of computation time or memory usage), other can be closer to the natural language or enabling a concise syntax. We don't claim to propose here the "best" R syntax but we tried to provide an readable syntax that could be re-used in other problems. In some cases an alternative syntax, usually more complex but more efficient/generalisable, is proposed in appendix.

First, load the necessary packages

```
> library(nlme)
> library(lattice)
>
> # optional
> library(data.table)
> library(ggplot2)
```

Question 1: Data import

```
> df.data_swabs <- read.table("http://publicifsv.sund.ku.dk/~jufo/courses/repeated15/swabs.txt",
+                         header = TRUE, na.strings = ".")
```

Rename the levels of the variable group:

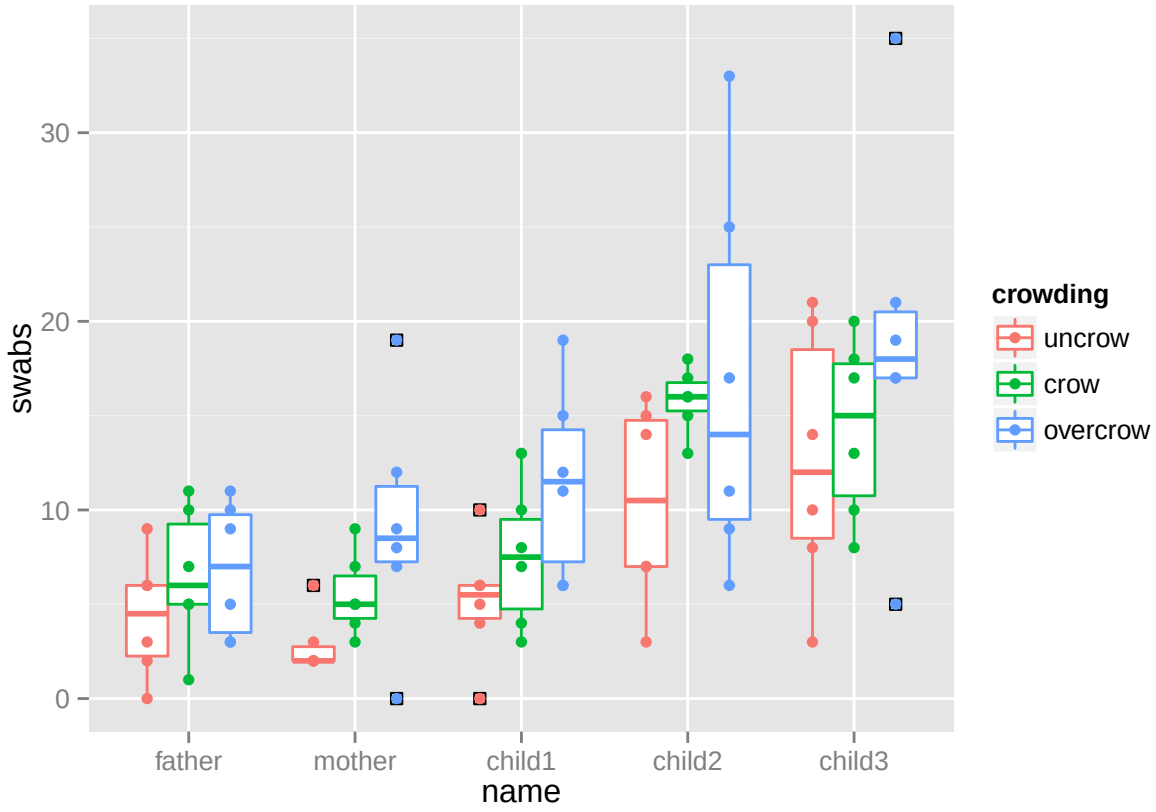
```
> df.data_swabs$crowding <- factor(df.data_swabs$crowding,
+                             levels = c("uncrow", "crow", "overcrow"))
> df.data_swabs$family <- factor(df.data_swabs$family)
> df.data_swabs$name <- factor(df.data_swabs$name,
+                           levels = c("father", "mother", "child1", "child2", "child3"))
```

Summary of the dataset

```
> summary(df.data_swabs)
```

```
     crowding      family       name        swabs
 uncrow  :30   1      : 5   father:18   Min.   : 0.000
 crow    :30   2      : 5   mother:18   1st Qu.: 5.000
 overcrow:30   3      : 5   child1:18   Median : 8.500
               4      : 5   child2:18   Mean   : 9.889
               5      : 5   child3:18   3rd Qu.:14.750
               6      : 5               Max.   :35.000
               (Other):60
```

```
> #### using ggplot2
> gg.base <- ggplot(df.data_swabs, aes(x = name, y = swabs, color = crowding))
> gg.base + geom_boxplot(outlier.shape = 0) + geom_point(position = position_dodge(width = 0.75))
```




```
> #### or using boxplot
> # par(mar = c(8,2,2,2))
> # boxplot(swabs ~ name + crowding, data = df.data_swabs,
> #         las = 2)
```

Question 2: mixed model

```
> df.data_swabs$crowding <- relevel(df.data_swabs$crowding, ref = "uncrow")
> df.data_swabs$name <- relevel(df.data_swabs$name, ref = "mother")
>
> lme.1 <- lme(swabs ~ crowding + name,
+              data = df.data_swabs,
+              random = ~1 | family)
```

Identical to SAS output, e.g. :

```
                                R       SAS
-2 log-likelihood    -5.271104e+02 -527.1000
sigma_B               4.334008e+00    4.3341
sigma_W               2.336964e+01   23.3696
betaOverCrow          5.600000e+00    5.6000
sd.betaOverCrow       1.732814e+00    1.7328
p_value.betaOverCrow  5.588554e-03    0.0056
```

```
> summary(lme.1)
```

```
Linear mixed-effects model fit by REML
 Data: df.data_swabs
       AIC      BIC    logLik
  545.1104 566.8799 -263.5552

Random effects:
 Formula: ~1 | family
        (Intercept) Residual
StdDev:    2.081828 4.834216

Fixed effects: swabs ~ crowding + name
                    Value Std.Error DF  t-value p-value
(Intercept)      3.011111  1.593730 68 1.889349  0.0631
crowdingcrow     2.866667  1.732814 15 1.654341  0.1188
crowdingovercrow 5.600000  1.732814 15 3.231737  0.0056
namefather       0.055556  1.611405 68 0.034476  0.9726
namechild1       2.222222  1.611405 68 1.379059  0.1724
namechild2       8.500000  1.611405 68 5.274899  0.0000
namechild3       9.500000  1.611405 68 5.895475  0.0000
 Correlation:
                 (Intr) crwdngc crwdngv nmfthr nmchl1 nmchl2
crowdingcrow     -0.544
crowdingovercrow -0.544  0.500
namefather       -0.506  0.000   0.000
namechild1       -0.506  0.000   0.000   0.500
namechild2       -0.506  0.000   0.000   0.500  0.500
namechild3       -0.506  0.000   0.000   0.500  0.500  0.500

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.25198520 -0.69457063 -0.01723823  0.57199493  2.80356511

Number of Observations: 90
Number of Groups: 18
```

```
> anova(lme.1, type = "marginal")
```

```
            numDF denDF   F-value p-value
(Intercept)     1    68  3.569639  0.0631
crowding        2    15  5.223048  0.0190
name            4    68 16.406611  <.0001
```

```
> intervals(lme.1)$fixed[c("crowdingovercrow","namechild3"),]
```

```
                    lower est.     upper
crowdingovercrow 1.906594  5.6  9.293406
namechild3       6.284491  9.5 12.715509
```

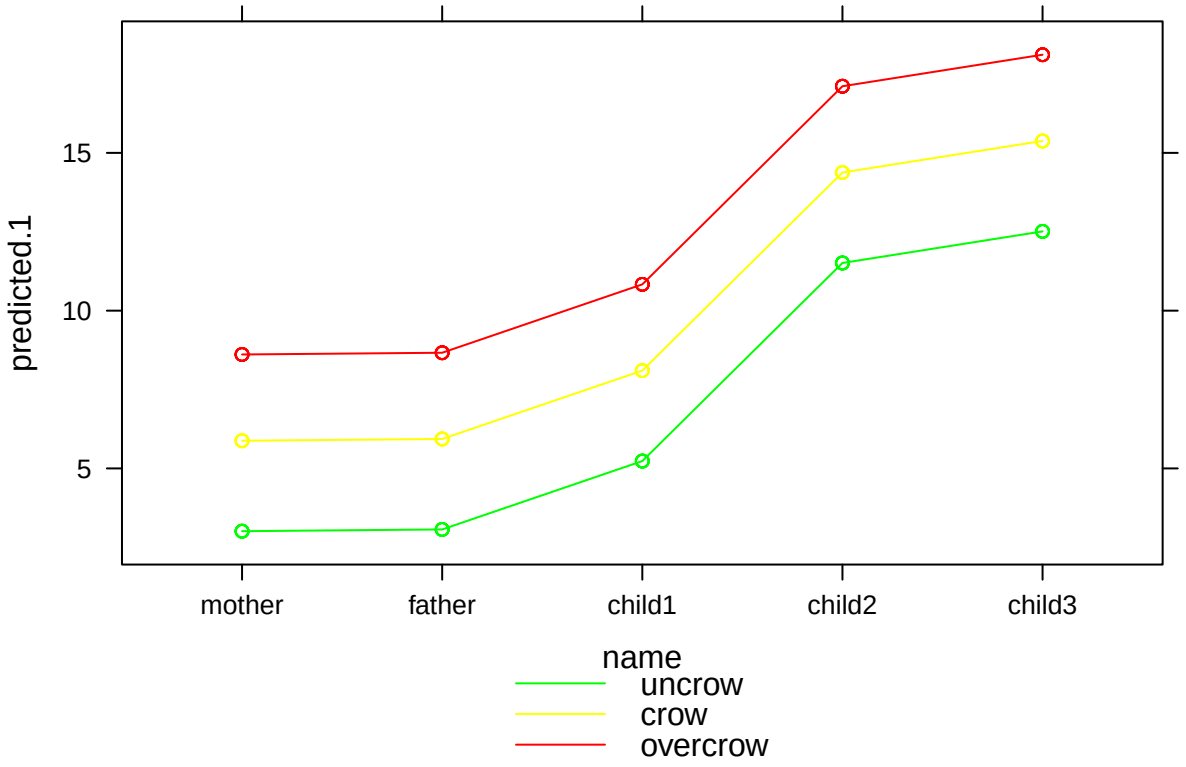
```
> # See appendix A for a display of the confidence intervals for the mean values
```

```
> lme.1interact <- lme(swabs ~ crowding*name,
+                      data = df.data_swabs,
+                      random = ~1 | family)
>
> anova(lme.1interact, type = "marginal")
```

```
              numDF denDF  F-value p-value
(Intercept)       1    60 1.647788  0.2042
crowding          2    15 2.075414  0.1601
name              4    60 4.214980  0.0045
crowding:name     8    60 0.358006  0.9384
```

Graphical display

```
> df.Pred_swabs <- data.frame(df.data_swabs,
+                             predicted.1 = predict(lme.1, level = 0, type = "response"))
> # level 0 indicates to use the marginal model (i.e. average over random effet)
>
> legend_tempo <- list(space="bottom",
+                      lines = list(col=c("green", "yellow", "red"), lty=1),
+                      text = list(levels(df.data_swabs$crowding))
+ )
>
> xyplot(predicted.1 ~ name, group = crowding,
+        data = df.Pred_swabs[order(df.Pred_swabs$name),],
+        type = "b", col = c("green", "yellow", "red"),
+        key = legend_tempo # optional argument
+ )
```




```
> # See appendix A for a display of the confidence intervals for the mean values
```

Potential interaction name crowding

Question 3: Ignore inter-family correlation

```
> lm.1 <- lm(swabs ~ crowding + name ,
+            data = df.data_swabs)
> summary(lm.1)
```

```
Call:
lm(formula = swabs ~ crowding + name, data = df.data_swabs)

Residuals:
     Min       1Q   Median       3Q      Max
-13.1111  -2.7861  -0.0833   2.8222  16.8889

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      3.01111    1.45679   2.067   0.0419 *
crowdingcrow     2.86667    1.34873   2.125   0.0365 *
```

```
crowdingovercrow  5.60000    1.34873   4.152 7.94e-05 ***
namefather        0.05556    1.74120   0.032   0.9746
namechild1        2.22222    1.74120   1.276   0.2054
namechild2        8.50000    1.74120   4.882 5.03e-06 ***
namechild3        9.50000    1.74120   5.456 4.93e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.224 on 83 degrees of freedom
Multiple R-squared:  0.4695,    Adjusted R-squared:  0.4311
F-statistic: 12.24 on 6 and 83 DF,  p-value: 8.078e-10
```

Question 4: Average within each family

New dataset:

```
> n.family <- nlevels(df.data_swabs$family)
>
> df.data_swabs.average <- data.frame(
+   crowding = tapply(df.data_swabs$crowding, df.data_swabs$family, "[",1),
+   swabs = tapply(df.data_swabs$swabs, df.data_swabs$family, mean),
+   family = tapply(df.data_swabs$family, df.data_swabs$family, "[",1)
+ )
>
> #### alternative using data.table
> # dt.data_swabs <- as.data.table(df.data_swabs)
> # dt.data_swabs.average <- dt.data_swabs[, .(swabs = mean(swabs), crowding = crowding[1]),
> #                                        by = family]
> # df.data_swabs.average <- as.data.frame(dt.data_swabs.average)
>
> df.data_swabs.average$crowding <- factor(df.data_swabs.average$crowding,
+                                          levels = 1:3, label = c("uncrow", "crow", "overcrow"))
> df.data_swabs.average$family <- factor(df.data_swabs.average$family)
```

Anova

```
> aov.average <- gls(swabs ~ crowding, data = df.data_swabs.average)
> anova(aov.average)
```

```
Denom. DF: 15
            numDF   F-value p-value
(Intercept)     1 195.40655  <.0001
crowding        2   5.22301   0.019
```

```
> intervals(aov.average)
```

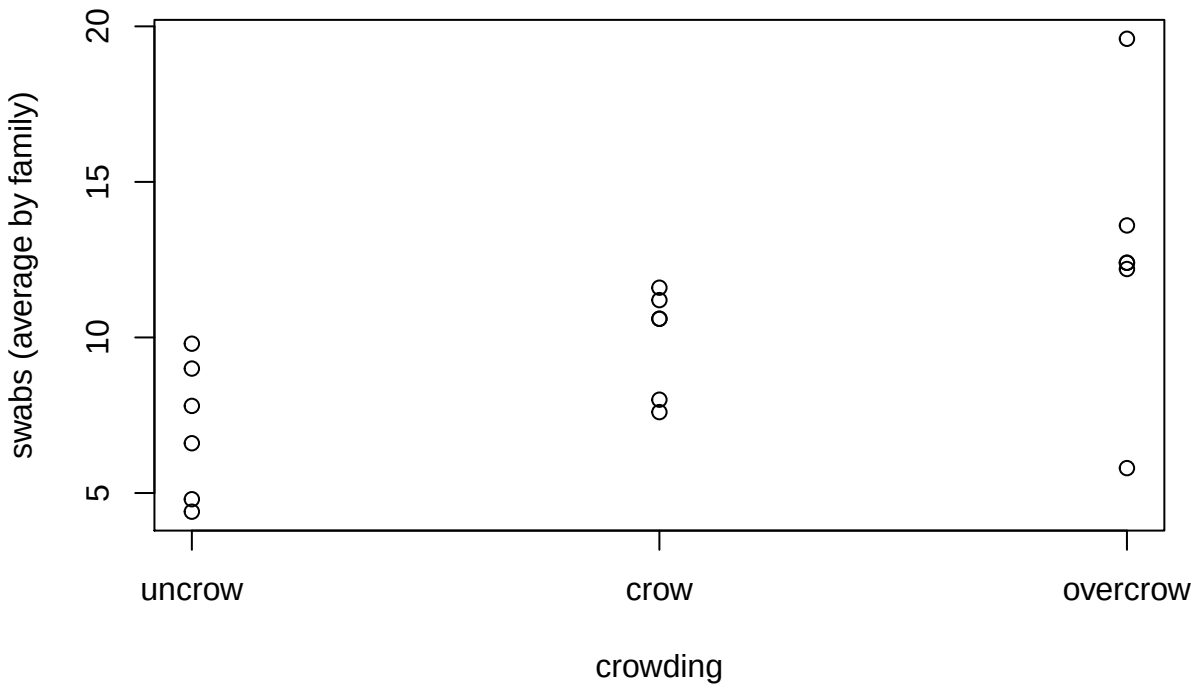
```
Approximate 95% confidence intervals

 Coefficients:
                    lower     est.    upper
(Intercept)       4.4550248 7.066667 9.678309
```

```
crowdingcrow     -0.8267527 2.866667 6.560086
crowdingovercrow  1.9065807 5.600000 9.293419
attr(,"label")
[1] "Coefficients:"

 Residual standard error:
   lower     est.    upper
2.217099 3.001333 4.645137
```

```
> #### display
> plot(x = as.numeric(df.data_swabs.average$crowding),
+      y = df.data_swabs.average$swabs,
+      ylab = "swabs (average by family)", xlab = "crowding", axes = FALSE)
> axis(1, at = 1:3, labels = c("uncrow", "crow", "overcrow"))
> axis(2)
> box()
```




```
> # alternative
> # plot(x = df.data_swabs.average$crowding, y = df.data_swabs.average$swabs)
```

Question 5: Ignore effect of crowding

```
> gls.2noCrow <- gls(swabs ~ family + name,
+                    data = df.data_swabs)
>
> anova(gls.2noCrow, type = "marginal")
```

```
Denom. DF: 68
            numDF   F-value p-value
(Intercept)     1 12.188865   8e-04
family         17  2.884815   1e-03
name            4 16.406637  <.0001
```

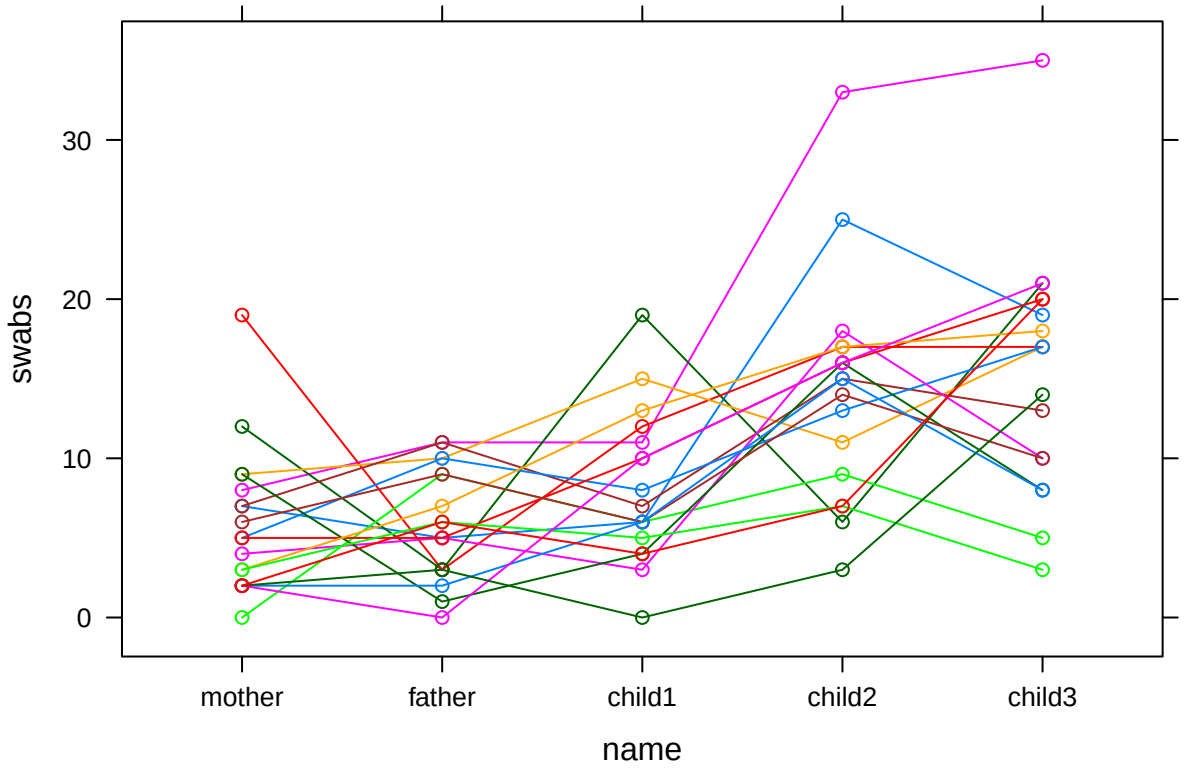
```
> intervals(gls.2noCrow)
```

```
Approximate 95% confidence intervals

 Coefficients:
                  lower          est.      upper
(Intercept)   3.5750772  8.344444e+00 13.1138117
family2       1.0990049  7.200000e+00 13.3009951
family3      -6.3009951 -2.000000e-01  5.9009951
family4      -4.9009951  1.200000e+00  7.3009951
family5      -6.1009951  5.615566e-15  6.1009951
family6     -12.7009951 -6.600000e+00 -0.4990049
family7      -7.9009951 -1.800000e+00  4.3009951
family8      -7.9009951 -1.800000e+00  4.3009951
family9     -10.5009951 -4.400000e+00  1.7009951
family10    -10.9009951 -4.800000e+00  1.3009951
family11     -7.3009951 -1.200000e+00  4.9009951
family12     -6.9009951 -8.000000e-01  5.3009951
family13    -13.7009951 -7.600000e+00 -1.4990049
family14     -9.5009951 -3.400000e+00  2.7009951
family15    -11.9009951 -5.800000e+00  0.3009951
family16     -8.7009951 -2.600000e+00  3.5009951
family17    -14.1009951 -8.000000e+00 -1.8990049
family18    -10.7009951 -4.600000e+00  1.5009951
namefather   -3.1599512  5.555556e-02  3.2710623
namechild1   -0.9932845  2.222222e+00  5.4377290
namechild2    5.2844932  8.500000e+00 11.7155068
namechild3    6.2844932  9.500000e+00 12.7155068
attr(,"label")
[1] "Coefficients:"

 Residual standard error:
   lower     est.    upper
4.140637 4.834212 5.809071
```

```
> #### display
> xyplot(swabs ~ name, groups = family,
+        data = df.data_swabs[order(df.data_swabs$name),],
+        type = "b" , xlab = "name", ylab = "swabs")
```


swabs
30
20
10
0
mother
father
child1
child2
child3
name

Appendix A: Compute the confidence intervals for predictions

"The general recipe for computing predictions from a linear or generalized linear model is to

1- figure out the model matrix X corresponding to the new data;

2- matrix-multiply X by the parameter vector β to get the predictions;

3- extract the variance-covariance matrix of the parameters V

4- compute XVX' to get the variance-covariance matrix of the predictions;

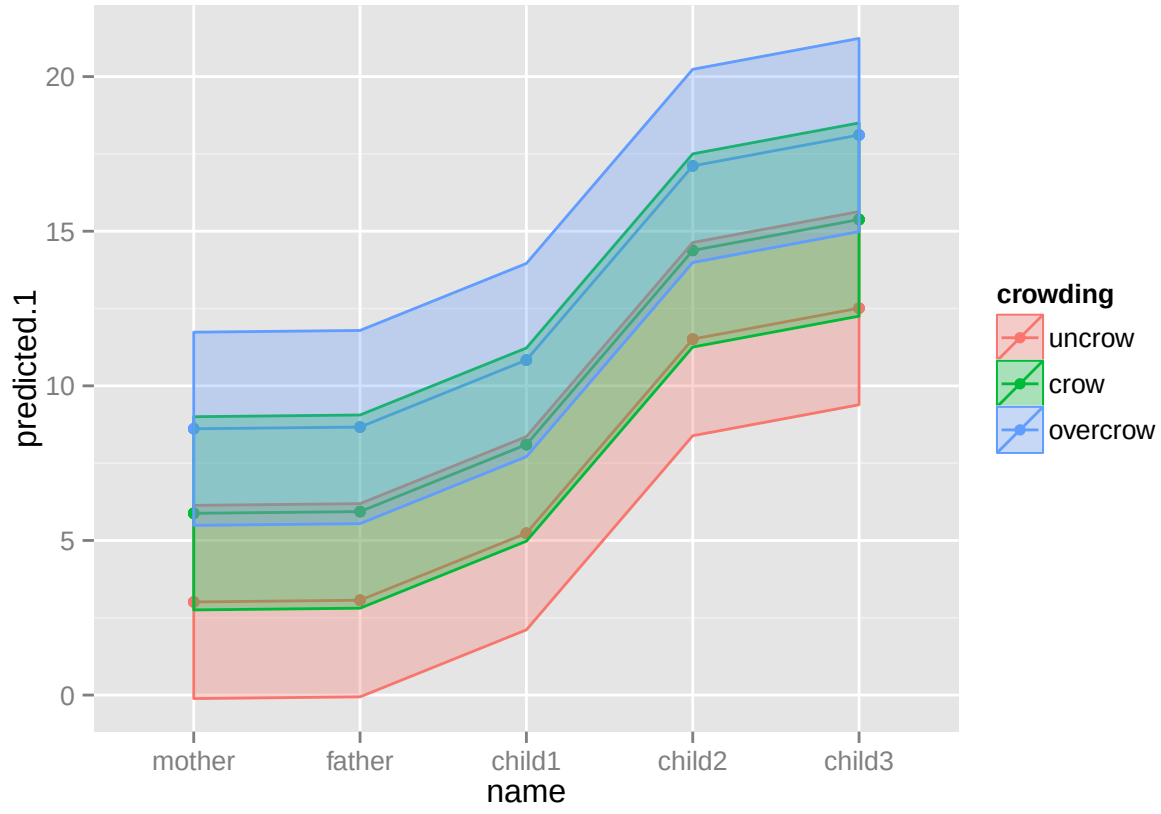
5- extract the diagonal of this matrix to get variances of predictions;

6- take the square-root of the variances to get the standard deviations (errors) of the predictions;

7- compute confidence intervals based on a Normal approximation;"

(from http://glmm.wikidot.com/faq)

```
> #### 1
> Xmatrix <- model.matrix( ~crowding + name, df.data_swabs)
>
> #### 2
> predictions <- predict(lme.1, level = 0)
> # same as Xmatrix %*% lme.1$coefficients$fixed
> # same as df.Pred_swabs$predicted.1
>
> #### 3
> VCOV.beta <- vcov(lme.1)
>
> #### 4
> VCOV.predictions <- Xmatrix %*% VCOV.beta %*% t(Xmatrix)
>
> #### 5
> VAR.predictions <- diag(VCOV.predictions)
>
> #### 6
> SE.predictions <- sqrt(VAR.predictions)
>
> #### 7
> quantile_norm <- qnorm(p = 0.975)
> df.Pred_swabs$predicted_Lower <- predictions - quantile_norm * SE.predictions
> df.Pred_swabs$predicted_Upper <- predictions + quantile_norm * SE.predictions
>
>
> #### display with ggplot
> gg.base <- ggplot(df.Pred_swabs,
+                   aes(x = name, y = predicted.1,
+                       group = crowding, color = crowding))
> gg.punctal <- gg.base + geom_line() + geom_point()
> gg.punctal + geom_ribbon(aes(ymin = predicted_Lower, ymax = predicted_Upper,
+                              fill = crowding, color = crowding),
+                          alpha = 0.3)
```




```
> #### display with lattice
> # xyplot_estimate <- xyplot(predicted.1 ~ name, group = crowding,
> #                           data = df.Pred_swabs[order(df.Pred_swabs$name),],
> #                           type = "b")
> #
> # xyplot_Lower <- xyplot(predicted_Lower ~ name,
> #                        group = crowding,
> #                        data = df.Pred_swabs[order(df.Pred_swabs$name),],
> #                        type = "l", lty = 3, col = c("blue","red","green"))
> #
> # xyplot_Upper <- xyplot(predicted_Upper ~ name,
> #                        group = crowding,
> #                        data = df.Pred_swabs[order(df.Pred_swabs$name),],
> #                        type = "l", lty = 3, col = c("blue","red","green"))
> #
> # library(latticeExtra)
> # xyplot_estimate + as.layer(xyplot_Lower) + as.layer(xyplot_Upper)
```