**Running Mixed Models in the R Package**

Alice Sweeting  
Victoria University  
Melbourne, Australia   
[Email](mailto:alice.sweeting@live.vu.edu.au)

This document provides model statements in R corresponding to the SAS programs in the accompanying resource on mixed modeling with SAS (authored by Will Hopkins). Some of his text has been copied from the Word docs in that resource. An introduction to R and useful code for physiology users can be found at [my blog](https://sportstatisticsrsweet.wordpress.com/).

There are two packages in R for mixed modeling, lme4 and nlme. The nlme concerns *non-linear mixed modeling*, which appears to be equivalent of the non-linear mixed-model procedure in SAS, Proc Nlmixed. However, this workshop does not use nlme. The programs shown here make use of the package lme4, which does general and generalized linear mixed modeling. This package will need to be installed into R prior to use. The .R script file accompanying this doc in the zip file contains all the relevant code for the analyses.

**Reliability analyses:**[One-way](#oneway)  
[Two-way](#twoway)  
[Two-way with Missing Values](#twowaymiss)  
[Two-way with a Fixed Effect](#twowaymissplus)   
[Extending the Reliability Mixed Model to Monitoring and Clustering](#extending)

**One-way Reliability Model**

1. Import the 2-way long spreadsheet from the workbook "reliability data.xlsx". This needs to be imported via a .csv or .txt file. Name the dataset rely2waylong.
2. Here's the code to run the simplest of all possible mixed models, a one-way reliability model:

require(lme4)

rely1way <- lmer(LnDepVar ~ 1 + (1 | AthleteID), data = rely2waylong)

summary(rely1way)

confint(rely2way, level=0.9)

Here is the output:

Random effects:

Groups Name Variance Std.Dev.

AthleteID (Intercept) 1101.0 33.18

Residual 100.8 10.04

Number of obs: 200, groups: AthleteID, 20

Fixed effects:

Estimate Std. Error t value

(Intercept) 597.728 7.453 80.2

Computing profile confidence intervals ...

5 % 95 %

.sig01 25.374884 43.07264

.sigma 9.230651 10.97980

(Intercept) 585.363355 610.09334

These are log-transforms, which need to be back-transformed to percents using 100\*exp(1101)-100, etc. See the “one-way” tab in the workbook "convert covparms to SDs.xlsx", where it is done with the values from SAS. The value for AthleteID is the SD of the athletes' mean values, free of measurement error. Residual is the error of measurement, and the fixed-effect intercept is the mean of the athletes' means.

**Two-way Reliability Model**

1. Here's the code with GameID included as a random effect. In other words, imagine that the games are also a random sample.

rely2way <- lmer(LnDepVar ~ 1 + (1 | AthleteID) + (1 | GameID),

data = rely2waylong)

summary(rely2way)

Here is the output:

Random effects:

Groups Name Variance Std.Dev.

AthleteID (Intercept) 1102.16 33.199

GameID (Intercept) 11.96 3.459

Residual 88.88 9.428

Number of obs: 200, groups: AthleteID, 20; GameID, 10

Fixed effects:

Estimate Std. Error t value

(Intercept) 597.728 7.533 79.35

Computing profile confidence intervals ...

5 % 95 %

.sig01 25.412518 43.110518

.sig02 1.932157 5.932842

.sigma 8.647601 10.332910

(Intercept) 585.234515 610.222186

You will have to back-transform these to percents.

You can derive an intraclass (retest) correlation coefficient from the variances as follows: AthleteID/(AthleteID+Residual).

Compare the Intercept and Residual here with those from the one-way analysis: the intercepts are practically the same (differences between players should not depend on whether some games are harder than others), but the residual here is smaller. I hope it's obvious why: the variation in the mean of each game from game to game contributed to the within-player variability in the one-way analysis, but it has been taken out of the picture here.

**Two-way Reliability Model with Missing Values**

1. Import the spreadsheet with missing values via the tab "2-way long missing". The code is the same, but the output is a little different, reflecting the loss of some data.

Random effects:

Groups Name Variance Std.Dev.

AthleteID (Intercept) 529.15 23.003

GameID (Intercept) 4.87 2.207

Residual 102.58 10.128

Number of obs: 94, groups: AthleteID, 20; GameID, 10

Fixed effects:

Estimate Std. Error t value

(Intercept) 607.632 5.359 113.4

Computing profile confidence intervals ...

5 % 95 %

.sig01 17.373146 30.118552

.sig02 0.000000 5.666069

.sigma 8.858848 11.734436

(Intercept) 598.740471 616.537848

The confidence limits for GameID are now quite different from those of SAS, which in this case are unrealistic. As noted in the documentation for this analysis in SAS, the solution is to allow negative values of the variances and their confidence limits, a feature of SAS that R and lme4 currently does not offer.

**Two-way Reliability Model with Missing Values and a Fixed Effect**

1. Use the same dataset as above, but add a term for LoseTieWin to the model:

rely2waymissplus <- lmer(LnDepVar ~ LoseTieWin + (1 | AthleteID) +

(1 | GameID), data = rely2waylongmiss)

summary(rely2waymissplus)

confint(rely2waymissplus, level=0.9)

Here is the output:

Random effects:

Groups Name Variance Std.Dev.

AthleteID (Intercept) 522.58 22.860

GameID (Intercept) 0.00 0.000

Residual 97.49 9.874

Number of obs: 94, groups: AthleteID, 20; GameID, 10

Fixed effects:

Estimate Std. Error t value

(Intercept) 610.490 5.392 113.23

LoseTieWinT -2.320 2.345 -0.99

LoseTieWinW -8.452 2.722 -3.11

Computing profile confidence intervals ...

5 % 95 %

.sig01 17.277740 29.898056

.sig02 0.000000 2.335827

.sigma 8.558855 11.218108

(Intercept) 601.581965 619.415344

LoseTieWinT -6.155122 1.514452

LoseTieWinW -12.915210 -4.003555

Note that R has produced zero variance for GameID. SAS by default also produced zero, but SAS has an option for allowing negative variance, which is the correct approach. Interestingly, R produced a small positive variance for the upper confidence limit, whereas SAS's upper confidence limit (with the negative-variance option) was slightly negative. Conclusion: R's confidence limits for random effects when the variances are close to zero ("close" in terms of the residual variance) are untrustworthy.

Notice that the fixed-effects part of the model produced an intercept, which is the first alphabetical level of the fixed effect LoseTieWin, while the other two levels are the differences of those levels from the first level (Tie-Lose, Win-Lose). To produce the three means, use this model:

rely2waymissplus <- lmer(LnDepVar ~ -1 + LoseTieWin + (1 | AthleteID) +

(1 | GameID), data = rely2waylongmiss)

summary(rely2waymissplus)

Here is the output:

Fixed effects:

Estimate Std. Error t value

LoseTieWinL 610.490 5.392 113.2

LoseTieWinT 608.170 5.387 112.9

LoseTieWinW 602.038 5.596 107.6

See the SAS documentation for extensive interpretation of the output.

1. One final cool feature of mixed models that R shares with SAS is the "solution" for the random effects, which here allow you to assess the performance of each athlete. You output these with this bit of code:

ranef(rely2waymissplus)

And here is the output:

$AthleteID

(Intercept)

Alex 8.684583

Ariel 30.737495

Ashley -20.704054

Bernie -16.217553

Casey 10.866093

Chris 16.591984

Corey 7.972682

Courtney 6.615328

Devon -18.530437

Drew 49.223536

Dylan -5.758964

Frances -17.695587

Gene 3.538779

Jaimie -1.032421

Jean 26.044018

Jesse -39.039007

Jo -6.806228

Jody -7.633829

Jordan 13.452205

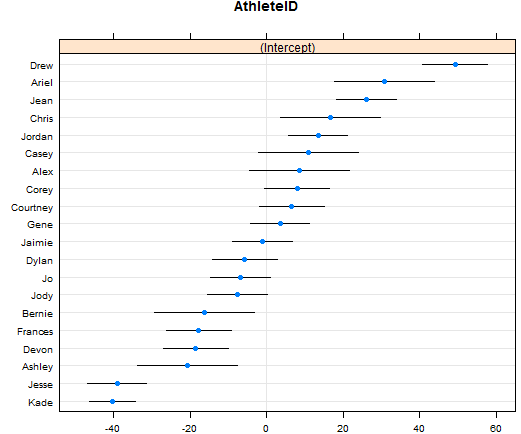
Kade -40.308619

You can even plot them in rank order, with confidence limits, using this code:

require(ggplot2)

require(lattice)

dotplot(ranef(rely2waymissplus, condVar=TRUE))



The plot can be saved in any format, including editable metafile. R beats SAS, for once.

**Extending the Reliability Mixed Model to Monitoring and Clustering**

1. Here's a fairly simple example of real data with clusters and trends. The data come from a study of the effects of caffeine on repeated sprints, done as a crossover with placebo and several doses of caffeine, each a week apart. On each of the three testing days, the athletes performed four sets of six all-out 30-m sprints. To keep it simple, data is only included for the placebo. The aim of the analysis is to quantify how much the subjects fatigue in each set of six sprints, taking into account and estimating individual differences in the fatigue. We'll also account for any consistent changes in mean sprint time between sets (i.e., is there fatigue between sets as well as within sets), and any variability in each subject's mean sprint time from set to set (which probably won't make sense to you until we develop the model). Oh, and most subjects make a bigger effort on the last sprint of a set, so we'll account for and estimate that, too, along with individual differences!
2. Save the "sets of reps long format.xlsx" worksheet as a .csv file and import into R. The variables Athlete, Set, Rep, Time and LnTime should be obvious enough. xVarRep6 is a dummy variable generated in Excel to model extra effects on the last (6th) sprint of each of the four sets. (A simple name like Rep6 would have done.) Check out its values. Ensure Athlete and Set are factors in R, which can be done by running the following line of code:

SetsRepsLongFormat$Athlete <- as.factor(SetsRepsLongFormat$Athlete)

SetsRepsLongFormat$Set <- as.factor(SetsRepsLongFormat$Set)

1. There are many ways to summarise data in R but I prefer the “plyr” package, which will need to be installed into R prior to use. The code below will create a new data.frame that contains summary data of the DV, Time, for each rep within each set:

require(plyr)

SummaryData <- ddply(SetsRepsLongFormat,.(Set, Rep), summarise,

Mean = mean(Time),

SD = sd(Time),

LowerCI = quantile(Time, 0.05),

UpperCI = quantile(Time, 0.95))

A cool part of R is its graphics packages and the ability to create publication worthy plots with a few lines of code. A package that I use nearly every day is “ggplot2” that has a great deal of development and user guides. Install the package and run the following lines of code to generate a figure:

SummaryData$Set <- as.factor(SummaryData$Set)

require(ggplot2)

ggplot(data = SummaryData, aes(x = Rep, y = Mean, colour = Set)) +

geom\_line() +

geom\_point() +

ylab("Mean Time (s)\n") +

xlab("\nRepetition") +

scale\_y\_continuous(expand = c(0, 0), limits = c(7.9, 8.55)) +

scale\_x\_continuous(expand = c(0, 0), limits = c(0.5, 6.5), breaks = c(1:6)) +

theme\_classic() +

theme(legend.position = "bottom",

axis.line = element\_line(color = "black", size = .7),

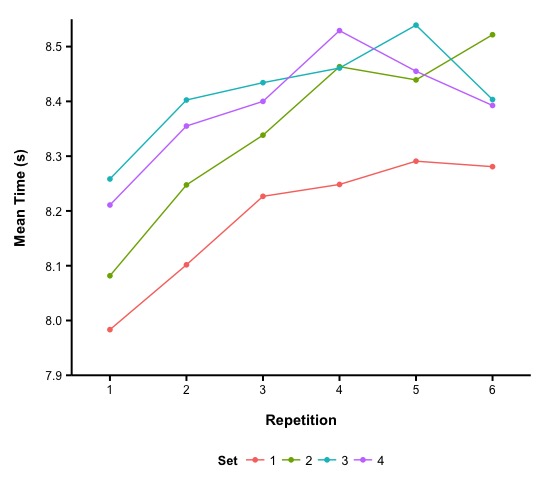
axis.ticks = element\_line(color = "black", size = .7),

axis.text.y = element\_text(colour="black", size=9),

axis.text.x = element\_text(colour="black", size=9),

axis.title.y = element\_text(face ="bold", colour="black", size=11),

axis.title.x = element\_text(face ="bold", colour="black", size=11))



There's obvious fatigue over the first five reps in each set, but in all but the second set the athletes have gone relatively faster on the last rep. So we'll develop a model that estimates linear fatigue in each set, apart from the last rep.

1. For reasons that will become clear, we need a new Rep variable that goes from -0.5 to 0.5 rather than 1 to 6. In R, run the code below to create a new column containing the above variable

SetsRepsLongFormat$RepRescaled <- ((SetsRepsLongFormat$Rep - 1)/5 - 0.5)

Check that the values do indeed go from -0.5 to +0.5.

1. Now let's build the model. Athlete is a factor variable, because it will be a random effect. Set is a factor variable, because we'll look at the mean for each set. RepRescaled in a covariate, because fatigue usually develops in a linear fashion with repeated sprinting, and we want an overall estimate of what I call linearized fatigue from first to last sprint. That is, we will estimate the linear component of fatigue over six sprints. If we had used Rep rather than RepRescaled, we'd have to multiply its coefficient by 5 (yes, 5, not 6) to get the overall fatigue in six sprints. OK that's not a big deal, but it makes the individual differences in the slope a lot easier to interpret, as we will see. xVarRep6 is a dummy continuous variable that will estimate the extra effort the athletes aka on the last sprint of each set.

This model implies we're going to estimate a different mean for each set, a different mean slope for each set, and a different mean time for the last rep averaged across all the sets. Because we're already estimating mean time in each rep in each set with the other two terms, the effect for this dummy will be the amount by which the athletes on average go faster than a linear fatigue model predicts for the last rep.in each set. This random-effect model generates a different random value for athlete (they differ in how fast they sprint), and a different random value for Athlete\*RepRescaled (they differ in how much they fatigue). How cool is that?! "Unstructured" allows these two effects to be correlated: faster athletes might fatigue more, for example. It's called an individual-slopes model.

Set (i.e., Athlete\*Set) represents the variability in an athlete's mean time between sets, which might not be much here, but it would be more if there were days rather than minutes between sets. xVarRep6 (i.e., Athlete\*xVarRep6) represents individual differences (consistent over the 4 sets) from the mean value provided by the fixed effect for xVarRep6.

To run the SAS equivalent model in R, run the following line of code:

ClusteredTrial <- lmer(LnTime ~ -1 + Set + RepRescaled + Set:RepRescaled + xVarRep6 + (0 + xVarRep6 | Athlete) + (1 | Set:Athlete) + (1 + RepRescaled | Athlete), data = SetsRepsLongFormat)

summary(ClusteredTrial)

1. Let's deal with the random effects first, because they also provide us with magnitude thresholds for interpreting their magnitude and the magnitude of the fixed effects. Here they are:

Random effects:

Groups Name Variance Std.Dev. Corr

Set.Athlete (Intercept) 1.5639 1.2505

Athlete (Intercept) 18.0996 4.2544

RepRescaled 3.0136 1.7360 -0.71

Athlete.1 xVarRep6 0.4713 0.6865

Residual 3.4592 1.8599

Number of obs: 288, groups: Set:Athlete, 48; Athlete, 12

See the SAS documentation for interpretation of the random effects.

1. Now the fixed effects:

Fixed effects:

Estimate Std. Error t value

Set1 210.4321 1.3003 161.83

Set2 212.3904 1.3003 163.34

Set3 213.1974 1.3003 163.96

Set4 212.8918 1.3003 163.72

RepRescaled 4.9994 0.8603 5.81

xVarRep6 -1.7675 0.4366 -4.05

Set2:RepRescaled 1.2762 0.9075 1.41

Set3:RepRescaled -1.6798 0.9075 -1.85

Set4:RepRescaled -1.4440 0.9075 -1.59

See the SAS documentation for interpretation of the fixed effects.