Practical 1: Random intercepts models

This session aims to provide a straightforward example of fitting a two-level multilevel model with a continuous response and a continuous predictor. It is concerned with the practicalities of model specification and estimation. The practical is conducted in R. As always with the program there are several ways of doing the same thing and we will try and guide you through a convenient route. We will consider the following models:

1. a random intercepts null model with school performance (at age 16) as the response; no predictor variables (apart from the Constant) and with the levels defined as pupils (level 1) within schools (level 2);
2. a model which additionally includes past performance (age 11) as a (pupil level) predictor variable;
3. a model which additionally includes average school performance as a (school level) predictor variable.

For any multilevel model, there is a basic sequence of procedures which we will follow:

* data input and exploration;
* model specification: response, predictors, levels, terms for the fixed and random parts of the model;
* estimation: the fitting of the specified model by a chosen procedure;
* examining the model estimates and values such as standard errors;
* estimating the residuals at each level for diagnosis of model ills, as well as to make substantive interpretations;
* graphing the results both to look at estimated residuals and predictions from the estimated model.

**Data input and manipulation**

We would recommend doing this analysis through RStudio. A sensible first step would be to create a project pointing at a file that contains the dataset “TutorialData.dta”. This will ensure you are starting R with a clean slate without any additional objects in the working environment. Alternatively, you can change the working directory to point at that file manually, for instance:

setwd(“full file path here”)

We will first install and load the packages that we will need to run the analysis in R

install.packages("directlabels", "haven", "tidyverse", "lme4", "merTools", "sjPlot")

library(haven)

library(tidyverse)

library(directlabels)

library(lme4)

library(merTools)

library(sjPlot)

Next, you will need to load the tutorial.dta data. To input it into R, you will need to use the command read\_dta in the package haven. You should call the dataframe tutorial.

tutorial <- read\_dta(tutorial.dta)

Take a look at your data, and look at the descriptive statistics / histograms for some of the key variables.

str(tutorial)

summary(tutorial)

*Self test: what is the range of the variable normexam? (this is our dependent variable) Is it Normally distributed?*

*Self test: how many students are in the dataset*

*Self test: What is the range of the variable “girl”? Can you explain why?*

*Self test: What is the normexam score of student 1 in school 1?*

**Model 1: a two-level null random intercepts model**

We are now ready to fit our first multilevel model. We want to specify a 2 level null random intercepts model, that is a model with no predictors (except for the constant). First we need to install the lme4 package

library(lme4)

nullmodel <- lmer(normexam ~ (1 | school), data=tutorial)

The lmer fuction is used here – that stands for ‘linear mixed effects regression’, another name for multilevel models. We then specify our dependent variable normexam. The intercept is included by default. In this model there are no explanatory variables. The ~ is used as it is in OLS regression with the lm function and is then followed by the X variables (in this case, there aren’t any).

We then specify the random part of the model. In doing this we have specified the structure of the response variable as being 2 levels, by stating that the highest level is defined by school (the lower level is assumed to just be the level of the observations – students in this case). By default we will have the intercept estimated in the fixed part of the model (like a standard intercept in a regression model), and that parameter is being allowed to vary at level 2 – this is done by including a “1” in the code (1 represents the intercept). Note that if you wanted to specify a single level (standard regression) model, you would not include the (1 | school)section. By default, the model will be fitted by restricted maximum likelihood estimation.

We can now look at the regression output

summary(nullmodel)

There are lots of parts to this output.

First, there are details about the convergence of the model, and then details of the (level 1) residuals. You can usually ignore these.

At the bottom are the fixed effects. These can be interpreted very similarly to regression coefficients in OLS regression, the only difference is a p value is not provided (you would need to work that out yourself based on the t-value). But for now, you can assume a t-value of greater than 1.96 implies a p value of less than 0.05 (that is, significance at the arbitrary 95% level).

*Self test: What is the coefficient associated with the intercept? What does it mean? Is it statistically significant at the 95% level?*

In between these two sections, we have the random effects. These are the estimates of the variance at the school level, and the variance at the pupil level. Next to these, are the standard deviations (simply the square root of the variance)

*Self test questions:*

*How big is the level 2 variance?*

*How big is the level 1 variance?*

At this stage we might want to estimate the variance partitioning coefficient – that is the proportion of the variance that exists at the school level. In this case the calculation for this is

VPC = school level variance / total variance

VPC = 0.172 / (0.172 + 0.848)

Which is about 0.17. As such, about 17% of the variance in the response variable normexam occurs at the school level. This could be interpreted as the amount that schools matter (although it could also be the result of selection effects into the school). There is much more variation within schools than there is between schools.

*Self test question: what might this mean, if you were a parent, in deciding which school to send your child to?*

Make sure to save your code, in case R crashes!

**Is this model better than a single level model?**

We probably want to know whether this model fits the data better than a single level model. Note that R doesn’t give us a standard error or any significance level associated with these variances (as it does with the fixed part of the model).

First, we should fit a single level model:

singlelevel <- lm(normexam~1, data=tutorial)

Note that we need to include a 1 in there, which represents the intercept (it is assumed when there are other predictor variables in the model, but needs to be specified here)

We then want to compare the models using one of the model fit statistics. We could, for example, compare the AIC values for the two models

AIC(nullmodel, singlelevel)

The AIC is a statistic that considers both how well the model fits the data, and how complex the model is. Lower scores indicate a better model fit given the complexity of the model. In this case, the AIC for the multilevel model is very much lower, meaning we can assume the multilevel structure is important. Note we could have used the BIC and gained a similar result.

**Adding a predictor variable**

We now want to include a predictor variable in the model: in this case, we want to include past performance as a predictor of future performance. This is interesting in its own right, but also means we are controlling (at least in part) for selection effects in considering school-level and student-level variance. Our measure of past performance is the variable called ‘standlrt’

randomint <- lmer(normexam ~ standlrt + (1 | school), data=tutorial)

summary(randomint)

*Self test questions:*

*Fill in the blanks: as \_\_\_\_\_\_\_ increases by 1, we would expect \_\_\_\_\_\_\_ to increase by 0.563 on average.*

*Is the effect of standlrt on normexam statistically significant at the 95% level?*

*What has happened to the level 1 and level 2 variance with the adding of the new variable? Can you explain why?*

*What is the VPC for the new model?*

*Self test: any questions at this point? Anything you don’t understand about what you are doing? Ask Andy now if so!*

**Examining Residuals**

The model that we have just fitted is very similar to the model used to get ‘value added’ scores in school league tables. We have estimated the variance of schools (how different they are from one another), once we have controlled for past performance. However we have not yet looked at the schools themselves, to see which schools are the best or the worst. We do this by, post-estimation of the model, calculating the residuals at the school level (labelled as u0j in the equations window). The residuals should also be examined to check the assumption that they are distributed Normally.

To calculate the residuals we can use the following code to produce estimates of our level 2 residuals, and a ggplot graph of them (you’ll need to install and load the merTools package):

reEX <- REsim(randomint)

plotREsim(reEX, labs=TRUE)

The best performing school is school number 53. It also reveals the uncertainty that exists around the rankings. Whilst there are clearly differences between the very worst and very best ranked schools, for most schools there is no statistically significant difference between themselves and the average school.

*Self test:*

*Which school is the worst performing? How much worse than average does this school perform?*

*Which schools do you think have the smallest sample size? Why do you say this?*

*Given the width of the confidence intervals, how useful are league tables in helping parents choose schools for their children?*

**Plotting random intercepts**

The next plot that we want to produce is a line graph showing the relationship between standlrt (our predictor variable) and normexam (our response), with a different line plotted for each school. To do this, we first want to create a prediction – that is what the model’s best guess of what the normexam score is, given their school and standlrt score.

predexam <- fitted(randomint)

This will produce a new variable that contains the predictions of normexam based on the model, for values of standlrt and each of the schools.

We then want to plot these predictions against standlrt, using ggplot:

install.packages("directlabels")

library("directlabels")

This will allow us to put the school label directly on the graph…

ggplot(tutorial, aes(x=standlrt, y=predexam, group=school, color=school)) +

geom\_line() +

scale\_colour\_discrete(guide = 'none') +

geom\_dl(aes(label=school), method="last.points")

The resultant graph visualises the relationship between standlrt and normexam in each school. Note however, that we are currently constraining these lines to be parallel. We can relax this assumption with a random slope model, but we do not explore this here.

We have (1) run a model, (2) made predictions for what that model thinks is happening, and (3) plotted those predictions. This gives us a much clearer picture of what is going on than we would get if we just plotted the raw data (which would be rather too messy to be able to understand.

**Adding higher level variables**

Adding higher (school) level variables is done in exactly the same way as level 1 variables. In this case, we can add the variable avslrt, which is a measure of the average school’s verbal reasoning ability.

randomint2 <- lmer(normexam ~ standlrt + avslrt + (1 | school), data=tutorial)

summary(randomint2)

*Self test questions:*

*How big is the school effect of verbal reasoning ability on normexam?*

*What has happened to the variance at level 2? Can you explain why?*

*What has happened to the variance at level 1? Can you explain why?*

Finally, it would be sensible to make a table of the models we have produced throughout this model building strategy. We can do this in the same way as a single-level model, using tab\_model:

tab\_model(singlelevel, nullmodel, randomint, randomint2)

Again, you might want to change some things in this table, which you can do within the tab\_model command options. The table automatically gives you the VPC (although it calls it the ICC, or intra-class correlation).

Make sure you now save your code (we will be using the same models/data in future sessions...)

Key learning outcomes from this session

* How to fit a 2-level random intercepts model
* How to calculate the VPC
* How to produce graphs to visualize the model’s residuals school effects
* How to add variables to the model, and interpret their parameter estimates.

## Practical 2: MAIHDA analysis of diabetes

This practical will follow the process outlined in more detail in:

Evans, Leckie, Subramanian, Bell and Merlo (2024) A tutorial for conducting intersectional multilevel analysis of individual heterogeneity and discriminatory accuracy (MAIHDA). *SSM – Population Health*, 26, 101664, <https://doi.org/10.1016/j.ssmph.2024.101664>

In this exercise sheet, we will go through the key steps of running a MAIHDA analysis, and understanding the outputs produced. However, we would encourage you to read the full paper, to understand a bit more depth in terms of the theoretical and methodological underpinnings of the analysis that we are doing.

The data used here are simulated, but are designed to produce realistic features of equivalent data from the United States of America.

The outcome of interest is HbA1c. This is a biomarker commonly used as an indicator of blood glucose control and diabetes. Generally, values of greater than 48 mmol/mol are indicative of a “diabetic range.

We additionally have “intersectional” identity characteristics:

* Sex (Male and Female),
* Ethnicity (White, Black, Hispanic),
* Education (Less than High School, Completed High School, Some college no degree, College degree or more)
* Income (Low, Low-middle, High-middle, and High income)
* Age (18-29, 30-44, 45-59, 60+)

The data can be found on that paper’s website, as well as in the google drive you have been provided with.

**Initial software setup, and data loading**

We would recommend doing this analysis through RStudio. A sensible first step would be to create a project pointing at a file that contains the dataset “TutorialData.dta”. This will ensure you are starting R with a clean slate without any additional objects in the working environment. Alternatively, you can change the working directory to point at that file manually.

We will first install and load the packages that we will need to run the analysis in R (you may have already installed some of these from this morning’s exercise.

install.packages("directlabels", "haven", "tidyverse", "ggeffects", "lme4", "merTools", "labelled", "sjPlot", "Metrics")

library(haven)

library(tidyverse)

library(ggeffects)

library(lme4)

library(merTools)

library(labelled)

library(sjPlot)

library(Metrics)

Next, we will load the data. As it is currently in .dta (stata) format, we will use read\_dta in the haven package to do so:

tut <- read\_dta("TutorialData.dta")

Because the haven package loads the data in a particular way, we will need to reattach factor labels from the original dataset. This can be done using the labelled package:

tut <- unlabelled(tut)

**Generating the stratum ID**

The strata identifiers (sex, ethnicity, education, income, and age) have been chosen based on theory. We have done so attempting to balance (a) wanting a good amount of nuance in understanding of the different categories, and (b) wanting the sample size in each strata to be so small that it is difficult to analyse.

A simple way to produce the stratum ID variable is to use numerical codes that will be unique to each strata. We do this here by creating a set of 5-digit codes, where each digit corresponds to a different variable.

tut$stratum <- 10000\*tut$sex + 1000\*tut$race + 100\*tut$education + 10\*tut$income + 1\*tut$age

This will produce a variable, stratum, where the first digit corresponds to the sex of the individual, the second digit corresponds to the strata ethnicity, etc. Since this is a categorical variable, we want to turn this into a factor variable.

We may then want to analyse the stratum data in more detail. We could tabulate it, as well as create a new variable that records the number of individuals in each stratum.

table(tut$stratum)

tut <- tut %>%

group\_by(stratum) %>%

mutate(strataN = n())

**Descriptive statistics**

It would be sensible to further consider the descriptive statistics of each of our variables, to ensure they are as we would expect. For the categorical stratum variables, these would be tabulations:

table(tut$sex)

table(tut$race)

table(tut$education)

table(tut$income)

table(tut$age)

Whereas for the outcome variable, we would look at continuous measures of mean and spread

summary(tut$HbA1c)

sd(tut$HbA1c)

We could additionally make a plot of our outcome variable as well.

ggplot(tut, aes(x=HbA1c)) +

geom\_histogram()

**Running our two MAIHDA models**

We can now run our two MAIHDA models – first a null model, with just an intercept in the fixed part of the model and random intercepts on strata, and then a main effects model, with main effects of the strata-defining variables included. Starting with the first model:

model1A <- lmer(HbA1c ~ (1|stratum), data=tut)

We are using the lmer package, and then defining the outcome variable (HbA1c). The section in brackets highlights the random effects – the level is defined by the variable “stratum”, and the 1 indicates the intercept as the only think allowed to vary at the stratum level (ie this is not a random slopes model. Finally we stata what dataframe the variables can be found in.

It will be useful for later to make predictions based on this model, for which we can use the predict function.

tut$m1Am <- predict(model1A)

Next, let’s run the main effects model

model1B <- lmer(HbA1c ~ sex + race + education + income + age + (1|stratum), data=tut)

Again, it will be helpful to make predictions based on this model. For these predictions, we additionally want to save predicted confidence intervals of those predictions, and we can do so using the predictInterval package:

m1Bm <- predictInterval(model1B, level=0.95, include.resid.var=FALSE)

This will create a separate dataframe, with predictions that we will later want to merge back into the main dataframe tut. As such, we will want to create an identifier for this new dataframe, as well as for the main dataframe, so they can be easily merged.

m1Bm <- mutate(m1Bm, id=row\_number())

tut$id <- seq.int(nrow(tut))

We can then merge the two dataframes together, and then rename some of the variables in the prediction dataframe so it is clear what they refer to,

tut2 <- merge(tut, m1Bm, by="id")

tut2 <- tut2 %>%

rename(

m1Bmfit=fit,

m1Bmupr= upr,

m1Bmlwr=lwr

)

Finally, given that many of the differences we are looking at are between strata, it makes sense to reduce the data down to the strata level, so that we can more efficiently plot strata differences.

stratum\_level <- aggregate(x=tut2[c("HbA1c”)],

by=tut2[c("sex", "race", "education", "income", "age",

"stratum", "strataN", "m1Am", "m1Bmfit", "m1Bmupr", "m1Bmlwr")], FUN=mean)

This will take the mean of of HbA1c, and use the remaining variables as the grouping variables (these variables all vary at the strata level, so this is effectively going to group by strata).

**Model analysis**

We can now analyse our model results. A convenient way to produce a regression results table is the tab\_model function (in sjPlot)

tab\_model(model1A, model1B, p.style="stars")

Looking at model 1A, we can see that the precision-weighted stratum mean of HbA1c is 40.79. Note that this will differ slightly from the overall sample mean, since it is the stratum mean weighted by stratum size, rather than the sample mean (weighted evenly across individuals). We can also see the stratum- and individual-level variances, which are 9.33 and 90.26 respectively. On the basis of this, we can calculate the VPC

VPC = 9.33 / (9.33 + 90.26)

That is, approximately 9.4% of the variance in HbA1c can be found at the stratum level.

Model 1B introduces the additive main effects of the stratum-level variables. We can see that individuals with the highest HbA1c levels are generally Male, Black, Low Educated, relatively low income, and old. It’s worth noting the differences in effect size of these variables, though, with a very pronounced effect of being 60+ and black that are much larger than the other variables. Note, as well, that it is not always appropriate to consider these inequalities through an intersectional lens. Sometimes these differences will be produced through biological effects, whereas at other times they may be produced through the effects of social injustice and discrimination. The model cannot tell us how these effects were produced.

We can see that, comparing model 1A and 1B, the stratum-level variance reduces substantially, from 9.33 to 0.80. This is a reduction of 91.4% - this statistic is the proportional change in variance, and tells us that ~90% of the inequalities between strata is driven by additive, rather than multiplicative effects. This might seem like a lot, but it about average for what is seen in MAIHDA studies in the literature, and the remaining 8.6% may well be very important in terms of shaping intersectional inequalities.

**Looking at specific strata**

The above model outputs tell us about intersectionality *generally* – that is: how big are inequalities generally, and to what extent are those inequalities additive or multiplicative. It would now be sensible to look at specific strata- that is, what strata would we expect to have the highest levels of HbA1c, and which strata the lowest.

We can first make a plot of our strata predictions, using the stratum\_level dataframe previously created. First, we create a rank variable for our strata estimates:

stratum\_level <- stratum\_level %>%

mutate(rank=rank(m1Bmfit))

We can then plot a caterpillar plot of our main effects model’s predictions against this rank. We have previously produced these predictions and their 95% confidence intervals, so this plot is fairly easy to produce:

ggplot(stratum\_level, aes(y=m1Bmfit, x=rank)) +

geom\_point() +

geom\_pointrange(aes(ymin=m1Bmlwr, ymax=m1Bmupr)) +

ylab("Predicted HbA1c, Model 1B") +

xlab("Stratum Rank") +

theme\_bw()

We might want to highlight the top and bottom strata here:

stratum\_level <- stratum\_level[order(stratum\_level$rank),]

head(stratum\_level)

tail(stratum\_level)

It can be seen that the Male, white, highly educated, high income, young group has the lowest predicted level of HbA1c, whilst the highest level of HbA1c is found in the low-income, low educated, black male group.

These results are based on both the main effects and the multiplicative differences from the main effects. However we might additionally want to consider just the multiplicative differences – that is which strata are particularly advantaged / disadvantaged in comparison to what would be expected based on just their main effects. This can be done by first making predictions of those multiplicative effects for each strata – that is the stratum random effects in model 1B:

m1Bu <- REsim(model1B)

We can then plot these using the plotREsim command

p <-plotREsim(m1Bu) +

xlab("Stratum Rank") +

ylab("Predicted stratum Random Effect in HbA1c (mmol/mol)") +

ggtitle("") +

theme(

strip.background=element\_blank(),

strip.text.x = element\_blank()

)

p

It can be seen from this “caterpillar plot” there are a few strata which are particularly high and low, although most overlap the 0 line suggesting they are not significantly different from the mean expected by their additive effect. We could next plot only those strata which have statistically significant multiplicative effects.

To do this, we are using the plot object “p” above, which contains the strata estimates as well as information on which strata confidence intervals cross the zero line. We can extract that data, and then filter based on that significance level.

m1Bucut <- p[["data"]]

m1Bucut <- m1Bucut %>%

filter(sig=="TRUE") %>%

mutate(xvar=as.factor(xvar))

We can then reproduce our plot using the ggplot command directly

ggplot(m1Bucut, aes(x=xvar, y=median, label=groupID)) +

geom\_point(size=3) +

geom\_pointrange(aes(ymin=ymin, ymax=ymax)) +

geom\_hline(yintercept=0, color="red", linewidth=1) +

geom\_text(hjust=0, vjust=5) +

xlab("Stratum Rank") +

ylab("Predicted stratum Random Effect in HbA1c")

It can be seen that strata 21223 (Male, White, High-School Education, Low-middle income, age 45-59) has the biggest positive multiplicative effect, and strata 22114 (Male, Black, Low education, low income, over 60) has the biggest negative multiplicative effect. However, this is not to say that those strata are particularly advantaged or disadvantaged, just that they are more (dis)advantaged than we might expect given their combination of additive identity characteristics. As such, these results should only be interpreted in the context of the main effects in model 1B and the overall predictions produced earlier.