Solutions to Lab 2

Multivariate Statistics with R

packages needed: `lme4`, `arm`

Complete pooling and no pooling models

Multilevel modelling can be thought of as a method for compromising between two extremes: *complete pooling* and *no pooling* (Gelman & Hill, 2007, Chapter 12). *Pooling* refers to how different aspects of the data are combined together to estimate a particular parameter of interest (such as how vocabulary scores increase with grade level). To work with an example, read in the vocabulary data that were introduced last week and transform it to long format.

```
### ANSWER ###

vocab <- read.table("vocabulary.txt", header = T)
vocab.long <- reshape(vocab, varying = names(vocab)[3:6], timevar = "Grade", direction = "long")
```

We implement a transformation of `Grade` to shift the intercept to school year 8, which was the first year of vocabulary measurement. Practically, we subtract 8 from all values of the `Grade` variable. You will learn more about such transformations in lecture 4.

```
# transformation of grade: grade 8 = 0
vocab.long$Grade <- vocab.long$Grade - min(vocab.long$Grade)
```

Complete pooling

The group indicator and/or categorical predictor (*e.g.*, `Subject`) is excluded from the model. All the data are treated as independent, ignoring betweensubject variation. Predicting vocabulary score from `Grade` and `Grade`^{2}, we have,[1]

```
vocab.pool <- lm(Vocab ~ 1 + Grade + I(Grade^2), data = vocab.long)
summary(vocab.pool)$coefficients[ , 1:2]
```

```
##                Estimate Std. Error
## (Intercept)  1.1869219  0.2460314
## Grade        1.4359062  0.3951044
## I(Grade^2)  -0.2303125  0.1262115
```

```
# summary(vocab.pool)$coefficients extracts the estimates for regression coefficients
# [ , 1:2] restricts the information to the estimates and their standard errors,
# dropping the t and p values
```

[1] Recall in the formula syntax, `I()` functions to keep arithmetic operators that we want to perform on the inputs (here we are squaring Grade) from interfering with the `+` and `*` as used in the formula.

No pooling

Separate models are fit within each group and/or within each level of the categorical predictor. Thus, a separate model is fit for each subject but no information is shared when making those estimates.

```
vocab.nopool <- lm(Vocab ~ 1 + Grade + I(Grade^2) + as.factor(Subject), data = vocab.long)
head(summary(vocab.nopool)$coefficients[ , 1:2])
```

```
##                         Estimate Std. Error
## (Intercept)            1.5997344 0.46598092
## Grade                  1.4359063 0.17848139
## I(Grade^2)            -0.2303125 0.05701379
## as.factor(Subject)2 -0.6375000 0.64503738
## as.factor(Subject)3 -1.8475000 0.64503738
## as.factor(Subject)4  0.8875000 0.64503738
```

R automatically dummy codes factors, so here `lm` has created 63 dummy-coded variables that, when fit, show how much above or below (intercept) subjects 2-64 are compared to subject 1, who acts as the reference group. While this gives us reasonable estimates, the model is not efficient because each of these predictors is treated as if they were providing completely independent information about the outcome.

Multilevel models as partial pooling models

Partial pooling joins the measurements of each subject together in determining how much the subjects differ from each other. This is what multilevel models (AKA linear mixed-effects models) do.

The `lmer()` function

We use the `lmer()` function for fitting linear mixed-effects model in R. It comes from the `lme4` package. When working with `lmer()`, it may be useful to also load the `arm` package[2], as it contains a few convenience functions (*e.g.*, `display`, `se.fixef`, `se.ranef`).

```
install.packages("arm")
install.packages("lme4")
library(arm)
library(lme4)
```

Here, we model the vocabulary data by specifying model 2 from the lecture – a quadratic model with random intercept. In particular, we specify a separate intercept for each subject by adding `(1 | Subject)` to the model formula.

```
vocab.M1 <- lmer(Vocab ~ 1 + Grade + I(Grade^2) + (1 | Subject), data = vocab.long)
```

Extracting information from the fitted model

The estimates of the *fixed-effects* parameters (population parameters) can be extracted from the fitted model object in several ways. First, they can be read off (with their standard errors) from the output provided by the generic summary function. Second, they can be read off from the output provided by the display function from the arm package – look for the `coef.est` and `coef.se` columns. Third, the estimates of the fixed-effects parameters can be extracted directly using the `fixef()` function from the `lme4` package, and their standard errors using the `se.fixef()` function from the `arm` package. In a similar manner, information

[2] `arm` is the companion `R` package to Gelman and Hill (2007), which is referred to as the "ARM book".

on the random effects is provided by passing the fitted model object to the `summary()` (look for "Variance" and "Std.Dev." under "Random effects") or `display()` (look for "Std.Dev." under "Error terms"). The actual random effects can be extracted with the `ranef()` function from the `lme4` package and their standard error with the `se.ranef()` function from the `arm` package.

```
summary(vocab.M1) # lme4 package
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Vocab ~ 1 + Grade + I(Grade^2) + (1 | Subject)
##    Data: vocab.long
##
## REML criterion at convergence: 865.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.1560 -0.6210  0.0127  0.6120  2.8883
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Subject  (Intercept) 3.2586   1.8052
##  Residual             0.8321   0.9122
## Number of obs: 256, groups:  Subject, 64
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.18692    0.25153   4.719
## Grade        1.43591    0.17848   8.045
## I(Grade^2)  -0.23031    0.05701  -4.040
##
## Correlation of Fixed Effects:
##            (Intr) Grade
## Grade      -0.304
## I(Grade^2)  0.227 -0.958
```

```
display(vocab.M1) # arm package
```

```
## lmer(formula = Vocab ~ 1 + Grade + I(Grade^2) + (1 | Subject),
##     data = vocab.long)
##             coef.est coef.se
## (Intercept)  1.19     0.25
## Grade        1.44     0.18
## I(Grade^2)  -0.23     0.06
##
## Error terms:
##  Groups   Name        Std.Dev.
##  Subject  (Intercept) 1.81
##  Residual             0.91
## ---
## number of obs: 256, groups: Subject, 64
## AIC = 875.6, DIC = 847.4
## deviance = 856.5
```

```
fixef(vocab.M1) # lme4 package
```

```
## (Intercept)       Grade  I(Grade^2)
##   1.1869219   1.4359063  -0.2303125
```

```
se.fixef(vocab.M1) # arm package
```

```
## (Intercept)       Grade  I(Grade^2)
##  0.25153224  0.17848139  0.05701379
```

```
ranef(vocab.M1) # lme4 package
```

```
## $Subject
##    (Intercept)
## 1   0.38803948
## 2  -0.21120393
## 3  -1.34859142
## 4   1.22228031
## 5  -3.67506582
## 6  -1.76688681
## 7  -1.10419410
## 8  -1.75983689
## 9  -1.75983689
## 10  0.99198284
##  [ reached getOption("max.print") -- omitted 54 rows ]
```

```
se.ranef(vocab.M1) # arm package
```

```
## $Subject
##    (Intercept)
## 1    0.4422129
## 2    0.4422129
## 3    0.4422129
## 4    0.4422129
## 5    0.4422129
## 6    0.4422129
## 7    0.4422129
## 8    0.4422129
## 9    0.4422129
## 10   0.4422129
##  [ reached getOption("max.print") -- omitted 54 rows ]
```

Question 1: How do the estimates, `ranef(vocab.M1)`, and the standard errors, `se.ranef(vocab.M1)`, for subject intercepts from the partial pooling (`lme`) model compare with those from the no-pooling model, `summary(vocab.nopool)$coefficients[ , 1:2]`?

```
### ANSWER ###

# Standard errors of subject intercepts are
# smaller in the partial pooling (lme) model.
se.ranef(vocab.M1)
```

```
## $Subject
##    (Intercept)
## 1    0.4422129
## 2    0.4422129
## 3    0.4422129
## 4    0.4422129
## 5    0.4422129
```

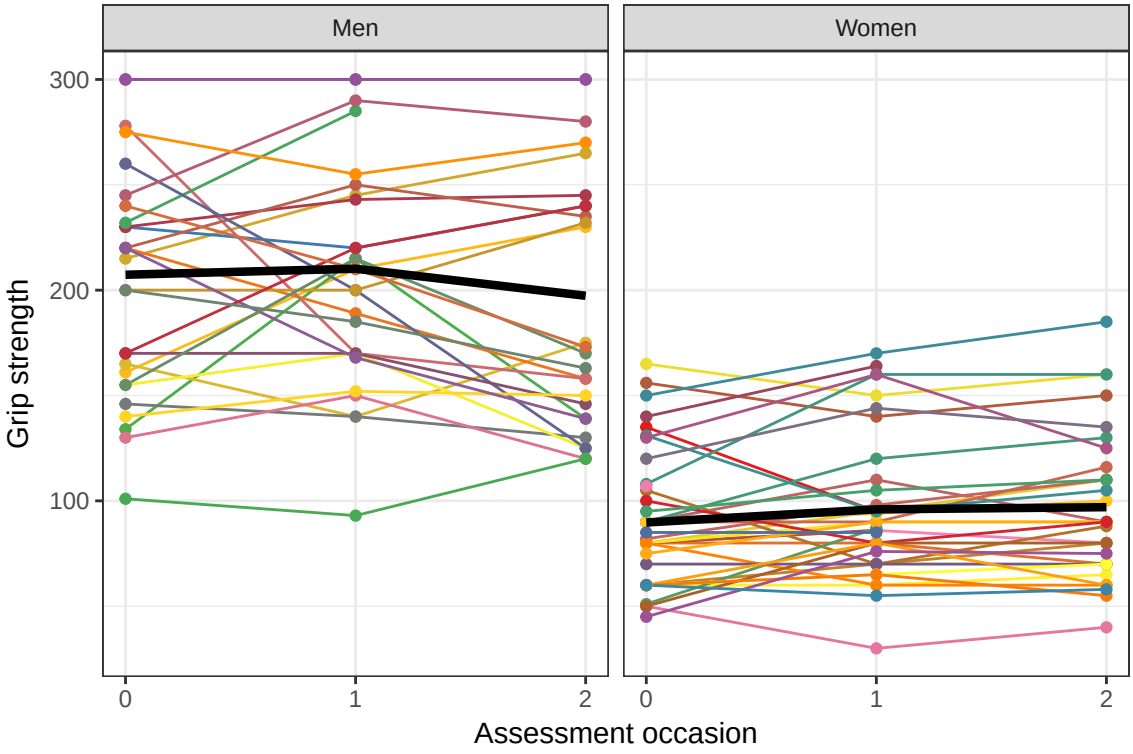
```
## 6    0.4422129
## 7    0.4422129
## 8    0.4422129
## 9    0.4422129
## 10   0.4422129
##  [ reached getOption("max.print") -- omitted 54 rows ]
```

```
summary(vocab.nopool)$coefficients[ , 1:2]
```

```
##                          Estimate Std. Error
## (Intercept)             1.5997344 0.46598092
## Grade                   1.4359063 0.17848139
## I(Grade^2)             -0.2303125 0.05701379
## as.factor(Subject)2    -0.6375000 0.64503738
## as.factor(Subject)3    -1.8475000 0.64503738
##  [ reached getOption("max.print") -- omitted 61 rows ]
```

Arthritis example revisited

Let's revisit the arthritis example from the lecture. In this study, 67 patients (29 men, 38 women) were treated for rheumatoid arthritis (Patel, 1991). To examine the effectiveness of the treatment, patients' *grip strength* was measured on 3 occasions. The plot below (see also lecture 2, slides 20-21) shows the individual functions, separately for male and female patients. The mean vector, the average performance for each group (bold black line), tracks "typical performance". Note that the variability of the patients' performance during the three occasions of the experiment, which is somewhat greater for the men than the women, is a salient feature of the data.



The data file (arthritis.txt) is available on LEARN. Download the file, read the file into `R` and store it in an object named `arth`. Then convert `arth` into long format, storing it in `arth.long`.

```
### ANSWER ###

arth <- read.table("arthritis.txt", header = TRUE)

arth.long <- reshape(
  arth, idvar = c("Subject"), varying = names(arth)[3:5],
  timevar = "time", direction = "long", times = 0:2, v.names = 'y')
```

In the lecture, calculations focused on the 29 men. Let's create a subset of `art.long` called `men` with only data from, well, men.

```
### ANSWER ###

men <- arth.long[arth.long$Gender == "Men", ]

summary(men)
```

```
##    Gender      Subject        time         y
##  Men  :87   Min.   : 1   Min.   :0   Min.   : 93.0
##  Women: 0   1st Qu.: 8   1st Qu.:0   1st Qu.:157.2
##             Median :15   Median :1   Median :210.0
##             Mean   :15   Mean   :1   Mean   :205.1
##             3rd Qu.:22   3rd Qu.:2   3rd Qu.:245.0
##             Max.   :29   Max.   :2   Max.   :300.0
##                                      NA's   :3
```

`summary(men)` tells us that there are 3 missing data points, coded as `NA`, in the outcome variable `y`. There are only three measurement points. A linear model, where we fit a regression line to these three data points, is about the most complex form the data can handle. In the lecture, we specified a linear mixed-effects model with two fixed effects (intercept, slope) and 2 random effects (intercept, slope). The model is a *random coefficient model* because it represents the special case that each predictor has both a fixed and random term.

We also need to tell `lmer()` what to do with the 3 missing values in the data. Here, we set the argument `na.action` to `na.exclude`. This omits the `NA`s for the analysis but predictions are padded to the correct length by inserting `NA`s for the omitted cases.

```
m1 <- lmer(y ~ 1 + time + (1 + time | Subject), data = men, na.action = na.exclude)
```

`m1` is an object of class `"mer"` (mixed-effects representation). There are many extractor functions that can be applied to such objects.

Question 2: Retrieve the two regression coefficients that represent the population parameters (fixed effects). Write the regression equation for the population and describe the modelled relationship.

```
### ANSWER ###

print(fixef(m1))
```

```
## (Intercept)        time
##  209.434275   -3.350259
```

```r
print(summary(m1))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ 1 + time + (1 + time | Subject)
##    Data: men
##
## REML criterion at convergence: 847.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.56090 -0.43564  0.07258  0.36889  2.40121
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  Subject  (Intercept) 2470.2   49.70
##           time         419.7   20.49    -0.12
##  Residual              430.7   20.75
## Number of obs: 84, groups:  Subject, 29
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  209.434      9.883  21.191
## time          -3.350      4.800  -0.698
##
## Correlation of Fixed Effects:
##      (Intr)
## time -0.251
```

The intercept is 209.4. The slope is negative at -3.35:

$$y = 209.4 - 3.37x$$

That is, male patients lost grip strength at the rate of -3.37 units per occasion (measurement point).

R: `coef()` = `fixef()` + `ranef()` and/or `ranef()` = `coef()` - `fixef()`

It is important to understand the relationship between *fixed effects*, *random effects*, and *individual regression coefficients*. In `R`, the population parameters (or fixed effects) are obtained by using the function `fixef()`. `ranef()` extracts the random effects from a fitted model object, and `coef()` the individual regression coefficients. The **individual regression coefficients** (intercept and slope) for a given subject (see below subject 12) are calculated as the **sum of the fixed effects and random effects**.

$$\begin{aligned}\hat{\beta}_{12} &= \hat{\beta} + \hat{b}_{12} \\ &= \begin{pmatrix} 209.434 \\ -3.35 \end{pmatrix} + \begin{pmatrix} 8.997 \\ -0.46 \end{pmatrix} \\ &= \begin{pmatrix} 218.43 \\ -3.81 \end{pmatrix}\end{aligned}$$

We can check this with `R` code:

```
# individual regression coefficients for subject 12
(coef_12 <- coef(m1)$Subject[12, ])
```

```
##    (Intercept)      time
## 12    218.4314 -3.806984
```

```
# fixed effects for m1
(fix <- fixef(m1))
```

```
## (Intercept)        time
##  209.434275   -3.350259
```

```
# random effects for subject 12
(ref_12 <- ranef(m1)$Subject[12, ])
```

```
##    (Intercept)       time
## 12    8.997113 -0.4567241
```

```
coef_12 == fix + ref_12
```

```
##    (Intercept) time
## 12        TRUE TRUE
```

Question 3: Apply `coef()`, `fixef()`, and `ranef()` to our fitted model object `m1`. For subject 1, retrieve the individual regression coefficients and write the regression equation. How well did this subject respond to the medical treatment?

```
### ANSWER ###

# intercept for subject 1:
coef(m1)$Subject[1,1]
```

```
## [1] 180.8095
```

```
# slope for subject 1:
coef(m1)$Subject[1,2]
```

```
## [1] 20.7875
```

$$\begin{aligned} y_{1j} &= (\beta_{10} + b_{10}) + (\beta_{21} + b_{11})x_j + \epsilon_{1j} \\ &= (209.434 + -28.625) + (-3.35 + 24.138)x_j + \epsilon_{1j} \\ &= 180.809 + 20.787x_j + \epsilon_{1j} \end{aligned}$$

Model parameters: variance-covariance matrices

For `lmer` model `m1`, a total of six parameters need to be estimated (see slide 23, lecture 2):

1. the population parameters (fixed effects), intercept and slope (β)
2. the covariance matrix of the individual coefficients (Φ)
3. the covariance matrix of the residuals (λ)

In `R`, the variance-covariance matrix for the individual coefficients (φ) is estimated by `lmer()`. It can be extracted from the fitted model object with the `VarCorr()` function. In the arthritis example, we modelled a linear relationship with two random effects per subject – an intercept and a slope. Therefore, the variance-covariance matrix for the individual coefficients has three unique elements (see slides 23 and 30, lecture 2).

The diagonal elements are the variances of the individual coefficients (β_{ij}). Are the individual differences large or small? The off-diagonal element describes the extent to which pairs of coefficients covary or correlate. For two parameters you only get one such pairing. It describes the relationship between individual intercepts and slopes.

Question 4: Use the VarCorr function to retrieve the estimates for the covariance matrix of the individual coefficients. You should be able to reproduce the numbers from the lecture slides. Conceptually, what do the diagonal elements and the off-diagonal element mean?

Tip:

1. By default, VarCorr provides standard deviations rather than variance. The help file tells you how to change it (it's well hidden); alternatively, remind yourself of the difference between variance and standard deviation.
2. VarCorr provides the correlation while the slide(s) give you the covariance, but they are closely related (look it up online)

Refresher:

$Var(x) = std.dev(x)^2$

$r = \frac{cov(x,y)}{std.dev(x) \times std.dev(y)} \implies cov(x,y) = r_{x,y} \times std.dev(x) \times std.dev(y)$

```
### ANSWER ###

print(VarCorr(m1))
```

```
##  Groups   Name        Std.Dev. Corr
##  Subject  (Intercept) 49.701
##           time        20.487   -0.123
##  Residual             20.752
```

Variance-covariance matrix Φ:

- diagonal elements:

$$\begin{aligned} Var(intercept) &= std.dev(intercept)^2 \\ &= 49.701^2 \\ &= 2470.189 \\ Var(slope) &= std.dev(slope)^2 \\ &= 20.487^2 \\ &= 419.7172 \end{aligned}$$

- off-diagonal element:

$$\begin{aligned} cov(intercept, slope) &= r_{(intercept,slope)} \times std.dev(intercept) \times std.dev(slope) \\ &= -0.123 \times 49.701 \times 20.487 \\ &= -125.2416 \end{aligned}$$

(These are the values from lecture slide 23.)

`VarCorr()` also provides the variances directly, but this behaviour is hidden in the help document:

Details

The print method for `VarCorr.merMod` objects has optional arguments [...] `comp`: the latter is a character vector with any combination of "Variance" and "Std.Dev.", to specify whether variances, standard deviations, or both should be printed.

```
print(VarCorr(m1), comp = c("Std.Dev", "Variance"))
```

```
##  Groups   Name        Std.Dev. Variance Corr
##  Subject  (Intercept) 49.701   2470.16
##           time        20.487    419.73  -0.123
##  Residual             20.752    430.66
```

Question 5: What's the variance of the residuals in model m1?

```
### ANSWER ###
print(summary(m1))
```

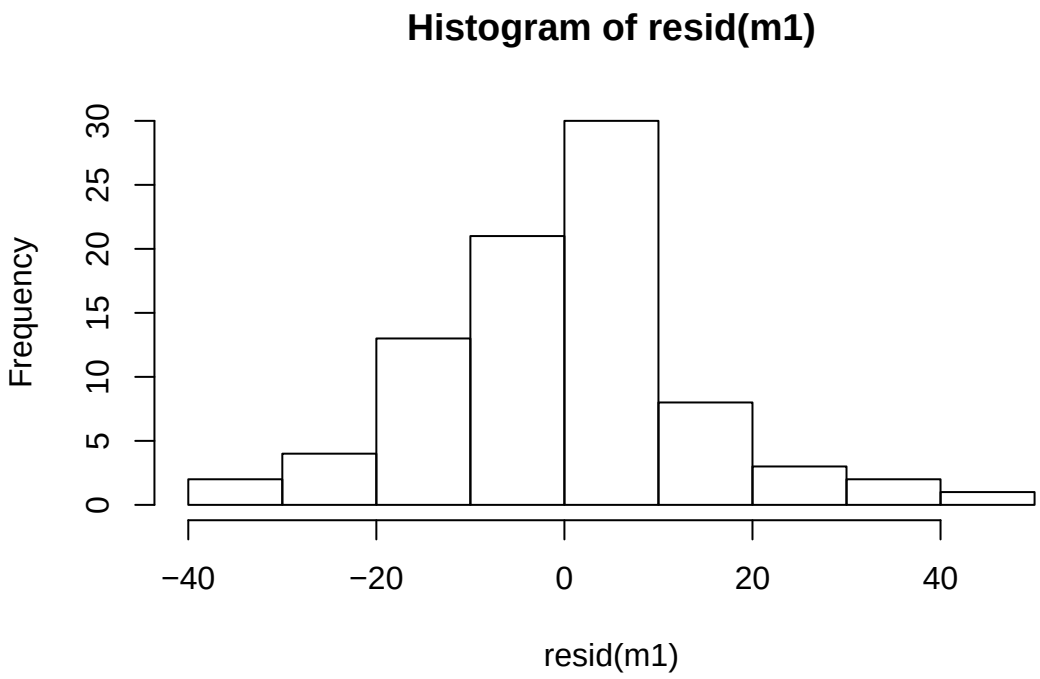
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ 1 + time + (1 + time | Subject)
##    Data: men
##
## REML criterion at convergence: 847.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.56090 -0.43564  0.07258  0.36889  2.40121
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  Subject  (Intercept) 2470.2   49.70
##           time         419.7   20.49    -0.12
##  Residual              430.7   20.75
## Number of obs: 84, groups:  Subject, 29
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  209.434      9.883  21.191
## time          -3.350      4.800  -0.698
##
## Correlation of Fixed Effects:
##      (Intr)
## time -0.251
```

```
# - look under Residual / Variance

# OR
attr(VarCorr(m1), "sc")^2
```

```
## [1] 430.6631
```

Question 6: Show graphically that the residuals follow a normal distribution, centred at a mean of zero. Model residuals can be extracted with the generic function `resid()`.

```
### ANSWER ###

hist(resid(m1))
```



Exercises

Using the vocabulary data, the goal of the exercises is to practise how to translate model equations into `R` implementations using `lmer()`. We start by specifying the simplest possible model, a model with a fixed effect for the intercept, and a random effect for the (by-subject) intercept.

model equation:

$y_{ij} = (\beta_0 + b_{i0}) + \epsilon_{ij}$

implementation in R:

```
fm2 <- lmer(Vocab ~ 1 + (1 | Subject), data = vocab.long)
```

(`fm1` stands for "fitted model 1")

Note that you always have to add at least one random effect; in our example `(1 | subject)` fits a separate intercept for each subject. Your task is to extend this model step by step by first adding more fixed effects (Exercises 1 and 2) and random effects (Exercises 3-5). The final model is the full model introduced in lecture 2. Exercise 1 Specify `fm2` with fixed effects for both intercept and slope.

Exercise 1: Specify `fm2` with fixed effects for both intercept and slope.

$y_{ij} = (\beta_0 + b_{i0}) + \beta_1 x_j + \epsilon_{ij}$

```
### ANSWER ###

fm2 <- lmer(Vocab ~ 1 + Grade + (1 | Subject), data = vocab.long)
```

Exercise 2: Specify `fm3` with fixed effects for intercept, slope, and quadratic term.

$y_{ij} = (\beta_0 + b_{i0}) + \beta_1 x_j + \beta_2 x_j^2 + \epsilon_{ij}$

Tip: You have to include a power term, grade squared. You have to use the `I()` operator, which forces computation of its argument before evaluation of the formula.

```
### ANSWER ###

fm3 <- lmer(Vocab ~ 1 + Grade + I(Grade^2) + (1 | Subject), data = vocab.long)
```

Exercise 3: Specify fm4 with a random effect for slope (only):

$y_{ij} = \beta_0 + (\beta_1 + b_{i1}) x_j + \beta_2 x_j^2 + \epsilon_{ij}$

```
### ANSWER ###

# 0 to suppress the intercept!
fm4 <- lmer(Vocab ~ 1 + Grade + I(Grade^2) + (0 + Grade | Subject), data = vocab.long)
```

Exercise 4: Specify fm5 with random effects for intercept and slope.

```
### ANSWER ###

fm5 <- lmer(Vocab ~ 1 + Grade + I(Grade^2) + (1 + Grade | Subject), data = vocab.long)
```

Exercise 5: Specify fm6 with random effects for intercept, slope, and quadratic term.

$y_{ij} = (\beta_0 + b_{i0}) + (\beta_1 x_j + b_{i1}) + (\beta_2 + b_{i2}) x_j^2 + \epsilon_{ij}$

```
### ANSWER ###

fm6 <- lmer(Vocab ~ 1 + Grade + I(Grade^2) + (1 + Grade + I(Grade^2) | Subject),
            data = vocab.long)
```

References

Gelman, A. & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models.* New York: Cambridge University Press.

Patel, H. I. (1991). Analysis of incomplete data from a clinical trial with repeated measurements. *Biometrika, 78*(3), 609-619.