Variance–Covariance Matrices

Daniel Anderson

Week 4

Agenda

- Review Homework 1
- Review the most important parts of Week 3 content
- Discuss Gelman and Hill notation – contrast with Raudenbush and Bryk
- Unstructured VCV Matrices and alternatives

Learning Objectives

- Understand at least the basics of the GH notation and why I view it as preferable
- Gain a deeper understanding of how the residual structure is different in multilevel models
- Understand that there are methods for changing the residual structure, and understand when and why this might be preferable
- Have a basic understanding of implementing alternative methods

Review Homework 1

Review Week 3 content

Data and model

```
library(tidyverse)
library(lme4)

popular <- read_csv(here::here("data", "popularity.csv"))

m <- lmer(popular ~ extrav + (extrav|class), popular,
          control = lmerControl(optimizer = "bobyqa"))
```

`extrav` is a measure of extraversion.

What is this model fitting, in plain English?

Model summary

```
arm::display(m)
```

```
## lmer(formula = popular ~ extrav + (extrav | class), data = popular,
##     control = lmerControl(optimizer = "bobyqa"))
##             coef.est coef.se
## (Intercept) 2.46     0.20
## extrav      0.49     0.03
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  class    (Intercept) 1.73
##           extrav      0.16     -0.97
##  Residual             0.95
## ---
## number of obs: 2000, groups: class, 100
## AIC = 5791.4, DIC = 5762
## deviance = 5770.7
```

Let's walk through

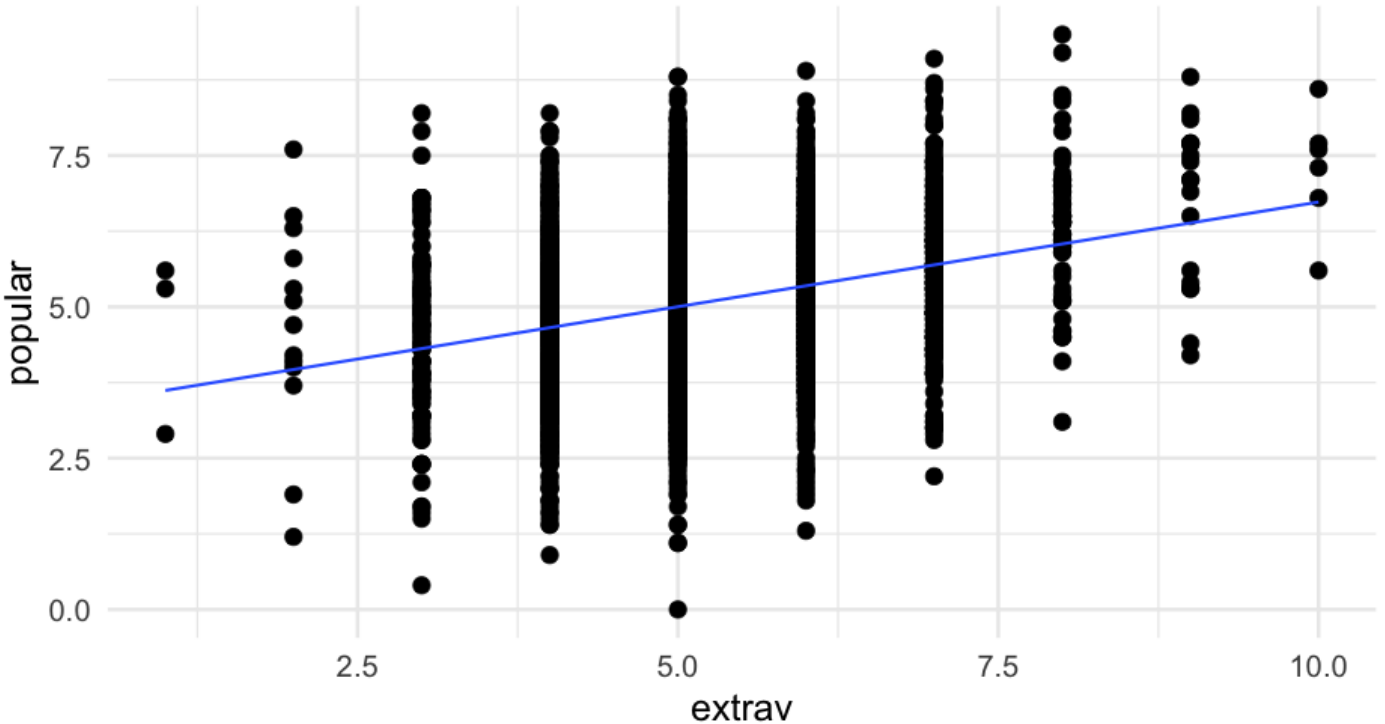
- By way of thinking through it, let's compare to simple linear regression

```
slr <- lm(popular ~ extrav, popular)
arm::display(slr)
```

```
## lm(formula = popular ~ extrav, data = popular)
##             coef.est coef.se
## (Intercept) 3.27     0.12
## extrav      0.35     0.02
## ---
## n = 2000, k = 2
## residual sd = 1.31, R-Squared = 0.10
```

Visually

```
ggplot(popular, aes(extrav, popular)) +
  geom_point() +
  geom_smooth(se = FALSE, method = "lm")
```



Making predictions

$$\widehat{\text{popular}} = 3.27 + 0.35(\text{extrav})$$

Scores of 0 to 5

$$\widehat{\text{popular}} = 3.27 + 0.35 \times 0 = 3.27$$

$$\widehat{\text{popular}} = 3.27 + 0.35 \times 1 = 3.62$$

$$\widehat{\text{popular}} = 3.27 + 0.35 \times 2 = 3.97$$

$$\widehat{\text{popular}} = 3.27 + 0.35 \times 3 = 4.32$$

$$\widehat{\text{popular}} = 3.27 + 0.35 \times 4 = 4.67$$

$$\widehat{\text{popular}} = 3.27 + 0.35 \times 5 = 5.02$$

Now for the mlm

It's more complicated now for a couple of reasons

- Each `class` has their own intercept and slope. Which class is the student in?
- What if we want to make a prediction for someone outside our sample? Let's walk through 4 predictions

Sample preds

```
sample_preds <- popular %>%
  group_by(class) %>%
  slice(1) %>%
  ungroup() %>%
  slice(1:4)

sample_preds
```

```
## # A tibble: 4 x 7
##   pupil class extrav sex    texp  popular popteach
##   <dbl> <dbl>  <dbl> <chr> <dbl>    <dbl>    <dbl>
## 1     1     1      5 girl     24 6.3              6
## 2     1     2      8 girl     14 6.4              6
## 3     1     3      5 boy      13 4.2              4
## 4     1     4      3 girl     20 4.100000         4
```

Coefficients

$$
\begin{aligned}
\widehat{\text{popular}}_i &\sim N\left(\mu, 0.95\right) \\
\mu &= 2.46_{\alpha_{j[i]}} + 0.49_{\beta_{1j[i]}}(\text{extrav}) \\
\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1.73 & -0.97 \\ -0.97 & 0.16 \end{pmatrix}\right), \text{for class j} = 1, \ldots, \text{J}
\end{aligned}
$$

Grab params

Fixed effects

```
f <- fixef(m)
f
```

```
## (Intercept)      extrav
##   2.4610234   0.4928571
```

classroom deviations

```
r <- ranef(m)
r
```

```
## $class
##     (Intercept)       extrav
## 1    0.34095954 -0.027014677
## 2   -1.17798954  0.096974668
## 3   -0.62937002  0.057097182
## 4    1.08529708 -0.099953042
## 5   -0.19489641  0.021601800
## 6   -0.98339230  0.083762531
## 7   -1.00065422  0.064825024
```

Predictions depend on classroom

Fixed effect part

Works just like simple linear regression

```
sample_preds[1, ]
```

```
## # A tibble: 1 x 7
##   pupil class extrav sex    texp popular popteach
##   <dbl> <dbl>  <dbl> <chr> <dbl>   <dbl>    <dbl>
## 1     1     1      5 girl     24     6.3        6
```

$$\widehat{\text{popular}} = 2.46 + 0.49 \times 5 = 4.91$$

```
f[1] + f[2]*5
```

```
## (Intercept)
##    4.925309
```

Random effects

We now have to add in the random effects *for the corresponding classroom*.

```
head(r$class)
```

```
##   (Intercept)      extrav
## 1   0.3409595 -0.02701468
## 2  -1.1779895  0.09697467
## 3  -0.6293700  0.05709718
## 4   1.0852971 -0.09995304
## 5  -0.1948964  0.02160180
## 6  -0.9833923  0.08376253
```

$$\widehat{\text{popular}} = (2.46 + 0.34) + (0.49 + -0.03) \times 5 = 5.10$$

In code

```
class1 <- r$class[1, ]
class1
```

```
##   (Intercept)      extrav
## 1   0.3409595 -0.02701468
```

```
(f[1] + class1[1]) + (f[2] + class1[2])*5
```

```
##   (Intercept)
## 1    5.131195
```

Predictions

```
sample_preds
```

```
## # A tibble: 4 x 7
##   pupil class extrav sex    texp  popular popteach
##   <dbl> <dbl>  <dbl> <chr> <dbl>    <dbl>    <dbl>
## 1     1     1      5 girl     24 6.3              6
## 2     1     2      8 girl     14 6.4              6
## 3     1     3      5 boy      13 4.2              4
## 4     1     4      3 girl     20 4.100000         4
```

```
head(r$class, n = 4)
```

```
##   (Intercept)      extrav
## 1   0.3409595 -0.02701468
## 2  -1.1779895  0.09697467
## 3  -0.6293700  0.05709718
## 4   1.0852971 -0.09995304
```

```
fixef(m)
```

```
## (Intercept)      extrav
##   2.4610234   0.4928571
```

$$\widehat{\text{popular}} = (2.46 + 0.34) + (0.49 + -0.03) \times 5 = 5.10$$

$$\widehat{\text{popular}} = (2.46 + -1.18) + (0.49 + 0.10) \times 8 = 6.00$$

$$\widehat{\text{popular}} = (2.46 + -0.63) + (0.49 + 0.06) \times 5 = 4.58$$

$$\widehat{\text{popular}} = (2.46 + 1.09) + (0.49 + -0.10) \times 3 = 4.72$$

```
predict(m, newdata = sample_preds)
```

```
##        1        2        3        4
## 5.131195 6.001688 4.581425 4.725033
```

What if we want to make a prediction outside of our classrooms?

Fixed effects only

$$\widehat{\text{popular}} = 2.46 + 0.49 \times \text{extraversion}$$

Plotting

We can use the `expand.grid()` function to create different conditions. Let's compare slopes across the first five classrooms

```
conditions <- expand.grid(extrav = 1:10, class = 1:5)
```

```
head(conditions)
```

```
##   extrav class
## 1      1     1
## 2      2     1
## 3      3     1
## 4      4     1
## 5      5     1
## 6      6     1
```

```
tail(conditions)
```

```
##    extrav class
## 45      5     5
## 46      6     5
## 47      7     5
## 48      8     5
## 49      9     5
## 50     10     5
```

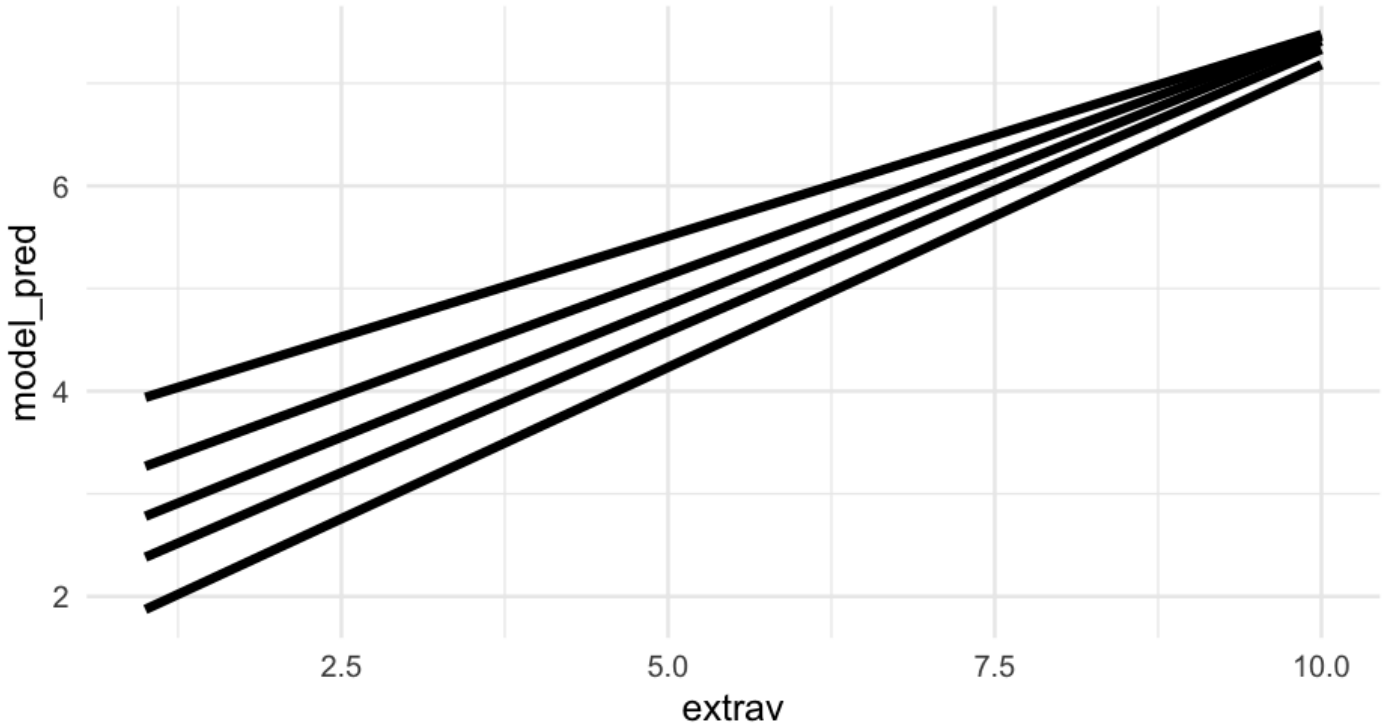
Make predictions

```
conditions %>%
  mutate(model_pred = predict(m, newdata = conditions))
```

```
##    extrav class model_pred
## 1       1     1   3.267825
## 2       2     1   3.733668
## 3       3     1   4.199510
## 4       4     1   4.665353
## 5       5     1   5.131195
## 6       6     1   5.597038
## 7       7     1   6.062880
## 8       8     1   6.528722
## 9       9     1   6.994565
## 10     10     1   7.460407
## 11      1     2   1.872866
## 12      2     2   2.462697
## 13      3     2   3.052529
## 14      4     2   3.642361
## 15      5     2   4.232193
## 16      6     2   4.822025
## 17      7     2   5.411856
## 18      8     2   6.001688
## 19      9     2   6.591520
## 20     10     2   7.181352
## 21      1     3   2.381608
```

Plot

```
conditions %>%
  mutate(model_pred = predict(m, newdata = conditions)) %>%
  ggplot(aes(extrav, model_pred)) +
  geom_line(aes(group = class))
```



One more quick example

Model an interaction

```
m2 <- lmer(popular ~ extrav*sex + (extrav|class), popular,
           control = lmerControl(optimizer = "bobyqa"))
```

Model summary

```
arm::display(m2)
```

```
## lmer(formula = popular ~ extrav * sex + (extrav | class), data = popular
##     control = lmerControl(optimizer = "bobyqa"))
##                coef.est coef.se
## (Intercept)    2.23     0.20
## extrav         0.41     0.03
## sexgirl        0.95     0.16
## extrav:sexgirl 0.06     0.03
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  class    (Intercept) 1.63
##           extrav      0.18     -0.94
##  Residual             0.74
## ---
## number of obs: 2000, groups: class, 100
## AIC = 4890.7, DIC = 4836.6
## deviance = 4855.7
```

Marginal effect

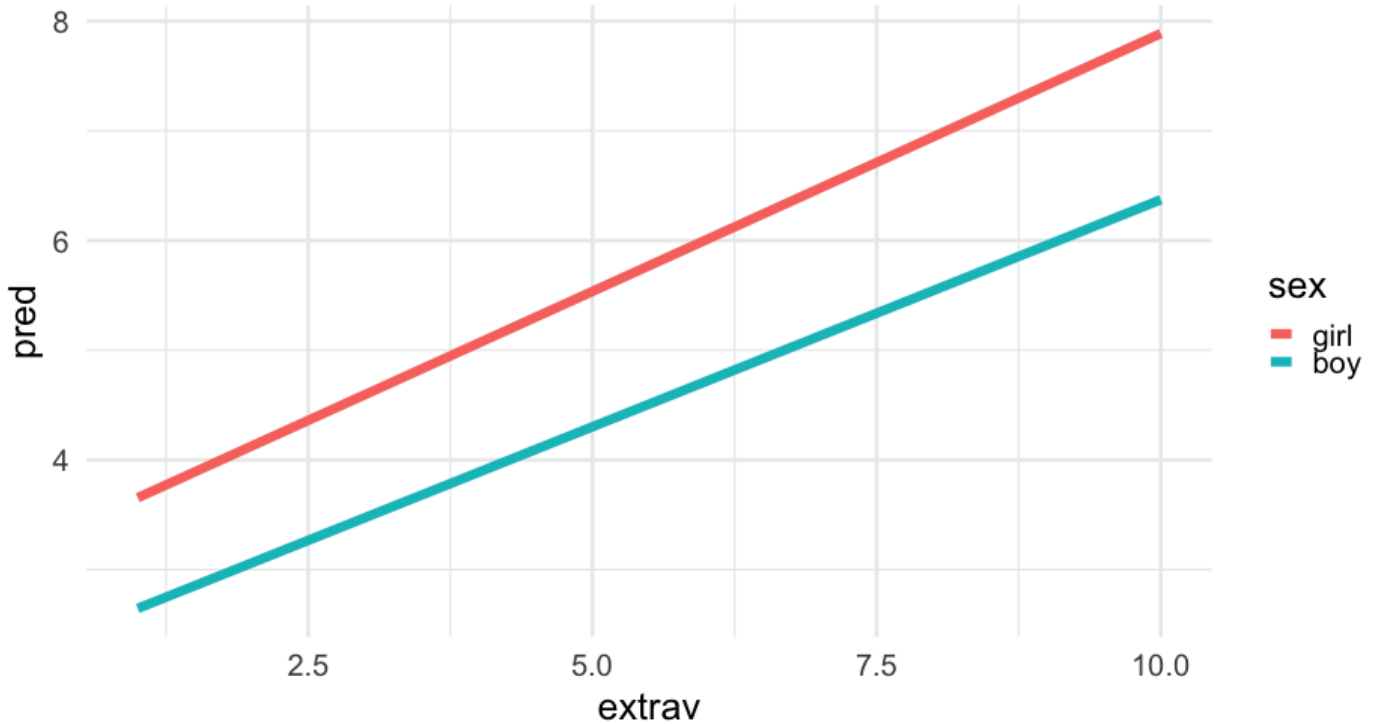
Let's look at the interaction between extraversion and sex

```
conditions2 <- expand.grid(extrav = 1:10,
                           sex = c("girl", "boy"),
                           class = 0) %>%
  mutate(pred = predict(m2,
                        newdata = .,
                        allow.new.levels = TRUE))
conditions2
```

```
##    extrav  sex class     pred
## 1       1 girl     0 3.654686
## 2       2 girl     0 4.124926
## 3       3 girl     0 4.595165
## 4       4 girl     0 5.065404
## 5       5 girl     0 5.535643
## 6       6 girl     0 6.005883
## 7       7 girl     0 6.476122
## 8       8 girl     0 6.946361
## 9       9 girl     0 7.416600
## 10     10 girl     0 7.886840
## 11      1  boy     0 2.645470
## 12      2  boy     0 3.059526
## 13      3  boy     0 3.473583
```

Plot

```
ggplot(conditions2, aes(extrav, pred)) +
  geom_line(aes(color = sex))
```



Questions?

Notation

Introducing the Gelman and Hill notation

Standard regression

Imagine we have a model like this

```
m <- lm(mpg ~ disp + hp + drat, data = mtcars)
```

We would probably display this model like this

$$\mathbf{mpg} = \alpha + \beta_1(\mathbf{disp}) + \beta_2(\mathbf{hp}) + \beta_3(\mathbf{drat}) + \epsilon$$

What we often don't show, is the distributional assumption of the residuals

$$\epsilon \sim N\left(0, \sigma\right)$$

A different view

The model on the previous slide could also be displayed like this

$$\begin{aligned} \hat{y} &= \alpha + \beta_1(\text{disp}) + \beta_2(\text{hp}) + \beta_3(\text{drat}) \\ \text{mpg} &\sim N\left(\hat{y}, \sigma\right) \end{aligned}$$

This makes the distributional assumptions clearer

Each `mpg` value is assumed generated from a normal distribution, with a mean structure according to $\hat{y}$, and an unknown standard deviation, σ.

Simulate

I'm not expecting you to follow along here

If we have a solid understanding of the distributional properties, we can simulate new data from the model

First let's set some population parameters

```
n <- 1000
intercept <- 100
b1 <- 5
b2 <- -3
b3 <- 0.5
sigma <- 4.5
```

Simulate

Next create some variables. The standard deviations relate to the standard errors – more variance in the predictor leads to lower standard errors.

```
set.seed(123)
x1 <- rnorm(n, sd = 1)
x2 <- rnorm(n, sd = 2)
x3 <- rnorm(n, sd = 4)
```

Create y–hat

```
yhat <- intercept + b1*x1 + b2*x2 + b3*x3
```

Generate data & test

```
sim <- rnorm(n, yhat, sigma)
summary(lm(sim ~ x1 + x2 + x3))
```

```
##
## Call:
## lm(formula = sim ~ x1 + x2 + x3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7528  -2.8505   0.0021   3.0387  13.0151
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 99.96508    0.14141  706.92   <2e-16 ***
## x1           4.99415    0.14306   34.91   <2e-16 ***
## x2          -3.01827    0.07027  -42.95   <2e-16 ***
## x3           0.55792    0.03613   15.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.466 on 996 degrees of freedom
## Multiple R-squared:  0.7514,    Adjusted R-squared:  0.7506
## F-statistic:  1003 on 3 and 996 DF,  p-value: < 2.2e-16
```

Generalizing

We can generalize this same basic approach to multilevel models

This is helpful because the error structure is more complicated

Using this approach helps us better understand the distributional assumptions of our model

Simple example

I know we hate the HSB data but bear with me for a minute.

Consider this simple model

```
library(lme4)
library(equatiomatic)
hsb_m0 <- lmer(math ~ ses + (1|sch.id), data = hsb)
```

R&B

In Raudenbush and Bryk notation, the model on the prior slide would look like this

$$\begin{aligned}\mathbf{math}_{ij} &= \beta_{0j} + \beta_{1j}(\mathbf{ses}) + e_{ij} \\ \beta_{0j} &= \gamma_{00} + u_{0j} \\ \beta_{1j} &= \gamma_{10}\end{aligned}$$

Generally, the distributional part is omitted, which in this case is

$$\begin{aligned}E\left(e_{ij}\right) = 0, \mathbf{Var}\left(e_{ij}\right) = \sigma^2 \\ E\left(u_{0j}\right) = 0, \mathbf{Var}\left(u_{0j}\right) = \tau_{00}\end{aligned}$$

Put differently

$$\begin{aligned}e_{ij} &\sim N\left(0, \sigma^2\right) \\ u_{0j} &\sim N\left(0, \tau_{00}\right)\end{aligned}$$

G&H

In Gelman & Hill notation, this same model can be communicated as

$$\begin{aligned} \mathbf{math}_i &\sim N\left(\alpha_{j[i]} + \beta_1(\mathbf{ses}), \sigma^2\right) \\ \alpha_j &\sim N\left(\mu_{\alpha_j}, \sigma^2_{\alpha_j}\right), \text{for sch.id j} = 1, \ldots, \text{J} \end{aligned}$$

This notation communicates the distributional assumptions

We can also still easily see what levels the predictors are at

It does look a little more complex, but it's not hiding anything

If you properly understand the notation, you can simultate data assuming this data generating process (which we'll do later)

Bonus

It works really well to communicate model results

$$\begin{aligned} \widehat{\text{math}}_i &\sim N\left(12.66_{\alpha_{j[i]}} + 2.39_{\beta_1}(\text{ses}), 6.09\right) \\ \alpha_j &\sim N\left(0, 2.18\right), \text{for sch.id j} = 1, \ldots, \text{J} \end{aligned}$$

Extra bonus!

You can use equatiomatic to give you the model formula. The above was generated with `extract_eq(hsb_m0, use_coef = TRUE)`

Quick simulation

We'll go over this in more detail later, but I want to give you the general idea.

First, set some parameters

```
j <- 30 # 30 schools
nj <- 50 # 50 students per school
```

Next, simulate the school distribution

```
# School distribution
a_j <- rnorm(j, 0, 2.18)
```

For each school, simulate nj obs from the level 1 model, adding in the school deviation

There are lots of ways to do this – I'm using a `for()` loop here in an effort to be transparent

```
school_scores <- vector("list", j)
ses <- vector("list", j)

for(i in 1:j) {
  ses[[i]] <- rnorm(nj)
  school_scores[[i]] <- rnorm(nj,
                              12.66 + 2.39*ses[[i]] + a_j[i],
                              6.09)
}
```

Put in a df

```
sim_df <- data.frame(
  scid = rep(1:j, each = nj),
  ses = unlist(ses),
  score = unlist(school_scores)
)
head(sim_df)
```

```
##   scid        ses     score
## 1    1 -0.9529766  9.789073
## 2    1  0.4057701 12.820300
## 3    1 -0.9839385 20.956667
## 4    1 -1.6101230 15.884079
## 5    1 -0.4301688  1.363627
## 6    1 -1.2242106  7.820335
```

Test it out

```
sim_m0 <- lmer(score ~ ses + (1|scid), data = sim_df)
summary(sim_m0)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: score ~ ses + (1 | scid)
##    Data: sim_df
##
## REML criterion at convergence: 9704.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1418 -0.6848  0.0030  0.6552  3.5886
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  scid     (Intercept)  5.685   2.384
##  Residual             36.187   6.016
## Number of obs: 1500, groups:  scid, 30
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  12.3901     0.4622   26.81
## ses           2.4682     0.1562   15.80
##
## Correlation of Fixed Effects:
```

Expanding the model

Let's add a school–level predictor

```
hsb_m1 <- lmer(math ~ ses + sector + (1|sch.id), data = hsb)
```

```
extract_eq(hsb_m1)
```

$$\begin{aligned} \text{math}_i &\sim N\left(\alpha_{j[i]} + \beta_1(\text{ses}), \sigma^2\right) \\ \alpha_j &\sim N\left(\gamma_0^\alpha + \gamma_1^\alpha(\text{sector}), \sigma^2_{\alpha_j}\right), \text{for sch.id j} = 1, \ldots, \text{J} \end{aligned}$$

Add in a random slope

```
hsb_m2 <- lmer(math ~ ses + sector + (ses|sch.id), data = hsb)
extract_eq(hsb_m2)
```

$$\text{math}_i \sim N\left(\alpha_{j[i]} + \beta_{1j[i]}(\text{ses}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{sector}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for sch.id j} = 1, \ldots, \text{J}$$

Include interaction

Include `sector` as a predictor of the relation between `ses` and `math`

```
hsb_m3 <- lmer(math ~ ses * sector + (ses|sch.id), data = hsb,
               control = lmerControl(optimizer = "nmkbw"))
extract_eq(hsb_m3)
```

$$\text{math}_i \sim N\left(\alpha_{j[i]} + \beta_{1j[i]}(\text{ses}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{sector}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{sector}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for sch.id j} = 1, \ldots, \text{J}$$

Even more complicated

This model doesn't actually fit well – I omitted some convergence warnings

```
hsb_m4 <- lmer(
  math ~ ses * sector + minority + female + meanses + size +
    (ses + minority + female|sch.id),
  data = hsb
)

extract_eq(hsb_m4)
```

$$
\begin{aligned}
\text{math}_i &\sim N\left(\mu, \sigma^2\right) \\
\mu &= \alpha_{j[i]} + \beta_{1j[i]}(\text{ses}) + \beta_{2j[i]}(\text{minority}) + \beta_{3j[i]}(\text{female}) \\
\begin{pmatrix} \alpha_j \\ \beta_{1j} \\ \beta_{2j} \\ \beta_{3j} \end{pmatrix} &\sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{sector}) + \gamma_2^{\alpha}(\text{meanses}) + \gamma_3^{\alpha}(\text{size}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{sector}) \\ \mu_{\beta_{2j}} \\ \mu_{\beta_{3j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} & \rho_{\alpha_j\beta_{2j}} & \rho_{\alpha_j\beta_{3j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} & \rho_{\beta_{1j}\beta_{2j}} & \rho_{\beta_{1j}\beta_{3j}} \\ \rho_{\beta_{2j}\alpha_j} & \rho_{\beta_{2j}\beta_{1j}} & \sigma^2_{\beta_{2j}} & \rho_{\beta_{2j}\beta_{3j}} \\ \rho_{\beta_{3j}\alpha_j} & \rho_{\beta_{3j}\beta_{1j}} & \rho_{\beta_{3j}\beta_{2j}} & \sigma^2_{\beta_{3j}} \end{pmatrix} \right), \text{for sch.id j} = 1, \dots, \text{J}
\end{aligned}
$$

Multiple levels

Let's go to a different dataset from equatiomatic

```
head(sim_longitudinal)
```

```
## # A tibble: 6 x 8
## # Groups:   school [1]
##     sid school district group  treatment  prop_low  wave    score
##   <int>  <int>    <int> <chr>  <fct>         <dbl> <dbl>    <dbl>
## 1     1      1        1 medium 1         0.1428571     0 102.2686
## 2     1      1        1 medium 1         0.1428571     1 102.0135
## 3     1      1        1 medium 1         0.1428571     2 102.5216
## 4     1      1        1 medium 1         0.1428571     3 102.2792
## 5     1      1        1 medium 1         0.1428571     4 102.2834
## 6     1      1        1 medium 1         0.1428571     5 102.7963
```

Four levels

Model doesn't really fit again

```
sl_m <- lmer(
  score ~ wave*treatment + group + prop_low +
    (wave|sid) + (wave + treatment| school) + (1|district),
  data = sim_longitudinal
)
extract_eq(sl_m)
```

$$\text{score}_i \sim N\left(\alpha_{j[i],k[i],l[i]} + \beta_{1j[i],k[i]}(\text{wave}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_{1k[i]}^{\alpha}(\text{treatment}_1) + \gamma_2^{\alpha}(\text{group}_{\text{low}}) + \gamma_3^{\alpha}(\text{group}_{\text{medium}}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{treatment}_1) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for sid j} = 1, \ldots$$

$$\begin{pmatrix} \alpha_k \\ \beta_{1k} \\ \gamma_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{prop\_low}) \\ \mu_{\beta_{1k}} \\ \mu_{\gamma_{1k}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & \rho_{\alpha_k\beta_{1k}} & \rho_{\alpha_k\gamma_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma^2_{\beta_{1k}} & \rho_{\beta_{1k}\gamma_{1k}} \\ \rho_{\gamma_{1k}\alpha_k} & \rho_{\gamma_{1k}\beta_{1k}} & \sigma^2_{\gamma_{1k}} \end{pmatrix}\right), \text{for school k} = 1, \ldots, \text{K}$$

$$\alpha_l \sim N\left(\mu_{\alpha_l}, \sigma^2_{\alpha_l}\right), \text{for district l} = 1, \ldots, \text{L}$$

Residual structures

Data

Willett, 1988

- n = 35 people
- Each completed a cognitive inventory on "opposites naming"
- At first time point, participants also completed a general cognitive measure

Read in data

```
willett <- read_csv(here::here("data", "willett-1988.csv"))
willett
```

```
## # A tibble: 140 x 4
##       id  time   opp   cog
##    <dbl> <dbl> <dbl> <dbl>
##  1     1     0   205   137
##  2     1     1   217   137
##  3     1     2   268   137
##  4     1     3   302   137
##  5     2     0   219   123
##  6     2     1   243   123
##  7     2     2   279   123
##  8     2     3   302   123
##  9     3     0   142   129
## 10     3     1   212   129
## # … with 130 more rows
```

Standard OLS

- We have four observations per participant.
- If we fit a standard OLS model, it would look like this

```
bad <- lm(opp ~ time, data = willett)
summary(bad)
```

```
##
## Call:
## lm(formula = opp ~ time, data = willett)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -88.374 -25.584   1.186  28.926  64.746
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  164.374      5.035   32.65   <2e-16 ***
## time          26.960      2.691   10.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35.6 on 138 degrees of freedom
```

Assumptions

As we discussed previously, this model looks like this

$$\mathbf{opp} = \alpha + \beta_1(\mathbf{time}) + \epsilon$$

where

$$\epsilon \sim (0, \sigma)$$

Individual level residuals

We can expand our notation, so it looks like a multivariate normal distribution

$$\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{pmatrix} \sim MVN \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_\epsilon & 0 & 0 & \dots & 0 \\ 0 & \sigma_\epsilon & 0 & 0 & 0 \\ 0 & 0 & \sigma_\epsilon & 0 & 0 \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_\epsilon \end{bmatrix} \right)$$

This is where the $i.i.d.$ part comes in. The residuals are assumed $i$ndependent and $i$dentically $d$istributed.

Multilevel model

Very regularly, there are reasons to believe the $i.i.d.$ assumption is violated. Consider our current case, with 4 time points for each individual.

- Is an observation for one time point for one individual *independent* from the other observations for that individual?

- Rather than estimating a single residual variance, we estimate an additional components associated with individuals, leading to a *block* diagonal structure

Block diagonal

$$\begin{pmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{14} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{23} \\ \epsilon_{24} \\ \vdots \\ \epsilon_{n1} \\ \epsilon_{n2} \\ \epsilon_{n3} \\ \epsilon_{n4} \end{pmatrix} \sim \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} \end{bmatrix} \right)$$

Correlations for off–diagonals estimated

Same variance components for all blocks

Out–of–block diagonals are still zero

Homogeneity of variance

As mentioned on the previous slide, we assume the same variance components across all student

This is referred to as the homogeneity of variance assumption – although the block (often referred to as the composite residual) may be heteroscedastic and dependent **within** a grouping factor (i.e., people) the entire error structure is repeated identically across units (i.e., people)

Block diagonal

Because of the homogeneity of variance assumption, we can re–express our block diagonal design as follows

$$r \sim N\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{\Sigma}_r & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_r & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_r & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{\Sigma}_r \end{bmatrix}\right)$$

Composite residual

We then define the composite residual, which is common across units

$$\boldsymbol{\Sigma_r} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} \end{bmatrix}$$

Let's try!

Let's fit a parallel slopes model with the Willett data. You try first.

```
w0 <- lmer(opp ~ time + (1|id), willett)
```

What does the residual variance–covariance look like? Let's use **sundry** to pull it

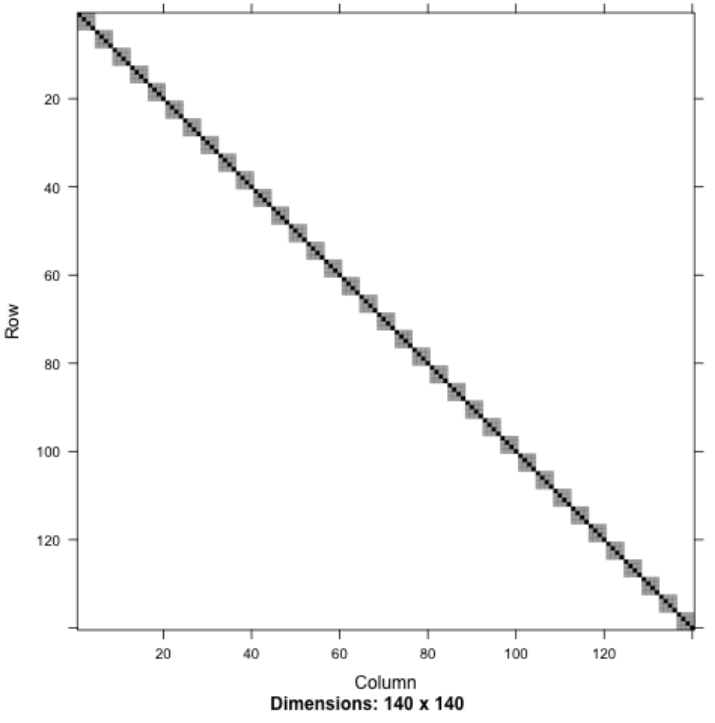
```
library(sundry)
w0_rvcv <- pull_residual_vcov(w0)
```

02:00

Image

Sparse matrix – we can view it with `image()`

```
image(w0_rvcv)
```



Pull first few rows/cols

```
w0_rvcv[1:8, 1:8]
```

```
## 8 x 8 sparse Matrix of class "dgCMatrix"
##           1         2         3         4         5         6         7
## 1 1280.7065  904.8054  904.8054  904.8054    .         .         .
## 2  904.8054 1280.7065  904.8054  904.8054    .         .         .
## 3  904.8054  904.8054 1280.7065  904.8054    .         .         .
## 4  904.8054  904.8054  904.8054 1280.7065    .         .         .
## 5    .         .         .         .      1280.7065  904.8054  904.8054
## 6    .         .         .         .       904.8054 1280.7065  904.8054
## 7    .         .         .         .       904.8054  904.8054 1280.7065
## 8    .         .         .         .       904.8054  904.8054  904.8054
```

Structure

On the previous slide, note the values on the diagonal are all the same, as are all the off–diagonals

- This is because we've only estimated one additional variance component

Understanding these numbers

Let's look at the model output

```
arm::display(w0)
```

```
## lmer(formula = opp ~ time + (1 | id), data = willett)
##             coef.est coef.se
## (Intercept) 164.37     5.78
## time         26.96     1.47
##
## Error terms:
##  Groups   Name        Std.Dev.
##  id       (Intercept) 30.08
##  Residual             19.39
## ---
## number of obs: 140, groups: id, 35
## AIC = 1308.3, DIC = 1315.9
## deviance = 1308.1
```

The diagonal values were $\mathbf{1280.7065389}$ while the off diagonal values were $\mathbf{904.8053852}$

Let's extract the variance components from our model.

```
vars_w0 <- as.data.frame(VarCorr(w0))
vars_w0
```

```
##        grp        var1 var2     vcov    sdcor
## 1       id (Intercept) <NA> 904.8054 30.07998
## 2 Residual        <NA> <NA> 375.9012 19.38817
```

Notice anything?

The diagonals are given by `sum(vars_w0)$vcov` while the off–diagonals are just the intercept variance

Including more complexity

Try estimating this model now, then look at the residual variance–covariance matrix again

$$\text{opp}_i \sim N\left(\alpha_{j[i]} + \beta_{1j[i]}(\text{time}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_{\alpha_j} \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for id j} = 1, \ldots, \text{J}$$

04:00

The composite residual

```
w1 <- lmer(opp ~ time + (time|id), willett)
w1_rvcv <- pull_residual_vcov(w1)
w1_rvcv[1:4, 1:4]
```

```
## 4 x 4 sparse Matrix of class "dgCMatrix"
##           1         2         3         4
## 1 1358.2469 1019.5095  840.2515  660.9934
## 2 1019.5095 1132.1294  925.7905  878.9310
## 3  840.2515  925.7905 1170.8089 1096.8686
## 4  660.9934  878.9310 1096.8686 1474.2856
```

Unstructured

The model we fit has an *unstructured* variance co–variance matrix. While each block is the same, every element of the block is now estimated.

What are these numbers?

They are the variance components, re–expressed as a composite residual

The diagonal is given by

$$\sigma^2 + \sigma^2_{\alpha_j} + 2\sigma^2_{01} w_i + \sigma^2_{\beta_1} w_i^2$$

where $\boldsymbol{w}$ represents the given wave (for our example)

Let's do this "by hand"

Get the pieces

```
vars_w1 <- as.data.frame(VarCorr(w1))

# get the pieces
int_var <- vars_w1$vcov[1]
slope_var <- vars_w1$vcov[2]
covar <- vars_w1$vcov[3]
residual <- vars_w1$vcov[4]
```

Calculate

```
diag(w1_rvcv[1:4, 1:4])
```

```
## [1] 1358.247 1132.129 1170.809 1474.286
```

```
residual + int_var
```

```
## [1] 1358.247
```

```
residual + int_var + 2*covar + slope_var
```

```
## [1] 1132.129
```

```
residual + int_var + (2*covar)*2 + slope_var*2^2
```

```
## [1] 1170.809
```

```
residual + int_var + (2*covar)*3 + slope_var*3^2
```

```
## [1] 1474.286
```

Off–diagonals

The off–diagonals are given by

$$\sigma^2_{\alpha_j} + \sigma_{01}(t_i + t'_i) + \sigma^2_{\beta_1} t_i t'_i$$

Calculate a few

```
w1_rvcv[1:4, 1:4]
```

```
## 4 x 4 sparse Matrix of class "dgCMatrix"
##           1         2         3         4
## 1 1358.2469 1019.5095  840.2515  660.9934
## 2 1019.5095 1132.1294  925.7905  878.9310
## 3  840.2515  925.7905 1170.8089 1096.8686
## 4  660.9934  878.9310 1096.8686 1474.2856
```

```
int_var + covar*(1 + 0) + slope_var*1*0
```

```
## [1] 1019.51
```

```
int_var + covar*(2 + 1) + slope_var*2*1
```

```
## [1] 925.7905
```

```
int_var + covar*(3 + 2) + slope_var*3*2
```

```
## [1] 1096.869
```

Positing other structures

The possibilities

There are a number of alternative structures. We'll talk about a few here.

If you want to go deeper, I suggest Singer & Willett, Chapter 7

Code to fit models with each type of structure, using the same Willett data we're using today, is available here

Structures we'll fit

- Unstructured (default with **lme4**, we've already seen this)
- Autoregressive
- Heterogeneous autoregressive
- Toeplitz

Outside of unstructured, we'll need to use the **nlme** package to estimate other structures

We'll also use a generalized least squares estimator, rather than maximum likelihood

Autoregressive

Autoregressive

- There are many types of autoregressive structures
    - If you took a class on time–series data you'd learn about others
- What we'll talk about is referred to as an AR1 structure
- Variances (on the diagonal) are constant
- Includes constant "band–diagonals"

Autoregressive

$$\boldsymbol{\Sigma_r} = \begin{bmatrix} \sigma^2 & \sigma^2\rho & \sigma^2\rho^2 & \sigma^2\rho^3 \\ \sigma^2\rho & \sigma^2 & \sigma^2\rho & \sigma^2\rho^2 \\ \sigma^2\rho^2 & \sigma^2\rho & \sigma^2 & \sigma^2\rho \\ \sigma^2\rho^3 & \sigma^2\rho^2 & \sigma^2\rho & \sigma^2 \end{bmatrix}$$

- Each band is forced to be lower than the prior by a constant fraction
    - estimated autocorrelation parameter $\boldsymbol{\rho}$. The error variance is multiplied by $\boldsymbol{\rho}$ for the first diagonal, by $\boldsymbol{\rho^2}$ for the second, etc.
- Uses only two variance components

Fit

First load **nlme**

```
library(nlme)
```

We'll use the `gls()` function. The interface is, overall, fairly similar to **lme4**

```
ar <- gls(opp ~ time,
          data = willett,
          correlation = corAR1(form = ~ 1|id))
```

Summary

Notice, you're not actually estimating random effect variances here, just an alternative residual covariance structure

```
summary(ar)
```

```
## Generalized least squares fit by REML
##   Model: opp ~ time
##   Data: willett
##        AIC      BIC    logLik
##   1281.465 1293.174 -636.7327
##
## Correlation Structure: AR(1)
##  Formula: ~1 | id
##  Parameter estimate(s):
##       Phi
## 0.8249118
##
## Coefficients:
##                Value Std.Error  t-value p-value
## (Intercept) 164.33842  6.136372 26.78104       0
## time         27.19786  1.919857 14.16661       0
##
```

Extract composite residual

```
cm_ar <- corMatrix(ar$modelStruct$corStruct) # all of them
cr_ar <- cm_ar[[1]] # just the first (they're all the same)
cr_ar
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.8249118 0.6804795 0.5613356
## [2,] 0.8249118 1.0000000 0.8249118 0.6804795
## [3,] 0.6804795 0.8249118 1.0000000 0.8249118
## [4,] 0.5613356 0.6804795 0.8249118 1.0000000
```

Multiply the correlation matrix by the model residual variance to get the covariance matrix

```
cr_ar * sigma(ar)^2
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1323.4596 1091.7375  900.5872  742.9050
## [2,] 1091.7375 1323.4596 1091.7375  900.5872
## [3,]  900.5872 1091.7375 1323.4596 1091.7375
## [4,]  742.9050  900.5872 1091.7375 1323.4596
```

Confirming calculations

```
sigma(ar)^2
```

```
## [1] 1323.46
```

```
sigma(ar)^2*0.8249118
```

```
## [1] 1091.737
```

```
sigma(ar)^2*0.8249118^2
```

```
## [1] 900.5871
```

```
sigma(ar)^2*0.8249118^3
```

```
## [1] 742.9049
```

Heterogenous autoregressive

- Same as autorgressive but allows each variance to differ
- Still one ρ estimated
    - Same "decay" across band diagonals
- Band diagonals no longer equivalent, because different variances

Heterogenous autoregressive

$$\boldsymbol{\Sigma_r} = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho^2 & \sigma_1\sigma_4\rho^2 \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho^2 \\ \sigma_3\sigma_1\rho^2 & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho^3 & \sigma_4\sigma_2\rho^2 & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$$

Fit

Note – `varIdent` specifies different variances for each wave (variances of the identity matrix)

```
har <- gls(
  opp ~ time,
  data = willett,
  correlation = corAR1(form = ~ 1|id),
  weights = varIdent(form = ~1|time)
)
```

Summary

```
summary(har)
```

```
## Generalized least squares fit by REML
##   Model: opp ~ time
##   Data: willett
##       AIC     BIC    logLik
##   1285.76 1306.25 -635.8798
##
## Correlation Structure: AR(1)
##  Formula: ~1 | id
##  Parameter estimate(s):
##       Phi
## 0.8173622
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | time
##  Parameter estimates:
##        0        1        2        3
## 1.000000 0.915959 0.985068 1.045260
##
## Coefficients:
##                Value Std.Error  t-value p-value
## (Intercept) 164.63344  5.959533 27.62523       0
## time         27.11552  1.984807 13.66154       0
##
##  Correlation:
```

Extract/compute composite residual

The below is fairly complicated, but you can work it out if you go line by line

```
cm_har <- corMatrix(har$modelStruct$corStruct)[[1]]
var_struct <- har$modelStruct$varStruct
vars <- coef(var_struct, unconstrained = FALSE, allCoef = TRUE)
vars <- matrix(vars, ncol = 1)

cm_har * sigma(har)^2 *
  (vars %*% t(vars)) # multiply by a mat of vars
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1308.6660  979.7593  861.2399  746.9588
## [2,]  979.7593 1097.9457  965.1295  837.0629
## [3,]  861.2399  965.1295 1269.8757 1101.3712
## [4,]  746.9588  837.0629 1101.3712 1429.8058
```

Toeplitz

- Constant variance
- Has identical band–diagonals, like autoregressive
- Relaxes assumption of each band being a parallel by a common fraction of the prior band
    - Each band determined empirically by the data

A bit of a compromise between prior two

Toeplitz

$$\boldsymbol{\Sigma}_{\boldsymbol{r}} = \begin{bmatrix} \sigma^2 & \sigma_1^2 & \sigma_2^2 & \sigma_3^2 \\ \sigma_1^2 & \sigma^2 & \sigma_1^2 & \sigma_2^2 \\ \sigma_2^2 & \sigma_1^2 & \sigma^2 & \sigma_1^2 \\ \sigma_2^2 & \sigma_2^2 & \sigma_1^2 & \sigma^2 \end{bmatrix}$$

Fit

```
toep <- gls(opp ~ time,
            data = willett,
            correlation = corARMA(form = ~ 1|id, p = 3))
```

Summary

```
summary(toep)
```

```
## Generalized least squares fit by REML
##   Model: opp ~ time
##   Data: willett
##        AIC      BIC    logLik
##   1277.979 1295.543 -632.9896
##
## Correlation Structure: ARMA(3,0)
##  Formula: ~1 | id
##  Parameter estimate(s):
##       Phi1       Phi2       Phi3
##  0.8039121  0.3665122 -0.3950326
##
## Coefficients:
##                Value Std.Error  t-value p-value
## (Intercept) 165.11855  6.122841 26.96764       0
## time         26.91997  2.070391 13.00236       0
##
##  Correlation:
##      (Intr)
## time -0.507
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -2.44029024 -0.71984566  0.01373249  0.77304950  1.75580973
```

Extract/compute composite residual

Same as with autoregressive – just multiply the correlation matrix by the residual variance

```
cr_toep <- corMatrix(toep$modelStruct$corStruct)[[1]]
cr_toep * sigma(toep)^2
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1333.6848 1105.7350  940.9241  634.8366
## [2,] 1105.7350 1333.6848 1105.7350  940.9241
## [3,]  940.9241 1105.7350 1333.6848 1105.7350
## [4,]  634.8366  940.9241 1105.7350 1333.6848
```

Comparing fits

```
library(performance)
compare_performance(ar, har, toep, w1,
                    metrics = c("AIC", "BIC"),
                    rank = TRUE) %>%
  as_tibble()
```

```
## # A tibble: 4 x 5
##   Name  Model        AIC      BIC Performance_Score
##   <chr> <chr>      <dbl>    <dbl>             <dbl>
## 1 toep  gls     1277.979 1295.543         0.9094409
## 2 w1    lmerMod 1278.823 1296.473         0.8196721
## 3 ar    gls     1281.465 1293.174         0.7759608
## 4 har   gls     1285.760 1306.250         0
```

We have slight evidence here that the Toeplitz structure fits better than the unstructured version, which was slightly better than the autoregressive model.

	Unstructured	Autoregressive (AR)	Heterogeneous AR	Toeplitz
(Intercept)	164.374	164.338	164.633	165.119
	(6.119)	(6.136)	(5.960)	(6.123)
time	26.960	27.198	27.116	26.920
	(2.167)	(1.920)	(1.985)	(2.070)
sd__(Intercept)	34.623			
cor__(Intercept).time	−0.450			
sd__time	11.506			
sd__Observation	12.629			
AIC	1278.8	1281.5	1285.8	1278.0
BIC	1296.5	1293.2	1306.3	1295.5
Log.Lik.	−633.411	−636.733	−635.880	−632.990
REMLcrit	1266.823			

Stepping back

Why do we care about all of this?

Overfitting

- If a simpler model fits the data just as well, it should be preferred
- Models that are overfit have "learned too much" from the data and won't generalize well
- Can think of it as fitting to the errors in the data, rather than the "true" patterns found in the population

Convergence issues

- As your models get increasingly complicated, you're likely to run into convergence issues.
- Simplifying your residual variance–covariance structure may help
    - Keeps the basic model structure intact

Aside – see here for convergence troubleshooting with **lme4**. The `allFit()` function is often helpful but very computationally intensive.

Summary

- As is probably fairly evident from the code – there are many more structures you could explore. However, most are not implemented through **lme4**.
- Simplifying the residual variance–covariance can sometimes lead to better fitting models
- May also be helpful with convergence and avoid overfitting

Next time

Modeling growth (part 1)

Also Homework 2 is assigned