SFS 2024 Short Course – Bayesian Applications in Environmental and Ecological Studies with R and Stan

Song S. Qian and Mark R. DuFour

6/1/2024

Hierarchical Structure and Regression Using Data from Multiple Sources

If we define "big data" as data obtained from multiple sources and representing multiple levels of aggregation, it becomes evident that most of the data we utilize in our work falls into the category of big data. In our context, the era of big data coincides with the era of hierarchical modeling. Failing to appropriately address the hierarchical structure inherent in the data can often result in misleading conclusions when working with big data.

When applying a regression model to data from multiple sources, we can either fitting regression models separately for each source or combining data together to fit one regression model. The former assumes data from different sources are completely different from each other (no pooling) and the later assumes that different sources are identical and can be treated as replicates (complete pooling). In practice, neither is satisfactory. When analyzing data from similar studies carried out in different locations, we cannot, on the one hand, assume that data are replicates because of regional differences in natural conditions. On the other hand, these studies targeted the same or similar relationships. As a result, the data are not entirely different, nor the same as replicates. When using the hierarchical modeling approach we explicitly incorporate the commonality and the potential differences in the same model to improve the overall optimal results.

In this section, we use the data from a USGS National Water Quality Assessment (NAWQA) study to illustrate the Bayesian hierarchical modeling approach of a linear regression problem. Here, we use data from a NAWQA study to illustrate our general approach in modeling: exploratory data analysis, proposing tentative models and fitting them using existing models in R, revising the model until a satisfactory one is achieved, and implementing the final model using Stan.

Effects of Urbanization on Stream Ecosystems (EUSE)

The EUSE project was briefly introduced in Qian (2016):

> The U.S. Geological Survey (USGS) is responsible for monitoring the country's natural resources. In 1991, USGS started a program to develop long-term consistent and comparable information on water quality and factors affecting aquatic ecosystems. The program is designed to understand the conditions of streams, rivers, and groundwater in the U.S. and their temporal trends. The program, known as the National Water Quality Assessment program (NAWQA), has both long-term monitoring networks and short-term topical studies. The project on the effects of urbanization on stream ecosystem (EUSE) is a topic study with an emphasis on the effects of various urbanization-induced changes in the landscape on water quality and aquatic ecosystem. The project, started in 1999, consists of a series of studies with a common design to examine the regional effects of urbanization on aquatic biota (fish, invertebrates, and algae), water chemistry, and physical habitat in nine metropolitan areas in different environmental settings. These studies are known as "urban gradient studies' ' because they were conducted on watersheds selected along urban gradients within their respective study areas. These study areas are in the metropolitan areas of Atlanta, Georgia (ATL); Boston, Massachusetts (BOS); Birmingham, Alabama (BIR);

Denver, Colorado (DEN); Dallas-Fort Worth, Texas (DFW); Milwaukee-Green Bay, Wisconsin (MGB); Portland, Oregon (POR); Raleigh, North Carolina (RAL); and Salt Lake City, Utah (SLC).

During the initial stage of EUSE, researchers developed a multimetric urban intensity index (UII) to identify representative gradients of urbanization within relatively homogeneous environmental settings associated with each urban area. Within each study area, 30 watersheds were selected to represent the urbanization gradient. These watersheds are of similar size and other natural characteristics, so that researchers can address several main questions:

- Do physical, chemical, and biological characteristics of streams respond to urban intensity?
- What are the rates of such response?
- What are typical indicators of changes caused by urbanization?
- What are typical characteristics of biological response to increased urban intensity?
- Do biological responses to urbanization vary by region?

Although these questions are ecological and environmental in nature, statistics played an important role in analyzing the data collected in the subsequent years.

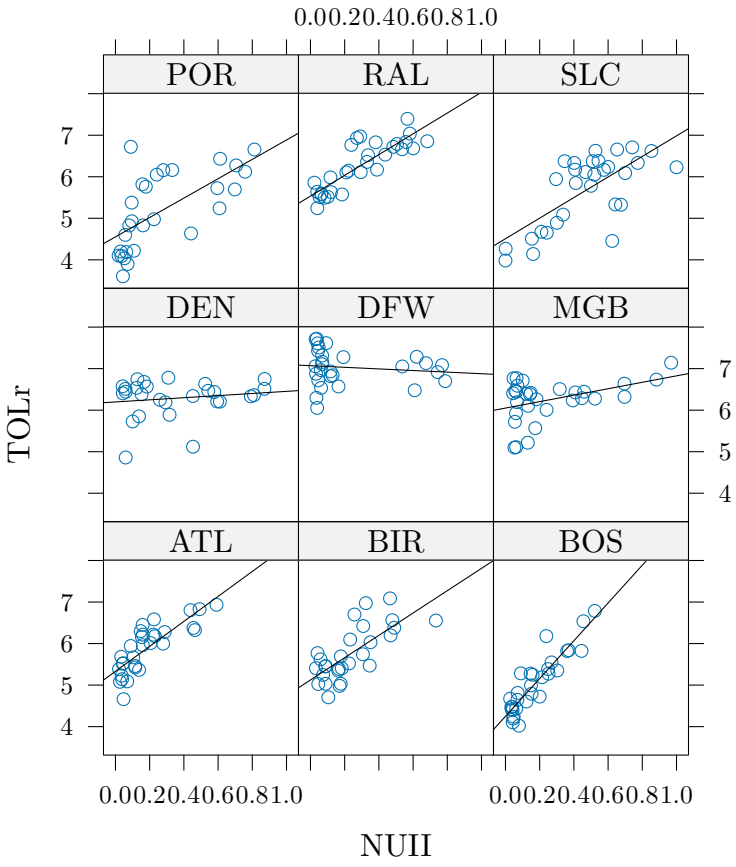
One indicator, average tolerance of macroinvertebrate taxa (TOLr), is of particular interest to USGS (Cuffney et al., 2005). The tolerance of a taxon is a measure of whether a taxon can survive in a polluted environment. In general, taxa with a higher tolerant score can be found in waters with poorer water quality. As such, the average tolerance scores of the sampled taxa from a water can be used as a biological measure of water quality. Initial analysis suggested that TOLr is linearly associated with watershed urbanization index (USGS' National Urban Intensity Index, NUII).

```
rtol2 <- read.csv(file=paste(dataDIR, "rtolforMS.csv", sep="/"),
                  header=T)

## environmental data
euse.env <- read.csv(paste(dataDIR, "EUSE_NAT_ENV.csv", sep="/"),
                     header=T)
names(euse.env)[13]<-"MAX.ELEV"
names(euse.env)[12]<-"MIN.ELEV"

AvePrec <- tapply(euse.env$AnnMeanP, euse.env$CITY, mean)
AveTemp <- tapply(euse.env$AnnMeanT, euse.env$CITY, mean)
AveElev <- tapply(euse.env$MEANELEV, euse.env$CITY, mean)
AveMaxT <- tapply(euse.env$AnnMaxT, euse.env$CITY, mean)
AveMaxP <- tapply(euse.env$AnnMaxP, euse.env$CITY, mean)
AveMinP <- tapply(euse.env$AnnMinP, euse.env$CITY, mean)
AvePdif <- tapply(euse.env$AnnMaxP-euse.env$AnnMinP, euse.env$CITY, mean)

city_ag<-read.csv(paste(dataDIR, "City_AG_Grassland.csv",
                        sep="/"), header=T, na.strings = ".")
city_ag<-cbind(city_ag[,1:2],city_ag[,3:5]/100)
city_ag[order(city_ag[ ,1]), ]
```

```
##   CITY SUID      PNLCD7     PNLCD8    PNLCD78
## 3  ATL ACFB 0.071979288 0.16511416 0.23709344
## 4  BIR MOBL 0.036328628 0.16987809 0.20620672
## 1  BOS NECB 0.003375359 0.07521565 0.07859101
## 6  DEN SPLT 0.635333298 0.22640489 0.86173819
## 7  DFW TRIN 0.208222135 0.54798380 0.75620594
## 5  MGB WMIC 0.009334345 0.76057259 0.76990694
## 9  POR WILL 0.074810446 0.12115522 0.19596567
## 2  RAL ALBE 0.050754986 0.21494498 0.26569997
```

```
## 8  SLC GRSL 0.000000000 0.02252253 0.02252253
```

```
ag<-city_ag[order(city_ag[ ,1]), ][,5]
ag.cat <- as.numeric(ag>0.5)
site <- as.numeric(ordered(rtol2$city))
ag.full <- as.vector(ag.cat[site])
temp.full <- as.vector(AveTemp[site])

rtol2$nuii <- rtol2$nuii/100

xyplot(richtol~nuii|city, data=rtol2,
       panel=function(x,y,...){
         panel.xyplot(x, y, ...)
         panel.lmline(x,y,...)},
       ylab="TOLr", xlab="NUII", aspect=1)
```



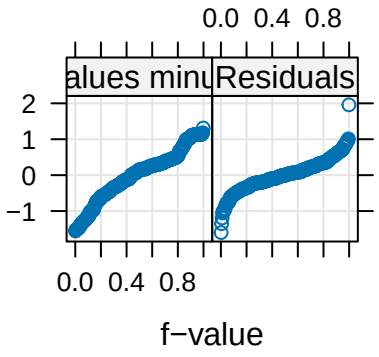
We start with a linear regression model.

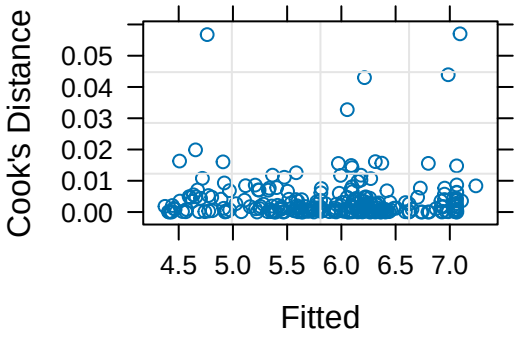
```
## No pooling:
euse.lm1 <- lm(richtol ~ nuii*city, data=rtol2)
display(euse.lm1, 4)
```

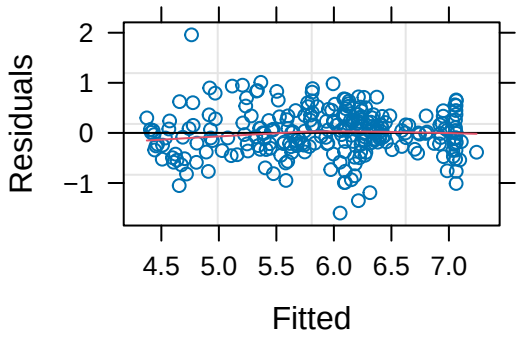
```
## lm(formula = richtol ~ nuii * city, data = rtol2)
##              coef.est coef.se
## (Intercept)   5.3318   0.1355
```

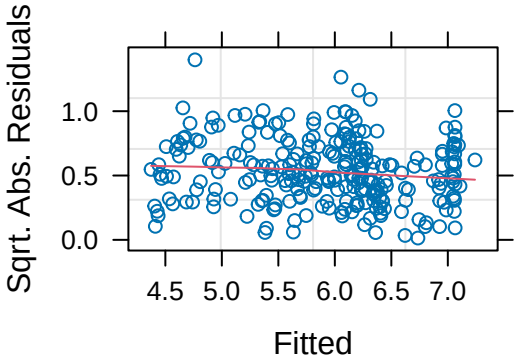
```
## nuii          3.0060   0.5584
## cityBIR      -0.2090   0.2054
## cityBOS      -1.0831   0.1942
## cityDEN       0.8660   0.2021
## cityDFW       1.7386   0.1788
## cityMGB       0.7183   0.1827
## cityPOR      -0.7788   0.1881
## cityRAL       0.2022   0.2054
## citySLC      -0.8238   0.2363
## nuii:cityBIR -0.3180   0.7696
## nuii:cityBOS  1.5421   0.8322
## nuii:cityDEN -2.7514   0.6521
## nuii:cityDFW -3.1988   0.6506
## nuii:cityMGB -2.2298   0.6504
## nuii:cityPOR -0.6731   0.6574
## nuii:cityRAL -0.5069   0.7078
## nuii:citySLC -0.5289   0.6679
## ---
## n = 261, k = 18
## residual sd = 0.4744, R-Squared = 0.72
```

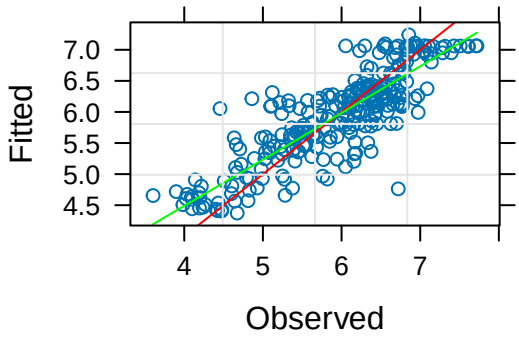
```
lm.plots(euse.lm1)
```

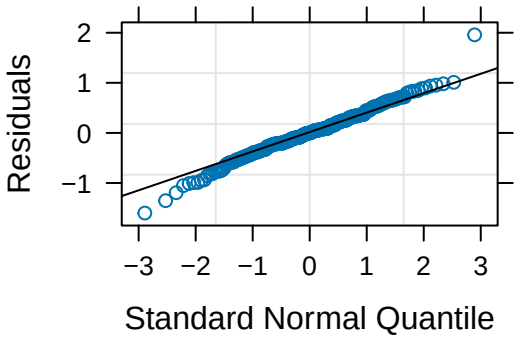













```
display(euse.lm11 <- update(euse.lm1,  .~. -1-nuii), 4)
```

```
## lm(formula = richtol ~ city + nuii:city - 1, data = rtol2)
##              coef.est coef.se
## cityATL       5.3318   0.1355
## cityBIR       5.1228   0.1544
## cityBOS       4.2486   0.1392
## cityDEN       6.1978   0.1499
## cityDFW       7.0704   0.1167
## cityMGB       6.0501   0.1227
## cityPOR       4.5529   0.1305
```

```
## cityRAL       5.5340   0.1543
## citySLC       4.5080   0.1936
## cityATL:nuii  3.0060   0.5584
## cityBIR:nuii  2.6880   0.5297
## cityBOS:nuii  4.5481   0.6171
## cityDEN:nuii  0.2546   0.3369
## cityDFW:nuii -0.1928   0.3340
## cityMGB:nuii  0.7762   0.3336
## cityPOR:nuii  2.3329   0.3470
## cityRAL:nuii  2.4990   0.4350
## citySLC:nuii  2.4770   0.3664
## ---
## n = 261, k = 18
## residual sd = 0.4744, R-Squared = 0.99
```

```
## Complete pooling
euse.lm2 <- lm(richtol~nuii, data=rtol2)
display(euse.lm2, 4)
```

```
## lm(formula = richtol ~ nuii, data = rtol2)
##             coef.est coef.se
## (Intercept) 5.5433   0.0752
## nuii        1.4018   0.2066
## ---
## n = 261, k = 2
## residual sd = 0.7963, R-Squared = 0.15
```

BHM using `lmer`

A hierarchical model:

$$\begin{pmatrix} y_{ij} \\ \begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \\ \epsilon_{ij} \end{pmatrix} \begin{matrix} = \\ \sim \\ \sim \end{matrix} \begin{matrix} \beta_{0j} + \beta_{1j} x_{ij} + \epsilon_{ij} \\ MVN\left[\begin{pmatrix}\mu_{\beta_0}\\ \mu_{\beta_1}\end{pmatrix}, \Sigma_\beta\right] \\ N(0, \sigma^2) \end{matrix}$$

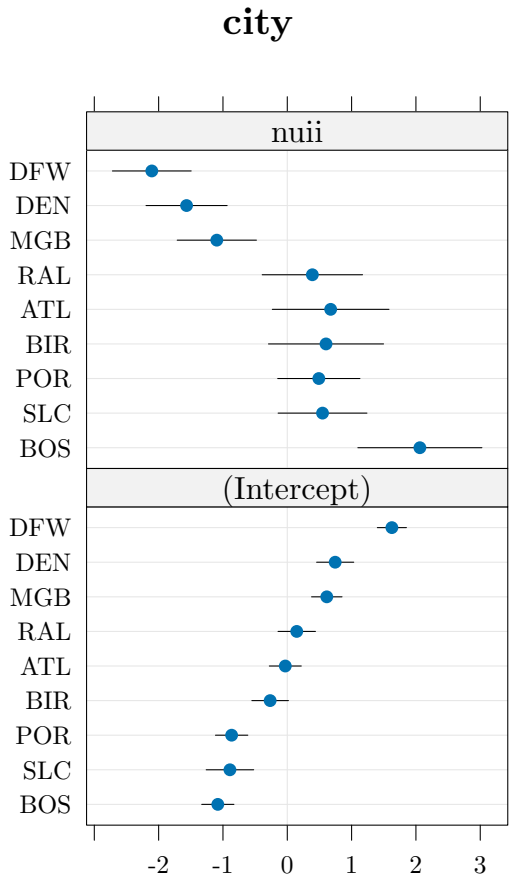
Can be implemented using restricted MLE (using function `lmer` from package `lme4`:

```
euse.lmer1 <- lmer(richtol ~ nuii + (1+nuii|city), data=rtol2)
display(euse.lmer1, 4)
```

```
## lmer(formula = richtol ~ nuii + (1 + nuii | city), data = rtol2)
##             coef.est coef.se
## (Intercept) 5.4184   0.3052
## nuii        1.9431   0.4790
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  city     (Intercept) 0.9040
##           nuii        1.3720   -0.8927
##  Residual             0.4747
## ---
## number of obs: 261, groups: city, 9
## AIC = 414.6, DIC = 399.1
## deviance = 400.9
```

```
dotplot(ranef(euse.lmer1))
```

```
## $city
```



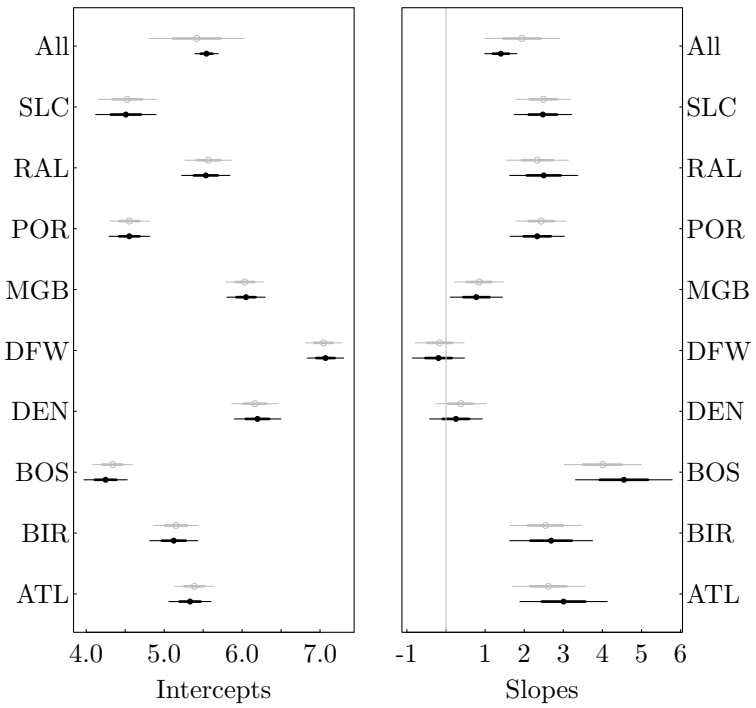
The common prior for model coefficients:

$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim MVN\left[\begin{pmatrix}\mu_{\beta_0}\\ \mu_{\beta_1}\end{pmatrix}, \Sigma_\beta\right]$$

represents the exchangeable assumption imposed on the model: we know that model coefficients (intercepts and slopes) are different from city to city (indexed by j). But we don't know how the coefficients vary by city, therefore, we assign the same prior distribution on all 9 models. The prior model coefficients $(\mu_{\beta_0}, \mu_{\beta_1}, \Sigma_\beta)$ are estimated from the combined data.

```
## comparing three linear models

## No pooling
ls2.int<-as.data.frame(rbind(summary(euse.lm11)$coef[1:9, 1:2],
                             summary(euse.lm2)$coef[1, 1:2]))
ls2.slp<-as.data.frame(rbind(summary(euse.lm11)$coef[10:18, 1:2],
                             summary(euse.lm2)$coef[2, 1:2]))

## partial pooling
lmer1.int<-as.data.frame(rbind(cbind(ranef(euse.lmer1)[[1]][,1]+
                                     fixef(euse.lmer1)[1],
                                se.ranef(euse.lmer1)[[1]][,1]),
```

```
                                  c(fixef(euse.lmer1)[1],
                                    se.fixef(euse.lmer1)[1])))
lmer1.slp<-as.data.frame(rbind(cbind(ranef(euse.lmer1)[[1]][,2]+
                                       fixef(euse.lmer1)[2],
                                    se.ranef(euse.lmer1)[[1]][,2]),
                                  c(fixef(euse.lmer1)[2],
                                    se.fixef(euse.lmer1)[2])))

par(mfrow=c(1,2), mar=c(4,4,0.5, 0.75), mgp=c(1.25,0.125,0),
    tck=0.01)
line.plots.compare(ls2.int[,1], ls2.int[,2], lmer1.int[,1],
                   lmer1.int[,2],
                   c(levels(factor(rtol2$city)), "All"),
                   "Intercepts", 2)
box()
par(mar=c(4,0.75,0.5,4))
line.plots.compare(ls2.slp[,1], ls2.slp[,2], lmer1.slp[,1], lmer1.slp[,2], c(levels(factor(rtol2$city))
box()
```




```
## the shrinkage effect of BHM
```

BHM with Group-level Predictor(s)

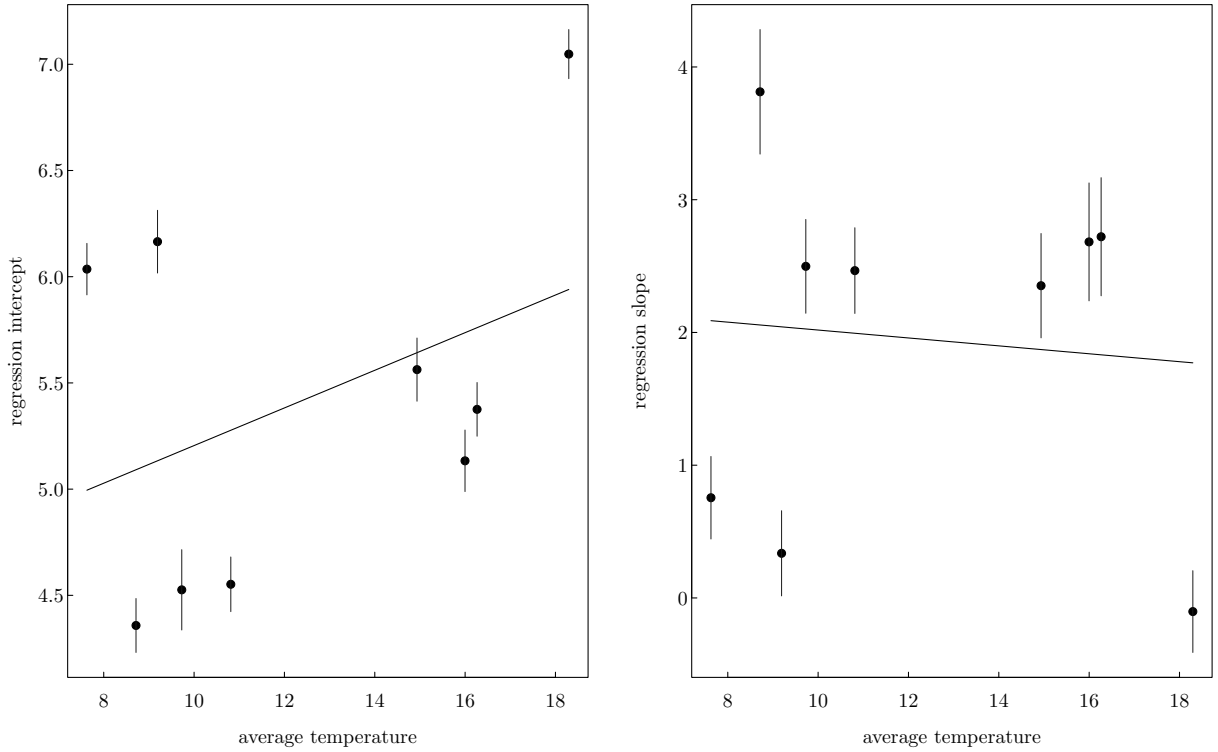
Why the differences?

The obvious difference among the 9 cities is the annual average temperature.

Let T_j be the annual average temperature of city j, we start with linear model:

$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim MVN\left[\begin{pmatrix}a_0 + a_1 \times T_j\\ b_0 + b_1 \times T_j\end{pmatrix}, \Sigma_\beta\right]$$

```
euse.lmer2 <- lmer(richtol ~ nuii + temp.full + nuii:temp.full +
                     (1+nuii|city), data=rtol2)
M2.coef <- coef (euse.lmer2)
a.hat.M2 <- M2.coef[[1]][,1] + M2.coef[[1]][,3]*AveTemp
b.hat.M2 <- M2.coef[[1]][,2] + M2.coef[[1]][,4]*AveTemp
a.se.M2 <- se.ranef(euse.lmer2)[[1]][,1]
b.se.M2 <- se.ranef(euse.lmer2)[[1]][,2]

# plot estimated intercepts and slopes
par (mfrow=c(1,2), mar=c(3,3,3,0.25), mgp=c(1.25,0.125,0),
     las=1, tck=0.01)
lower <- a.hat.M2 - a.se.M2
upper <- a.hat.M2 + a.se.M2
plot (AveTemp, a.hat.M2, ylim=range(lower,upper),
      cex.lab=0.75, cex.axis=0.75,
      xlab="average temperature",
      ylab="regression intercept", pch=20)
curve (fixef(euse.lmer2)["(Intercept)"] +
         fixef(euse.lmer2)["temp.full"]*x, lwd=1,
       col="black", add=TRUE)
segments (AveTemp, lower, AveTemp, upper, lwd=.5, col="gray10")
text(AveTemp, lower, levels(rtol2$city), adj=c(.5,1),cex=0.5)

lower <- b.hat.M2 - b.se.M2
upper <- b.hat.M2 + b.se.M2

plot (AveTemp, b.hat.M2, ylim=range(lower,upper), cex.lab=0.75,
      cex.axis=0.75,
      xlab="average temperature", ylab="regression slope",
      pch=20)
curve (fixef(euse.lmer2)["nuii"] +
         fixef(euse.lmer2)["nuii:temp.full"]*x, lwd=1,
       col="black", add=TRUE)
segments (AveTemp, lower, AveTemp, upper, lwd=.5, col="gray10")
text(AveTemp, lower, levels(rtol2$city), adj=c(.5,1),cex=0.5)
```



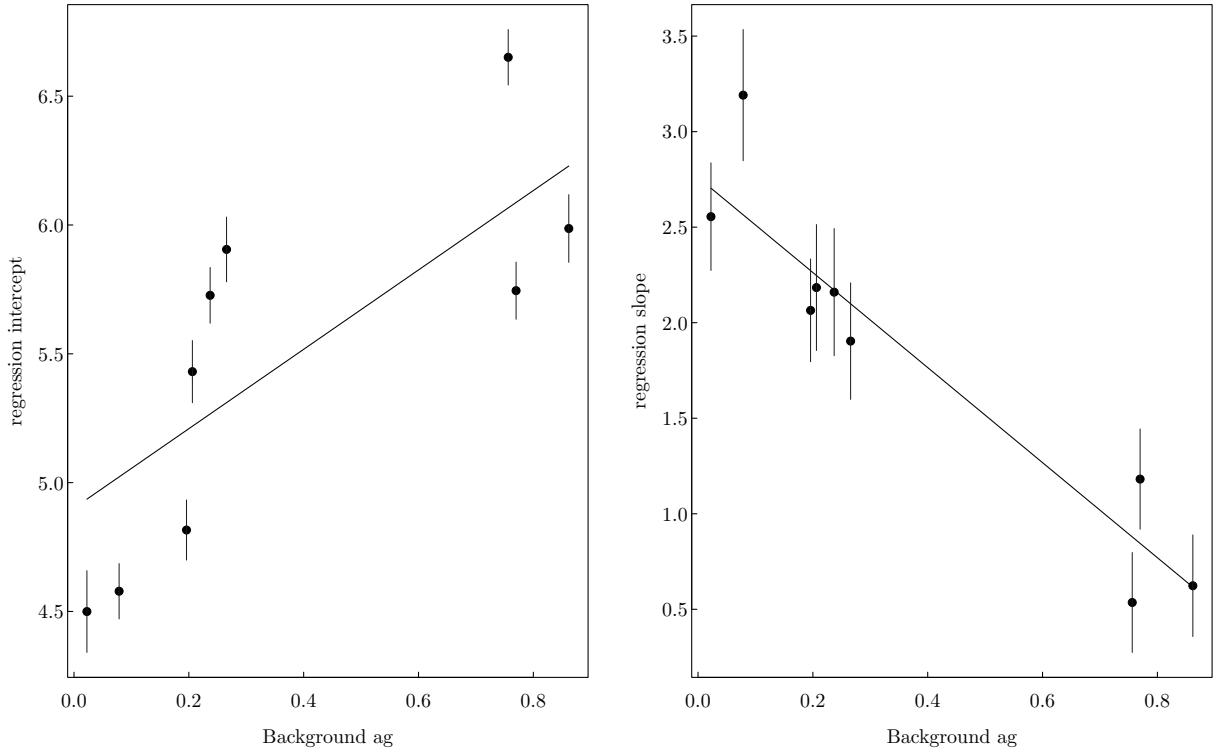
The 9 cities can be grouped into 2 groups, grouped by background agriculture land use. Now let Ag_j be the historical agricultural land use percentage:

$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim MVN\left[\begin{pmatrix}a_0 + a_1 \times Ag_j\\ b_0 + b_1 \times Ag_j\end{pmatrix}, \Sigma_\beta\right]$$

```
euse.lmer3 <- lmer(richtol ~ nuii+ag.full+nuii:ag.full+
                     (1+nuii|site), data=rtol2)
display(euse.lmer3, 4)  ## display no plonger works for lmer
```

```
## lmer(formula = richtol ~ nuii + ag.full + nuii:ag.full + (1 +
##     nuii | site), data = rtol2)
##              coef.est coef.se
## (Intercept)   4.9012   0.2144
## nuii          2.7601   0.2602
## ag.full       1.5407   0.3687
## nuii:ag.full -2.4875   0.4122
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  site     (Intercept) 0.5029
##           nuii        0.4408   -0.4876
##  Residual             0.4755
## ---
## number of obs: 261, groups: site, 9
## AIC = 404.9, DIC = 380.1
## deviance = 384.5
```

```
M3.coef <- coef (euse.lmer3)
a.hat.M3 <- M3.coef[[1]][,1] + M3.coef[[1]][,3]*ag
b.hat.M3 <- M3.coef[[1]][,2] + M3.coef[[1]][,4]*ag
a.se.M3 <- se.ranef(euse.lmer3)[[1]][,1]
b.se.M3 <- se.ranef(euse.lmer3)[[1]][,2]

# plot estimated intercepts and slopes

par (mfrow=c(1,2), mar=c(3,3,3,0.25), mgp=c(1.25,0.125,0),
     las=1, tck=0.01)
lower <- a.hat.M3 - a.se.M3
upper <- a.hat.M3 + a.se.M3
plot (ag, a.hat.M3, ylim=range(lower,upper), cex.lab=0.75,
      cex.axis=0.75,
      xlab="Background ag", ylab="regression intercept",
      pch=20)
curve (fixef(euse.lmer3)["(Intercept)"] +
         fixef(euse.lmer3)["ag.full"]*x, lwd=1, col="black",
       add=TRUE)
segments (ag, lower, ag, upper, lwd=.5, col="gray10")
text(ag, lower, levels(rtol2$city), adj=c(.5,1),cex=0.5)

lower <- b.hat.M3 - b.se.M3
upper <- b.hat.M3 + b.se.M3

plot (ag, b.hat.M3, ylim=range(lower,upper), cex.lab=0.75,
      cex.axis=0.75, xlab="Background ag",
      ylab="regression slope", pch=20)
curve (fixef(euse.lmer3)["nuii"] +
         fixef(euse.lmer3)["nuii:ag.full"]*x,
       lwd=1, col="black", add=TRUE)
segments (ag, lower, ag, upper, lwd=.5, col="gray10")
text(ag, lower, levels(rtol2$city), adj=c(.5,1),cex=0.5)
```



Adding both `ag` and `temp.full`:

$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim MVN\left[\begin{pmatrix}a_0 + a_1 \times T_j + a_2 \times Ag_j\\ b_0 + b_1 \times T_j + b_2 \times Ag_j\end{pmatrix}, \Sigma_\beta\right]$$

```
ag.cat2 <- ag.full>0.5

euse.lmer4 <- lmer(richtol ~ nuii+temp.full+nuii:temp.full+
                     (1+nuii|site)+
                     (1+nuii|ag.cat2), data=rtol2)
```

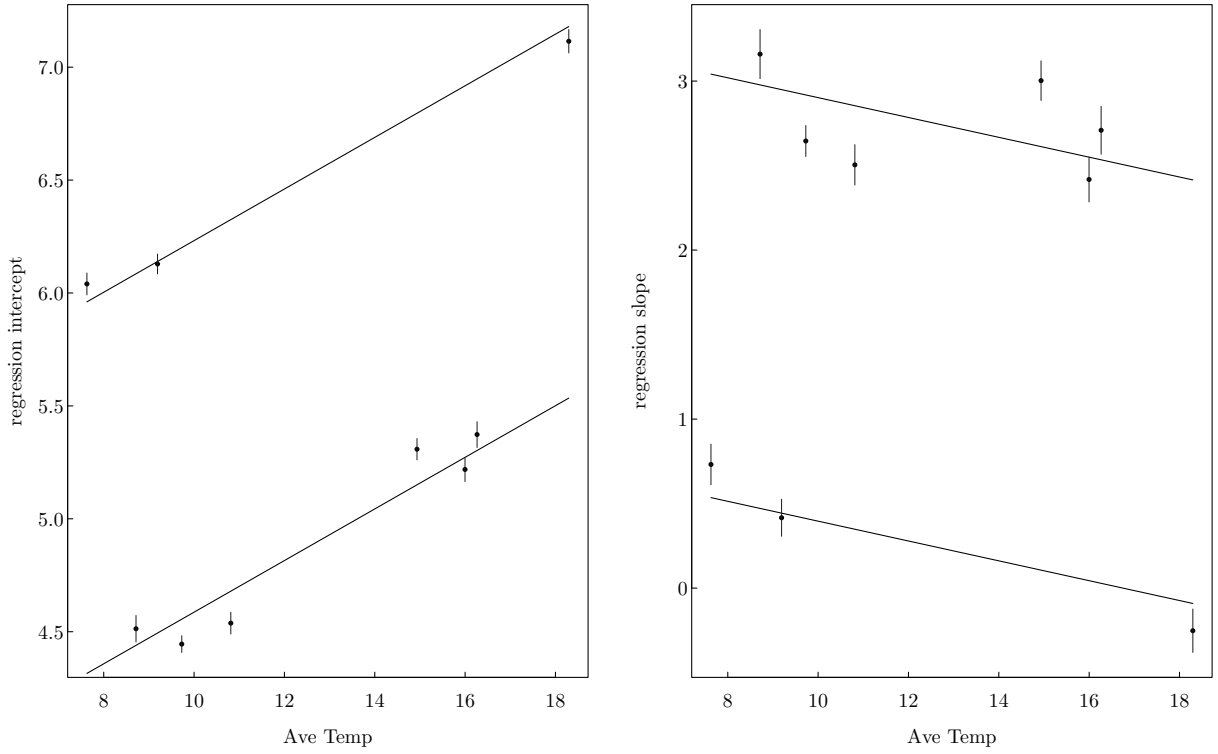
```
## boundary (singular) fit: see help('isSingular')
```

```
display(euse.lmer4, 4)
```

```
## lmer(formula = richtol ~ nuii + temp.full + nuii:temp.full +
##     (1 + nuii | site) + (1 + nuii | ag.cat2), data = rtol2)
##                coef.est coef.se
## (Intercept)     4.2663   0.8501
## nuii            2.2360   1.3738
## temp.full       0.1143   0.0158
## nuii:temp.full -0.0587   0.0437
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  site     (Intercept) 0.1182
##           nuii        0.2925   1.0000
##  ag.cat2  (Intercept) 1.1662
```

```
##           nuii        1.7760   -1.0000
##  Residual             0.4761
## ---
## number of obs: 261, groups: site, 9; ag.cat2, 2
## AIC = 410, DIC = 364.7
## deviance = 376.4
```

```
M4.fixef <- fixef (euse.lmer4)
M4.ranef <- ranef (euse.lmer4)
M4.sefixef <- se.fixef (euse.lmer4)
M4.seranef <- se.ranef (euse.lmer4)

a.hat.M4 <- M4.fixef[1] + M4.ranef[[1]][,1] +
  M4.ranef[[2]][c(1,1,1,2,2,2,1,1,1),1] +
  M4.fixef[3]*AveTemp
b.hat.M4 <- M4.fixef[2] + M4.ranef[[1]][,2] +
  M4.ranef[[2]][c(1,1,1,2,2,2,1,1,1),2] +
  M4.fixef[4]*AveTemp

a.se.M4 <- M4.seranef[[1]][,1]
b.se.M4 <- M4.seranef[[1]][,2]

# plot estimated intercepts and slopes

par (mfrow=c(1,2), mar=c(3,3,3,0.25), mgp=c(1.25,0.125,0),
     las=1, tck=0.01)
lower <- a.hat.M4 - a.se.M4
upper <- a.hat.M4 + a.se.M4
plot (AveTemp, a.hat.M4, ylim=range(lower,upper),
      cex.lab=0.75,  cex.axis=0.75, cex=0.5,
      xlab="Ave Temp", ylab="regression intercept", pch=20)
curve (fixef(euse.lmer4)[1] + fixef(euse.lmer4)[3]*x +
         M4.ranef[[2]][1,1], lwd=1, col="black", add=TRUE)
curve (fixef(euse.lmer4)[1] + fixef(euse.lmer4)[3]*x +
         M4.ranef[[2]][2,1], lwd=1, col="black", add=TRUE)
segments (AveTemp, lower, AveTemp, upper, lwd=.5, col="gray10")
text(AveTemp, lower, levels(rtol2$city), adj=c(.5,1),cex=0.5)

lower <- b.hat.M4 - b.se.M4
upper <- b.hat.M4 + b.se.M4

plot (AveTemp, b.hat.M4, ylim=range(lower,upper), cex.lab=0.75,
      cex.axis=0.75, cex=0.5,
      xlab="Ave Temp", ylab="regression slope", pch=20)
curve (fixef(euse.lmer4)[2] + fixef(euse.lmer4)[4]*x +
         M4.ranef[[2]][1,2], lwd=1, col="black", add=TRUE)
curve (fixef(euse.lmer4)[2] + fixef(euse.lmer4)[4]*x +
         M4.ranef[[2]][2,2], lwd=1, col="black", add=TRUE)
segments (AveTemp, lower, AveTemp, upper, lwd=.5, col="gray10")
text(AveTemp, lower, levels(rtol2$city), adj=c(.5,1),cex=0.5)
```



BHM using Stan

Model `euse.lmer4` is the best model –

But the algorithm used in `lme4` has difficulty in estimating the variances in the model. Using Stan, we need to have an efficient way to model the multivariate normal distribution, especially the covariance matrix Σ_β. The conjugate prior for the covariance matrix is the inverse Wishart distribution.

Stan reference manual recommends using a specification of the covariance matrix that is computationally suitable for Stan. The covariance matrix is decomposed into a vector of scales (standard deviation) and a correlation matrix:

$$\Sigma_\beta = \text{diag\_matrix}(\tau) \times \Omega \times \text{diag\_matrix}(\tau)$$

where Ω is a correlation matrix and τ is the coefficient scale matrix (standard deviation of β_0 and β_1). Stan User's Manual recommends a half-Cauchy distribution as the prior for τ, specifically, $\tau \sim \text{Cauchy}(0, 2.5)$. The correlation matrix Ω is recommended using the LJK prior with a shape > 1. When written in Stan code, the variance-covariance matrix part can be expressed as follows:

```
...
parameters{
  vector[K] beta[Nreg];
  corr_matrix[K] Omega;
  row_vector[K] mu_b;
  vector[K] tau;
...
model{
  tau ~ cauchy(0,2.5);
```

```
  Omega ~ ljk_corr(2);
  beta ~ multi_normal(mu_b, quad_form_diag(Omega,tau));
}
...
```

To improve computational efficiency, we can parameterize the correlation matrix Ω in terms of its Cholesky factor $\mathbf{L}$ (i.e., $\mathbf{L}\mathbf{L}' = \Omega$)using Stan's Cholesky LJK correlation distribution. As in a univariate normal random variable $y \sim N(\mu, \sigma^2)$ with known μ and σ where we can draw random numbers of y by drawing a standard normal random variable ($z \sim N(0,1)$) and calculate $y = \mu + z\sigma$, a multivariate normal vector $\mathbf{y} \sim MVN(\mu, \Sigma)$ can be drawn by using z: $\mathbf{y} = \mu + z \times \sqrt{\Sigma}$. Because $\mathbf{L}$ is the square root of the correlation matrix, the covariance matrix $\Sigma = (\text{diag\_matrix}(\tau) \cdot \mathbf{L})(\mathbf{L} \cdot \text{diag\_matrix}(\tau))^T$. In Stan, such computation is made simple with the following distribution and function:

```
parameter{
  matrix[K,Nreg] z;
  cholesky_factor_corr[K] L_Omega;
  row_vector[K] mu_b;
  vector<lower=0>[K] tau;
}
transformed parameters{
  matrix[Nreg,K] beta;
  beta = rep_matrix(mu_b,Nreg)+
         (diag_pre_multiply(tau,L_Omega)*z)';
}
model {
  //prior
  to_vector(z) ~ std_normal();
  L_Omega ~ ljk_corr_cholesky(2);
  //likelihood
  ...
}
```

The Stan model without group-level predictors We demonstrate Stan implementations of linear BHM.

```
euse_stan1 <- "
data {
  int<lower=0> N;
  int<lower=0> Nreg;
  int<lower=1> K;
  int<lower=1,upper=Nreg> region[N];
  matrix[N,K] x;
  vector[N] y;
}
parameters {
  vector[K] beta[Nreg];
  corr_matrix[K] Omega;
  row_vector[K] mu_b;
  vector<lower=0>[K] tau;
  real<lower=0,upper=10> sigma_y;
}
transformed parameters {
  vector[N] mu_y;
  for (i in 1:N)
    mu_y[i] = x[i] * beta[region[i]];
```

```
}
model {
  tau ~ cauchy(0,2.5);
  Omega ~ lkj_corr(2);
  beta ~ multi_normal(mu_b, quad_form_diag(Omega, tau));
  y ~ normal(mu_y, sigma_y);
}
"
fit1 <- stan_model(model_code=euse_stan1)
stan_in1 <- function(data=rtol2, chains=nchains){
    y <- data$richtol
    x <- as.matrix(cbind(1, data$nuii))
    n <- dim(data)[1]
    k <- dim(x)[2]
    reg <- as.numeric(ordered(data$city))
    nreg <- max(reg)
    stan_data <- list(N=n, Nreg=nreg, K=k, x=x, y=y, region=reg)
    stan_inits <- list()
    for (i in 1:chains)
        stan_inits[[i]] <- list(beta=matrix(rnorm(nreg*k),
                                            nrow=nreg),
                                mu_b=rnorm(k), tau=runif(k),
                                sigma_y=runif(1))
    stan_pars <- c("beta","mu_b","tau","sigma_y")
    return(list(data=stan_data, inits=stan_inits,
                pars=stan_pars, n.chains=chains))
}

input.to.stan <- stan_in1()
fit2keep <- sampling(fit1, data=input.to.stan$data,
                     init=input.to.stan$inits,
                     pars=input.to.stan$pars,
                     iter=niters,thin=nthin,
                     chains=input.to.stan$n.chains)#,
#                     control=list(adapt_delta = 0.9))
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##            mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1,1]  5.37    0.00 0.13  5.12  5.28  5.37  5.46  5.63  2318    1
## beta[1,2]  2.75    0.01 0.51  1.78  2.40  2.75  3.10  3.72  2562    1
## beta[2,1]  5.15    0.00 0.15  4.87  5.05  5.15  5.25  5.44  2489    1
## beta[2,2]  2.56    0.01 0.48  1.61  2.24  2.55  2.89  3.54  2535    1
## beta[3,1]  4.33    0.00 0.14  4.07  4.24  4.33  4.43  4.59  2402    1
## beta[3,2]  4.07    0.01 0.59  2.95  3.65  4.06  4.47  5.21  2502    1
## beta[4,1]  6.15    0.00 0.15  5.87  6.05  6.16  6.25  6.43  2526    1
## beta[4,2]  0.38    0.01 0.32 -0.24  0.16  0.38  0.60  1.03  2412    1
## beta[5,1]  7.03    0.00 0.12  6.80  6.95  7.03  7.11  7.26  2530    1
## beta[5,2] -0.09    0.01 0.33 -0.74 -0.31 -0.09  0.13  0.57  2806    1
## beta[6,1]  6.03    0.00 0.12  5.78  5.95  6.03  6.11  6.26  2494    1
## beta[6,2]  0.85    0.01 0.32  0.22  0.64  0.85  1.07  1.46  2280    1
```

```
## beta[7,1]  4.57    0.00 0.13  4.31  4.48  4.57  4.66  4.83  2396    1
## beta[7,2]  2.34    0.01 0.34  1.68  2.11  2.34  2.56  3.00  2571    1
## beta[8,1]  5.55    0.00 0.15  5.25  5.45  5.55  5.65  5.85  2528    1
## beta[8,2]  2.41    0.01 0.42  1.58  2.14  2.40  2.68  3.26  2492    1
## beta[9,1]  4.55    0.00 0.19  4.18  4.42  4.55  4.68  4.91  2721    1
## beta[9,2]  2.43    0.01 0.35  1.76  2.20  2.42  2.66  3.12  2694    1
## mu_b[1]    5.42    0.01 0.33  4.75  5.21  5.42  5.63  6.07  2638    1
## mu_b[2]    1.96    0.01 0.52  0.90  1.63  1.96  2.30  2.99  2732    1
## tau[1]     0.94    0.00 0.25  0.59  0.77  0.89  1.06  1.55  2594    1
## tau[2]     1.44    0.01 0.42  0.84  1.15  1.37  1.66  2.41  2597    1
## sigma_y    0.48    0.00 0.02  0.44  0.46  0.48  0.49  0.52  2558    1
## lp__      54.03    0.07 3.66 45.79 51.89 54.45 56.69 60.08  2392    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:46:37 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Now processing Stan output using `rv`

```
## EUSE no group-level predictor
euse_multilevel1 <- rvsims(as.matrix(as.data.frame(
    rstan::extract(fit2keep, pars=c("beta","mu_b",
                   "tau", "sigma_y")))))

## compare to lmer results
### 1. Intercepts
cbind(coef(euse.lmer1)$city[,1],
summary(euse_multilevel1[1:9])$mean)
```

```
##           [,1]     [,2]
##  [1,] 5.387641 5.371383
##  [2,] 5.152254 5.150745
##  [3,] 4.338540 4.334409
##  [4,] 6.163262 6.152162
##  [5,] 7.044866 7.029762
##  [6,] 6.032836 6.025164
##  [7,] 4.553434 4.569121
##  [8,] 5.565650 5.550108
##  [9,] 4.527003 4.546437
```

```
### 2. Slopes
cbind(coef(euse.lmer1)$city[,2],
summary(euse_multilevel1[10:18])$mean)
```

```
##             [,1]        [,2]
##  [1,]  2.6176101  2.75392117
##  [2,]  2.5456275  2.56160087
##  [3,]  4.0047871  4.07081848
##  [4,]  0.3774699  0.38240449
##  [5,] -0.1629494 -0.09025824
##  [6,]  0.8476555  0.85539728
##  [7,]  2.4334600  2.34083821
##  [8,]  2.3330872  2.40859144
##  [9,]  2.4912182  2.43164830
```

```
### 3. fixed effect standard deviation
cbind(fixef(euse.lmer1),
summary(euse_multilevel1[19:20])$mean)
```

```
##                 [,1]     [,2]
## (Intercept) 5.418387 5.418311
## nuii        1.943107 1.960139
```

```
### 4. Variance components
cbind(attr(summary(euse.lmer1)$varcor$city, "stddev"),
summary(euse_multilevel1[21:22])$mean)
```

```
##                  [,1]      [,2]
## (Intercept) 0.9040097 0.9436777
## nuii        1.3720008 1.4452367
```

```
### 5 Residual stddev:
c(attr(summary(euse.lmer1)$varcor, "sc"),summary(euse_multilevel1[23])$mean)
```

```
## [1] 0.4746692 0.4773307
```

The same model can be programmed using Cholesky factorization:

```
euse_stan15 <- "
data {
  int<lower=0> N;
  int<lower=0> Nreg;
  int<lower=1> K;
  int<lower=1,upper=Nreg> region[N];
  matrix[N,K] x;
  vector[N] y;
}
parameters {
  matrix[K,Nreg] z;
  cholesky_factor_corr[K] L_Omega;
  row_vector[K] mu_b;
  vector<lower=0>[K] tau;
  real<lower=0,upper=10> sigma_y;
}
transformed parameters {
  matrix[Nreg,K] beta;
  beta = rep_matrix(mu_b, Nreg) +
              (diag_pre_multiply(tau,L_Omega)*z)';
}
model {
  to_vector(z) ~ std_normal();
  L_Omega ~ lkj_corr_cholesky(2);
  y ~ normal(rows_dot_product(beta[region], x), sigma_y);
}
"
fit15 <- stan_model(model_code=euse_stan15)
stan_in15 <- function(data=rtol2, chains=nchains){
    y <- data$richtol
    x <- as.matrix(cbind(1, data$nuii))
    n <- dim(data)[1]
    k <- dim(x)[2]
```

```
    reg <- as.numeric(ordered(data$city))
    nreg <- max(reg)
    stan_data <- list(N=n, Nreg=nreg, K=k, x=x, y=y, region=reg)
    stan_inits <- list()
    for (i in 1:chains)
        stan_inits[[i]] <- list(mu_b=rnorm(k), tau=runif(k),
                                sigma_y=runif(1))
    stan_pars <- c("beta","mu_b","tau","sigma_y")
    return(list(data=stan_data, inits=stan_inits,
                pars=stan_pars, n.chains=chains))
}

input.to.stan <- stan_in15()
fit2keep <- sampling(fit15, data=input.to.stan$data,
                     init=input.to.stan$inits,
                     pars=input.to.stan$pars,
                     iter=niters,thin=nthin,
                     chains=input.to.stan$n.chains)
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##            mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1,1]  5.37    0.00 0.13  5.11  5.27  5.37  5.46  5.64  2752    1
## beta[1,2]  2.76    0.01 0.53  1.74  2.39  2.75  3.12  3.83  2725    1
## beta[2,1]  5.14    0.00 0.15  4.86  5.05  5.14  5.24  5.42  2568    1
## beta[2,2]  2.59    0.01 0.48  1.67  2.25  2.58  2.92  3.55  2608    1
## beta[3,1]  4.33    0.00 0.14  4.07  4.24  4.33  4.43  4.60  2599    1
## beta[3,2]  4.09    0.01 0.59  2.96  3.68  4.08  4.48  5.29  2605    1
## beta[4,1]  6.16    0.00 0.15  5.87  6.06  6.16  6.25  6.44  2660    1
## beta[4,2]  0.37    0.01 0.33 -0.29  0.15  0.37  0.59  1.02  2559    1
## beta[5,1]  7.03    0.00 0.12  6.80  6.96  7.03  7.11  7.26  2485    1
## beta[5,2] -0.10    0.01 0.33 -0.73 -0.33 -0.10  0.11  0.53  2471    1
## beta[6,1]  6.04    0.00 0.12  5.79  5.95  6.04  6.12  6.28  2501    1
## beta[6,2]  0.84    0.01 0.33  0.20  0.61  0.84  1.06  1.47  2591    1
## beta[7,1]  4.57    0.00 0.13  4.31  4.48  4.57  4.66  4.83  2493    1
## beta[7,2]  2.35    0.01 0.34  1.68  2.13  2.35  2.57  3.04  2419    1
## beta[8,1]  5.55    0.00 0.15  5.26  5.44  5.55  5.65  5.85  2162    1
## beta[8,2]  2.41    0.01 0.42  1.60  2.13  2.41  2.68  3.26  2380    1
## beta[9,1]  4.54    0.00 0.19  4.15  4.41  4.54  4.66  4.93  2201    1
## beta[9,2]  2.45    0.01 0.37  1.72  2.21  2.45  2.70  3.19  2344    1
## mu_b[1]    5.41    0.01 0.35  4.73  5.19  5.42  5.63  6.08  2134    1
## mu_b[2]    1.96    0.01 0.55  0.87  1.62  1.95  2.32  3.09  2430    1
## tau[1]     0.98    0.01 0.28  0.60  0.79  0.93  1.11  1.70  2598    1
## tau[2]     1.53    0.01 0.48  0.88  1.20  1.44  1.75  2.76  2490    1
## sigma_y    0.48    0.00 0.02  0.44  0.46  0.48  0.49  0.52  2813    1
## lp__      53.15    0.10 4.71 43.21 50.11 53.44 56.60 61.57  2353    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:50:10 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
euse_multilevel15 <- rvsims(as.matrix(as.data.frame(
    rstan::extract(fit2keep,
                   pars=c("beta","mu_b","tau","sigma_y")))))
```

Stan model with group-level predictors We now add `temp` and `ag` as group-level predictors (the "best" model `euse.lmer4`). The code below is written in matrix notation. The model and code consider the potential of an interaction term between ag and T:

$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim MVN\left[\begin{pmatrix}a_0 + a_1 \times T_j + a_2 \times Ag_j + a_3 Ag_j \times T_j\\ b_0 + b_1 \times T_j + b_2 \times Ag_j + b_3 Ag_j \times T_j\end{pmatrix}, \Sigma_\beta\right]$$

However, in the matrix notation, the above equation is simply

$$\beta_j = MVN(\mathbf{gr}_{\mathbf{pred}_j} \times \gamma, \Sigma_\beta)$$

where β_j is the jth row of the regression coefficients for region j (there are 9 regions each with 2 regression coefficients, intercept β_0 and slope β_1),

$$\gamma = \begin{pmatrix} a_0 & b_0\\ a_1 & b_1\\ a_2 & b_2\\ a_3 & b_3\end{pmatrix}$$

is the group-level model coefficient matrix, and

$$\mathbf{gr}_{\mathbf{pred}_j} = \begin{pmatrix} 1 & T_1 & Ag_1 & T_1 \times Ag_1\\ \cdots & & & \\ 1 & T_9 & Ag_9 & T_9 \times Ag_9\end{pmatrix}$$

is the design matrix of the group-level predictor values for region j.

```
euse_stan2 <- "
data {
  int<lower=0> N;
  int<lower=0> Nreg;
  int<lower=1> K; // individual predictor
  int<lower=1> J; // group-level predictor
  int<lower=1,upper=Nreg> region[N];
  matrix[N,K] x;
  vector[N] y;
  row_vector[J] gr[Nreg];
}
parameters {
  vector[K] beta[Nreg];
  matrix[J,K] gamma;
  corr_matrix[K] Omega;
  vector<lower=0>[K] tau;
  real<lower=0,upper=10> sigma_y;
}
transformed parameters {
  vector[N] mu_y;
  row_vector[K] u_gamma[Nreg];
  for (j in 1:Nreg)
    u_gamma[j] = gr[j] * gamma;
  for (i in 1:N)
    mu_y[i] = x[i] * beta[region[i]];
```

```
}
model {
  tau ~ cauchy(0,2.5);
  Omega ~ lkj_corr(2);
  beta ~ multi_normal(u_gamma, quad_form_diag(Omega, tau));
  y ~ normal(mu_y, sigma_y);
}
"

fit2 <- stan_model(model_code=euse_stan2)

stan_in2 <- function(data=rtol2, Ag=ag.cat, Temp=AveTemp,
                     y="richtol", x="nuii", regn="city",
                     Log=F, int=F, chains=nchains){
  ## The default is not including the interaction
    if (Log){
        y <- log(data[,y])
        x <- log(data[,x])
    } else {
        y <- data[,y]
        x <- data[,x]
    }
    x <- as.matrix(cbind(1, x))
    n <- dim(data)[1]
    k <- dim(x)[2]
    reg <- as.numeric(ordered(data[,regn]))
    nreg <- max(reg)
    if (int) regPred <- cbind(1, 2*(Ag-0.5), Temp-mean(Temp),
                              2*(Ag-0.5)*Temp-mean(Temp))
    else
        regPred <- cbind(1, 2*(Ag-0.5), Temp-mean(Temp))
    nregpred <- dim(regPred)[2]
    stan_data <- list(N=n, Nreg=nreg, K=k, J=nregpred, x=x, y=y,
                      region=reg, gr = regPred)
    stan_inits <- list()
    for (i in 1:chains)
        stan_inits[[i]] <- list(beta=matrix(rnorm(nreg*k),
                                            ncol=k),
                                gamma=matrix(rnorm(nregpred*k),
                                             ncol=k),
                                tau=runif(k), sigma_y=runif(1))
    stan_pars <- c("beta","gamma","tau","sigma_y")
    return(list(data=stan_data, inits=stan_inits,
                pars=stan_pars, n.chains=chains))
}

input.to.stan <- stan_in2() ## no interaction
fit2keep <- sampling(fit2, data=input.to.stan$data,
                     init=input.to.stan$inits,
                     pars=input.to.stan$pars,
                     iter=niters, thin=nthin,
                     chains=input.to.stan$n.chains,
                     control=list(adapt_delta = 0.95,
```

```
                                    max_treedepth=15))
```

```
## Warning: There were 249 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: The largest R-hat is 1.18, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
print(fit2keep)
```

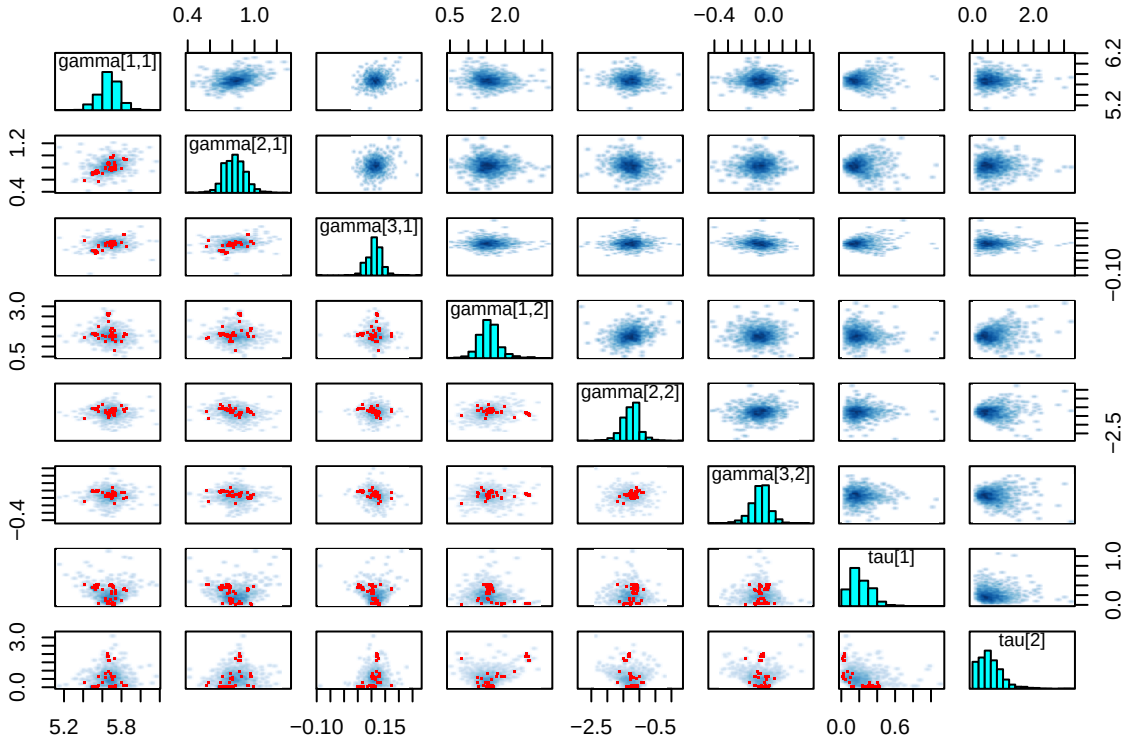
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1,1]   5.35    0.00 0.10  5.14  5.28  5.34  5.41  5.56  1430 1.01
## beta[1,2]   2.80    0.01 0.41  2.03  2.54  2.74  3.06  3.65  1624 1.01
## beta[2,1]   5.18    0.02 0.12  4.96  5.09  5.19  5.26  5.39    40 1.11
## beta[2,2]   2.53    0.01 0.35  1.82  2.31  2.54  2.73  3.25  2148 1.00
## beta[3,1]   4.44    0.02 0.13  4.18  4.35  4.44  4.53  4.68    34 1.14
## beta[3,2]   3.54    0.08 0.59  2.67  3.09  3.51  3.95  4.75    49 1.09
## beta[4,1]   6.15    0.01 0.12  5.93  6.06  6.14  6.23  6.39   493 1.02
## beta[4,2]   0.38    0.02 0.28 -0.18  0.18  0.39  0.59  0.88   266 1.04
## beta[5,1]   7.08    0.03 0.14  6.80  7.00  7.09  7.18  7.34    26 1.18
## beta[5,2]  -0.17    0.07 0.38 -0.90 -0.43 -0.20  0.06  0.59    31 1.16
## beta[6,1]   6.05    0.02 0.12  5.82  5.97  6.04  6.13  6.29    33 1.14
## beta[6,2]   0.73    0.01 0.30  0.17  0.54  0.71  0.93  1.35  1258 1.01
## beta[7,1]   4.55    0.01 0.11  4.34  4.46  4.55  4.62  4.76   133 1.05
## beta[7,2]   2.46    0.02 0.30  1.86  2.24  2.47  2.71  2.99   227 1.03
## beta[8,1]   5.39    0.03 0.16  5.09  5.27  5.38  5.50  5.67    40 1.11
## beta[8,2]   2.84    0.01 0.38  2.12  2.61  2.79  3.08  3.67   657 1.02
## beta[9,1]   4.46    0.00 0.14  4.20  4.38  4.46  4.55  4.74   959 1.01
## beta[9,2]   2.60    0.01 0.27  2.05  2.43  2.62  2.77  3.11  1246 1.01
## gamma[1,1]  5.67    0.02 0.10  5.50  5.61  5.68  5.73  5.88    43 1.11
## gamma[1,2]  1.56    0.01 0.29  1.02  1.39  1.55  1.68  2.25   682 1.01
## gamma[2,1]  0.82    0.01 0.10  0.64  0.75  0.82  0.88  1.02    95 1.06
## gamma[2,2] -1.26    0.01 0.28 -1.86 -1.43 -1.24 -1.08 -0.71   505 1.02
## gamma[3,1]  0.11    0.00 0.03  0.06  0.10  0.11  0.13  0.16    30 1.13
## gamma[3,2] -0.06    0.00 0.07 -0.23 -0.10 -0.06 -0.01  0.07   367 1.02
## tau[1]      0.22    0.01 0.12  0.04  0.13  0.19  0.29  0.48   119 1.05
## tau[2]      0.58    0.05 0.40  0.04  0.30  0.52  0.79  1.61    57 1.09
## sigma_y     0.48    0.00 0.02  0.44  0.46  0.48  0.49  0.52  1309 1.01
## lp__       75.14    0.83 6.60 62.44 70.70 74.86 79.77 87.54    63 1.10
##
```

```
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:53:57 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
pairs(fit2keep, pars=c("gamma","tau"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```




```
euse_multilevel2 <- rvsims(as.matrix(as.data.frame(
    rstan::extract(fit2keep, pars=c("beta","gamma",
                    "tau", "sigma_y")))))
```

We can also use Cholesky decomposition:

```
## Cholesky decomposition
euse_stan3 <- "
data {
  int<lower=0> N;
  int<lower=0> Nreg;
```

```
  int<lower=1> K; // individual predictor
  int<lower=1> J; // group-level predictor
  int<lower=1,upper=Nreg> region[N];
  matrix[N,K] x;
  vector[N] y;
  matrix[Nreg,J] gr;
}
parameters {
  matrix[K,Nreg] z;
  cholesky_factor_corr[K] L_Omega;
  vector<lower=0,upper=pi()/2>[K] tau_unif;
  matrix[J,K] gamma;
  real<lower=0,upper=10> sigma_y;
}
transformed parameters {
  matrix[Nreg,K] beta;
  vector<lower=0>[K] tau;

  for (k in 1:K)
    tau[k] = 2.5*tan(tau_unif[k]);
  beta = gr*gamma + (diag_pre_multiply(tau,L_Omega)*z)';
}
model {
  to_vector(z) ~ std_normal();
  L_Omega ~ lkj_corr_cholesky(2);
  to_vector(gamma) ~ normal(0,5);
  y ~ normal(rows_dot_product(beta[region], x), sigma_y);
}
"
fit3 <- stan_model(model_code=euse_stan3)
stan_in3 <- function(data=rtol2, Ag=ag.cat, Temp=AveTemp,
                     y="richtol", x="nuii", regn="city",
                     Log=F, int=F, chains=nchains){
    if (Log){
        y <- log(data[,y])
        x <- log(data[,x])
    } else {
        y <- data[,y]
        x <- data[,x]
    }
    x <- as.matrix(cbind(1, x))
    n <- dim(data)[1]
    k <- dim(x)[2]
    reg <- as.numeric(ordered(data[,regn]))
    nreg <- max(reg)
    if (int)
        regPred <- cbind(1, 2*(Ag-0.5), Temp-mean(Temp),
                         2*(Ag-0.5)*(Temp-mean(Temp)))
    else regPred <- cbind(1, 2*(Ag-0.5), Temp-mean(Temp))
    nregpred <- dim(regPred)[2]
    stan_data <- list(N=n, Nreg=nreg, K=k, J=nregpred, x=x, y=y,
                      region=reg, gr = regPred)
    stan_inits <- list()
```

```
    for (i in 1:chains)
        stan_inits[[i]] <- list(z=matrix(rnorm(k*nreg), nrow=k),
                                gamma=matrix(rnorm(nregpred*k),
                                             ncol=k),
                                sigma_y=runif(1))
    stan_pars <- c("beta","gamma","tau","sigma_y")
    return(list(data=stan_data, inits=stan_inits,
                pars=stan_pars, n.chains=chains))
}

input.to.stan <- stan_in3()
fit2keep <- sampling(fit3, data=input.to.stan$data,
                     init=input.to.stan$inits,
                     pars=input.to.stan$pars,
                     iter=niters,thin=nthin,
                     chains=input.to.stan$n.chains,
                     control=list(adapt_delta = 0.9))
```

```
## Warning: There were 3 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(fit2keep)
```

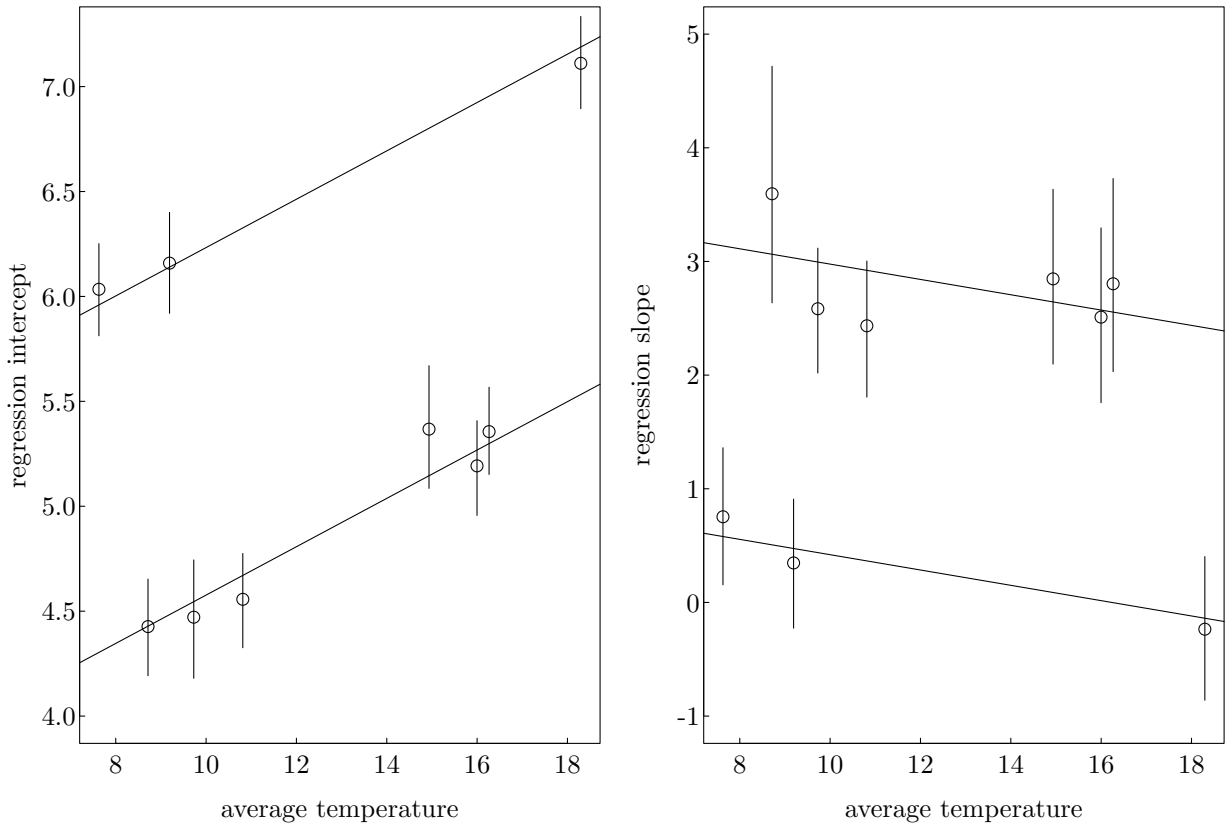
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1,1]   5.36    0.00 0.11  5.15  5.28  5.35  5.43  5.57  2442    1
## beta[1,2]   2.80    0.01 0.44  2.03  2.49  2.77  3.09  3.72  2493    1
## beta[2,1]   5.19    0.00 0.11  4.96  5.12  5.20  5.27  5.41  2490    1
## beta[2,2]   2.51    0.01 0.39  1.76  2.25  2.50  2.76  3.29  2564    1
## beta[3,1]   4.43    0.00 0.12  4.19  4.35  4.43  4.51  4.65  2497    1
## beta[3,2]   3.59    0.01 0.56  2.64  3.18  3.57  3.97  4.73  2433    1
## beta[4,1]   6.16    0.00 0.12  5.92  6.07  6.16  6.24  6.39  2461    1
## beta[4,2]   0.35    0.01 0.28 -0.21  0.16  0.34  0.54  0.91  2478    1
## beta[5,1]   7.11    0.00 0.11  6.89  7.04  7.11  7.18  7.33  2350    1
## beta[5,2]  -0.23    0.01 0.33 -0.86 -0.46 -0.24 -0.02  0.41  2452    1
## beta[6,1]   6.03    0.00 0.11  5.81  5.96  6.03  6.11  6.25  2415    1
## beta[6,2]   0.76    0.01 0.31  0.16  0.55  0.75  0.96  1.37  2200    1
## beta[7,1]   4.56    0.00 0.12  4.33  4.48  4.56  4.64  4.77  2330    1
## beta[7,2]   2.43    0.01 0.31  1.80  2.22  2.44  2.66  3.00  2373    1
## beta[8,1]   5.37    0.00 0.15  5.09  5.26  5.36  5.47  5.67  2307    1
## beta[8,2]   2.85    0.01 0.39  2.10  2.59  2.84  3.10  3.63  2560    1
## beta[9,1]   4.47    0.00 0.14  4.18  4.38  4.48  4.57  4.74  2191    1
## beta[9,2]   2.58    0.01 0.28  2.03  2.38  2.59  2.78  3.12  2240    1
## gamma[1,1]  5.68    0.00 0.10  5.48  5.63  5.68  5.74  5.89  2543    1
## gamma[1,2]  1.54    0.01 0.30  0.95  1.37  1.52  1.70  2.17  2551    1
## gamma[2,1]  0.83    0.00 0.10  0.62  0.77  0.83  0.88  1.03  2408    1
## gamma[2,2] -1.28    0.01 0.31 -1.92 -1.43 -1.26 -1.11 -0.73  1738    1
## gamma[3,1]  0.12    0.00 0.03  0.07  0.10  0.11  0.13  0.16  2410    1
## gamma[3,2] -0.07    0.00 0.08 -0.23 -0.11 -0.07 -0.02  0.08  1696    1
```

```
## tau[1]      0.20    0.00 0.13  0.02  0.12  0.18  0.26  0.51  2203    1
## tau[2]      0.61    0.01 0.37  0.06  0.36  0.56  0.79  1.50  1634    1
## sigma_y     0.48    0.00 0.02  0.44  0.46  0.48  0.49  0.52  2652    1
## lp__       48.02    0.10 4.82 37.58 45.05 48.46 51.41 56.32  2422    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:57:39 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
euse_multilevel3 <- rvsims(as.matrix(as.data.frame(
    rstan::extract(fit2keep, pars=c("beta","gamma",
                   "tau", "sigma_y")))))
```

We don't have warnings for this model. We now use the output to generate the same plot we see before:

```
beta_rv <- summary(rvmatrix(euse_multilevel3[1:18], nrow=9))
gamma_rv <- summary(rvmatrix(euse_multilevel3[19:24], nrow=3))
gr <- input.to.stan$data$gr

par(mfrow=c(1,2), mar=c(3, 3, 1, 0.25), mgp=c(1.5,0.125,0), las=1, tck=0.01)
plot(AveTemp,beta_rv[1:9, 1], xlab="average temperature",
     ylab="regression intercept", ylim=c(4,7.25))
a <- sum(gamma_rv[1:2,1]*c(1,-1)) - gamma_rv[3,1]*mean(AveTemp)
abline(a, gamma_rv[3,1])
a <- sum(gamma_rv[1:2,1]*c(1,1)) - gamma_rv[3,1]*mean(AveTemp)
abline(a, gamma_rv[3,1])
segments(y0=beta_rv[1:9, 4], y1=beta_rv[1:9, 8], x0=AveTemp, x1=AveTemp)

plot(AveTemp,beta_rv[10:18, 1], xlab="average temperature",
     ylab="regression slope", ylim=c(-0.01, 0.05)*100)
a <- sum(gamma_rv[4:5,1]*c(1,-1)) - gamma_rv[6,1]*mean(AveTemp)
abline(a, gamma_rv[6,1])
a <- sum(gamma_rv[4:5,1]*c(1,1)) - gamma_rv[6,1]*mean(AveTemp)
abline(a, gamma_rv[6,1])
segments(y0=beta_rv[10:18, 4], y1=beta_rv[10:18, 8],
         x0=AveTemp, x1=AveTemp)
```



The effort of using Stan (all the above code) is substantially larger than using `lmer`. This is why we often don't use Stan until we are satisfied with the final choice of the model. We then implement the final model in Stan. Let's compare the results from the two methods. The comparison is harder than the comparison of the model without group-level predictors.

```
## fixed effects:
fixef(euse.lmer4) ## 4 coefficients:
```

```
##    (Intercept)           nuii      temp.full nuii:temp.full
##     4.26630050     2.23602590     0.11429150    -0.05870001
```

```
## y = beta0+beta1*nuii + beta2* temp + beta3*(nuii*temp)
##or =(beta0+beta2*temp) + (beta1+beta3*temp)*nuii
## now compared to the Stan model output:

euse_multilevel3[1:18]
```

```
##          name  mean   sd    1%  2.5%   25%   50%    75% 97.5%  99% sims
##  [1] beta.1.1  5.36 0.11  5.10  5.15  5.28  5.35  5.428  5.57 5.60 4000
##  [2] beta.2.1  5.19 0.12  4.90  4.96  5.12  5.20  5.271  5.41 5.44 4000
##  [3] beta.3.1  4.43 0.12  4.15  4.19  4.35  4.43  4.506  4.65 4.70 4000
##  [4] beta.4.1  6.16 0.12  5.88  5.92  6.08  6.16  6.239  6.40 6.46 4000
##  [5] beta.5.1  7.11 0.11  6.85  6.89  7.04  7.11  7.183  7.33 7.37 4000
##  [6] beta.6.1  6.03 0.11  5.76  5.81  5.96  6.04  6.109  6.25 6.30 4000
##  [7] beta.7.1  4.56 0.12  4.29  4.33  4.48  4.56  4.634  4.78 4.81 4000
##  [8] beta.8.1  5.37 0.15  5.05  5.09  5.26  5.37  5.470  5.67 5.72 4000
##  [9] beta.9.1  4.47 0.14  4.13  4.18  4.38  4.47  4.568  4.74 4.80 4000
## [10] beta.1.2  2.80 0.44  1.90  2.03  2.49  2.77  3.091  3.73 3.95 4000
```

```
## [11] beta.2.2  2.51 0.39  1.60  1.76  2.24  2.49  2.762  3.30 3.47 4000
## [12] beta.3.2  3.60 0.56  2.50  2.64  3.18  3.58  3.974  4.72 5.00 4000
## [13] beta.4.2  0.35 0.28 -0.37 -0.23  0.16  0.34  0.533  0.91 1.01 4000
## [14] beta.5.2 -0.24 0.32 -0.97 -0.86 -0.46 -0.24 -0.019  0.40 0.48 4000
## [15] beta.6.2  0.75 0.31  0.06  0.16  0.55  0.74  0.957  1.36 1.45 4000
## [16] beta.7.2  2.43 0.31  1.69  1.81  2.22  2.44  2.658  3.00 3.08 4000
## [17] beta.8.2  2.85 0.39  1.94  2.10  2.59  2.84  3.102  3.63 3.78 4000
## [18] beta.9.2  2.58 0.29  1.91  2.02  2.39  2.59  2.786  3.12 3.25 4000
```

```
## the regional coefficients after Temp and Ag already considered

fixed <-fixef(euse.lmer4)
se.fixed <- se.fixef(euse.lmer4)
rndm <- ranef(euse.lmer4)
se.rndm <- se.ranef(euse.lmer4)
int <- fixed[1]+fixed[3]*AveTemp + rndm$ag.cat2[ag.cat+1,1]
## but the uncertainty cannot easily estimated
se.int <- sqrt(se.fixed[1]^2+se.fixed[3]^2*AveTemp^2 +
               rndm$ag.cat2[ag.cat+1,1]^2)
## assuming independent terms (not correct)

slp <- fixed[2]+fixed[4]*AveTemp + rndm$ag.cat2[ag.cat+1,2]
se.slp <- sqrt(se.fixed[2]^2+se.fixed[4]^2*AveTemp^2 +
               se.rndm$ag.cat2[ag.cat+1,2]^2)

## slopes:
cbind(slp, summary(euse_multilevel3[10:18])$mean)
```

```
##             slp
## ATL  2.53417089  2.8036219
## BIR  2.54986815  2.5100374
## BOS  2.97737651  3.5953999
## DEN  0.44362536  0.3473847
## DFW -0.09080119 -0.2353754
## MGB  0.53537442  0.7536755
## POR  2.85419563  2.4337952
## RAL  2.61224694  2.8467031
## SLC  2.91793519  2.5839397
```

```
cbind(se.slp, summary(euse_multilevel3[10:18])$sd)
```

```
##       se.slp
## ATL 1.550961 0.4400966
## BIR 1.545632 0.3932944
## BOS 1.429955 0.5550352
## DEN 1.437120 0.2820423
## DFW 1.594963 0.3225765
## MGB 1.419521 0.3061054
## POR 1.457101 0.3115872
## RAL 1.525160 0.3885048
## SLC 1.442389 0.2855143
```

```
## intercepts
cbind(int, summary(euse_multilevel3[1:9])$mean)
```

```
##          int
```

```
## ATL 5.302546 5.355899
## BIR 5.271982 5.192500
## BOS 4.439605 4.426577
## DEN 6.139436 6.158621
## DFW 7.179988 7.110970
## MGB 5.960797 6.034130
## POR 4.679443 4.556632
## RAL 5.150528 5.367639
## SLC 4.555340 4.471406
```

```
cbind(se.int, summary(euse_multilevel3[1:9])$sd)
```

```
##       se.int
## ATL 1.210609 0.1074639
## BIR 1.209719 0.1151690
## BOS 1.191017 0.1168709
## DEN 1.191910 0.1213494
## DFW 1.217817 0.1123362
## MGB 1.189156 0.1113299
## POR 1.195301 0.1152235
## RAL 1.206325 0.1501875
## SLC 1.192971 0.1424983
```

```
se.fixef(euse.lmer4)
```

```
##    (Intercept)           nuii      temp.full nuii:temp.full
##     0.85014125     1.37377372     0.01579465     0.04373228
```

```
se.ranef(euse.lmer4)
```

```
## $site
##   (Intercept)       nuii
## 1  0.05727721 0.14176072
## 2  0.05363971 0.13275794
## 3  0.05873822 0.14537672
## 4  0.04426109 0.10954592
## 5  0.05194224 0.12855670
## 6  0.04853583 0.12012587
## 7  0.04822998 0.11936889
## 8  0.04742250 0.11737040
## 9  0.03712977 0.09189593
##
## $ag.cat2
##       (Intercept)      nuii
## FALSE  0.07274118 0.1107782
## TRUE   0.08436722 0.1284836
```