Multilevel Modelling with Rstanarm

This notebook introduces the basic syntax for running multilevel models using MCMC simulation via the Rstanarm package in R.

The syntax should be used in conjuction with the short videos on the course Learn site which will talk you though the syntax and the basics of interpreting the output.

The syntax includes examples of variance compoenent models, random intercept models and random slope models. Each of these models will be estimated for a continuous outcome and a binary outcome. The logic of the syntax presented can be expanded to any outcome variable supported by Rstanarm (notable count models and ordinal outcomes). An example of the syntax for ordinal outputs will be included in the notebook associated with that week of the course. However, those interested in how to estimate models for such outcomes are encouraged to consult the Rstanarm documentation at https://cran.r-project.org/web/packages/rstanarm/rstanarm.pdf

A fuller version of the syntax of models with continuous outcomes can be found at https://mc-stan.org/users/documentation/case-studies/tutorial_rstanarm.html

Two datasets will be used. For the continuous outcome, the "Gcsemv" dataset from the "mlmRev" package. This dataset concerns the academic preformance of 1,905 students from 73 schools in England, in a GCSE science subject. The following variables are included in the dataset:-

```
school School ID - a factor identifying which school the student attended
student Student ID - a factor giving each student an ID within there school
gender - a factor recording the gender of student
written - a continuous variable recording the student's total score on written paper
course - a continuous variable recording the student's total score on coursework paper
```

Full details of this dataset, which is commonly used for comparing multilevel modelling software can be found at http://www.bristol.ac.uk/cmm/learning/mmsoftware/data-rev.html#gcsemv

The dataset is installed using the syntax below.

In [8]:
```
install.packages("mlmRev")
library(mlmRev)
```

```
data(Gcsemv, package = "mlmRev")
summary(Gcsemv)
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' ...
 done
```

```
     school        student      gender      written            course     
 68137  : 104   77      :  14   F:1128   Min.   : 0.60   Min.   :  9.25  
 68411  :  84   83      :  14   M: 777   1st Qu.:37.00   1st Qu.: 62.90  
 68107  :  79   53      :  13            Median :46.00   Median : 75.90  
 68809  :  73   66      :  13            Mean   :46.37   Mean   : 73.39  
 22520  :  65   27      :  12            3rd Qu.:55.00   3rd Qu.: 86.10  
 60457  :  54   110     :  12            Max.   :90.00   Max.   :100.00  
 (Other):1446   (Other):1827             NA's   :202     NA's   :180     
```

The following commands set "Male" to the reference category for the "Gender variable" and create a new dataframe called "GCSE" which includes both the recoded variable relating to gender and excludes the exam score variable, i.e. this session will concentrate on coursework grades.

In [9]:
```
# Make Male the reference category and rename variable
Gcsemv$female <- relevel(Gcsemv$gender, "M")


# Use only total score on coursework paper
GCSE <- subset(x = Gcsemv,
               select = c(school, student, female, course))

# Check structure of data frame
str(GCSE)
```

```
'data.frame':	1905 obs. of  4 variables:
 $ school : Factor w/ 73 levels "20920","22520",..: 1 1 1 1 1 1 1 1 1 2 ...
 $ student: Factor w/ 649 levels "1","2","3","4",..: 16 25 27 31 42 62 101 113 146 1 ...
 $ female : Factor w/ 2 levels "M","F": 1 2 2 2 1 2 2 1 1 2 ...
 $ course : num  NA 71.2 76.8 87.9 44.4 NA 89.8 17.5 32.4 84.2 ...
```

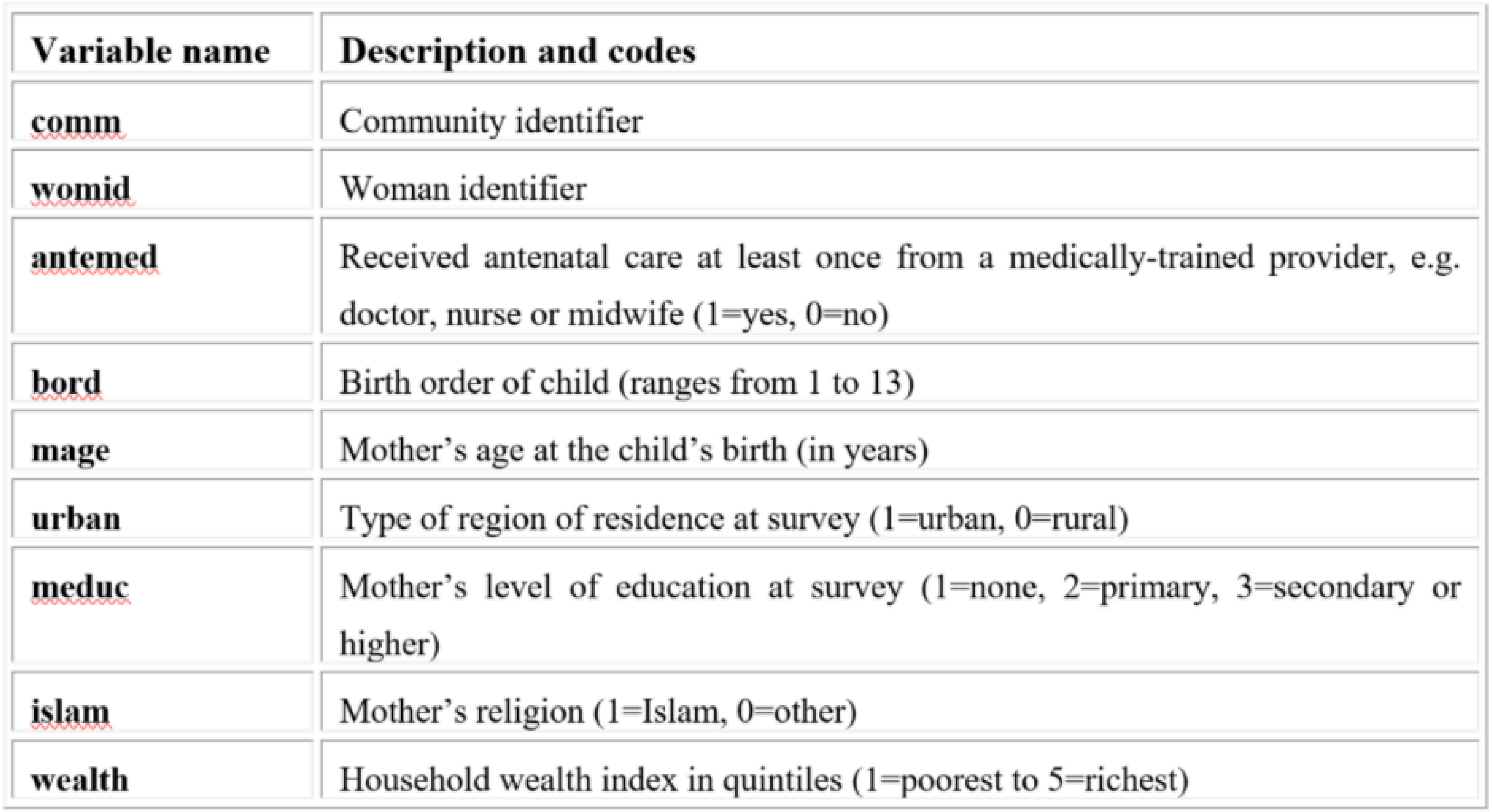
When considering binary outcomes, the Bangladesh Demographic and Health Survey 2004, introduced previously on the course, will be used.

The response variable (antemed) is a binary indicator of whether a woman received antenatal care from a medically-trained provider (a doctor, nurse or midwife) at least once before her most recent live birth.

In this practical, multilevel models are used to explore between-community variance in antenatal care. The data have a two-level hierarchical structure with 5366 women at level 1, nested within 361 communities at level 2. In rural areas a community corresponds to a village, while an urban community is a neighbourhood based on census definitions.

A range of predictorvariables will be considered. At level 1, variables such as a woman's age at the time of the birth and education. Level 2 variables include an indicator of whether the region of residence is classified as urban or rural. Further community-level measures can be derived by aggregating woman-level variables, for example the proportion of respondents in the community who are in the top quintile of a wealth index.

The file contains the following variables:



Variable name	**Description and codes**
comm	Community identifier
womid	Woman identifier
antemed	Received antenatal care at least once from a medically-trained provider, e.g. doctor, nurse or midwife (1=yes, 0=no)
bord	Birth order of child (ranges from 1 to 13)
mage	Mother's age at the child's birth (in years)
urban	Type of region of residence at survey (1=urban, 0=rural)
meduc	Mother's level of education at survey (1=none, 2=primary, 3=secondary or higher)
islam	Mother's religion (1=Islam, 0=other)
wealth	Household wealth index in quintiles (1=poorest to 5=richest)

This dataset is stored in the Stata data file "antenatal.dta". It is loaded using the syntax below, where the last command ensures that the variable "comm", which identifies the community in which each woman lives, is treated as a factor variable

In [10]:
```
require(devtools)
install_version("foreign", version = "0.8-76")
library (foreign)
bang <- read.dta ("antenatal.dta")
bang$comm <- as.factor (bang$comm)
```

```
Downloading package from url: https://cran.r-project.org/src/contrib/Archive/foreign/foreign_0.8-76.tar.gz
```

In [11]:
```
summary (bang) # check the file has laoded correctly by viewing descriptive statistics
```

```
      comm          womid          antemed           bord              mage      
 41     :  25   Min.   :   1   Min.   :0.000   Min.   : 1.000   Min.   :13.00  
 201    :  24   1st Qu.:1342   1st Qu.:0.000   1st Qu.: 1.000   1st Qu.:19.00  
 227    :  24   Median :2684   Median :1.000   Median : 2.000   Median :23.00  
 236    :  24   Mean   :2684   Mean   :0.513   Mean   : 2.868   Mean   :23.63  
 237    :  24   3rd Qu.:4025   3rd Qu.:1.000   3rd Qu.: 4.000   3rd Qu.:28.00  
 344    :  24   Max.   :5366   Max.   :1.000   Max.   :13.000   Max.   :49.00  
 (Other):5221                                                                  
     urban             meduc          islam            wealth     
 Min.   :0.0000   meduc_1:1866   Min.   :0.0000   Min.   :1.000  
 1st Qu.:0.0000   meduc_2:1649   1st Qu.:1.0000   1st Qu.:2.000  
 Median :0.0000   meduc_3:1851   Median :1.0000   Median :3.000  
 Mean   :0.3138                  Mean   :0.9096   Mean   :3.008  
 3rd Qu.:1.0000                  3rd Qu.:1.0000   3rd Qu.:4.000  
 Max.   :1.0000                  Max.   :1.0000   Max.   :5.000  
```

Running Null (or VPC) Models

As shown in previous sessions, the package "lme4" is used to run multilevel models in a traditional frequentist way. If you are using MCMC estimation you do not need to estimate these models as well. However, in this tutorial estimating these models alongside the MCMC version might help in learning how to read the MCMC based output.

The package "rstanarm" is needed to estimate multilevel models using MCMC methods (as implemented by MC-Stan - see https://mc-stan.org/)

The package "Bayesplot" is used to view diagnostic plots associated with MCMC estimation to ensure models have successfully converged. Convergance means that the simulation appears to have run long enough to provide a reliable estimate of the posterior distribution (for example the histogram is smooth like the one shown at the end of the first lecture video this week). Judging convergance is guided by statistical output, but always involves an element of judgement.

In [12]:
```
library(lme4)
library(rstanarm)
library (bayesplot)
```

Estimating a VPC Model for a Continuous Outcome Using LME4

The syntax below should be familar from earlier in the course. It runs a VPC model for the continuous outcome variable "course" where the value of the intercept is allowed to vary between schools. This model serves as a reference model for the MCMC based estimation that follows.

In [13]:
```
gcsenull <- lmer (course~1+(1|school), data=GCSE)
summary (gcsenull)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: course ~ 1 + (1 | school)
   Data: GCSE

REML criterion at convergence: 14103.3

Scaled residuals: 
    Min      1Q  Median      3Q     Max 
-4.9696 -0.5082  0.1111  0.6744  2.7622 

Random effects:
 Groups   Name        Variance Std.Dev.
 school   (Intercept)  76.51    8.747  
 Residual             190.77   13.812  
Number of obs: 1725, groups:  school, 73

Fixed effects:
            Estimate Std. Error t value
(Intercept)   73.722      1.118   65.92
```

Estimating a VPC Model for a Continuous Outcome Using RStanarm

The syntax below creates the same VPC model as above, using the default settings of rstanarm. When considering a continuous outcome, and hence a linear multilevel model, the appropriate command from the rstanarm package is "stan_lmer". The syntax for the "stan_lmer" command mirrors that of the "lmer" command, so the formula for the model is identical to the one used above, but in this case follows the argument "formula =". The "seed" argument is used to define the "random" part of the MCMC chain estimation. It is not needed for estimation, but hopes ensure the replicability of your results if using multiple computers etc.

As with the "lmer" command, we store the resultant model in an object (in this case called "M1_stanlmer") to allow us to view the results and do any post estimation analysis.

In [14]:
```
M1_stanlmer <- stan_lmer(formula = course ~ 1 + (1 | school),
                         data = GCSE,
                         seed = 349)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.000155 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.55 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 6.391 seconds (Warm-up)
Chain 1:                3.328 seconds (Sampling)
Chain 1:                9.719 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.000112 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.12 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 4.284 seconds (Warm-up)
Chain 2:                2.331 seconds (Sampling)
Chain 2:                6.615 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 0.000136 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.36 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3: 
Chain 3: 
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 4.705 seconds (Warm-up)
Chain 3:                3.331 seconds (Sampling)
Chain 3:                8.036 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 0.000119 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.19 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4: 
Chain 4: 
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 3.779 seconds (Warm-up)
Chain 4:                2.479 seconds (Sampling)
Chain 4:                6.258 seconds (Total)
Chain 4: 
```

Before considering the estimates provided by the MCMC simulation, it is important to check the model has estimated "successfully". Looking at the output above reveals the default MCMC estimation settings used by the "stan_lmer" command. The estimations provided are going to be based on 4 MCMC chains. That is to say, 4 seperate MCMC simulations are run, and the results combined together. Running multiple chains helps speed up the simulation process if you are using a modern computer - since each chain be run on a seperate processing core (and hence multiple chains run at the same time). Comparing the reults between chains (see below) can also prove a useful diagnostic tool. Assuming the posterier distributions provided by the different chains are broadly similar, this suggests they have not been influenced by the random parameter value that the estimation began at - indicating the model is likely to have done a good job exploring possible parameter values.

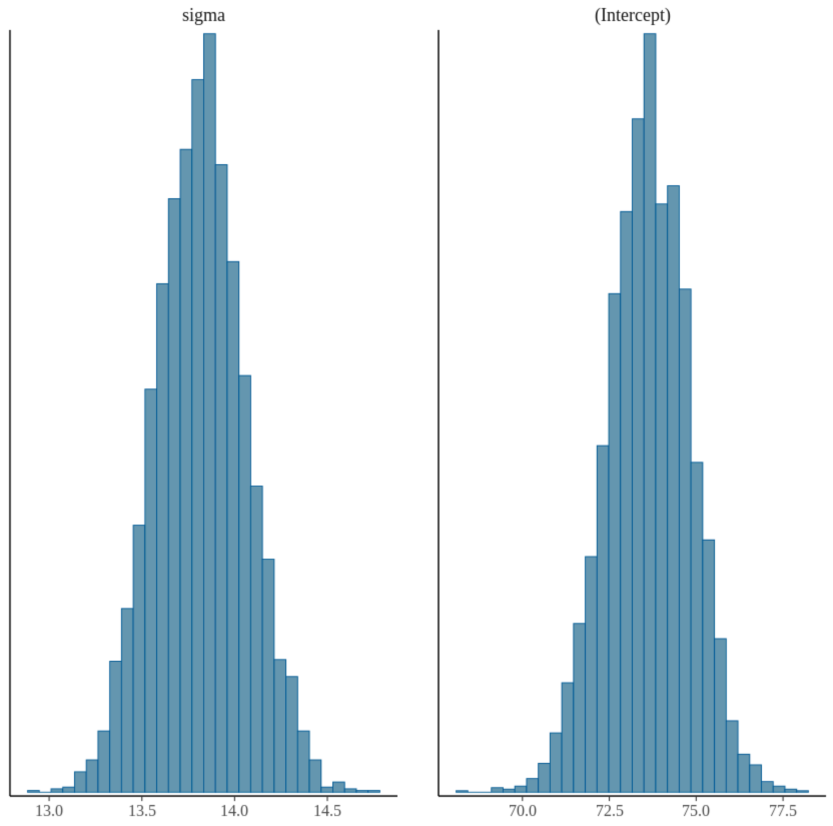
In the above case, each chain has involved 2000 MCMC steps, with the first 1000 steps been thrown away as "warm-up" or "burn-in"(in MCMC estimation, early estimates are typically excluded from the calculation of the posterier distribution since they will reflect the random starting point of the MCMC chain, rather than the "true" value of the parameter under investigation.

The command below provides a histrogram for the MCMC estimates associated with the model parameters "Intercept" and "Sigma" (the name given to the error term). The smoother the shape of the histogram the better the model is likely to have estimated. A smother histrogram can be achieved by increasing the number of MCMC steps used to estimate the posterier distribution.

In this case, the histograms are not unreasonable, although it could be argued that they could be smoother. This is a subjective judgement, and a useful process might be to increase the number of MCMC steps used and see if the histograms show a noteable improvment.

```
In [15]: mcmc_hist (M1_stanlmer,  pars = c("sigma", "(Intercept)"))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

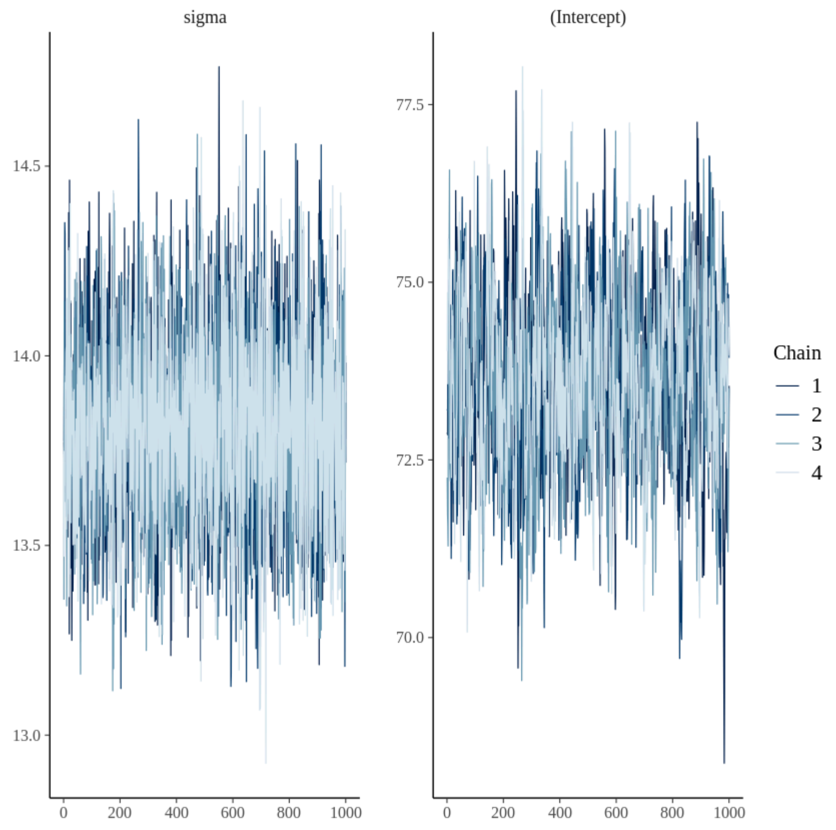

sigma
(Intercept)
13.0
13.5
14.0
14.5
70.0
72.5
75.0
77.5

The "mcmc_trace" command, provided as part of the "bayesplot" package provides a second useful diagnostic plot. A traceplot, involves plotting the values of each MCMC step (on the y-axis) against the count of steps on the x-axis. In essence it provides a record of how the MCMC chain explored the possible parameter values.

Two points of interest exist for these plots. Firstly, the overall plot should have the appearance of a "hairy catterpillar". That is to say that most of estimates should be concentrated around the middle of the resultant distribution, but with examples of outlying high/low values also having been tried. Without these outlying values having been tried it could be argued that the parameter space has not been fully explored. This might be due to auto-correlation (see below), or just the result of the MCMC chain not having been run for sufficent time.

Secondly, you will see that the MCMC estimates for each of the 4 chains that were estimated are plotted on the same graph. In the graphs below the trace plots for each of the 4 chains overlap, suggesting they are not been unduly influenced by their random starting values. If it were possible to distinglish between the values explored by the different chains, this would suggest the chains need to be run for longer, and the warm-up period increased - giving the chain more chance to move away from its starting position.

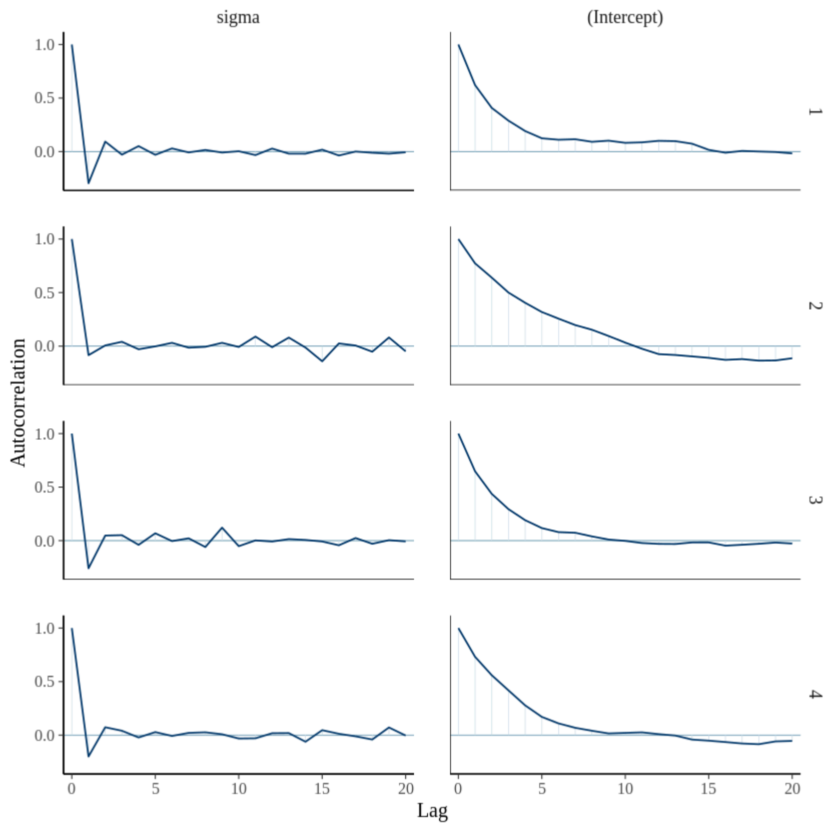
In [16]:
```
mcmc_trace (M1_stanlmer, , pars = c("sigma", "(Intercept)"))
```



The final diagnostic plot to consider is the auto-correlation plot. Recall that when an MCMC chain is estimated the value at one step is related to the value at the previous step. This relationship between the values tested at adjacent steps can create what is known as chain memory. If chains become "stuck" in one area of the parameter space (because the move from one step to the next is not very big) this can prevent the simulation from exploring a full range of possible values - and is known as auto-correlation. You can view auto-correlation plots using the "mcmc_acf" function of the Bayesplot package (shown below)

In [17]:
```
mcmc_acf (M1_stanlmer, , pars = c("sigma", "(Intercept)"))
```

```
Warning message:
“The `facets` argument of `facet_grid()` is deprecated as of ggplot2 2.2.0.
ℹ Please use the `rows` argument instead.
ℹ The deprecated feature was likely used in the bayesplot package.
  Please report the issue at <https://github.com/stan-dev/bayesplot/issues/>.”
```



Consider the auto-correlation plots above (for each variable requested, we get one plot per chain). The chains associated with the "sigma" parameter respresent a good auto-correlation plot. While there is correlation between the values explored at different steps, the correlation soon falls towards zero as the number of steps between the values been compared increases. Constrast this with the plots associated with the "intercept" parameter. In this case, the correlation takes much longer to fall towards zero. Two responses are appropriate to this, firstly protentially running the chains for longer, but secondly since the estimates provided by each step of the chain telling the computer not to keep the values provided by each step of the MCMC chain, but instead only store each x step. This is set using the "thin" argument shown below (in this case storing the value of ever 10th step).

The command below re-estimates the VPC model estimated above, but shows the additional arguments that can be used to control the MCMC estimation process. The "iter" argument defines the total number of steps each MCMC chain should estimate (in this case increasing from the default of 2000 to 5000). The "warmup" argument says how many of those steps (counting from the start of the chain) should be disregarded. In this acse, the first 2000 steps of each chain are now thrown away. This would, if needed, reduce the influence of teh random starting values used by each chain. The "thin" argument means that only every 10th MCMC step has its value recorded, to reduce auto-correlation as described above.

The rest of the command (so the model forumla etc) remain unchanged.

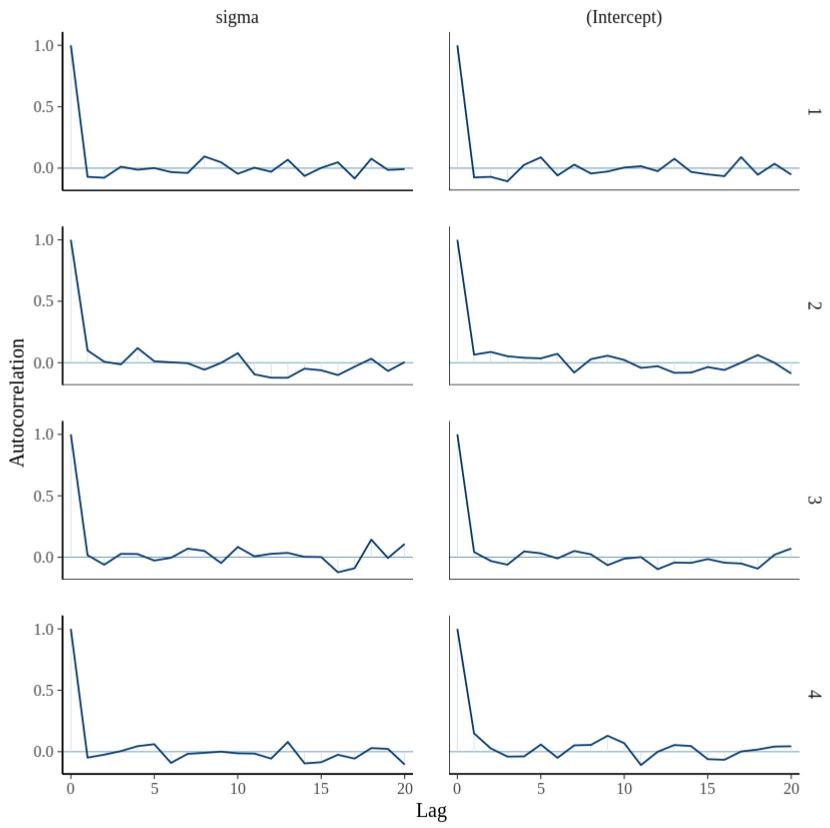
Increasing these values will make each MCMC chain longer, and hence increase the time needed to estimate the model.

In [18]:
```
M1b_stanlmer <- stan_lmer(formula = course ~ 1 + (1 | school),
                          data = GCSE, iter = 5000, warmup=2000, thin=10,
                          seed = 349)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.000144 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.44 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 8.754 seconds (Warm-up)
Chain 1:                9.175 seconds (Sampling)
Chain 1:                17.929 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.000111 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.11 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 6.833 seconds (Warm-up)
Chain 2:                8.635 seconds (Sampling)
Chain 2:                15.468 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 0.000109 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.09 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3: 
Chain 3: 
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 7.47 seconds (Warm-up)
Chain 3:                9.063 seconds (Sampling)
Chain 3:                16.533 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 0.000109 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.09 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4: 
Chain 4: 
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 4: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 5.882 seconds (Warm-up)
Chain 4:                7.263 seconds (Sampling)
Chain 4:                13.145 seconds (Total)
Chain 4: 
```

When we reviewed the diagnostics associated with the first model, the main concern was the auto-correlation associated with the intercept term. It therefore makes sense to recheck this diagnostic to see if our revised setting have improved the situation. Running the command below shows the auto-correlation plots associated with the new model. The plot associated with the intercept term is much improved, likely as a result of the use of thinning.

In [19]:
```
mcmc_acf (M1b_stanlmer, , pars = c("sigma", "(Intercept)"))
```



Having contented ourselves that the MCMC simuation is now likely reliable we can turn our attention to the substantive output of the model. Normally, the results of models estimated with rstanarm can be viewed using the "summary()" command shown below. However, as you will see this provides a large amount of output (including an intercept value for each school). A summary version of the output can be accessed through the "print" command shown below.

In [20]:
```
summary (M1b_stanlmer)
```

```
Model Info:
 function:     stan_lmer
 family:       gaussian [identity]
 formula:      course ~ 1 + (1 | school)
 algorithm:    sampling
 sample:       1200 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 1725
 groups:       school (73)

Estimates:
                                       mean   sd    10%   50%   90%
(Intercept)                           73.7    1.1  72.2  73.6  75.1
b[(Intercept) school:20920]          -10.1    4.7 -16.1 -10.1  -4.2
b[(Intercept) school:22520]          -16.5    2.1 -19.3 -16.5 -13.8
b[(Intercept) school:22710]            8.0    3.3   3.9   8.0  12.2
b[(Intercept) school:22738]           -0.5    4.6  -6.5  -0.5   5.3
b[(Intercept) school:22908]           -7.1    5.0 -13.3  -7.3  -0.5
b[(Intercept) school:23208]            5.6    3.0   1.9   5.6   9.4
b[(Intercept) school:25241]           -1.7    5.4  -8.7  -1.8   5.1
b[(Intercept) school:30474]            8.1    2.4   5.0   8.1  11.1
b[(Intercept) school:35270]          -11.1    4.0 -16.4 -11.1  -6.1
b[(Intercept) school:37224]           -6.8    2.3  -9.7  -6.8  -4.0
b[(Intercept) school:47627]          -10.3    4.7 -16.3 -10.1  -4.4
b[(Intercept) school:50627]          -17.0    3.0 -20.9 -16.9 -13.3
b[(Intercept) school:50631]           14.2    3.0  10.3  14.2  18.1
b[(Intercept) school:60421]            4.7    5.1  -1.8   4.7  10.9
b[(Intercept) school:60427]           12.9    3.0   9.1  12.9  16.5
b[(Intercept) school:60437]           14.1    3.3  10.0  14.1  18.3
b[(Intercept) school:60439]            1.8    3.9  -3.1   1.7   6.6
b[(Intercept) school:60441]           -3.5    3.3  -7.6  -3.5   0.7
b[(Intercept) school:60455]            6.9    2.5   3.8   6.9  10.0
b[(Intercept) school:60457]           11.3    2.1   8.6  11.2  14.1
b[(Intercept) school:60501]            2.7    2.8  -0.9   2.8   6.2
b[(Intercept) school:60729]           11.8    5.0   5.7  11.8  18.3
b[(Intercept) school:60741]           -1.2    2.7  -4.6  -1.2   2.4
b[(Intercept) school:63619]           -3.3    5.2 -10.1  -3.1   3.1
b[(Intercept) school:63833]           -2.6    3.8  -7.6  -2.5   2.3
b[(Intercept) school:64251]           11.8    2.7   8.5  11.8  15.2
b[(Intercept) school:64321]          -10.0    2.7 -13.6 -10.0  -6.6
b[(Intercept) school:64327]            5.1    2.7   1.5   5.1   8.6
b[(Intercept) school:64343]            3.9    2.5   0.7   3.9   7.0
b[(Intercept) school:64359]            6.2    3.5   1.7   6.1  10.6
b[(Intercept) school:64428]           -3.6    5.6 -10.7  -3.5   3.7
```

```
b[(Intercept) school:65385]             7.7    5.4   1.0   7.6  14.3
b[(Intercept) school:66365]             4.8    5.4  -2.2   4.5  11.9
b[(Intercept) school:67051]            -5.9    4.1 -11.2  -5.9  -0.5
b[(Intercept) school:67105]           -10.4    2.6 -13.8 -10.4  -6.9
b[(Intercept) school:67311]            10.3    3.8   5.6  10.3  15.1
b[(Intercept) school:68107]             0.4    1.9  -2.0   0.5   2.9
b[(Intercept) school:68111]             3.7    2.7   0.2   3.7   7.2
b[(Intercept) school:68121]             1.1    3.1  -2.8   1.1   5.1
b[(Intercept) school:68125]             4.2    2.3   1.4   4.2   7.2
b[(Intercept) school:68133]             2.5    2.4  -0.5   2.5   5.7
b[(Intercept) school:68137]           -10.7    1.7 -13.0 -10.7  -8.6
b[(Intercept) school:68201]             3.0    5.5  -3.8   3.1   9.9
b[(Intercept) school:68207]           -13.1    5.1 -19.9 -13.0  -6.7
b[(Intercept) school:68217]             3.2    3.0  -0.6   3.3   7.2
b[(Intercept) school:68227]            -5.1    3.5  -9.7  -5.1  -0.8
b[(Intercept) school:68233]            -9.7    2.5 -12.8  -9.5  -6.6
b[(Intercept) school:68237]             5.5    2.6   2.2   5.5   8.8
b[(Intercept) school:68241]             0.9    3.4  -3.4   0.9   5.1
b[(Intercept) school:68255]            10.3    2.9   6.7  10.3  14.0
b[(Intercept) school:68271]            -2.2    3.7  -6.8  -2.2   2.7
b[(Intercept) school:68303]            -5.6    2.4  -8.6  -5.6  -2.6
b[(Intercept) school:68321]             4.7    2.2   1.8   4.7   7.6
b[(Intercept) school:68329]             6.0    4.9  -0.2   6.0  12.4
b[(Intercept) school:68405]             5.8    2.4   2.6   5.8   8.9
b[(Intercept) school:68411]           -14.0    1.9 -16.4 -13.9 -11.5
b[(Intercept) school:68417]             0.7    2.4  -2.4   0.8   3.6
b[(Intercept) school:68531]             1.4    3.3  -2.8   1.5   5.5
b[(Intercept) school:68611]            -7.2    4.1 -12.3  -7.2  -2.2
b[(Intercept) school:68629]             1.7    2.7  -1.7   1.7   5.2
b[(Intercept) school:68711]           -17.9    4.5 -23.6 -18.1 -12.2
b[(Intercept) school:68723]            -3.5    4.7  -9.8  -3.3   2.6
b[(Intercept) school:68805]             9.5    2.7   6.0   9.5  12.8
b[(Intercept) school:68809]            -2.6    2.0  -5.3  -2.7   0.0
b[(Intercept) school:71927]            -5.7    3.1  -9.7  -5.8  -1.6
b[(Intercept) school:74330]            -7.2    5.5 -14.4  -7.4   0.2
b[(Intercept) school:74862]            -0.2    2.4  -3.3  -0.2   2.8
b[(Intercept) school:74874]            11.5    3.0   7.6  11.7  15.3
b[(Intercept) school:76531]           -11.1    5.4 -18.0 -11.0  -4.2
b[(Intercept) school:76631]             2.5    2.6  -0.9   2.5   5.8
b[(Intercept) school:77207]            -2.0    3.1  -5.9  -2.1   2.0
b[(Intercept) school:84707]             4.3    7.8  -5.2   4.1  14.6
b[(Intercept) school:84772]             9.0    3.3   4.9   8.9  13.3
sigma                                  13.8    0.2  13.5  13.8  14.1
Sigma[school:(Intercept),(Intercept)]  78.8   15.5  59.7  77.0  99.1
```

```
Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 73.4    0.5 72.8  73.4  74.0

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stan
reg')).

MCMC diagnostics
                                      mcse Rhat n_eff
(Intercept)                           0.0  1.0  1106
b[(Intercept) school:20920]           0.1  1.0  1098
b[(Intercept) school:22520]           0.1  1.0  1151
b[(Intercept) school:22710]           0.1  1.0  1201
b[(Intercept) school:22738]           0.1  1.0  1079
b[(Intercept) school:22908]           0.1  1.0  1245
b[(Intercept) school:23208]           0.1  1.0  1178
b[(Intercept) school:25241]           0.2  1.0  1176
b[(Intercept) school:30474]           0.1  1.0  1180
b[(Intercept) school:35270]           0.1  1.0  1148
b[(Intercept) school:37224]           0.1  1.0  1047
b[(Intercept) school:47627]           0.1  1.0  1103
b[(Intercept) school:50627]           0.1  1.0  1219
b[(Intercept) school:50631]           0.1  1.0  1164
b[(Intercept) school:60421]           0.1  1.0  1222
b[(Intercept) school:60427]           0.1  1.0  1187
b[(Intercept) school:60437]           0.1  1.0   946
b[(Intercept) school:60439]           0.1  1.0  1156
b[(Intercept) school:60441]           0.1  1.0  1133
b[(Intercept) school:60455]           0.1  1.0  1084
b[(Intercept) school:60457]           0.1  1.0  1205
b[(Intercept) school:60501]           0.1  1.0  1140
b[(Intercept) school:60729]           0.2  1.0  1104
b[(Intercept) school:60741]           0.1  1.0  1229
b[(Intercept) school:63619]           0.1  1.0  1250
b[(Intercept) school:63833]           0.1  1.0  1218
b[(Intercept) school:64251]           0.1  1.0  1163
b[(Intercept) school:64321]           0.1  1.0  1164
b[(Intercept) school:64327]           0.1  1.0  1199
b[(Intercept) school:64343]           0.1  1.0  1163
b[(Intercept) school:64359]           0.1  1.0  1285
b[(Intercept) school:64428]           0.2  1.0  1272
b[(Intercept) school:65385]           0.2  1.0  1238
b[(Intercept) school:66365]           0.2  1.0  1097
```

```
b[(Intercept) school:67051]          0.1  1.0  1112
b[(Intercept) school:67105]          0.1  1.0  1348
b[(Intercept) school:67311]          0.1  1.0  1143
b[(Intercept) school:68107]          0.1  1.0  1084
b[(Intercept) school:68111]          0.1  1.0  1211
b[(Intercept) school:68121]          0.1  1.0  1112
b[(Intercept) school:68125]          0.1  1.0  1024
b[(Intercept) school:68133]          0.1  1.0  1149
b[(Intercept) school:68137]          0.1  1.0   999
b[(Intercept) school:68201]          0.2  1.0  1167
b[(Intercept) school:68207]          0.1  1.0  1192
b[(Intercept) school:68217]          0.1  1.0  1263
b[(Intercept) school:68227]          0.1  1.0  1200
b[(Intercept) school:68233]          0.1  1.0  1176
b[(Intercept) school:68237]          0.1  1.0  1189
b[(Intercept) school:68241]          0.1  1.0  1209
b[(Intercept) school:68255]          0.1  1.0  1162
b[(Intercept) school:68271]          0.1  1.0  1269
b[(Intercept) school:68303]          0.1  1.0  1177
b[(Intercept) school:68321]          0.1  1.0  1184
b[(Intercept) school:68329]          0.1  1.0  1256
b[(Intercept) school:68405]          0.1  1.0  1128
b[(Intercept) school:68411]          0.1  1.0  1076
b[(Intercept) school:68417]          0.1  1.0  1212
b[(Intercept) school:68531]          0.1  1.0  1329
b[(Intercept) school:68611]          0.1  1.0  1074
b[(Intercept) school:68629]          0.1  1.0  1157
b[(Intercept) school:68711]          0.2  1.0   863
b[(Intercept) school:68723]          0.1  1.0  1207
b[(Intercept) school:68805]          0.1  1.0  1180
b[(Intercept) school:68809]          0.1  1.0  1025
b[(Intercept) school:71927]          0.1  1.0  1081
b[(Intercept) school:74330]          0.2  1.0  1174
b[(Intercept) school:74862]          0.1  1.0  1056
b[(Intercept) school:74874]          0.1  1.0  1102
b[(Intercept) school:76531]          0.2  1.0  1151
b[(Intercept) school:76631]          0.1  1.0  1317
b[(Intercept) school:77207]          0.1  1.0  1135
b[(Intercept) school:84707]          0.2  1.0  1250
b[(Intercept) school:84772]          0.1  1.0  1266
sigma                                0.0  1.0  1214
Sigma[school:(Intercept),(Intercept)] 0.5  1.0  1163
mean_PPD                             0.0  1.0  1206
log-posterior                        0.3  1.0  1142
```

```
For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the poten
tial scale reduction factor on split chains (at convergence Rhat=1).
```

```
In [21]: print(M1b_stanlmer, digits = 2)
```

```
stan_lmer
 family:       gaussian [identity]
 formula:      course ~ 1 + (1 | school)
 observations: 1725
------
            Median MAD_SD
(Intercept) 73.60   1.13

Auxiliary parameter(s):
      Median MAD_SD
sigma 13.82   0.25

Error terms:
 Groups   Name        Std.Dev.
 school   (Intercept)  8.88
 Residual             13.81
Num. levels: school 73

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Estimating a VPC Model for a Binary Outcome Using LME4

Turning to models for a binary output, the command below shows how a VPC model for a binary outcome would be estimated using the "glmer" command provided in the lme4 package. Variation is expected to occur between communities, so the "comm" variable is used to define the random intercept in this case.

Compare this command to the one provided in the next section which estimates the same model using the "stan_glmer" command from the rstanarm package.

You should note that with the exception of needing to specific a link function (in this case a logit transformation), the naming conventions and basic model setup of the MCMC commands (in the rstanarm package) are identical to those used in lme4.

In [22]:
```
binnull <- glmer (antemed~1+(1|comm), data=bang, family=binomial)
summary (binnull)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: antemed ~ 1 + (1 | comm)
   Data: bang

     AIC      BIC   logLik deviance df.resid
  6639.5   6652.7  -3317.8   6635.5     5364

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.7779 -0.7458  0.3423  0.7118  2.6784

Random effects:
 Groups Name        Variance Std.Dev.
 comm   (Intercept) 1.464    1.21
Number of obs: 5366, groups:  comm, 361

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.14809    0.07178   2.063   0.0391 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Estimating a VPC Model for a Binary Outcome Using Rstanarm

The command below estimates the VPC model for the outcome variable "antemed" using MCMC estimation. Typically speaking, models for non-normal outcomes require more robust chain settings than models for continuous outcomes. Therefore, while the model below is estimated using the default MCMC setting, it might well be that the lenght of chains etc, needs to be increased if the model is to be used in practice.

In [23]:
```
BINM1_stanglmer <- stan_glmer(formula = antemed~1+(1|comm), family = binomial(link = "logit"),
                         data = bang,
                         seed = 349)
```

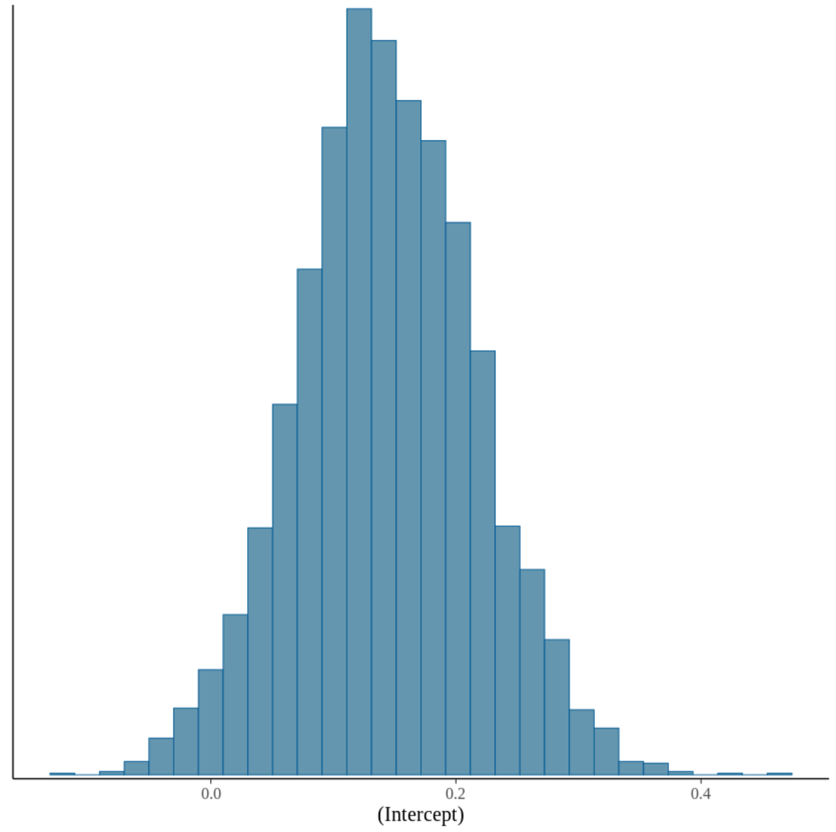
```
SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.00076 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 7.6 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 24.778 seconds (Warm-up)
Chain 1:                20.878 seconds (Sampling)
Chain 1:                45.656 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.000643 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.43 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 25.325 seconds (Warm-up)
Chain 2:                20.868 seconds (Sampling)
Chain 2:                46.193 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000664 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 6.64 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 24.477 seconds (Warm-up)
Chain 3:                20.54 seconds (Sampling)
Chain 3:                45.017 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000637 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 6.37 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 33.162 seconds (Warm-up)
Chain 4:                21.643 seconds (Sampling)
Chain 4:                54.805 seconds (Total)
Chain 4: 
```

The syntax below is used to access the diagnostic plots associated with the logit VPC model estimated above. The output provided is identical to that discussed for the continuous outcome model above. You might like to consider the output and decide whether you think the MCMC settings used were sufficient. You can edit the command above to change the settings if you wish to practice.

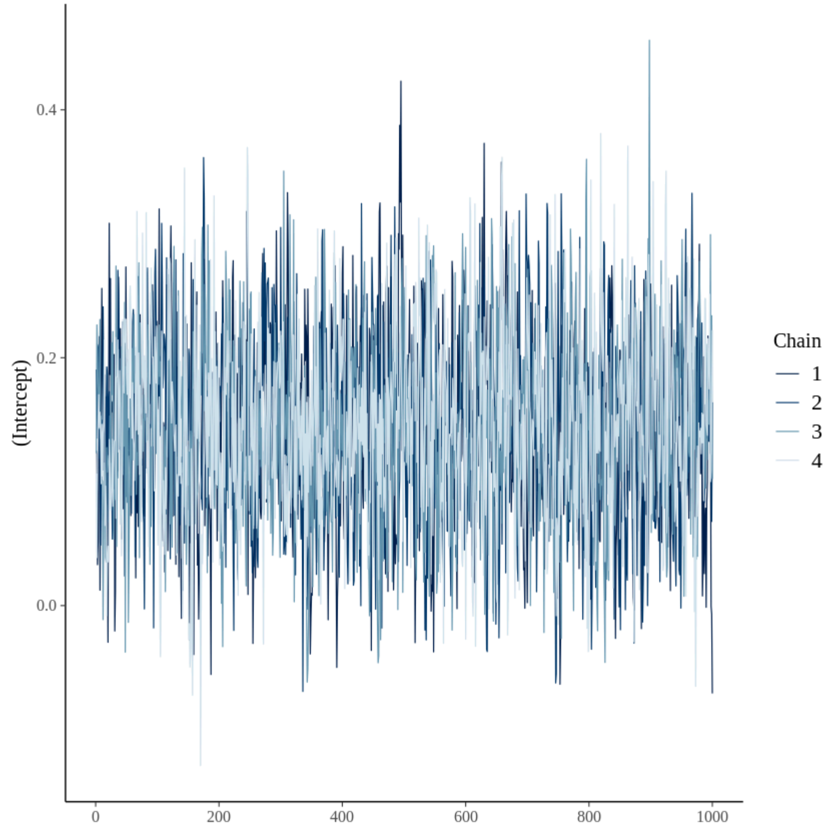
Recall that when estimating a multilevel model for a binary outcome, there is no variable at level 1. This applies whether the model is estimated using lme4 or MCMC techniques. Hence there is no sigma parameter to be considered in daignostic terms.

In [24]:
```
mcmc_hist (BINM1_stanglmer, pars = c("(Intercept)"))
```

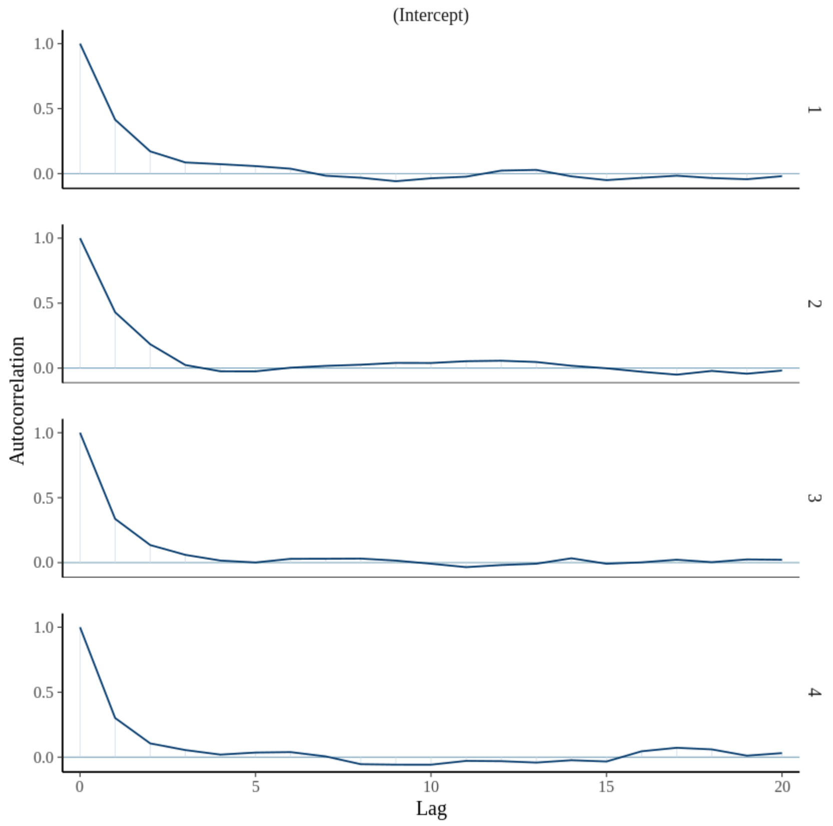
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



In [25]:
```
mcmc_trace (BINM1_stanglmer, pars = c("(Intercept)"))
```



```
In [26]: mcmc_acf (BINM1_stanglmer, pars = c("(Intercept)"))
```



In [27]:
```
print(BINM1_stanglmer, digits = 2)
```

```
stan_glmer
 family:       binomial [logit]
 formula:      antemed ~ 1 + (1 | comm)
 observations: 5366
------
            Median MAD_SD
(Intercept) 0.14   0.07

Error terms:
 Groups Name        Std.Dev.
 comm   (Intercept) 1.23
Num. levels: comm 361

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Estimating a Random Intercept Model for a Continuous Outcome

The remainder of this notebook shows the required syntax to estimate random intercept and random slope models using MCMC simulation through the rstanarm package. In each case the equivilent lme4 syntax is shown first. Comparing these syntax should help to illutrate the similarity between the two packages, and help reassure you that estimating models through rstanarm is not too difficult if required.

However, it should be noted that the MCMC based models (particularly for binary outcomes) can take a long time to estimate. You can refer to the videos on Learn if you wish to hear more about these models.

Since we established that the VPC model for our continuous outcome required MCMC settings in excess of those used by rstanarm by default, the commands for the random intercept and random slope versions of the model use the settings we used for the VPC model above as their starting point.

Having run an MCMC model, you might want to practice accessing and interpreting diagnostic output by replicating the plot commands we used above for the new model.

A reminder that moving from a VPC model to a random intercept model is simply a case of adding the names of the explanatory variables to the model formula, i.e, the following command is the basic syntax needed to run an MCMC simulation for the continuous outcome "course" as a function of the explanatory variable "female" with random intercepts at school level.

M2_stanlmer <- stan_lmer(formula = course ~ female + (1 | school), data = GCSE)

In [33]:
```
gcseri <- lmer (course~female+(1|school), data=GCSE)
summary (gcseri)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: course ~ female + (1 | school)
   Data: GCSE

REML criterion at convergence: 14006.3

Scaled residuals: 
    Min      1Q  Median      3Q     Max 
-4.7797 -0.5394  0.1266  0.6798  2.6752 

Random effects:
 Groups   Name        Variance Std.Dev.
 school   (Intercept)  77.93    8.828  
 Residual             180.07   13.419  
Number of obs: 1725, groups:  school, 73

Fixed effects:
            Estimate Std. Error t value
(Intercept)  69.7282     1.1921  58.490
femaleF       6.7394     0.6782   9.937

Correlation of Fixed Effects:
        (Intr)
femaleF -0.336
```

In [34]:
```
M2_stanlmer <- stan_lmer(formula = course ~ female + (1 | school), data = GCSE, iter = 5000, warmup=2000, thin=10, seed = 349)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.000171 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.71 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 7.582 seconds (Warm-up)
Chain 1:                10.69 seconds (Sampling)
Chain 1:                18.272 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.000159 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.59 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 8.52 seconds (Warm-up)
Chain 2:                7.489 seconds (Sampling)
Chain 2:                16.009 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 0.000118 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.18 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3: 
Chain 3: 
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 10.299 seconds (Warm-up)
Chain 3:                6.397 seconds (Sampling)
Chain 3:                16.696 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 0.00013 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.3 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4: 
Chain 4: 
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 4: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 7.899 seconds (Warm-up)
Chain 4:                8.59 seconds (Sampling)
Chain 4:                16.489 seconds (Total)
Chain 4: 
```

In [35]:
```
print(M2_stanlmer, digits = 2)
```

```
stan_lmer
 family:       gaussian [identity]
 formula:      course ~ female + (1 | school)
 observations: 1725
------
            Median MAD_SD
(Intercept) 69.76   1.19 
femaleF      6.74   0.64 

Auxiliary parameter(s):
      Median MAD_SD
sigma 13.42   0.24 

Error terms:
 Groups   Name        Std.Dev.
 school   (Intercept)  8.97   
 Residual             13.42   
Num. levels: school 73 

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Estimating a Random Intercept Model for a Binary Outcome

The syntax in this section expands the previously estimated logit VPC model to include the effect of a women's religion (recorded by the variable "islam"). As with the continuous example above, this is achieved by simply adding the explanatory variable to the fixed part of the regression formula.

Again, it is likely that the default setting for rstanarm might be too low to ensure a reliable simulation. You might want to produce and interpret the associated diagnostics and reflect on whether you would change the model settings if doing a piece of substantive research.

Since random intercept and random slope models are more complex, the model settings are increased from the defaults used by rstanarm.

In [36]:
```
ri1 <- glmer (antemed~ 1+islam + (1|comm), data=bang, family=binomial)
summary (ri1)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: antemed ~ 1 + islam + (1 | comm)
   Data: bang

     AIC      BIC   logLik deviance df.resid 
  6639.0   6658.8  -3316.5   6633.0     5363 

Scaled residuals: 
    Min      1Q  Median      3Q     Max 
-2.7626 -0.7462  0.3451  0.7142  2.6776 

Random effects:
 Groups Name        Variance Std.Dev.
 comm   (Intercept) 1.447    1.203   
Number of obs: 5366, groups:  comm, 361

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)  
(Intercept)   0.3558     0.1489   2.389   0.0169 *
islam        -0.2296     0.1442  -1.592   0.1114  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
      (Intr)
islam -0.877
```

In [37]:
```
BINM2_stanglmer <- stan_glmer(formula = antemed~islam+(1|comm), family = binomial(link = "logit"),
                         data = bang, iter = 5000, warmup=2000, thin=10,
                         seed = 349)
```

```
SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.000835 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 8.35 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 58.382 seconds (Warm-up)
Chain 1:                64.298 seconds (Sampling)
Chain 1:                122.68 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.000657 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.57 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 59.423 seconds (Warm-up)
Chain 2:                65.055 seconds (Sampling)
Chain 2:                124.478 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 0.000808 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 8.08 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3: 
Chain 3: 
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 57.033 seconds (Warm-up)
Chain 3:                66.037 seconds (Sampling)
Chain 3:                123.07 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 0.001065 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10.65 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4: 
Chain 4: 
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 4: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 60.225 seconds (Warm-up)
Chain 4:                65.679 seconds (Sampling)
Chain 4:                125.904 seconds (Total)
Chain 4: 
```

In [38]:
```
print(BINM2_stanglmer, digits = 2)
```

```
stan_glmer
 family:       binomial [logit]
 formula:      antemed ~ islam + (1 | comm)
 observations: 5366
------
            Median MAD_SD
(Intercept)  0.35   0.15 
islam       -0.23   0.15 

Error terms:
 Groups Name        Std.Dev.
 comm   (Intercept) 1.23    
Num. levels: comm 361 

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Estimating a Random Slope Model for a Continuous Outcome

Finally, the syntax for random slope models is presented below. As a reminder, random slope models allow the impact of explanatory variables to vary between clusters of cases. Hence in the lmer command below the effect of a student's gender is allowed to vary between schools. This is achieved by adding that variable to the random part of the model,

+(1 + female|school)

Again, comparing that syntax to the "stan_lmer" command given below shows that exactly the same change is needed, with the random part of the model formula updated to include both the intercept and teh explanatory variable.

In [39]:
```
gcseri <- lmer (course~female+(1 + female|school), data=GCSE)
summary (gcseri)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: course ~ female + (1 + female | school)
   Data: GCSE

REML criterion at convergence: 13967.3

Scaled residuals: 
    Min      1Q  Median      3Q     Max 
-4.6938 -0.5233  0.1246  0.6522  2.6804 

Random effects:
 Groups   Name        Variance Std.Dev. Corr 
 school   (Intercept) 104.90   10.242        
          femaleF      49.32    7.023   -0.52
 Residual             169.79   13.030        
Number of obs: 1725, groups:  school, 73

Fixed effects:
            Estimate Std. Error t value
(Intercept)   69.420      1.363  50.937
femaleF        7.133      1.141   6.252

Correlation of Fixed Effects:
        (Intr)
femaleF -0.575
```

In [40]:
```
M3_stanlmer <- stan_lmer(formula = course ~ female + (1 + female | school),
                            data = GCSE, iter = 5000, warmup=2000, thin=10, seed = 349)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.000403 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.03 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 56.769 seconds (Warm-up)
Chain 1:                64.289 seconds (Sampling)
Chain 1:                121.058 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.000342 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.42 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 51.91 seconds (Warm-up)
Chain 2:                63.717 seconds (Sampling)
Chain 2:                115.627 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 0.000383 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 3.83 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3: 
Chain 3: 
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 56.757 seconds (Warm-up)
Chain 3:                64.444 seconds (Sampling)
Chain 3:                121.201 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 0.000336 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 3.36 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4: 
Chain 4: 
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 4: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 52.27 seconds (Warm-up)
Chain 4:                63.408 seconds (Sampling)
Chain 4:                115.678 seconds (Total)
Chain 4: 
```

In [41]:
```
print(M3_stanlmer, digits = 2)
```

```
stan_lmer
 family:       gaussian [identity]
 formula:      course ~ female + (1 + female | school)
 observations: 1725
------
            Median MAD_SD
(Intercept) 69.47   1.34 
femaleF      7.05   1.13 

Auxiliary parameter(s):
      Median MAD_SD
sigma 13.03   0.21 

Error terms:
 Groups   Name        Std.Dev. Corr 
 school   (Intercept) 10.32         
          femaleF      7.12    -0.48
 Residual             13.04         
Num. levels: school 73 

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Estimating a Random Slope Model for a Binary Outcome

The logic for adding a random slope also applies to the the stan_glmer command as demonstrated below.

Please note the MCMC based model is very slow, taking around 1 hour to estimate on Noteable. It is also possible that running this model will produce various error messages around low effective sample sizes. These errors are indicative of a model that should have its chains run for longer to ensure the results are robust. The syntax etc is discussed in the accompanying video if you would prefer.

In [42]:
```
ri2 <- glmer (antemed~ 1+islam + (1+islam|comm), data=bang, family=binomial)
summary (ri2)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: antemed ~ 1 + islam + (1 + islam | comm)
   Data: bang

     AIC      BIC   logLik deviance df.resid 
  6641.2   6674.1  -3315.6   6631.2     5361 

Scaled residuals: 
    Min      1Q  Median      3Q     Max 
-2.7383 -0.7457  0.3368  0.7160  2.6534 

Random effects:
 Groups Name        Variance Std.Dev. Corr 
 comm   (Intercept) 2.2317   1.4939        
        islam       0.2466   0.4966   -0.73
Number of obs: 5366, groups:  comm, 361

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)  
(Intercept)   0.3468     0.1717    2.02   0.0434 *
islam        -0.2218     0.1631   -1.36   0.1739  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
      (Intr)
islam -0.910
```

In [43]:
```
BINM3_stanglmer <- stan_glmer(formula = antemed~islam+(1+islam|comm), family = binomial(link = "logit"),
                         data = bang, iter = 5000, warmup=2000, thin=10,
                         seed = 349)
```

```
SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 0.001652 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 16.52 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1: 
Chain 1: 
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1: 
Chain 1:  Elapsed Time: 350.256 seconds (Warm-up)
Chain 1:                263.228 seconds (Sampling)
Chain 1:                613.484 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 0.001496 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 14.96 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2: 
Chain 2: 
Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 339.18 seconds (Warm-up)
Chain 2:                549.224 seconds (Sampling)
Chain 2:                888.404 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.001456 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 14.56 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 387.004 seconds (Warm-up)
Chain 3:                558.664 seconds (Sampling)
Chain 3:                945.668 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.00141 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 14.1 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 4: Iteration: 2001 / 5000 [ 40%]  (Sampling)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 402.604 seconds (Warm-up)
Chain 4:                557.3 seconds (Sampling)
Chain 4:                959.904 seconds (Total)
Chain 4: 
```

```
Warning message:
“There were 19 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 message:
“Examine the pairs() plot to diagnose sampling problems
”
Warning message:
“Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be unreliable.
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess”
Warning message:
“Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quantiles may be unreliable.
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess”
```

In [44]:
```
print(BINM3_stanglmer, digits = 2)
```

```
stan_glmer
 family:       binomial [logit]
 formula:      antemed ~ islam + (1 + islam | comm)
 observations: 5366
------
            Median MAD_SD
(Intercept)  0.36   0.17 
islam       -0.22   0.17 

Error terms:
 Groups Name        Std.Dev. Corr 
 comm   (Intercept) 1.461         
        islam       0.618    -0.58
Num. levels: comm 361 

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

```
In [ ]:
```