### 

Introduction to Bayes Regression

Solution to Exercises  
Full resource: https://www.ncrm.ac.uk/resources/online/all/?id=20843

Before you run the analyses

Before you can run the exercises, you will need to download and install the package “rethinking” and other packages related to “Stan”, as well as the “ggplot2” package. These also require the installation of Tool Chain ++. See this webpage for instructions:

<https://github.com/rmcelreath/rethinking#installation>

The installation of these packages may not always be straightforward. In my experience, if you are using Windows 10, one of the problems is the fact that Windows may automatically install R packages in a “One Drive” directory or in a “Temporary” directory. This means that R may not be able to find the installed packages in the *default* directory. If this is the case, either instructing R where to install a package, or copying the package across in the default directory, usually solves the problem.

While not necessary, using some R interface, e.g. *RStudio*, can greatly help familiarise and use R.

Attached with the online material, you will also find an R script with the solutions that I will describe in what follows.

1.Build a linear regression model where math scores in grade 12 are a function of math scores in grade 8.

1a. Consider the metric of outcome and predictors: would it be sensible to centre and/or standardise some of these variables?

The variables are already standardised with reference to population norms where the average score is 50. However, in order to provide a more intuitive metric to the predictor, I standardised its scores subtracting 50 from the mean and dividing by 10. This effectively means that the scores are standardised referring to the expected “normative” mean and SD, ensuring that individuals that have the normative mean have score 0, while a score=1 indicates a score that is 1 SD-unit above the normative mean. As you will see in reply to the next question, this makes it easier to specify priors.

The script to transform the scores is this:

d$smath8<-(d$math8-50)/10

In the reminder of these exercises I will thus work with variables “math12” and “smath8”, indicating standardised grade 12 math scores and standardised grade 8 math scores, respectively.

There are other ways in which the scores could have been centred or standardised, the key is that transformation can help handling the variables in the model, as shown in what follows.

1b. Create and check priors for the parameters in the model (intercept, slopes, SD).

The model with which I will work can be initially described like this:

*= a + b*

*a*

*b*

σ

The previous statements assume that ***math12*** scores are normally distributed around a mean μ (or “mu”) and a standard deviation (SD) σ (or “sigma”). This seems a fair assumption considering most scores of pupils will cluster around the average and extremely good or extremely poor scores are unlikely.

The average score is a linear function of other parameters. Parameter “a” is the intercept, the average score when the predictor ***smath8*** is at its average value, i.e. equal to 0 (see the standardisation in the previous point 1a). Parameter “b” is the slope, the expected change in m12 for a 1-SD increase in ***smath8*** (see standardisation in previous point 1a). For the time being, I will assume these parameters are normally distributed, and I will specify the average and SD of these distributions in what follows.

Parameter σ is the SD of the ***math12*** scores. SD cannot be negative, so the assumptions should take this into account. A simple assumption would have been to consider values from 0 to an upper bound equally probable, thus using a “uniform” distribution. However, I can safely assume that extreme values of SD are increasingly unlikely. I also know that the “normative” SD of the math scores is 10, so I would expect a mean value of the SD to be close to 10. A log-normal distribution can ensure that the distribution is positive and that large values become increasingly unlikely. Therefore I used a log-normal distribution such as this

*σ*

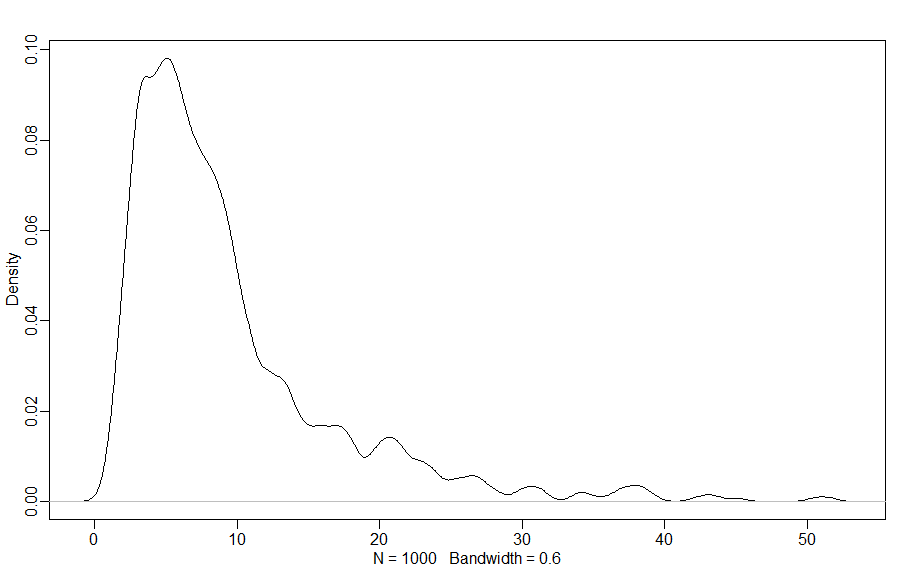
Which just means that the logarithm of *σ* is expected to be normally distributed with mean=2 and SD=0.7.

To visualise a random sample of this distribution, I used this script:

example.sigma<-rlnorm(1000, 2, .7)

dens(example.sigma)

which provides a similar representation:



As for the prior of parameter “a”, since the ***math12*** scores are standardised to expect them to have mean=50 and SD=10, it makes sense to assume the distribution of average ***math12*** scores when **smath8** is average will have similar values. This seems a fair assumption, however, it is always useful to visualise and check the assumptions. The script below creates 1000 random values from a normal distribution with mean=50 and SD=10 and from an log-normal distribution with mean=2 and SD=.7, and then uses these to create 1000 random samples from a normal distribution with these parameters.

sample\_mu<-rnorm(1000, 0, 1)

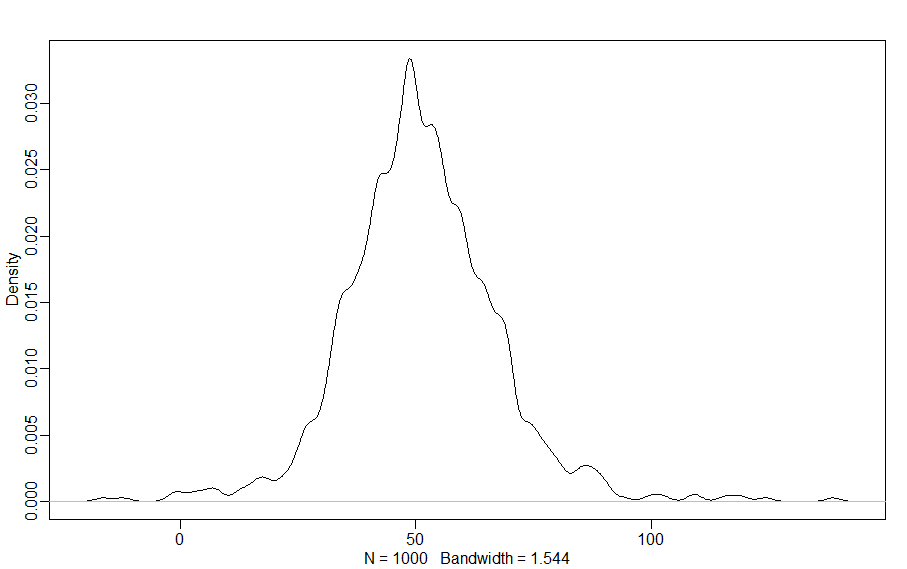
sample\_sigma<-rexp(1000, 1)

prior\_m1<-rnorm(1000, sample\_mu, sample\_sigma)

The command below plots this distribution:

dens(prior\_m1)

creating a similar graph:



This command:

quantile(prior\_m1, c(0.1, 0.90))

returns the values of the distribution corresponding to the 10th and 90th centile:

> quantile(prior\_m1, c(0.1, 0.90))

10% 90%

33.61550 69.43051

the results indicate that 80% of the expected average scores of ***math12*** when ***smath8***=0 (i.e. ***smath8*** is equal to the average) are spread between 33.62 and 69.43. It is overall a very large spread for average values, indeed some of the possible scores are even negative!

I might have a more sensible prior if the SD of the expected average scores is narrower, say SD=5. This creates a narrower range of expected scores, see the 10th and 90th centiles, but it still quite broad.

> quantile(prior\_m1, c(0.1, 0.90))

10% 90%

37.68915 64.91960

Next, I may start assuming the slope has mean=0 and SD that is equal to the normative SD in the outcome ***math12***, i.e. SD=10. By assuming the average of the slope is 0, I am taking a “sceptical” view of the possible association between math scores: I am assuming that the association may be negative or positive, or may be null. If I knew more about this phenomenon, I could provide more specific assumptions.

An important issue to remember when using a ***broad prior*** like this one: it is recommended that sensitivity analyses are run to test if other broad priors produce similar results, see recommendations by Kruschke here: <https://www.nature.com/articles/s41562-021-01177-7>

A SD =10 for the slope would mean that 95% of the slopes will predict a rate of change in ***math12*** scores associated with 1SD increase in ***math8*** that varies between -20 and +20, which are quite large changes. To visualise these, you can use this script:

N<-100

a<-rnorm(N, 50, .7 )

b<-rnorm(N, 0, 10)

#Create an empty plot

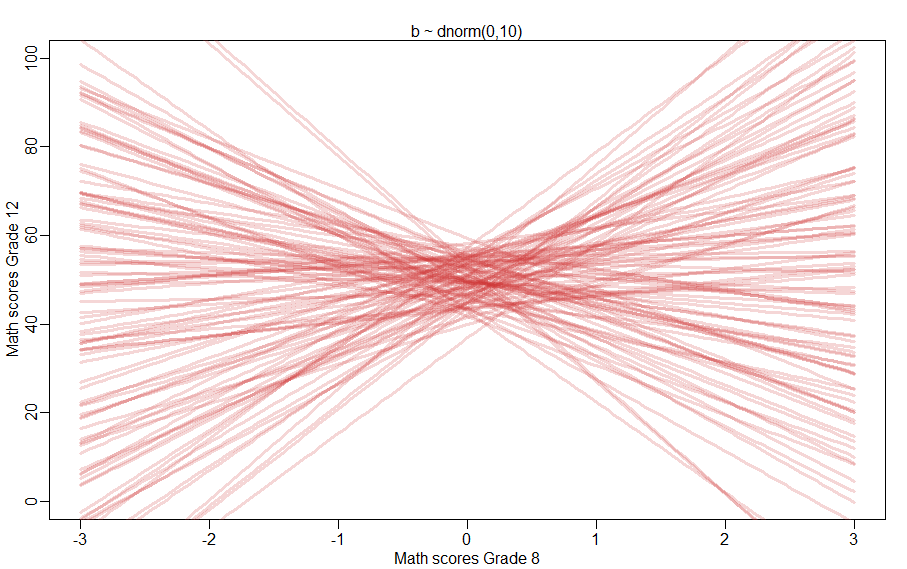
plot(NULL, xlim=c(-3,3), ylim=c(0,100), xlab="Math scores Grade 8", ylab="Math scores Grade 12")

mtext("b ~ dnorm(0,1)")

#the function below creates and plots lines using the linear model

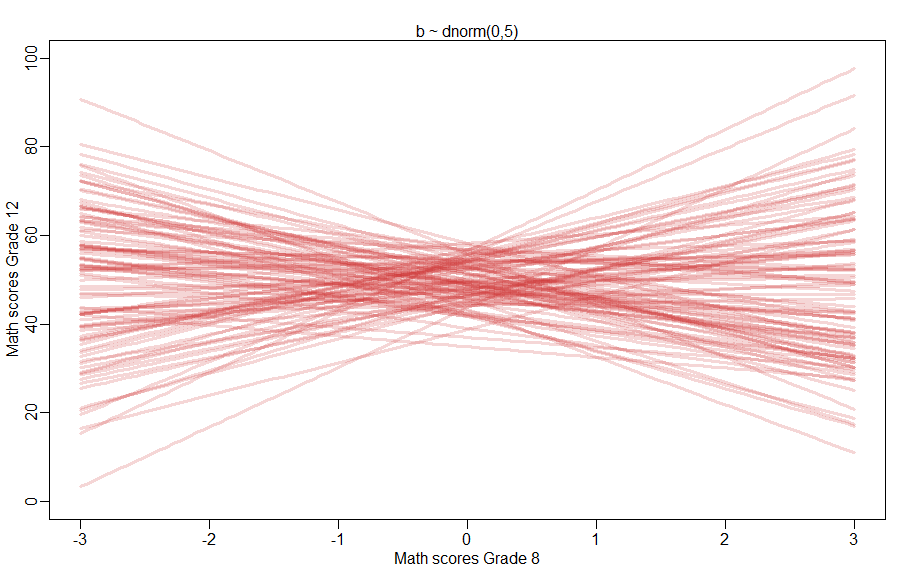
for (i in 1:N) curve(a[i] + b[i]\*(x), from=-3, to=3 add=TRUE, col=col.alpha("brown3",0.2), lwd=3)

The script creates 100 lines from random normal distributions of “a” and “b”, and plot these lines assuming the predictor takes the values on the “x” axis. The graph will look like this:



This shows very unplausible extreme values and extreme associations.

A more sensible prior may assume that the SD of the slope is 5, i.e. 95% of the slopes are expected to indicate changes in in ***math12*** scores for a 1SD increase in ***smath8*** that range between -10 and +10. This is still a broad, “sceptical” assumption, but it seems to produce more sensible expectations:



So, the final model I will use to analyse the data, and that I would report in the Methods section is the following:

*= a + b*

*a*

*b*

σ

Run the linear regression model specified above using the function “ulam”. Before running it, make sure that you use an ad-hoc list with the variables of interest. Run the model specifying only 1 chain. Inspect the output and consider algorithm diagnostics.

When using “ulam” it is advisable to ensure that only the variables in the model are in the dataset, thus avoiding potential problems. To do so, you can use the command:

dl <- list(

math12 = d$math12,

smath8 = d$smath8 )

which creates a list called “dl” with the two variables in the model, those created standardising the original variables in dataset “d”.

The command to run the model specified before is this:

m1<- ulam(

alist(

math12 ~ dnorm( mu, sigma) ,

mu<- a + b\*(smath8),

a ~ dnorm(50, 5),

b ~ dnorm(0, 5),

sigma ~ dlnorm(2, .7)

), data=dl, chains=1)

I have called this first model “m1”.

The command “ulam” allows to run a Hamilton Monte Carlo (HMC) approximation. This is an efficient algorithm. For more information about “ulam” command, see here:

<https://rdrr.io/github/rmcelreath/rethinking/man/ulam.html>

The information after “alist” specifies the likelihood function of the data, the linear model, and the priors for parameters “a”, “b” and “sigma”. The other options specify the dataset to be used and the number of chains, in this case 1.

The command “show” allows to get information about the running of the algorithm. This confirms that algorithm run is the Hamilton Monte Carlo approximation. It provides also information about the samples used and the process that followed.

The command “precis” from package “rethinking” provides summary information from the posterior distribution, but also some diagnostic information:

> precis(m1)

mean sd 5.5% 94.5% rhat ess\_bulk

a 49.52 0.18 49.22 49.80 1 506.96

b 8.20 0.17 7.93 8.49 1 434.90

sigma 5.41 0.11 5.23 5.58 1 543.18

R-hat is a convergence diagnostic. It compares the between- and with-chain estimates for model parameters, so if they agree, the ratio is 1. R-hat larger than one indicates issues with convergence, and the recommendation is that results should be used only if R-Hat < 1.05. In this case, there seems there were no problems with convergence.

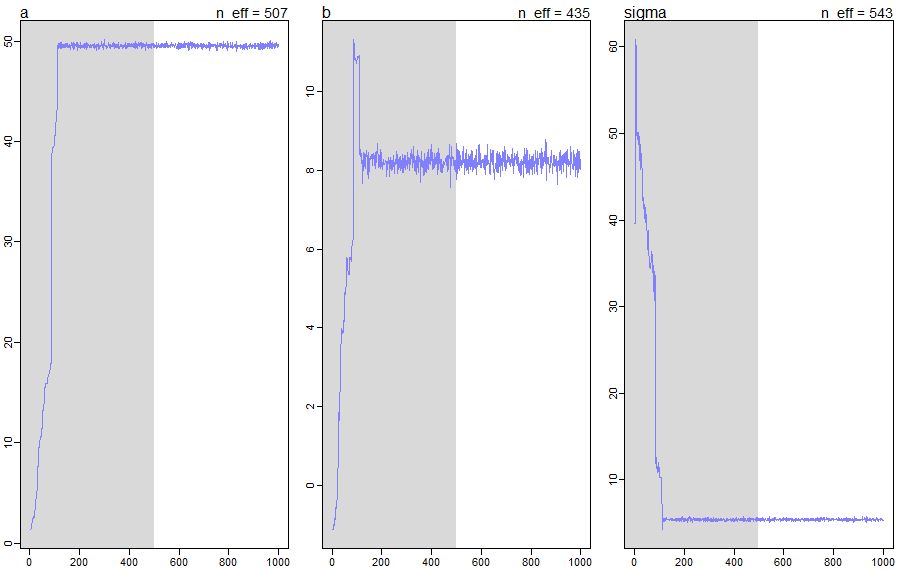
The “ess\_bulk” output is the Bulk Effective Sample Size. It is an index of sampling efficiency in the estimation. The recommendation is that the values of this index should be at least 100 per Markov Chain in order to be reliable. Since I only had a chain, this also seems to indicate reliable results.

See this page:

<https://mc-stan.org/rstan/reference/Rhat.html>

for more information about the diagnostic statistics.

Another command that can be useful is “traceplot”: This draws a traceplot corresponding the samples of the Markov chains in sequential order. In this way, this provides an idea about the sampling behaviour and convergence to stable values. The grey areas represent the initial “adaptation” samples, whereby the chain is learning to sample from the posterior in more efficient ways. This means these samples are not reliable to use for inference and are discarded by commands like “extact.samples” which return only the samples the follow the “adaptation” learning phase. The traceplot created is as follows:



The model is relatively simple, so even with a single chain there seems to have been very little worries about convergence and reliability of the estimation.

See Chapter 9 of McElreath’s “Statistical Rethinking” for more details about HMC and related problems and solutions.

Run the linear regression model specified above again. Use the function “ulam” but specify 6 chains. Inspect the output and consider algorithm diagnostics.

A key recommendation when conducting Bayesian analyses such as this is that one should use *at least*  4 chains. In this example, the following commands invokes 6 chains (the “core” option just allows to distribute the process across different cores in your machine, making it faster):

m1<- ulam(

alist(

math12 ~ dnorm( mu, sigma) ,

mu<- a + b\*(smath8),

a ~ dnorm(50, 5),

b ~ dnorm(0, 5),

sigma ~ dlnorm(2, .7)

), data=dl, chains=6, core=6)

The “show” command informs that this time the HMC approximation used 3000 samples. The command “precis” also shows that the R-hat and the ESS Bulk outputs indicate convergence and reliable solutions:

> precis(m1)

mean sd 5.5% 94.5% rhat ess\_bulk

a 49.51 0.18 49.23 49.8 1 2840.80

b 8.21 0.18 7.92 8.5 1 2839.18

sigma 5.41 0.12 5.22 5.6 1 3028.30

The command “traceplot” shows the sequence of all chains at one (using different colours for each chain). It is possible to visualise these chains singularly by using the “traceplot” options. The output seems to indicate reliable solutions:

A graph of a person with a number of numbers

Description automatically generated with medium confidence

Another way to visualise the chains runs is option “trankplot” (see Chapter 9 of McElreath’s “Statistical Rethinking”). These plots take the samples for each individual parameter in the model and rank these samples from the lowest to the largest. These ranks are then drawn in histograms for each chain. If the chains are efficiently exploring the same parameter space, the ranks should be similar across chains, and thus largely overlapping. This seems to be the case for the model tested here:

A group of colorful lines

Description automatically generated

After inspecting the results in tabular form, plot the observed results, the average slope, the 95% credibility interval around the mean, and the 95% credibility interval of math12 scores.

The “precis” command above provides the marginal posterior distribution of the parameters in the model.

> precis(m1)

mean sd 5.5% 94.5% rhat ess\_bulk

a 49.51 0.18 49.23 49.8 1 2840.80

b 8.21 0.18 7.92 8.5 1 2839.18

sigma 5.41 0.12 5.22 5.6 1 3028.30

The expected average of ***math12*** scores is 49.51 for pupils who had average scores in maths at grade 8: there is 89% probability that this average will vary between 49.23 and 49.80. The SD of the average scores expected for pupils who had average scores in math at grade 8 is 0.18.

The slope indicates a strong association between math scores in grade 12 and those in grade 8: A 1 SD-unit increase in math scores at grade 8 is associated with a 8.21 increase in math scores in grade 12. It is estimated that the size of this effect varies between 7.92 and 8.50 with 89% probability.

It is possible to use other commands to describe key statistics from the posterior distribution, e.g.:

round(vcov(m1), 3)

allows to report variance-covariance matrix of the parameters in the model.

To visualise the predicted average scores and their uncertainty according to the estimated posterior distribution, we need to sample from the posterior distribution. The “rethinking” package “extract.samples” does that (and, as explained above, automatically discards samples form the “adaptation” learning phase of the Markov chains).

I will use this command to extract samples from the posterior distribution of model m1. The “mu.link” is a function that uses the linear expression specified to calculate the values of the dependent based on the parameters “a” and “b” extracted from the posterior distribution. Using this and the command “sapply”, I can then calculate the expected values of parameter μ (“mu”) for each values of the predictor I have listed in sequence. Using the “apply” commands I can then calculate the mean of these predicted values, and their Credibility Intervals at p = .95, as you can see here:

post<-extract.samples(m1)

post<-extract.samples(m1)

# Create a sequence of values for the predictor

smath8.seq<-seq(from=-2, to=3, by=.1 )

# The following creates a function to calculate the expected values of dependent

#based on the independent and the liner function parameters from the posterior

mu.link<-function(smath8) post$a + post$b\*smath8

#This now calculates the value of "mu" for each value of the sequence smath8

mu<-sapply(smath8.seq, mu.link)

#now I can calculate the mean of these values

mu.mean<-apply(mu, 2, mean)

#and now I can calculate the CIs considering p=.95

mu.ci<-apply(mu,2,PI, prob=0.95)

Calculating the credibility interval of the predicted scores can be achieved by using the function “sim”, which would simulate data from the model, thus allowing to calculate 95% credibility intervals in a similar way as above. In the script I used a different method that is akin to what I showed above:

#create a function to calculate predicted scores of math12 considering parameters from posterior

premath12<-function(smath8) rnorm(3000, (post$a + post$b\*smath8) , post$sigma)

#calculate predicted math12 scores based on the function above and the sequence of math8 scores

predict.math12<-sapply(smath8.seq, premath12)

#calculate the 95% CI of these predicted scores

math12.ci<-apply(predict.math12,2,PI, prob=0.95)

Finally, I put all these in a graph:

# plot the observed data

plot(math12 ~ smath8, data=d, pch=19, col="firebrick2", xlab="Std Math Scores Grade 8",

ylab="Math Scores Grade 12", cex.lab=1.5, cex=1.5)

#### Add information about uncertainty around the central values ####

shade(mu.ci, smath8.seq, col=col.alpha("peachpuff", .75))

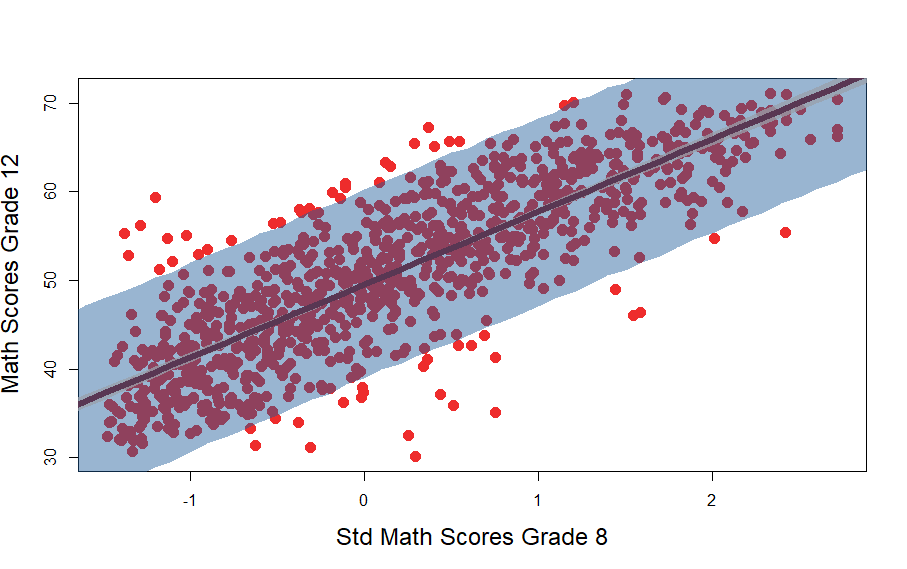
### Add the mean of mu

lines(smath8.seq, mu.mean, lwd=6, col="firebrick4")

### add CIs for the predicted scores

shade(math12.ci, smath8.seq, col=col.alpha("#1C5A99", .45))

The plot will look like this:



Further models might consider testing a quadratic association between math scores in different grades: all this would require is adding a parameter for the square of the predictor.