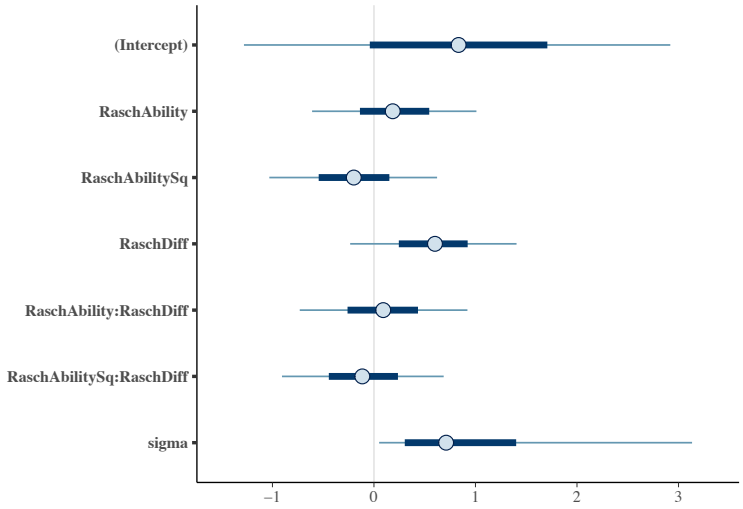
Model Selection and Prior Predictive Checking

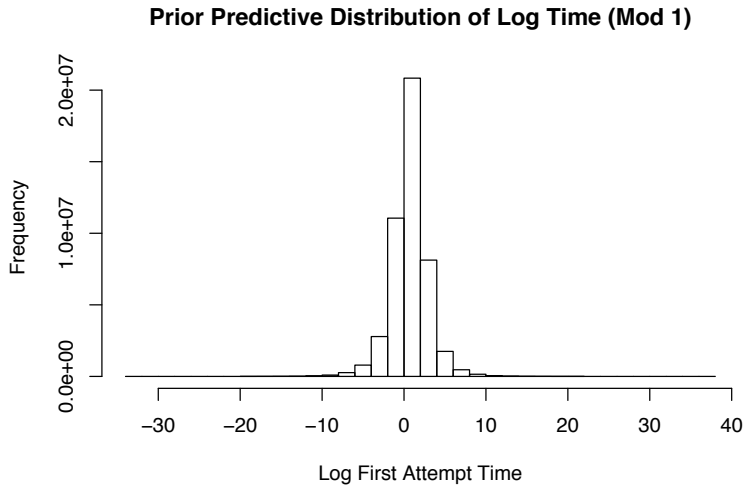
I chose to model log response time because of previous research that raw response times tend to be very right skewed. I set my priors such that they are centered at the (approximate) estimated coefficients from my previous analysis, which involved running identical models using OLS regression for the first cohort of students. All of the prior distributions (for the beta coefficients and intercept) are normal and I made sure that the standard deviations were wide enough that both negative and positive coefficients were reasonably plausible.

I then used the following code to sample from the prior distributions of the parameters, not conditioning on the data, and produce plots of the implied priors and prior predictive distributions. For both models, the priors on the coefficients and intercept look reasonable. The prior predictive distributions also look good, although they imply slightly more variance in the outcome variable than I would realistically expect. In my prior research, I did not observe log first response times that differed from the mean by more than ~4 units, whereas the prior predictive distributions allow values as large as +/-10 (albeit with low probability). However, I was not able to shrink this variance without implying over-confidence in the other parameters of the model. Therefore, I left everything as is.

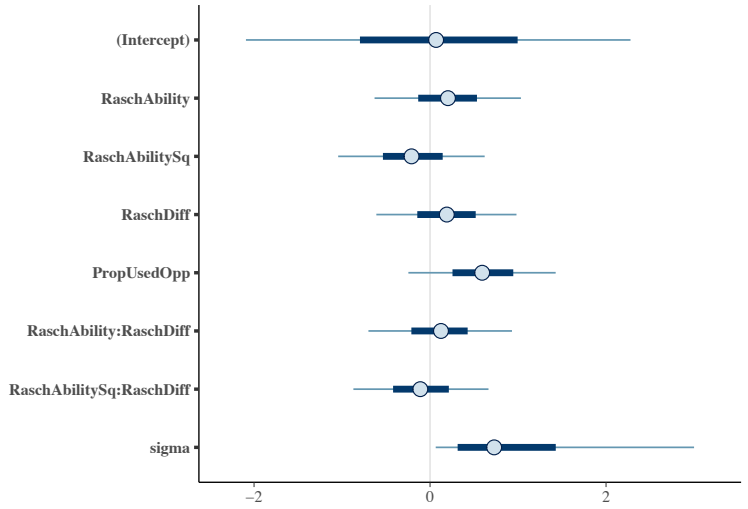
```
prior1 <- stan_glm(Log_Att1_Time_MC~(RaschAbility+RaschAbilitySq)*RaschDiff,
                   data=dat,
                   refresh=0,
                   family="gaussian",
                   prior_PD = TRUE,
                   prior = normal(location=c(0.2, -0.2, 0.6, 0.1, -0.1),
                                  scale=c(0.5, 0.5, 0.5, 0.5, 0.5),
                                  autoscale=FALSE),
                   prior_intercept = normal(.7, .5, autoscale=FALSE))
prior2 <- stan_glm(Log_Att1_Time_MC~(RaschAbility+RaschAbilitySq)*RaschDiff+PropUsedOpp,
                  data=dat,
                  refresh=0,
                  family="gaussian",
                  prior_PD = TRUE,
                  prior = normal(location=c(0.2, -0.2, 0.2, 0.6, 0.1, -0.1),
                                 scale=c(0.5, 0.5, 0.5, 0.5, 0.5, 0.5),
                                 autoscale=FALSE),
                  prior_intercept = normal(.5, .5, autoscale=FALSE))

plot(prior1)
```



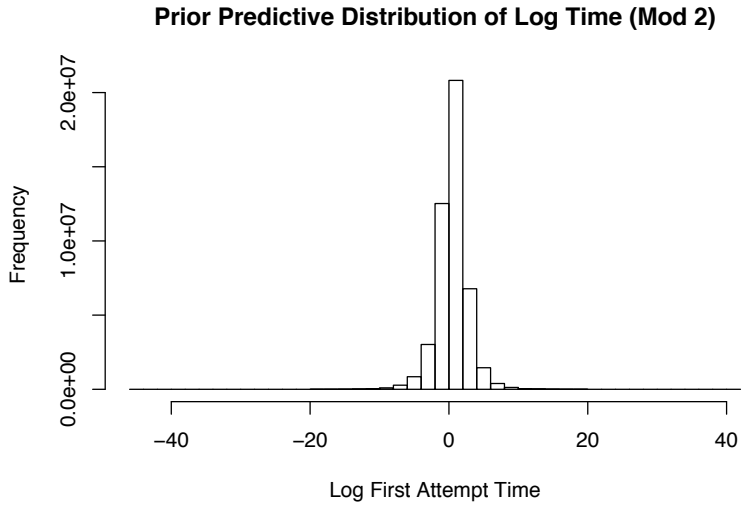
```
prior_preds <- posterior_predict(prior1)
hist(prior_preds, main="Prior Predictive Distribution of Log Time (Mod 1)",
     xlab="Log First Attempt Time", breaks=50)
```




```
plot(prior2)
```




```
prior_preds <- posterior_predict(prior2)
hist(prior_preds, main="Prior Predictive Distribution of Log Time (Mod 2)",
     xlab="Log First Attempt Time", breaks=50)
```



Posterior Distributions of the Parameters (Conditional on the Data)

Next, I ran the models using stan_glm, conditioning on the data, and inspected the resulting posterior distributions of the parameters. Based on median parameter values, the models look very similar to my original findings with run 1 of the course. The signs of the mean (and median) coefficients are all identical to what I found in my original analysis. The main differences are that a) these models suggest a slightly smaller coefficient on difficulty and b) these models suggest a slightly larger positive coefficient on ability but a slightly smaller negative coefficient on ability squared.

```
mod1 <- stan_glm(Log_Att1_Time_MC~(RaschAbility+RaschAbilitySq)*RaschDiff,
                 data=dat,
                 refresh=0,
                 family="gaussian",
                 #prior_PD = TRUE,
                 prior = normal(location=c(0.2, -0.2, 0.6, 0.1, -0.1),
                                scale=c(0.5, 0.5, 0.5, 0.5, 0.5),
                                autoscale=FALSE),
                 prior_intercept = normal(.7, .5, autoscale=FALSE))
summary(mod1, digits=4)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      Log_Att1_Time_MC ~ (RaschAbility + RaschAbilitySq) * RaschDiff
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 11616
##  predictors:   6
##
## Estimates:
##                            mean        sd          2.5%        25%
## (Intercept)                  -0.2217      0.0133     -0.2481     -0.2307
```

```
## RaschAbility                   0.3154      0.0186      0.2792      0.3030
## RaschAbilitySq                -0.0945      0.0089     -0.1118     -0.1007
## RaschDiff                      0.4442      0.0191      0.4070      0.4315
## RaschAbility:RaschDiff         0.3004      0.0258      0.2475      0.2833
## RaschAbilitySq:RaschDiff      -0.1001      0.0125     -0.1251     -0.1083
## sigma                          0.9283      0.0059      0.9167      0.9244
## mean_PPD                       0.0003      0.0122     -0.0238     -0.0078
## log-posterior             -15624.6139      1.8622 -15629.0087 -15625.6635
##                              50%         75%         97.5%
## (Intercept)                 -0.2219     -0.2125     -0.1955
## RaschAbility                 0.3157      0.3278      0.3514
## RaschAbilitySq              -0.0945     -0.0884     -0.0771
## RaschDiff                    0.4443      0.4565      0.4824
## RaschAbility:RaschDiff       0.3010      0.3175      0.3509
## RaschAbilitySq:RaschDiff    -0.1002     -0.0920     -0.0756
## sigma                        0.9282      0.9322      0.9401
## mean_PPD                     0.0004      0.0087      0.0232
## log-posterior           -15624.2801 -15623.2102 -15621.9837
##
## Diagnostics:
##                           mcse   Rhat   n_eff
## (Intercept)               0.0002 0.9998 4775
## RaschAbility              0.0004 1.0004 2679
## RaschAbilitySq            0.0002 1.0006 2729
## RaschDiff                 0.0003 1.0018 3127
## RaschAbility:RaschDiff    0.0005 1.0025 2311
## RaschAbilitySq:RaschDiff  0.0002 1.0008 2689
## sigma                     0.0001 1.0000 4210
## mean_PPD                  0.0002 0.9997 3622
## log-posterior             0.0432 1.0008 1857
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
mod2 <- stan_glm(Log_Att1_Time_MC~(RaschAbility+RaschAbilitySq)*RaschDiff+PropUsedOpp,
                 data=dat,
                 refresh=0,
                 family="gaussian",
                 #prior_PD = TRUE,
                 prior = normal(location=c(0.2, -0.2, 0.2, 0.6, 0.1, -0.1),
                                scale=c(0.5, 0.5, 0.5, 0.5, 0.5, 0.5),
                                autoscale=FALSE),
                 prior_intercept = normal(.5, .5, autoscale=FALSE))
summary(mod2, digits=4)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      Log_Att1_Time_MC ~ (RaschAbility + RaschAbilitySq) * RaschDiff +
##     PropUsedOpp
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 11616
```

```
##  predictors:   7
##
## Estimates:
##                              mean        sd          2.5%        25%
## (Intercept)                -0.3544     0.0999     -0.5443     -0.4231
## RaschAbility                0.3139     0.0182      0.2787      0.3018
## RaschAbilitySq             -0.0950     0.0086     -0.1118     -0.1010
## RaschDiff                   0.4434     0.0188      0.4068      0.4303
## PropUsedOpp                 0.1397     0.1048     -0.0612      0.0667
## RaschAbility:RaschDiff      0.2998     0.0262      0.2489      0.2820
## RaschAbilitySq:RaschDiff   -0.0996     0.0126     -0.1249     -0.1079
## sigma                       0.9282     0.0061      0.9164      0.9241
## mean_PPD                    0.0003     0.0122     -0.0239     -0.0077
## log-posterior          -15625.4252     2.0183 -15630.3196 -15626.4548
##                              50%         75%         97.5%
## (Intercept)                -0.3530     -0.2857     -0.1620
## RaschAbility                0.3137      0.3258      0.3499
## RaschAbilitySq             -0.0949     -0.0890     -0.0786
## RaschDiff                   0.4432      0.4562      0.4808
## PropUsedOpp                 0.1381      0.2123      0.3429
## RaschAbility:RaschDiff      0.2998      0.3175      0.3523
## RaschAbilitySq:RaschDiff   -0.0994     -0.0913     -0.0747
## sigma                       0.9282      0.9323      0.9401
## mean_PPD                    0.0005      0.0087      0.0238
## log-posterior          -15625.0890 -15623.9716 -15622.4978
##
## Diagnostics:
##                            mcse   Rhat   n_eff
## (Intercept)                0.0015 1.0003 4450
## RaschAbility               0.0004 1.0005 2625
## RaschAbilitySq             0.0002 1.0007 2702
## RaschDiff                  0.0003 0.9998 3725
## PropUsedOpp                0.0016 1.0003 4416
## RaschAbility:RaschDiff     0.0005 1.0001 2607
## RaschAbilitySq:RaschDiff   0.0002 1.0000 2975
## sigma                      0.0001 0.9992 4731
## mean_PPD                   0.0002 1.0002 4268
## log-posterior              0.0478 1.0010 1780
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

Next, I inspected the draws from the posterior distributions of beta coefficients to see what proportion were greater than or less than zero. In both models, 100% of the draws from the posterior distribution of $\beta_{ability}$ were greater than 0; whereas 100% of the draws from the posterior distribution of $\beta_{ability^2}$ were negative, painting the same picture as in my original analysis (with relatively high confidence): for lower ability levels there appears to be a positive association between ability and response time, but this flips to a negative association among higher ability students. Also in both models, 100% of the draws for $\beta_{difficulty*ability}$ were positive and 100% of draws for $\beta_{difficulty*ability^2}$ were negative, suggesting that the positive association between ability and response time and negative association between ability^2 and response time are both more pronounced for more difficult questions. Finally, in model 2, about 91% of the draws from the posterior distribution of coefficients on PropUsedOpp were positive, suggesting that there is likely a positive association between resilience and response time.

```
coefs1 <- as.matrix(mod1)
coefs1 <- as.data.frame(coefs1)
```

```r
coefs2 <- as.matrix(mod2)
coefs2 <- as.data.frame(coefs2)
##Model 1
#proportion of coefficients on Ability > 0
sum(coefs1$RaschAbility>0)/length(coefs1$RaschAbility)
```

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

```r
#proportion of coefficients on Ability^2 < 0
sum(coefs1$RaschAbilitySq<0)/length(coefs1$RaschAbilitySq)
```

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

```r
#proportion of coefficients on difficulty > 0
sum(coefs1$RaschDiff>0)/length(coefs1$RaschDiff)
```

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

```r
#proportion of coefficients on the interaction between difficulty and ability > 0
sum(coefs1$'RaschAbility:RaschDiff'>0)/length(coefs1$'RaschAbility:RaschDiff')
```

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

```r
#proportion of coefficients on the interaction between difficulty and ability^2 < 0
sum(coefs1$'RaschAbilitySq:RaschDiff'<0)/length(coefs1$'RaschAbilitySq:RaschDiff')
```

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

```r
##Model 2
#proportion of coefficients on Ability > 0
sum(coefs2$RaschAbility>0)/length(coefs2$RaschAbility)
```

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

```r
#proportion of coefficients on Ability^2 < 0
sum(coefs2$RaschAbilitySq<0)/length(coefs2$RaschAbilitySq)
```

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

```r
#proportion of coefficients on difficulty > 0
sum(coefs2$RaschDiff>0)/length(coefs2$RaschDiff)
```

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

```r
#proportion of coefficients on the interaction between difficulty and ability > 0
sum(coefs2$'RaschAbility:RaschDiff'>0)/length(coefs2$'RaschAbility:RaschDiff')
```

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

```r
#proportion of coefficients on the interaction between difficulty and ability^2 < 0
sum(coefs2$'RaschAbilitySq:RaschDiff'<0)/length(coefs2$'RaschAbilitySq:RaschDiff')
```

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

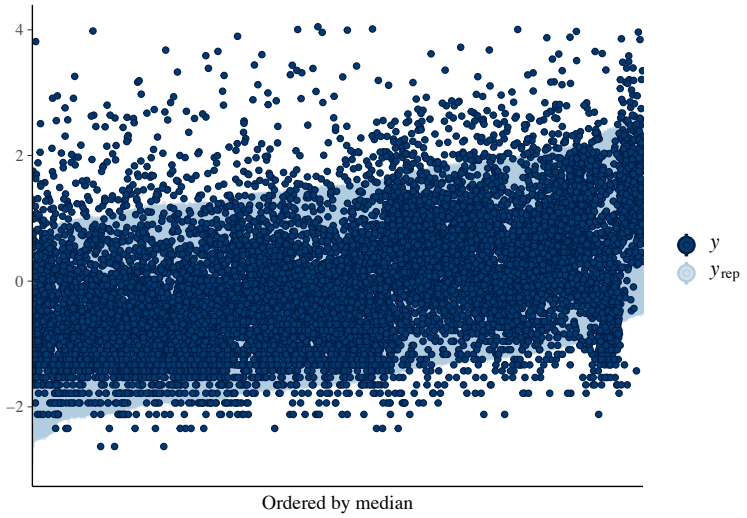
```r
#proportion of coefficients on PropUsedOpp > 0
sum(coefs2$PropUsedOpp>0)/length(coefs2$PropUsedOpp)
```

```
## [1] 0.9125
```

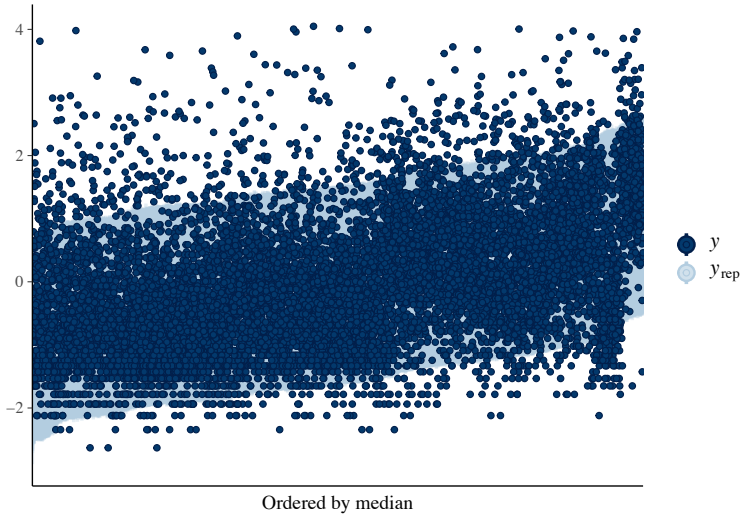
Posterior Predictive Checking

Based on the following plots, there are some large outliers in the data which are not estimated well by either model; however, most of the true data falls within the range estimated by the models. Note: there are 11,616 observations in this dataset.

```
pp_check(mod1, plotfun = "loo_intervals", order = "median")
```

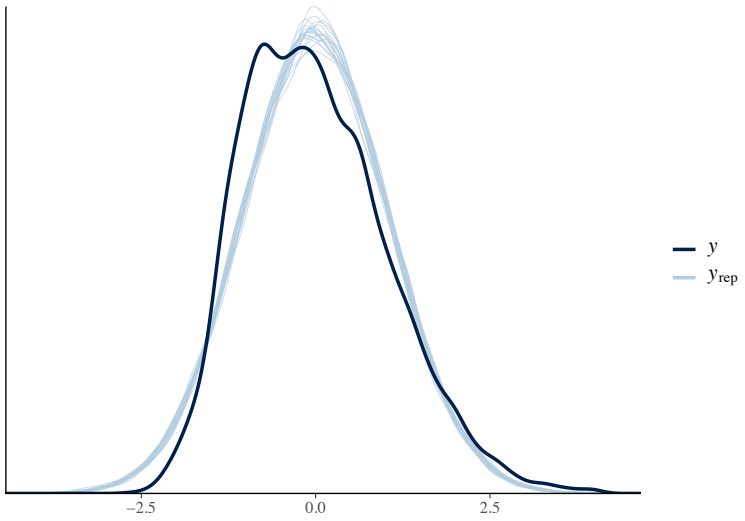



```
pp_check(mod2, plotfun = "loo_intervals", order = "median")
```

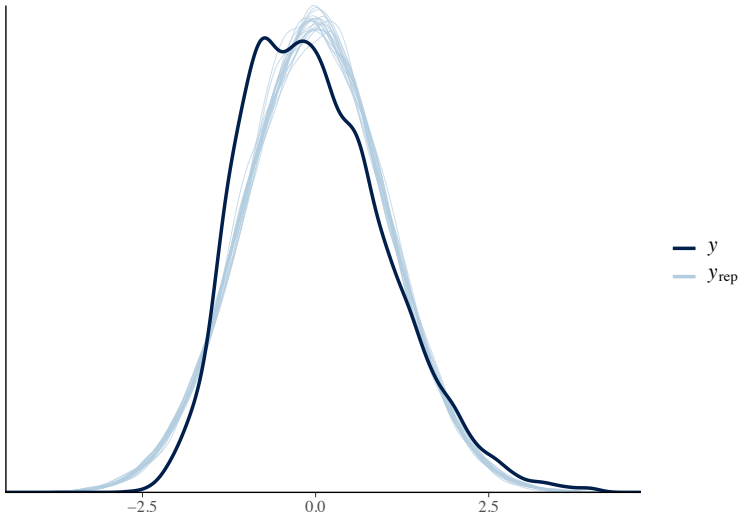


The following plots show posterior predictive distributions (in light blue) plotted against the true distribution of the outcome variable. The plots suggest that both models are predicting more symmetric distributions of the outcome than are realistic.

```
require(ggplot2)
pp_check(mod1, plotfun = "dens_overlay", nreps = 20)
```




```
pp_check(mod2, plotfun = "dens_overlay", nreps = 20)
```



Model Comparison by ELPD

The result below suggests that these two models are very similar in terms of Expected Log Predictive Density (with model 1 coming out slightly ahead). Therefore, I would definitely opt for the simpler model (without resilience). This is contrary to my conclusions based on the previous run of the course.

```
require(loo)
loo::loo_compare(loo(mod1),loo(mod2))
```

```
##        elpd_diff se_diff
## model1  0.0       0.0
## model2 -0.4       1.4
```

Model Comparison by Posterior Probability

Finally, the following results suggest that, under the strong assumption that one of these two models is correct, there is a higher posterior probability (~81%) of model 1 (without resilience) being correct than model 2. This assumes equal prior probabilities for the two models.

```
require(bridgesampling)
mod1 <- update(mod1, diagnostic_file = "mod1.csv")
mod2 <- update(mod2, diagnostic_file = "mod2.csv")
bridge_fit1 <- bridge_sampler(mod1, silent=TRUE)
bridge_fit2 <- bridge_sampler(mod2, silent=TRUE)
em1 <- error_measures(bridge_fit1)
em2 <- error_measures(bridge_fit2)
em1
```

```
## $re2
## [1] 3.579483e-06
##
## $cv
## [1] 0.001891952
##
## $percentage
## [1] "0.189%"
```

```
em2
```

```
## $re2
## [1] 3.019448e-06
##
## $cv
## [1] 0.001737656
##
## $percentage
## [1] "0.174%"
```

```
#note: errors are sufficiently small
post_prob(bridge_fit1, bridge_fit2)
```

```
## bridge_fit1 bridge_fit2
##   0.7225201   0.2774799
```

Concluding Thoughts

In conclusion, the results from this analysis are very similar to my original analysis using a different cohort of students. However, this data and approach suggest that there is more uncertainty around the association between student resilience and log response time, holding student ability and question difficulty constant, than I originally thought. I am also now far less convinced that this measurement of student resilience improves my ability to model and predict students' response times.