BDA Project

Dylan, Francesco, Pragati

Contents

1) Introduction

This report presents how to detect the "Quantitative Response" using Bayesian Inference.Bayesian modeling provides a principled way to quantify uncertainty and incorporate prior knowledge into the model.

What is more, **Stan**'s main inference engine, **Hamiltonian Monte Carlo sampling**, is friendly to diagnostics, which means we can verify whether our inference is reliable. Stan is an expressive probabilistic programing language that abstracts the inference and allows users to focus on the modeling. The resulting code is readable and easily extensible, which makes the modeler's work more transparent and flexible.

For this project **QSAR aquatic toxicity Data Set** has been chosen.

Motivation

This dataset was used to develop quantitative regression QSAR models to predict acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow) on a set of 908 chemicals.

To predict acute aquatic toxicity towards Daphnia Magna, LC50 data, which is the concentration that causes death in 50% of test D. magna over a test duration of 48 hours, was used as model response.

The model comprised 8 molecular descriptors: TPSA(Tot) (Molecular properties), SAacc (Molecular properties), H-050 (Atom-centred fragments), MLOGP (Molecular properties), RDCHI (Connectivity indices), GATS1p (2D autocorrelations), nN (Constitutional indices), C-040 (Atom-centred fragments).

The problem

Modeling idea

In this report, We focus on Bayesian inference with MCMC. Bayesian inference gives us a principled quantification of uncertainty and the ability to incorporate domain knowledge in the form of priors, while MCMC is a reliable and flexible algorithm.

In addition, Stan provides diagnostic tools to evaluate both the inference (e.g. accuracy of the MCMC, convergence of chains) and the model (e.g. posterior predictive checks).

This reports use 3 different type of models i.e

1) Linear Model
2) Hierarchical Model
3) Gaussian Process

Further section explains How to the Stan model was run, Convergence diagnostics,Posterior predictive checks,Model comparison (e.g. with LOO-CV),Predictive performance assessment,Sensitivity analysis with respect to prior choices, all these steps has been performed separately on these 3 models

Illustrative figure

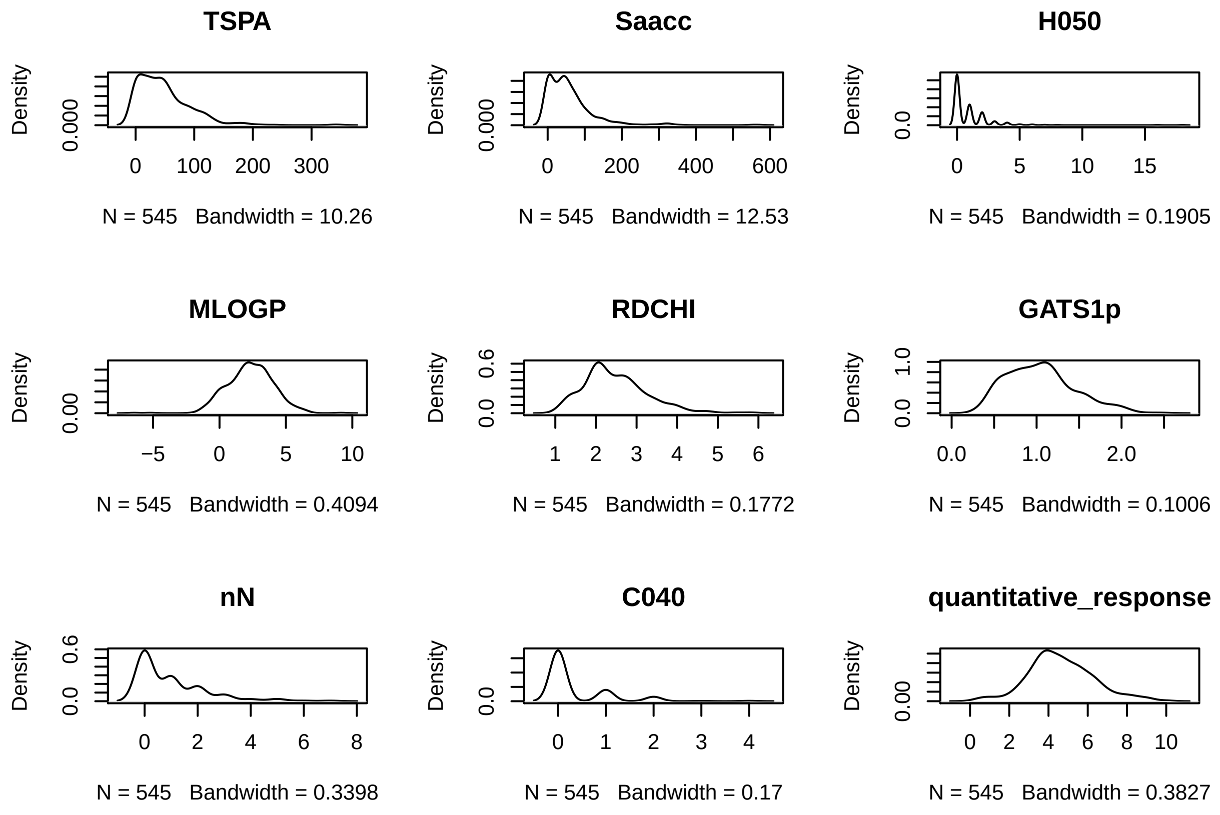
Lets visualizing the density plot of each variable in order to check the range of the variables in the dataset

QSAR toxicity Dataset

```r
QSAR <- read.csv(file = 'qsar_aquatic_toxicity.csv')
colnames(QSAR)
```

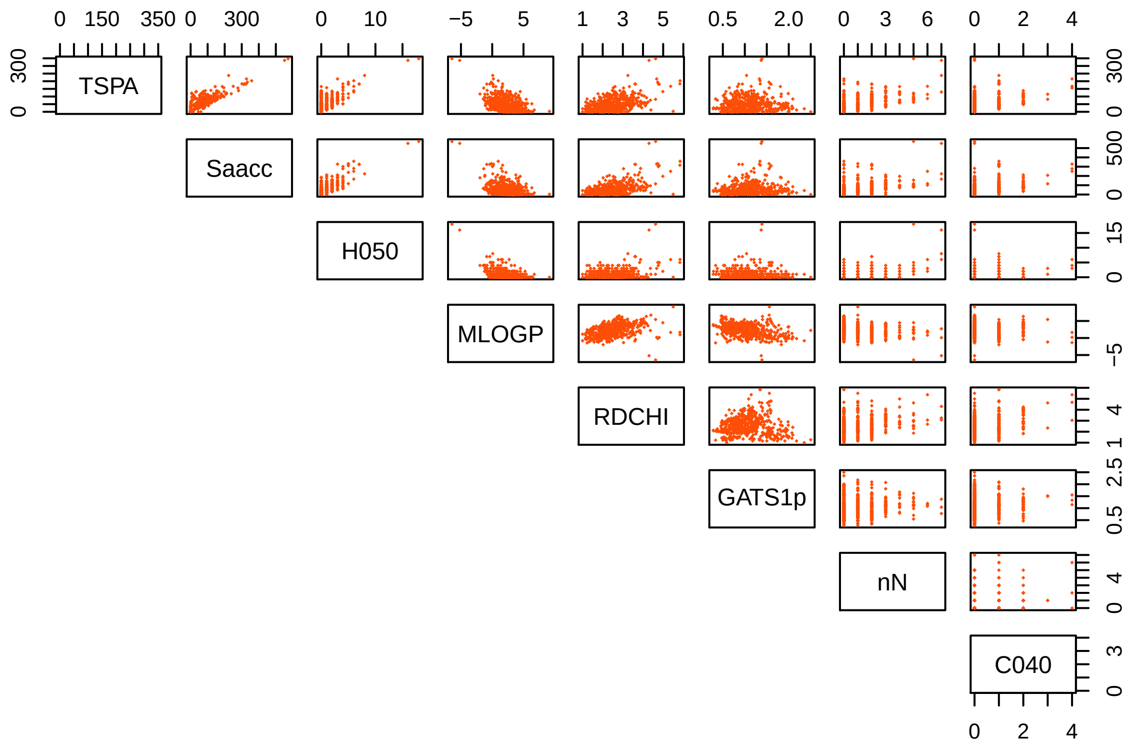
```
## [1] "TSPA"                     "Saacc"                    "H050"
## [4] "MLOGP"                    "RDCHI"                    "GATS1p"
## [7] "nN"                       "C040"                     "quantitative_response"
```

```r
par(mfrow=c(3,3))
cols = colnames(QSAR)

for(col in cols)
{
plot(density(QSAR[col][,1]), main=col)

}
```



Now lets visualize the pair plot in order to see if there is any collinearity in the data set or not.

```
#PairPlot(QSAR, colnames(QSAR)[1:8],
#"Pair Plotting of each pair of features", alpha = 0.8,  point_color = "blue")
pairs(QSAR[,1:8], pch = 18,  cex = 0.4, col = "#FC4E07", lower.panel=NULL)
```



```r
#heatmap(as.matrix(QSAR))
#boxplot.default(QSAR, horizontal = TRUE)
```

2) Description of the data and the analysis problem.

The dataset has been obtained from UCI dataset archives

(https://archive.ics.uci.edu/ml/datasets/QSAR+aquatic+toxicity)

The following table explains the explanatory and the target variables present in the dataset

feature number	Feature name	Feature Description
1	TSPA	Tot Molecular properties
2	SAACC	Molecular properties
3	H050	Atom-centred fragments
4	MLOGP	Molecular properties
5	RDCHI	Connectivity indices
6	GATS1p	2D autocorrelations
7	nN	Constitutional indices
8	C040	Atom-centred fragments
9	Quantitative Response	acute aquatic toxicity

In order, to analyze the problem in detail, the linear Regression model has been implemented. This has been implemented so that later the bayesian models can be compared with this linear regression model. This model gave us the base estimate and helped in visualizing the data more effectively.

Clearly we can see almost all the variables are statistically significant.

```r
#creating linear model
fullmodel=lm(quantitative_response~TSPA+Saacc+H050+MLOGP+RDCHI+GATS1p+nN+C040, data =QSAR)
summary(fullmodel)
```

```
##
## Call:
## lm(formula = quantitative_response ~ TSPA + Saacc + H050 + MLOGP +
##     RDCHI + GATS1p + nN + C040, data = QSAR)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5076 -0.7615 -0.1023  0.6109  4.9580
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.705317   0.244832  11.050  < 2e-16 ***
## TSPA         0.027341   0.002670  10.240  < 2e-16 ***
## Saacc       -0.015030   0.002094  -7.179 2.36e-12 ***
## H050         0.038974   0.059859   0.651 0.515266
## MLOGP        0.446237   0.063325   7.047 5.66e-12 ***
## RDCHI        0.514491   0.135630   3.793 0.000166 ***
## GATS1p      -0.570089   0.153962  -3.703 0.000235 ***
## nN          -0.232559   0.049547  -4.694 3.41e-06 ***
## C040        -0.030181   0.090962  -0.332 0.740175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.203 on 536 degrees of freedom
## Multiple R-squared:  0.4866, Adjusted R-squared:  0.479
## F-statistic: 63.51 on 8 and 536 DF,  p-value: < 2.2e-16
```

```r
#plot(QSAR$quantitative_response,fullmodel$res,
#ylab="e_bar", xlab="y_bar", main="Residual plot")
#abline(0,0)
```

3) Description of the 3 models used

For the sake of this project, we have implemented Linear(Non Hierarchical), Hierarchical model and Gaussian Process

a) Linear (Non Hierarchical)

In the linear model, quantitative response is normally distributed with mean mu and variance sigma. Mean mu is computed by the dot product between coeffiencients and explanatory variables.

$QuantitativeResponse_i \sim \mathcal{N}\ (\mu\ , sigma\ )$

where

μ = alpha + beta[1] * TSPA + beta[2] * Saacc + beta[3] * H050 + beta[4] * MLOGP+ beta[5] * RDCHI +beta[6] * GATS1p + beta[7] * nN + beta[8] * C040;

which is written in stan code like the following for simplicity

μ_i = alpha + dotProduct($x_{i,:}$,Transpose(beta));

Here alpha and beta are the prior which are used in this linear model where alpha is a real number and beta is vector of real numbers.

b) Hierarchical

In the hierarchical model, the data has been categoried according to C040 explanatory variables. C040 represents the number of carbon atoms. After thorough reading of research papers, our team analysed the C040 maybe be an interesting variable. Therefore hierarchical model has been constructed based on C040, i.e C040 will decide the prior based on C040 variable.

In the hierarchical model, quantitative response is normally distributed with mean mu and variance sigma similar to linear model.

$QuantitativeResponse_i \sim \mathcal{N}\ (\mu\ , sigma\ )$

However, the major difference here is of mu and the priors . Here

$\mu_i = alpha[C040_{i+1}] + dotProduct(x_{ind,:}, beta[C040_{ind+1},]')$

This means mu is dependent on the the coefficient alpha and beta which maps to corresponding C040 value.

Here alpha is vector of dimension NC(i.e number of distinct values of C040) and beta is a matrix of dimension NC x J (number of distinct values of C040 X explanatory variables)

4) Informative or weakly informative priors

For this report, we used the following priors

1) Weakly informative prior, very weak: normal(0, 10);
2) Generic weakly informative prior: normal(0, 1);

The recommendations have been followed as per

https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations

5) Stan code

a) Linear (Non Hierarchical)

Stan Code

```r
code <- file("linear_model_split.stan")
writeLines(readLines(code))
```

```
## data {
##
## int < lower =1> N_train; // number of data points in train set
## vector [N_train] qr_train;// quantitative response for train set
## int <lower=1> J; //number of features
## vector [J] x_train [N_train]; //train dataset of explanatory variables
##
## int < lower =1> N_test; //number of data points in test set
##
## vector [J] x_test [N_test];///train dataset of explanatory variables
##
##
## }
## parameters {
##   real alpha;
##   vector [J] beta;
##   //beta is the vector containing cofficients for 8 exlanatory variables
##   real < lower =0> sigma ;
## }
## transformed parameters {
##   vector [N_train] mu_train;
##   for (i in 1:N_train)
##   mu_train[i] = alpha + dot_product(x_train[i,:],beta');
##
## }
## model {
## //priors
## alpha~ normal(0,1);
## for(j in 1:J)
## {
## beta[j]~normal(0,1);
## }
## sigma ~ normal(0,100);
## qr_train ~ normal (mu_train , sigma );
##
## }
## generated quantities {
##
## // mu vector for test set
## vector [N_test] mu_test;
## //predicting the quantitive response for test set
## vector [N_test] qr_test;
## //log liklihood for train set
## vector[N_train] log_lik;
## //liklihood for train set
## vector[N_train] gen_lik;
##
## for (i in 1:N_test)
## {mu_test[i] = alpha + dot_product(x_test[i,:],beta');};
##
## for (ind in 1:N_test)
## {
## qr_test[ind]= normal_rng (mu_test[ind] ,sigma);
## };
##
## for (ind in 1:N_train)
## {
## log_lik[ind]= normal_lpdf(qr_train[ind] | mu_train[ind] ,sigma);
## gen_lik[ind]= normal_rng (mu_train[ind] ,sigma);
## };
##
## }
```

b) Hierarchical

```
code_hierarchial <- file("hierarchial.stan")
writeLines(readLines(code_hierarchial))
```

```
## data {
```

```
##
## int < lower =1> N; // number of data points
## vector [N] qr; //quantitative response
## int <lower=1> J; //J will be 7 excluding c040
## //here instead of 8 of linear model
## vector [J] x [N]; // dataset of explanatory variables
## int C040[N]; //data for C040 feature
## int <lower = 0> nc; //number of distinct C040(5 here)
##
## }
## parameters {
##   vector[nc] alpha; // 5 values for 5 distince C040
##   vector[J] beta [nc]; //5x7 matrix
##   real < lower =0> sigma ;
##
##   //hyperparameters declaration
##   vector [J] mu_coff;
##   vector <lower =0> [J] tau_coff;
##   real <lower= 0> tau_a;
##   real mu_a;
## }
## transformed parameters {
##   vector [N] mu;
##   for(ind in 1:N)
##   {
##   mu[ind]= alpha[C040[ind]+1] + dot_product(x[ind,:],beta[C040[ind]+1,]');
##
##   };
##
## }
## model {
## //setting priors for paramters and hyperparameters
## mu_a~normal(0,1);
## tau_a~ normal(0,1);
##
## for(j  in 1:J)
## {
## mu_coff[j]~normal(0,1);
## tau_coff[j]~ normal(0,1);
## }
##
## alpha~ normal(mu_a,tau_a);
##
## for(j in 1:J)
## {
## beta[,j] ~ normal(mu_coff[j],tau_coff[j]);
## }
##
## sigma ~ normal(0,100);
##
## //liklihood
## qr ~ normal (mu , sigma );
##
## }
## generated quantities {
## //log liklihood for data set;
## vector[N] log_lik;
## //liklihood for the dataset
## vector[N] gen_lik;
## for (ind in 1:N)
## {
## log_lik[ind]= normal_lpdf(qr[ind] | mu[ind] ,sigma);
## gen_lik[ind]= normal_rng (mu[ind] ,sigma);
## };
##
## }
```

6) How the Stan model was run

a) Linear (Non Hierarchical)

The Linear model has been run for 1000 interations and with 4 monte carlo chains. The dataset has been divided into test and train sets for predictive checks.

```
N_test=20
n=dim(QSAR)[1]
N_train = n-N_test
qr_train=QSAR$quantitative_response[1:N_train]
J=(dim(QSAR)[2]-1)
x_train= QSAR[1:N_train,1:(dim(QSAR)[2]-1)]
#N_test
x_test= QSAR[(N_train+1):n,1:(dim(QSAR)[2]-1)]

qsar_data_check <-list(N_train=N_train,qr_train=qr_train,
                       J=J,x_train=x_train,N_test=N_test, x_test=x_test)

linear_model <-stan(file = 'linear_model_split.stan' ,
                    data = qsar_data_check, chains=4, iter=1000)
params=extract(linear_model, permuted=FALSE, inc_warmup=TRUE)
```

b) Hierarchical

The Hierarchical model will be run for 2000 iterations and 4 chains. Initially the model was run with default adapt_delta, but there were too many divergences. So adapt_delta was set to 0.95 in order to make it less divergent.

```
n=dim(QSAR)[1]

qr=QSAR$quantitative_response
J=(dim(QSAR)[2]-2)
x= QSAR[,1:(dim(QSAR)[2]-2)]
C040=as.integer(QSAR$C040)
nc = length(unique(QSAR$C040))

qsar_data <-list(N=n,qr=qr,J=J,x=x,C040=C040,nc=nc)

hierarchial <-stan(file = 'hierarchial.stan' , data = qsar_data, chains=4,
                   iter=2000, control=list(adapt_delta=0.95))
#print(hierarchial)
```

7) Convergence diagnostics

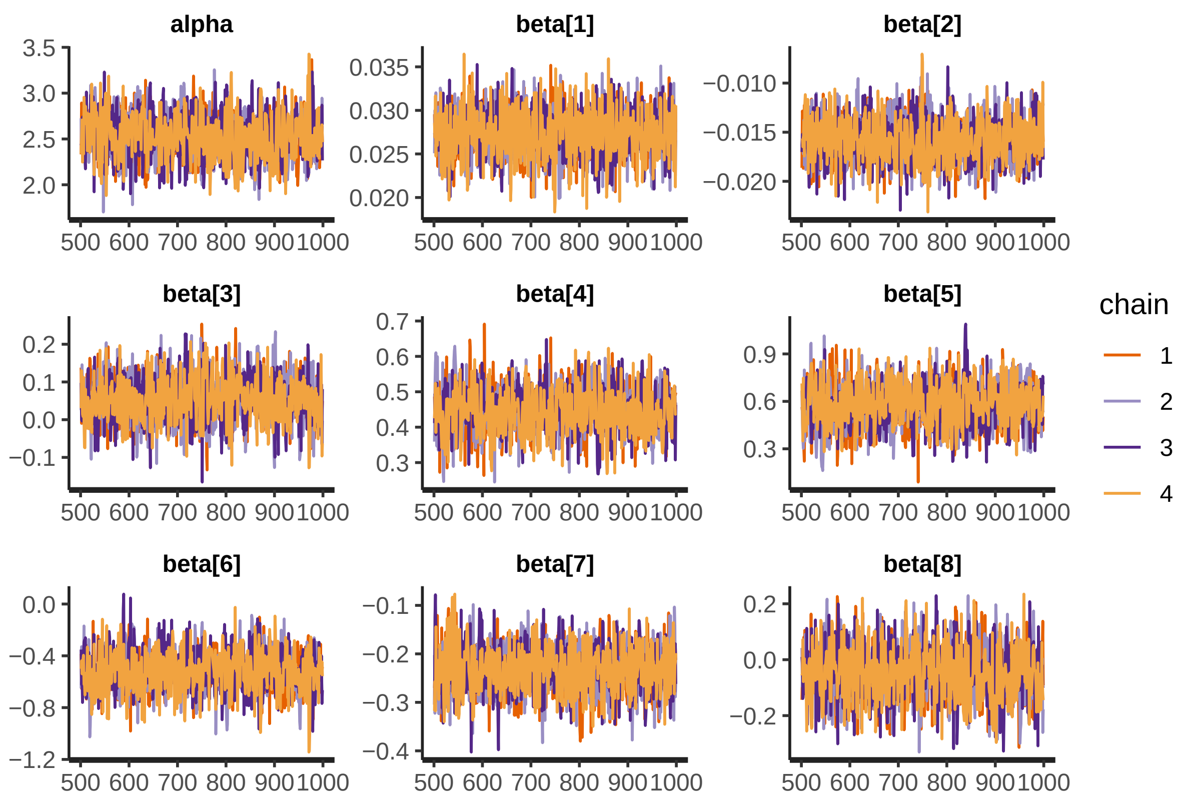
Here we will discuss 5 types of convergence tests 1) Traceplots 2) $\hat{R}$ 3) n_{eff} 4) Bulk ESS and Tail ESS 5) Divergences

Linear (Non Hierarchical)

1) Traceplots

The traceplots clearly show that the parameters have converged.

```
traceplot(linear_model, pars=c("alpha","beta"))
```



2) $\hat{R}$

```
Rhat_linear= summary(linear_model)$summary[,'Rhat']
print(max(Rhat_linear))
```

```
## [1] 1.006528
```

From printed output we can see that $\hat{R} < 1.01$ for all the parameters

3) n_{eff}

```
neff=summary(linear_model)$summary[,'n_eff']
val=neff/1000
#print(val)
which(val < 0.01)
```

```
## named integer(0)
```

$\frac{samples}{totalIterations} > 0.01$ for all parameters, this means samples are not biased and true effect of sample size is not overestimated.

4) Bulk ESS and Tail ESS

```
bulk_ess=monitor(extract(linear_model,permute=FALSE, inc_warmup=FALSE))[,'Bulk_ESS']
length(which(bulk_ess < 100))
```

BUlk ESS over 100 for all the parameters i.e reliable

5) Divergences

```
get_num_divergent(linear_model)
```

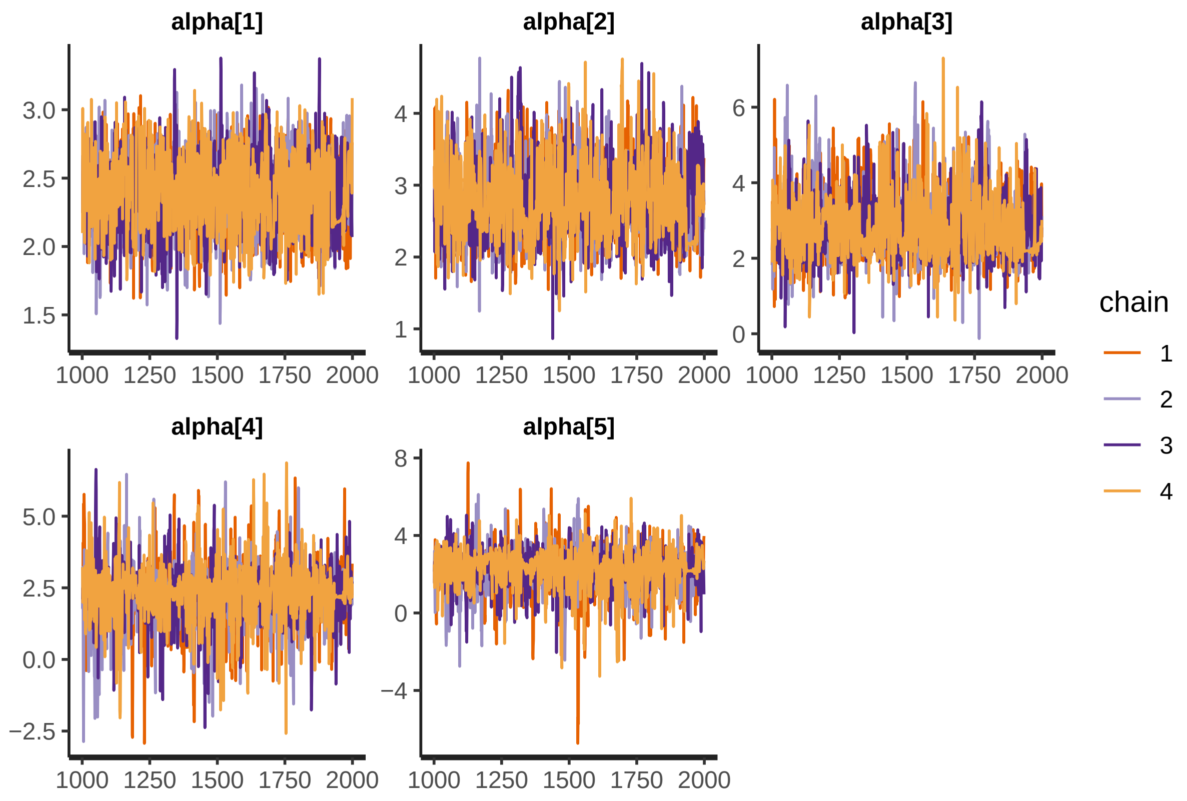
```
## [1] 0
```

No divergences in the linear model

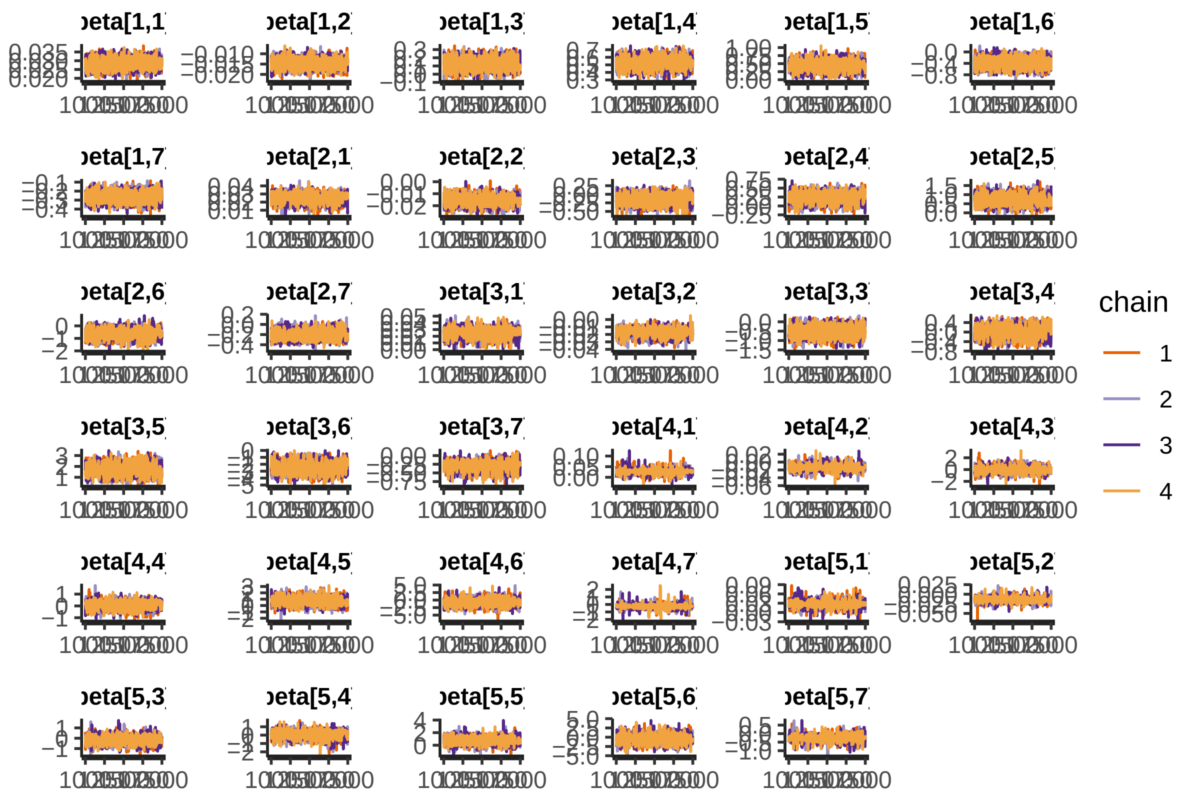
Hierarchical

The plots are seem to be converging

```
traceplot(hierarchial, pars=c("alpha"))
```



```r
traceplot(hierarchial, pars=c("beta"))
```



2) $\hat{R}$

```
Rhat_h= summary(hierarchial)$summary[,'Rhat']
print(max(Rhat_h))
```

```
## [1] 1.02795
```

From printed output we can see that $\hat{R} < 1.03$ for all the parameters. This implies Hierarchical model have not converged properly.

3) n_{eff}

```
neff=summary(hierarchial)$summary[,'n_eff']
val=neff/2000
#print(val)
length(which(val < 0.01))
```

```
## [1] 0
```

$\frac{samples}{totalIterations} > 0.01$ for all parameters, this means samples are not biased and true effect of sample size is not overestimated.

4) Bulk ESS and Tail ESS

```
bulk_ess=monitor(extract(hierarchial,permute=FALSE, inc_warmup=FALSE))[,'Bulk_ESS']
length(which(bulk_ess < 100))
```

BUlk ESS are not reliable. ### 5) Divergences

```
get_num_divergent(hierarchial)
```

```
## [1] 281
```

Initially the divergences were more than 1000, but after setting the **adapt_delta=0.95**, the divergences were reduced. So we evaluated the hierarchical model was not satisfactory.

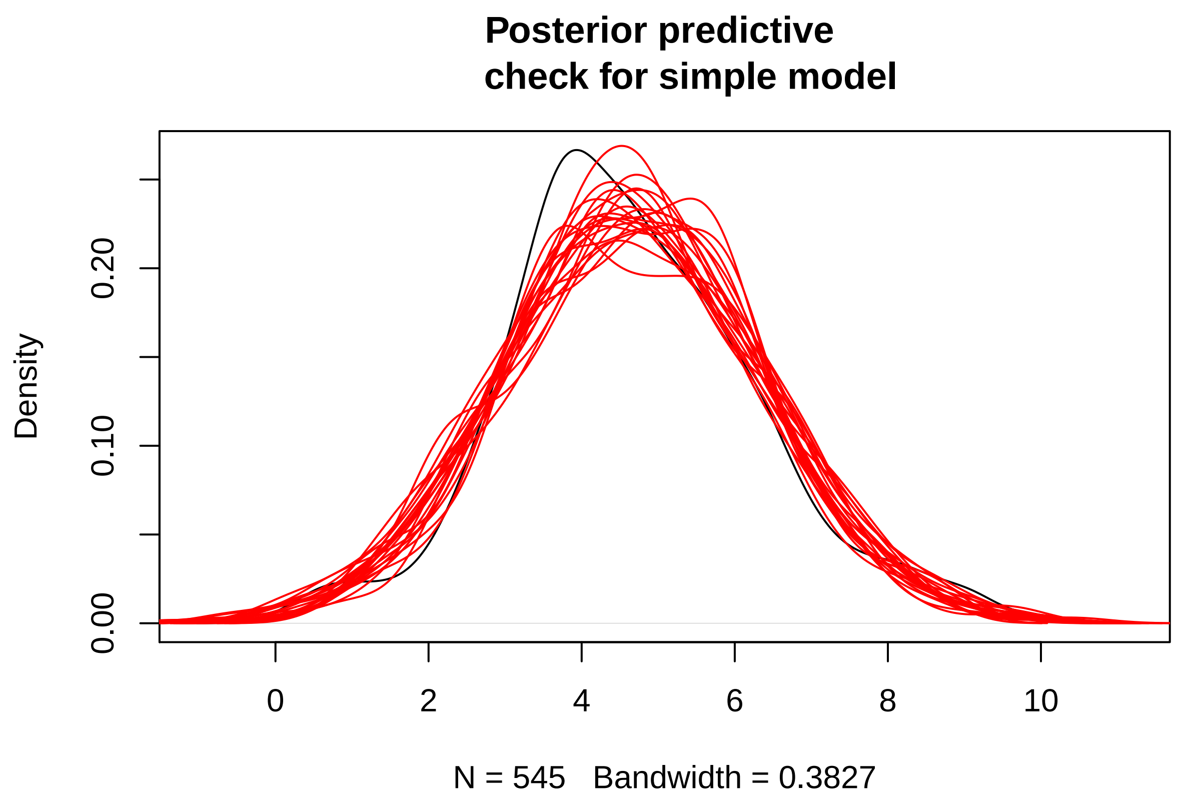
8) Posterior predictive checks

In order to check the Posterior, we extract the values of the quantitative response of the last 20 interactions and compare it with the actual quantitative response.

The black density plot is the plot for actual quantitative response. The red lines are the plot for generated quantitative response.

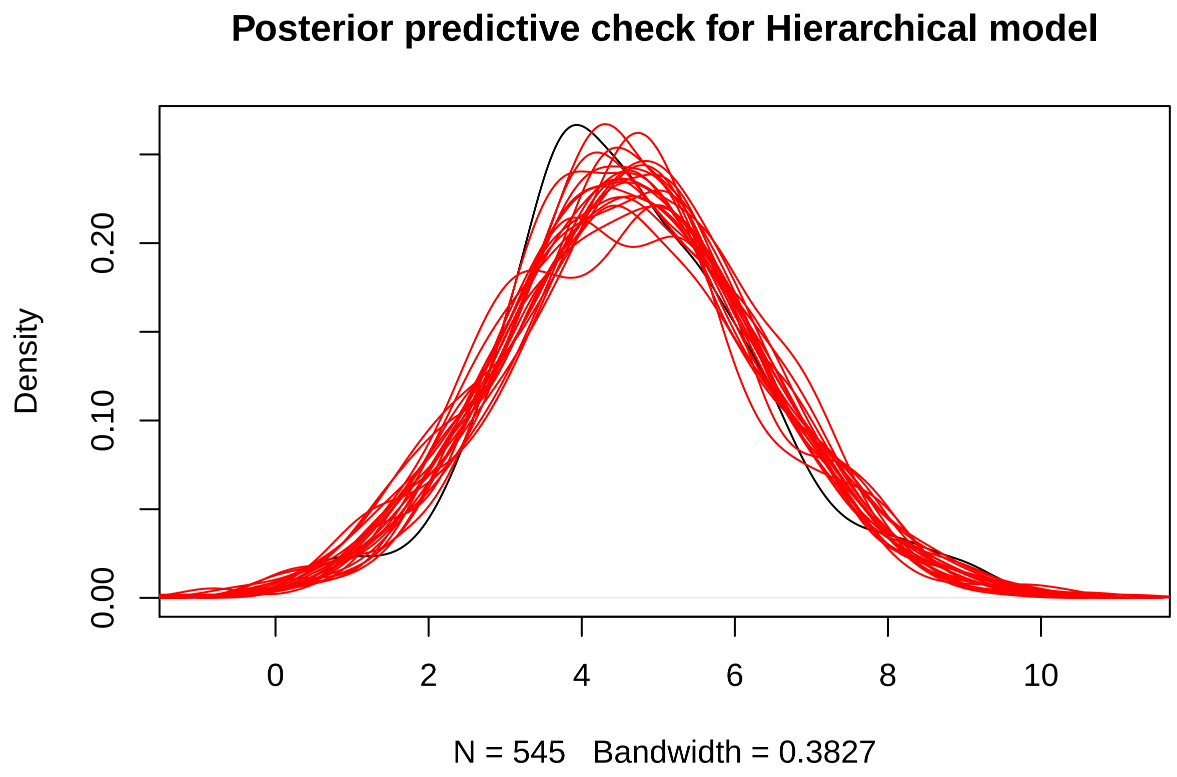
Linear (Non Hierarchical)

```
# instead of log, use rng
plot(density(QSAR$quantitative_response),main="Posterior predictive
     check for simple model")
params<-extract(linear_model)
for (ind in 1980:2000)
{
  lines(density(params$gen_lik[ind,]), col='red');
}
```



Hierarchical

```r
# instead of log, use rng
plot(density(QSAR$quantitative_response),main="Posterior predictive check for Hierarchical model")
params<-extract(hierarchial)
for (ind in 3980:4000)
{
  lines(density(params$gen_lik[ind,]), col='red');
}
```



We see the posterior check for the linear model are more close to the actual quantitative response.

9) Model comparison

In this section, we will compute the PSIS_LOO values using the loo library and then compare the models

Linear (Non Hierarchical)

```r
loo_model_linear <- loo(extract_log_lik(linear_model))
```

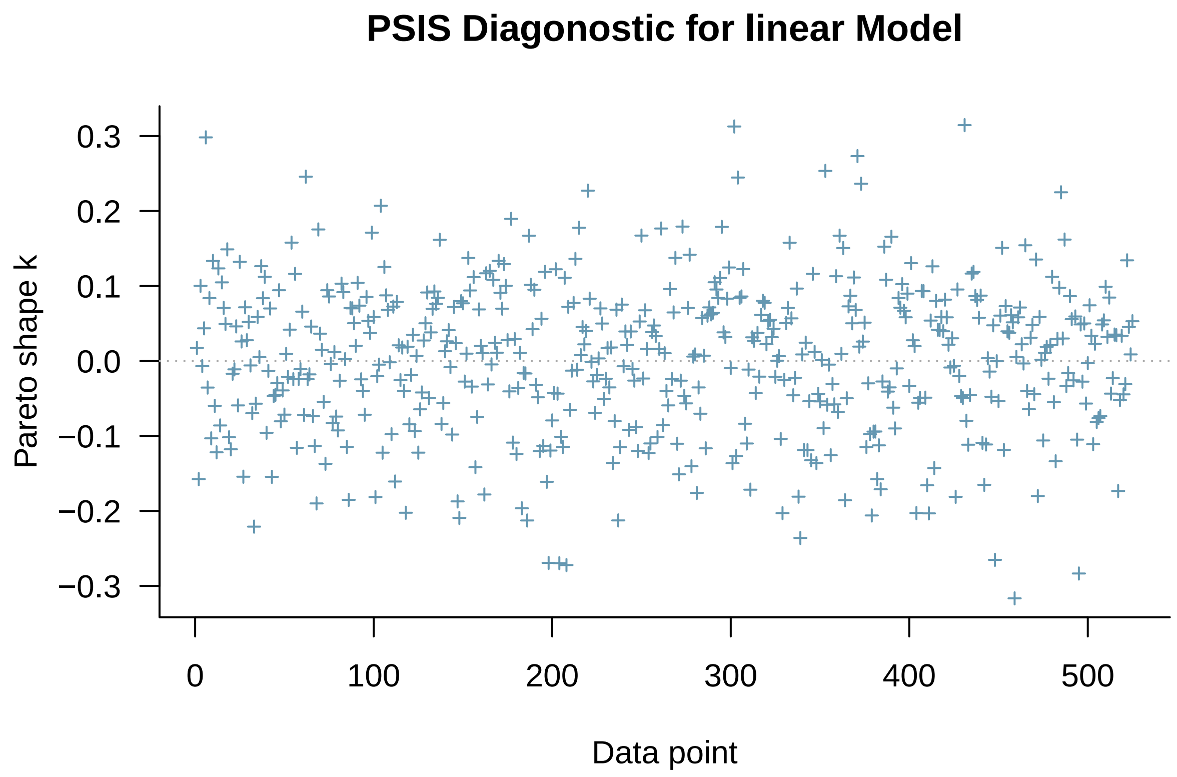
```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```r
print(loo_model_linear)
```

```
##
```

```
## Computed from 2000 by 525 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -850.9 21.1
## p_loo        12.5  1.5
## looic      1701.7 42.1
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_model_linear, main = "PSIS Diagonostic for linear Model")
```



Hierarchical

```
loo_model_hierarchial <- loo(extract_log_lik(hierarchial))
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

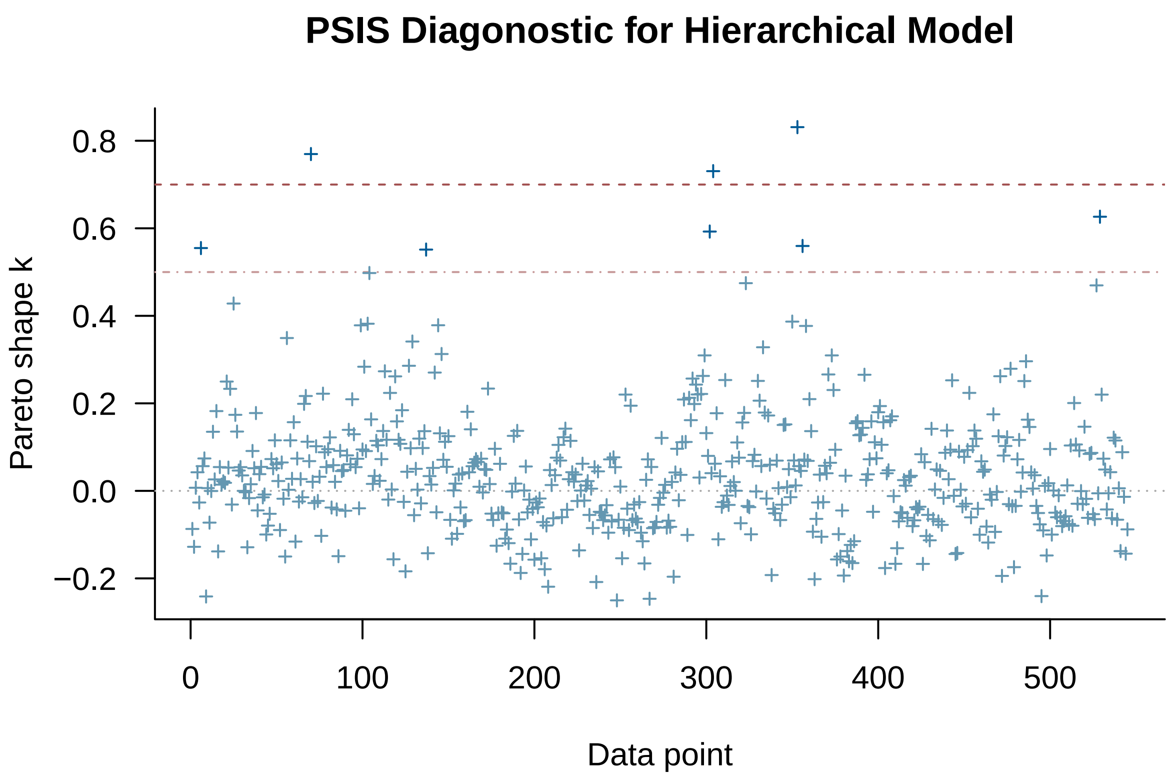
```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
loo_model_hierarchial
```

```
##
## Computed from 4000 by 545 log-likelihood matrix
```

```
##
##          Estimate   SE
## elpd_loo   -871.1 22.0
## p_loo        27.1  3.0
## looic      1742.1 44.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     537   98.5%   1247
##  (0.5, 0.7]   (ok)         5    0.9%   589
##    (0.7, 1]   (bad)        3    0.6%   152
##    (1, Inf)   (very bad)   0    0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```r
plot(loo_model_hierarchial, main = "PSIS Diagonostic for Hierarchical Model")
```



From the plot, we see the LOO values of the Linear model are better.

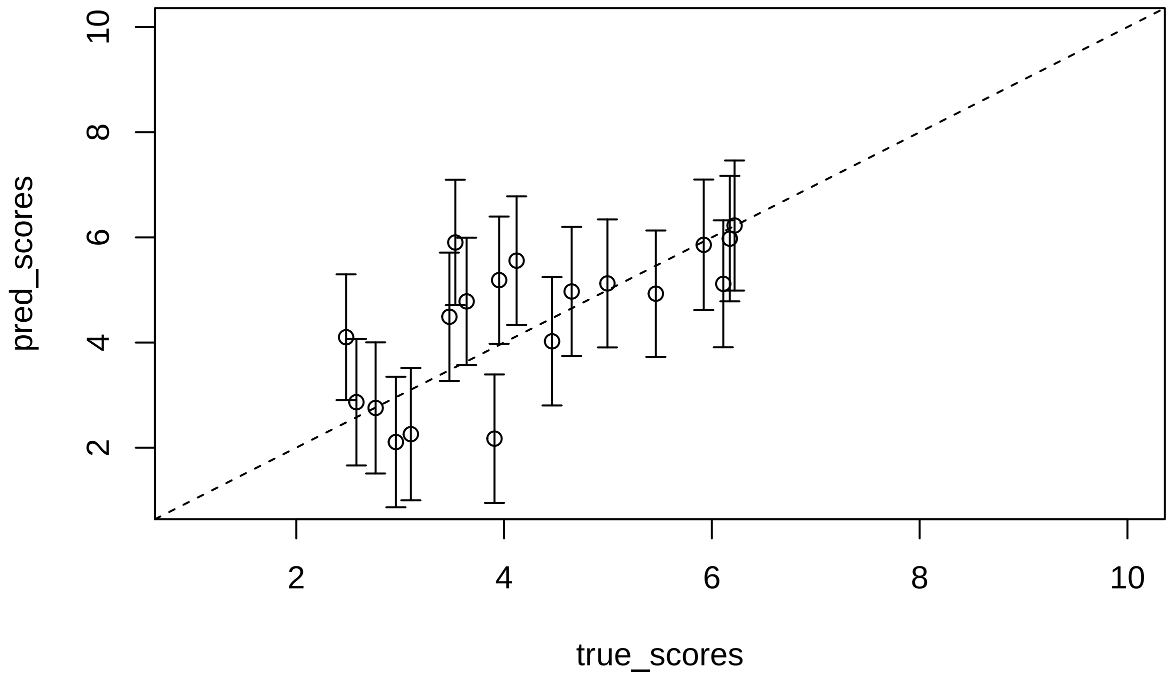
10) Predictive performance assessment

Here the predictive performance is done only for the Linear Model as it performed best for our dataset.

Here, we are comparing the actual quantitative response with the predicted quantitative response.

The plot indicates the actual target variable along with the predicted values with uncertainty.

```r
new_params= extract(linear_model)
pred_scores = colMeans(new_params$qr_test)
pred_error = sapply(1:N_test, function(x) sd(new_params$qr_test[,x]))
true_scores = QSAR$quantitative_response[(N_train+1):n]
plot(true_scores,pred_scores,xlim=range(1:10), ylim=range(1:10))
abline(a=0,b=1,lty="dashed")
arrows(true_scores,pred_scores+pred_error,true_scores,
       pred_scores-pred_error, length =0.05, angle = 90, code =3)
```



11) Sensitivity analysis

with respect to prior choices (i.e. checking whether the result changes a lot if prior is changed)

2 types of priors

12) Discussion of issues and potential improvements.

1) Based on our understanding, the hierarchical model was built on the basis of C040 variable. However this model performed worse than the linear model. One potential improvement can be building hierarchical model based on some other parameter like H050 or some variable. Based on deep understanding on the data, a better way of hierarchical modeling can be constructed.

2) For the Gaussian model which we implemented, we faced the issue of computational time. It took forever to execute our code. Thats why we chose the subset of dataset to work upon. In future maybe with the help of better computation power, we could build a better model.

13) Conclusion

1) One important thing we noticed here was that changing priors didn't change the results significantly. This also comes by intuition as the dataset we used in this report was large enough for Bayesian inference. And when the dataset is large enough, priors doesnt make much difference.
2) We concluded that not always the complex model is better. Most often complex models fails to outperform due to lack of domain knowledge. Like in our case the hierarchical model didn't performed good as expected. Maybe it will perform better when grouping method changes, but we have to check that.

The code for model is present at the following Github link, feel free to contribute.

https://github.com/PragatiGupta97/Bayesian-World-Bank-Youth-Unemployment-Rates

14) Self-reflection of what the group learned while making the project.

1) This project acted like a crash course for the BDA Course for us. This helped us in reviving all the concepts which we already covered in the course but somehow forgotten over time.
2) Developing an end to end project with real dataset helped us in connecting the knowledge which we gained over the course. There were some knowledge gaps in between, but this project helped us in connected all the dots.
3) Overall the experience was quite rich.