Data Analysis Project - Bayesian Statistics V2

Ken Wood

August 6, 2023

Executive Summary

The purpose of this project was to fit a series of regression models to a dataset containing housing features and a corresponding sale price as the response variable. Three models were constructed using both `R` and `JAGS`. One of the `JAGS` models used 3 features to predict sale prices while the final iteration used 4 features. A number of evaluation metrics (including the deviation information criterion (DIC)) were generated to gauge the accuracy of each model. It was determined that incorporating the 4th feature reduced the DIC and, hence, was a better for the data.

Introduction

The Ames Housing dataset, which is available on Kaggle.com, was compiled by Dean De Cock for use in data science education. It's an incredible dataset resource for data scientists and statisticians looking for a modernized and expanded version of the often-cited Boston Housing dataset.

The subject dataset contains 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, along with the sale price of each home. **For the purposes of this project, we will construct two models that leverage a subset (specifically, 3-4) of the most informative explanatory variables understand their ability to accurately predict sale prices for homes given their features.**

The features to be included are:

- `LotArea` - Lot size in square feet
- `OverallCond` - Rates the overall condition of the house (10=Excellent, 1=Very Poor)
- `GrLivArea` - Above grade (ground) living area in square feet

We will create an additional iteration of the model that incorporates the feature `TotRmsAbvGrd`, the total number of rooms above grade.

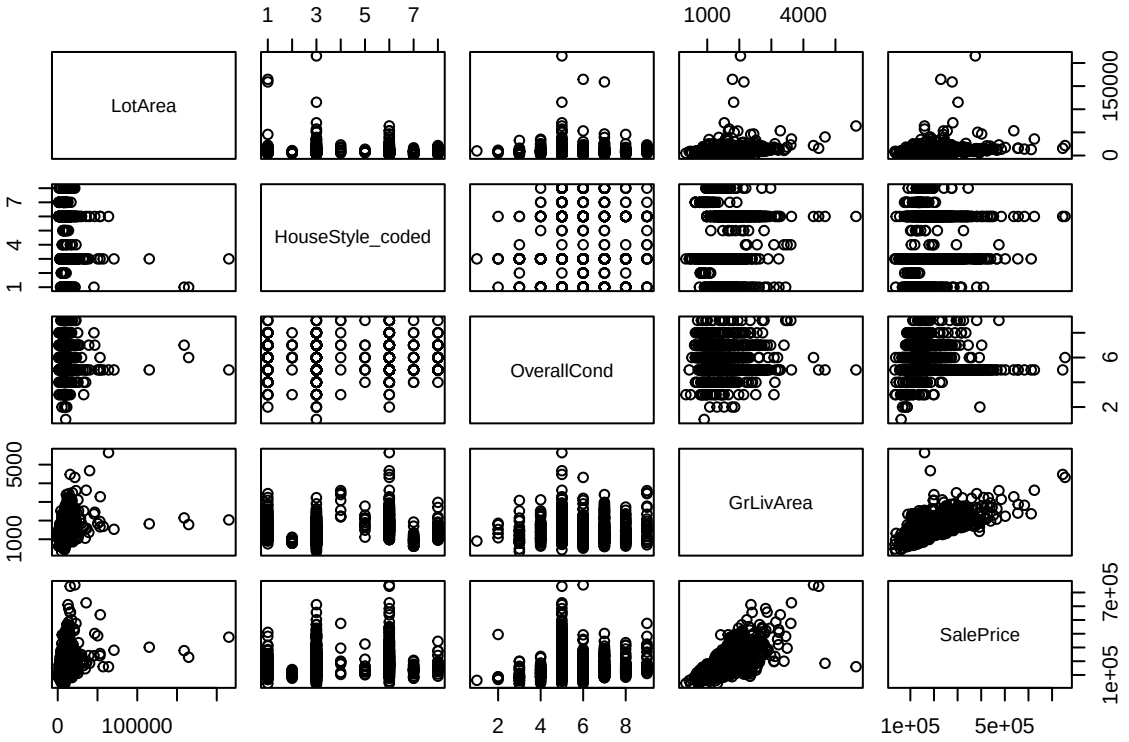
Load and Explore the Dataset

```
dat = read.csv(file="data_files/housing-price-data.csv", header=TRUE)

# Subset the raw data to features of interest and the response variable, SalePrice
dat1=dat[,c("LotArea","HouseStyle","OverallCond","GrLivArea","SalePrice")]

# Code the categorical variable `HouseStyle`.
dat1$HouseStyle_coded = factor(dat1$HouseStyle)
newdat1 = dat1[,c("LotArea","HouseStyle_coded","OverallCond","GrLivArea","SalePrice")]
head(newdat1)
```

```
##   LotArea HouseStyle_coded OverallCond GrLivArea SalePrice
## 1    8450           2Story           5      1710    208500
## 2    9600           1Story           8      1262    181500
## 3   11250           2Story           5      1786    223500
## 4    9550           2Story           5      1717    140000
## 5   14260           2Story           5      2198    250000
## 6   14115           1.5Fin           5      1362    143000
```

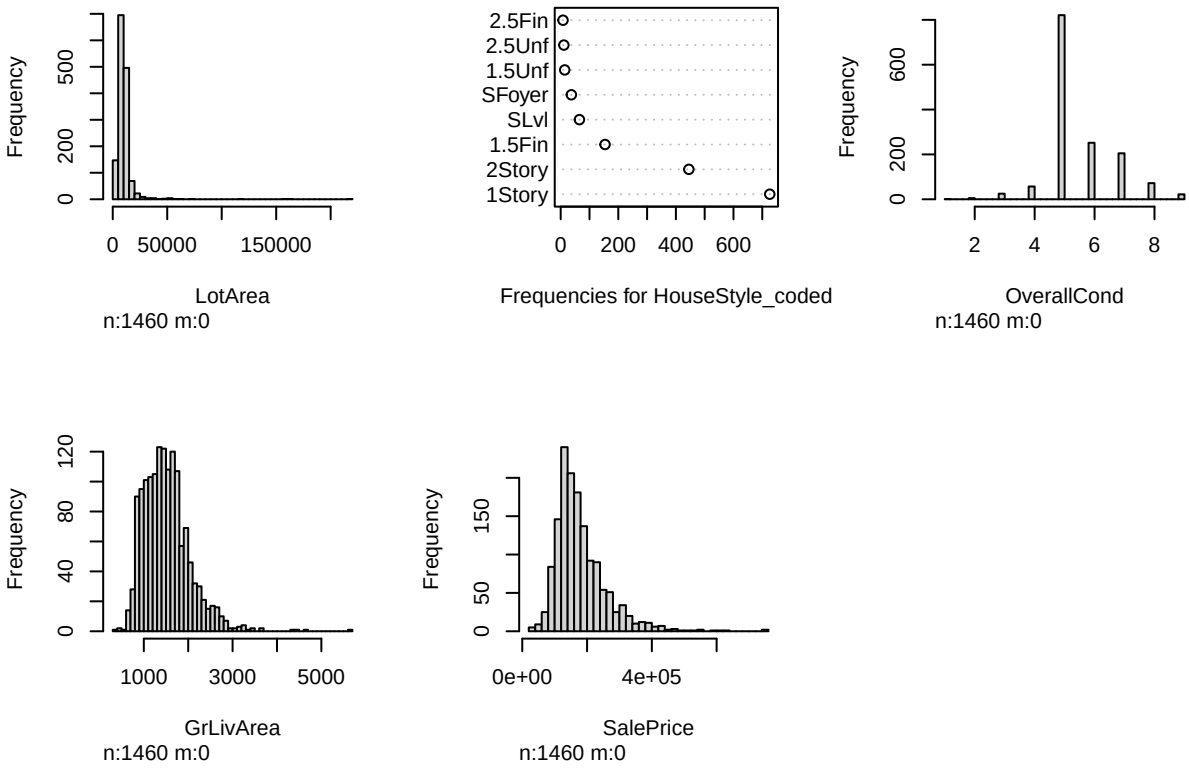
Now, let's look at the relationships among the features and their distributions:

```
library(Hmisc)
pairs(newdat1)
```



It looks like most, if not all, of the numeric features have a somewhat linear relationship with `SalePrice`.

```
hist.data.frame(newdat1)
```



It also appears that all of our non-categorical variables (including the response variable, `SalePrice`) have a somewhat normal distribution.

Postulate a Model

Since each of the features appear to have linear relationship with `SalePrice`, let's start off by postulating a heirarchical `JAGS` linear model of the descriptor features. **We will model with non-informative prior distributions (i.e., large σ^2) for the model coefficients.** The model will take the form of

$y_i \sim N(\mu_i, \sigma^2), \mu_i = \beta_0 + \beta x_{1i} + ... + \beta_k x_{ki}, \beta_k \sim N(0, 1e6)$

Where k is the number of descriptor variables in the data set and i is the number of observations.

Also, $y_i|x_i, \beta, \sigma^2 \overset{ind}{\sim} N(\beta_0 + \beta x_{1i} + ... + \beta_k x_{ki}, \sigma^2)$,

where the noninformative prior for σ^2 is modeled using an $InverseGamma(\alpha, \beta)$ distribution.

```
library("rjags")
newdat1 = na.omit(newdat1)

mod1_jags_string = " model {
    for (i in 1:n) {
        y[i] ~ dnorm(mu[i], prec)
        mu[i] = b0 + b[1]*LotArea[i] + b[2]*OverallCond[i]+ b[3]*HouseStyle_coded1.5Unf[i]
        +b[4]*HouseStyle_coded1Story[i]+b[5]*HouseStyle_coded2.5Fin[i]+ b[6]*HouseStyle_coded2.5Unf[i]
        + b[7]*HouseStyle_coded2Story[i]+ b[8]*HouseStyle_codedSFoyer[i]+ b[9]*HouseStyle_codedSLvl[i]
    }
    b0 ~ dnorm(0.0, 1.0/1.0e6)
    for (i in 1:9) {
        b[i] ~ dnorm(0.0, 1.0/1.0e6)
    }
    prec ~ dgamma(5/2.0, 5*10.0/2.0)
    sig2 = 1.0 / prec
    sig = sqrt(sig2)
} "
data_jags = list(y=newdat1$SalePrice,LotArea=newdat1$LotArea,OverallCond=newdat1$OverallCond,
                 HouseStyle_coded1.5Unf=as.numeric(newdat1$HouseStyle_coded=="1.5Unf"),HouseStyle_coded1Story=as
                 HouseStyle_coded2.5Fin=as.numeric(newdat1$HouseStyle_coded=="2.5Fin"),HouseStyle_coded2.5Unf=as
                 HouseStyle_coded2Story=as.numeric(newdat1$HouseStyle_coded=="2Story"),HouseStyle_codedSFoyer=as
                 HouseStyle_codedSLvl=as.numeric(newdat1$HouseStyle_coded=="SLvl"),n=nrow(newdat1))
params1 = c("b0","b", "sig")
inits1 = function() {
    inits = list("b0"=rnorm(1,0.0,100.0), "b"=rnorm(9,0.0,100.0), "prec"=rgamma(1,1.0,1.0))
}
mod1_jags = jags.model(textConnection(mod1_jags_string), data=data_jags, inits=inits1, n.chains=3)
```

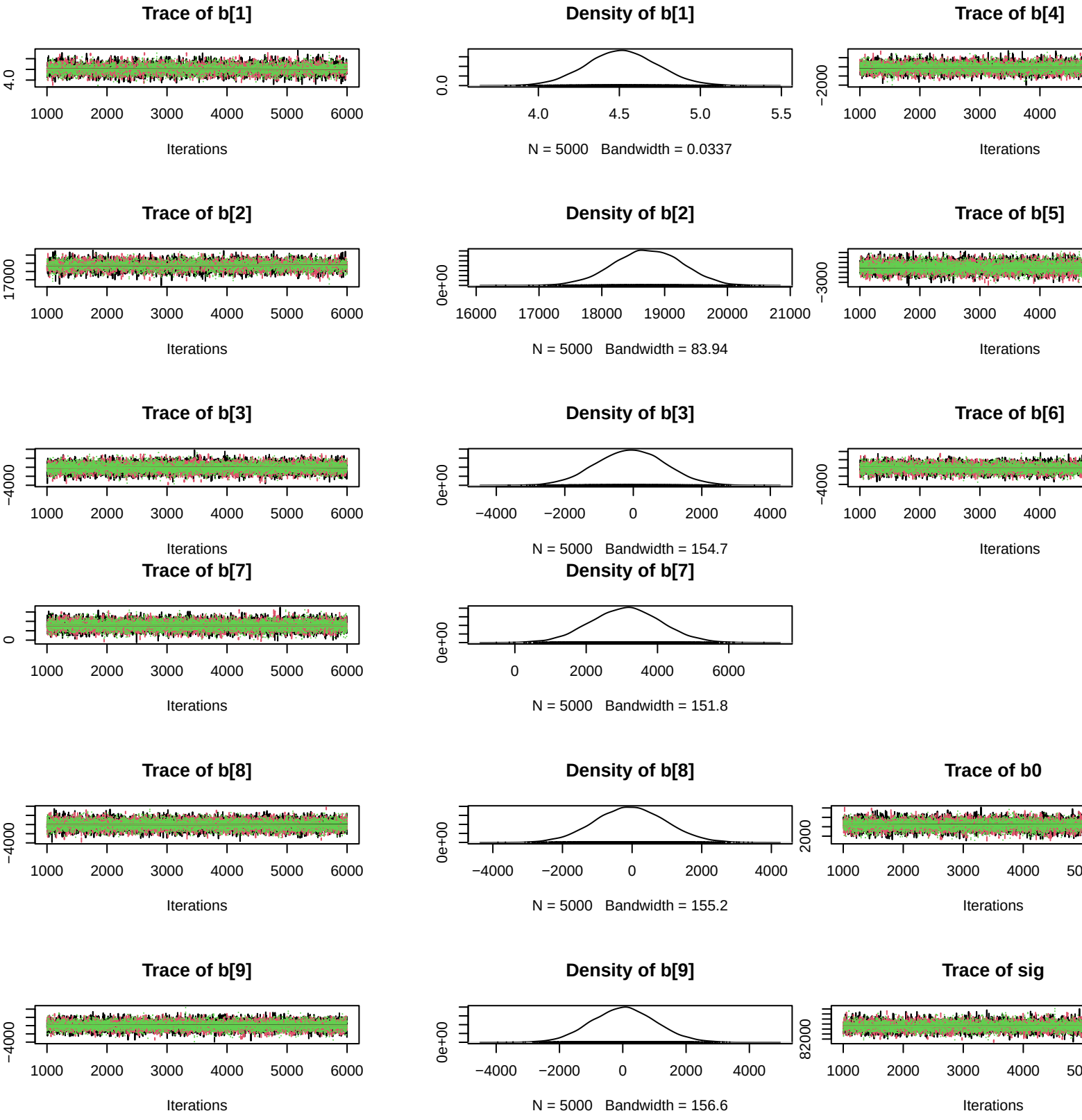
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1460
##    Unobserved stochastic nodes: 11
##    Total graph size: 17039
##
## Initializing model
```

Fit the Model using the Monte Carlo-Markov Chain (MCMC) Sampler

```
update(mod1_jags, 1000) # burn-in

mod1_jags_sim = coda.samples(model=mod1_jags,
                        variable.names=params1,
                        n.iter=5000)

mod1_jags_csim = do.call(rbind, mod1_jags_sim) # combine multiple chains
```

Check the Model by Examining Convergence Diagnostics

```
# gelman.diag(mod1_jags_sim)
# autocorr.diag(mod1_jags_sim)
# autocorr.plot(mod1_jags_sim)
# effectiveSize(mod1_jags_sim)
plot(mod1_jags_sim)
```



We can observe that the above traces for the $b[i]$s converge and the densities are all normal. **Since we have convergence, we can conclude that our assumed prior probability distributions for the coefficients were reasonable.** We can get a posterior summary of the parameters in our model.

```
summary(mod1_jags_sim)
```

```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
```

```
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean        SD  Naive SE Time-series SE
## b[1]     4.517    0.2175  0.001776       0.003026
## b[2] 18728.251  541.8517  4.424201       7.789448
## b[3]   -78.955  998.3323  8.151349       8.298431
## b[4]  1841.044  975.3587  7.963771       8.220008
## b[5]    33.146  997.9295  8.148060       8.112532
## b[6]    14.112 1000.3479  8.167806       8.167776
## b[7]  3114.012  980.0718  8.002253       8.202599
## b[8]   -51.577 1001.7787  8.179489       8.168780
## b[9]    33.778 1010.7271  8.252553       8.195989
## b0    4372.687  995.3406  8.126922       9.157956
## sig  87120.831 1739.7686 14.205151      17.459194
##
## 2. Quantiles for each variable:
##
##           2.5%       25%       50%       75%    97.5%
## b[1]     4.094     4.369     4.514     4.661     4.95
## b[2] 17642.194 18363.512 18731.602 19105.570 19763.27
## b[3] -2044.636  -761.251   -75.937   601.185  1874.55
## b[4]   -73.025  1183.912  1841.652  2497.476  3750.51
## b[5] -1927.512  -635.709    34.142   707.466  2001.06
## b[6] -1956.082  -659.364    20.151   698.547  1969.87
## b[7]  1191.680  2456.902  3121.087  3773.137  5052.39
## b[8] -2007.729  -737.091   -52.317   627.458  1924.65
## b[9] -1903.465  -652.223    32.255   710.465  2052.98
## b0    2423.465  3701.896  4376.390  5039.488  6322.26
## sig  83816.181 85926.547 87089.461 88281.096 90614.25
```

We notice that the standard deviation is quite high for the coefficients associated with the coded categorical variable `HouseStyle_coded`. This makes sense since the x_i for this feature can only take on values of 0 or 1.

Residual checks

In a Bayesian model, we have distributions for residuals, but we'll simplify and look only at the residuals evaluated at the posterior mean of the parameters.

```
X = cbind(rep(1.0, data_jags$n), data_jags$LotArea,data_jags$OverallCond,data_jags$HouseStyle_coded1.5Unf,
          data_jags$HouseStyle_coded1Story,data_jags$HouseStyle_coded2.5Fin,data_jags$HouseStyle_coded2.5Unf,
          data_jags$HouseStyle_coded2Story,data_jags$HouseStyle_codedSFoyer,data_jags$HouseStyle_codedSLvl)
head(X)
```

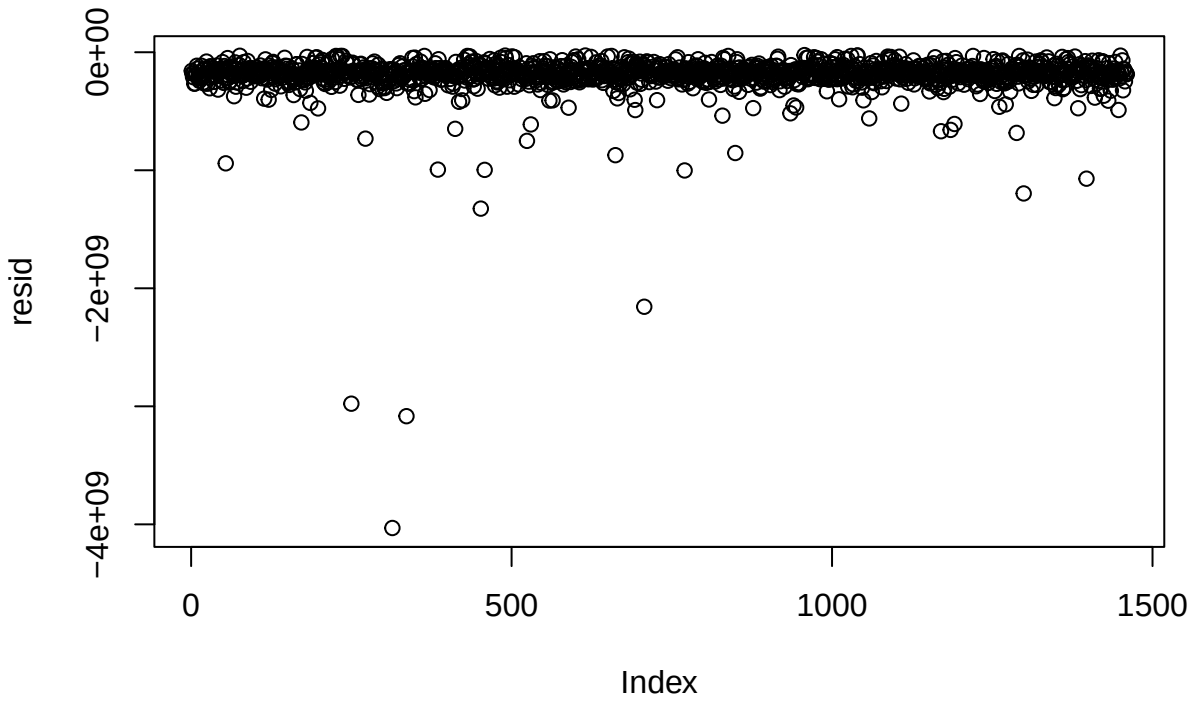
```
##      [,1]  [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1  8450    5    0    0    0    0    1    0     0
## [2,]    1  9600    8    0    1    0    0    0    0     0
## [3,]    1 11250    5    0    0    0    0    1    0     0
## [4,]    1  9550    5    0    0    0    0    1    0     0
## [5,]    1 14260    5    0    0    0    0    1    0     0
## [6,]    1 14115    5    0    0    0    0    0    0     0
```

```
(pm_params = colMeans(mod1_jags_csim)) # posterior mean
```

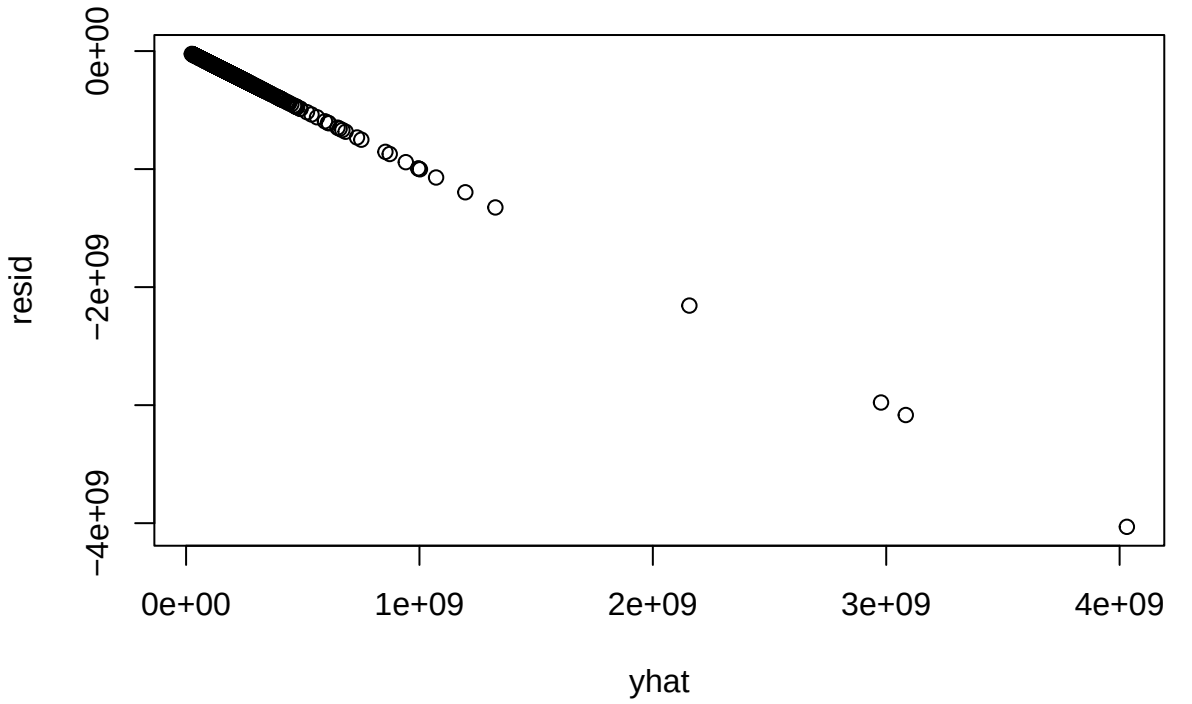
```
##         b[1]         b[2]         b[3]         b[4]         b[5]         b[6]
##     4.516578 18728.250717   -78.955222  1841.044123    33.146095    14.112366
##         b[7]         b[8]         b[9]           b0          sig
##  3114.011976   -51.577004    33.777542  4372.687403 87120.831409
```

Create an inference vector, $\hat{y}$, generated by the model and examine the residuals.

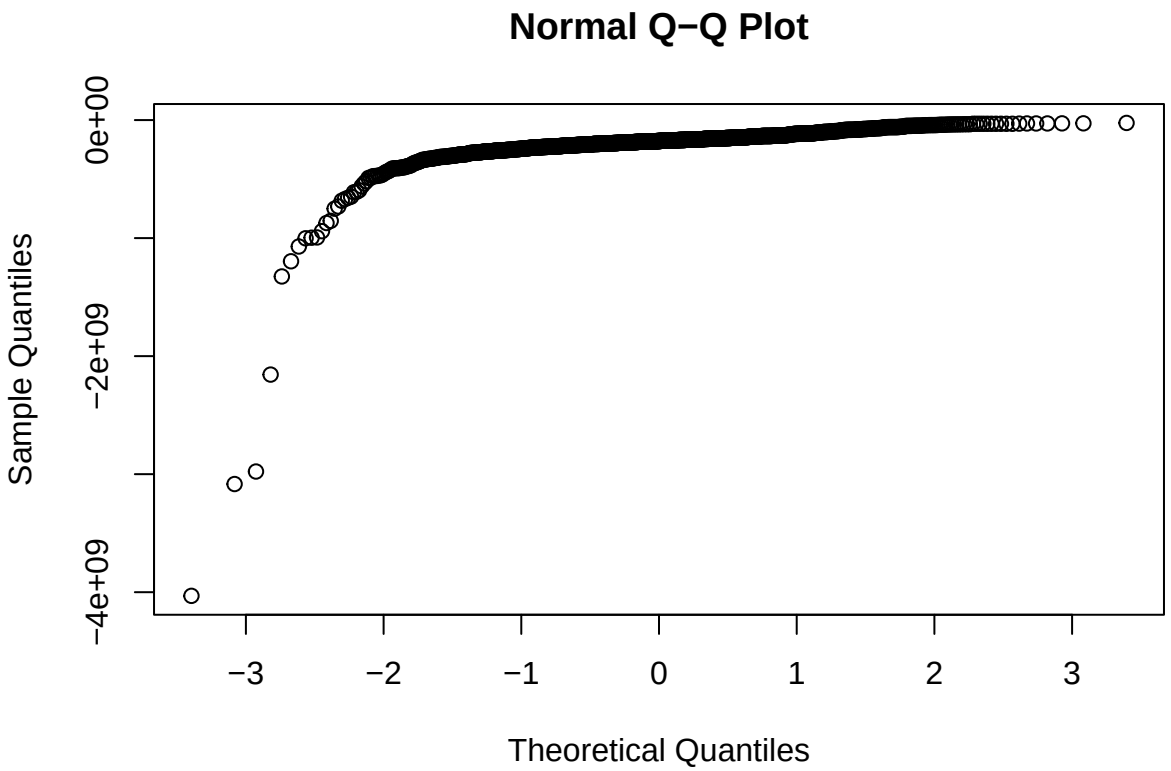
```
yhat = drop(X %*% pm_params[1:10])
resid = data_jags$y - yhat
plot(resid) # residuals against data index
```




```
plot(yhat, resid) # residuals against predicted values
```




```
qqnorm(resid) # checking normality of residuals
```



We see that the residuals are mostly concentrated about zero, however, there are a few outliers. Having this information would have us conclude that there are other features that could be added to the model to reduce the error of its inference generation.

Iterate with another model

As mentioned previously, we will build another linear model that incorporates the numeric feature `TotRmsAbvGrd`, which is the total rooms above grade (does not include bathrooms).

```
#Adjust the dataset to include `TotRmsAbvGrd`.
dat2=dat[,c("LotArea","HouseStyle","OverallCond","GrLivArea","TotRmsAbvGrd","SalePrice")]
dat2$HouseStyle_coded = factor(dat2$HouseStyle)
newdat2 = dat2[,c("LotArea","HouseStyle_coded","OverallCond","GrLivArea","TotRmsAbvGrd","SalePrice")]

# head(newdat2)
```

The `JAGS` model becomes:

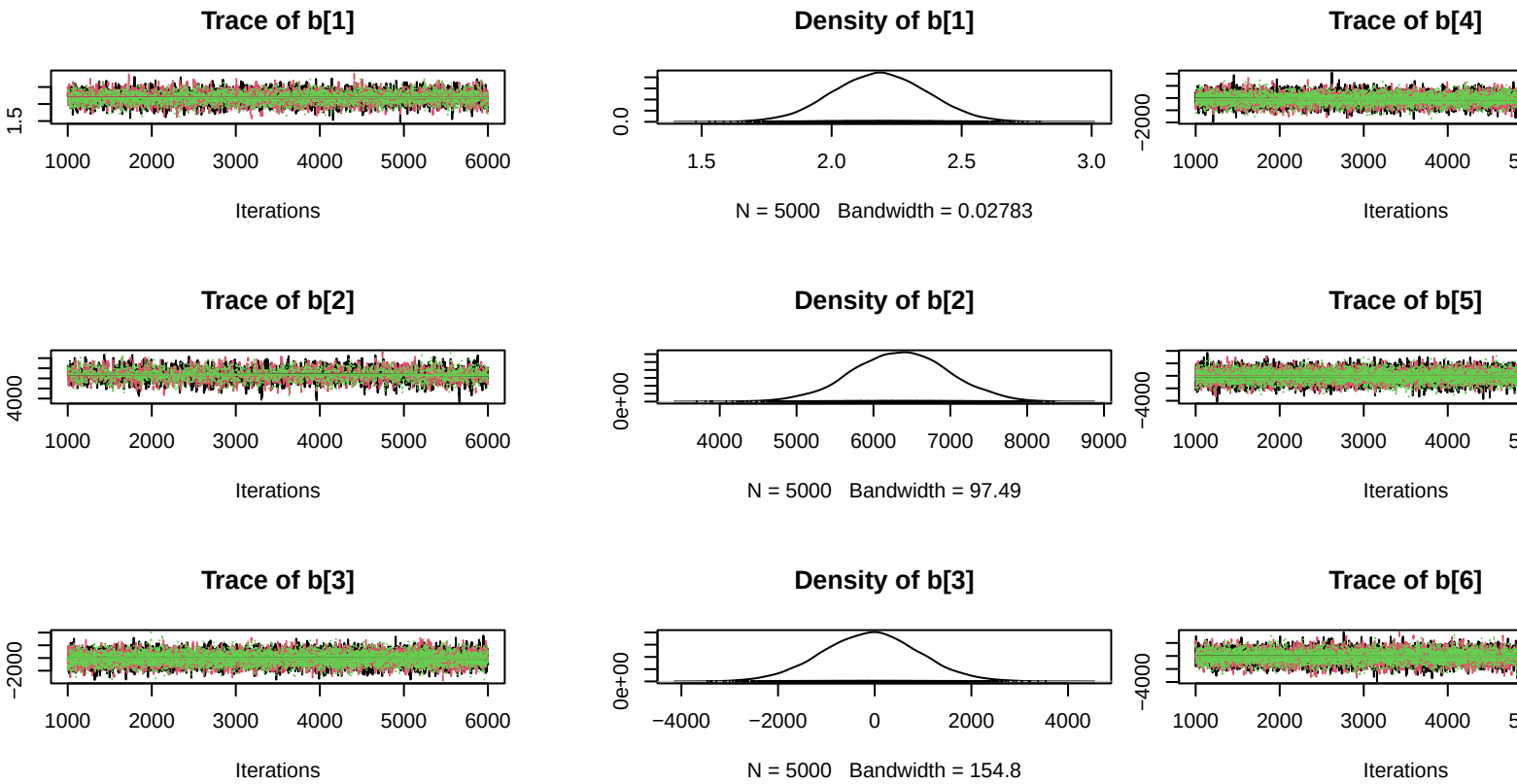
```
mod2_jags_string = " model {
    for (i in 1:n) {
        y[i] ~ dnorm(mu[i], prec)
        mu[i] = b0 + b[1]*LotArea[i] + b[2]*OverallCond[i]+ b[3]*HouseStyle_coded1.5Unf[i]
        + b[4]*HouseStyle_coded1Story[i]+ b[5]*HouseStyle_coded2.5Fin[i]+ b[6]*HouseStyle_coded2.5Unf[i]
        + b[7]*HouseStyle_coded2Story[i]+ b[8]*HouseStyle_codedSFoyer[i]+ b[9]*HouseStyle_codedSLvl[i]
        + b[10]*TotRmsAbvGrd[i]
    }
    b0 ~ dnorm(0.0, 1.0/1.0e6)
    for (i in 1:10) {
        b[i] ~ dnorm(0.0, 1.0/1.0e6)
    }
    prec ~ dgamma(5/2.0, 5*10.0/2.0)
    sig2 = 1.0 / prec
    sig = sqrt(sig2)
} "
data2_jags = list(y=newdat2$SalePrice, LotArea=newdat2$LotArea,OverallCond=newdat2$OverallCond,
                  HouseStyle_coded1.5Unf=as.numeric(newdat2$HouseStyle_coded=="1.5Unf"),HouseStyle_coded1Story=as
                  HouseStyle_coded2.5Fin=as.numeric(newdat2$HouseStyle_coded=="2.5Fin"),HouseStyle_coded2.5Unf=as
                  HouseStyle_coded2Story=as.numeric(newdat2$HouseStyle_coded=="2Story"),HouseStyle_codedSFoyer=as
                  HouseStyle_codedSLvl=as.numeric(newdat2$HouseStyle_coded=="SLvl"),TotRmsAbvGrd=dat2$TotRmsAbvGr
```

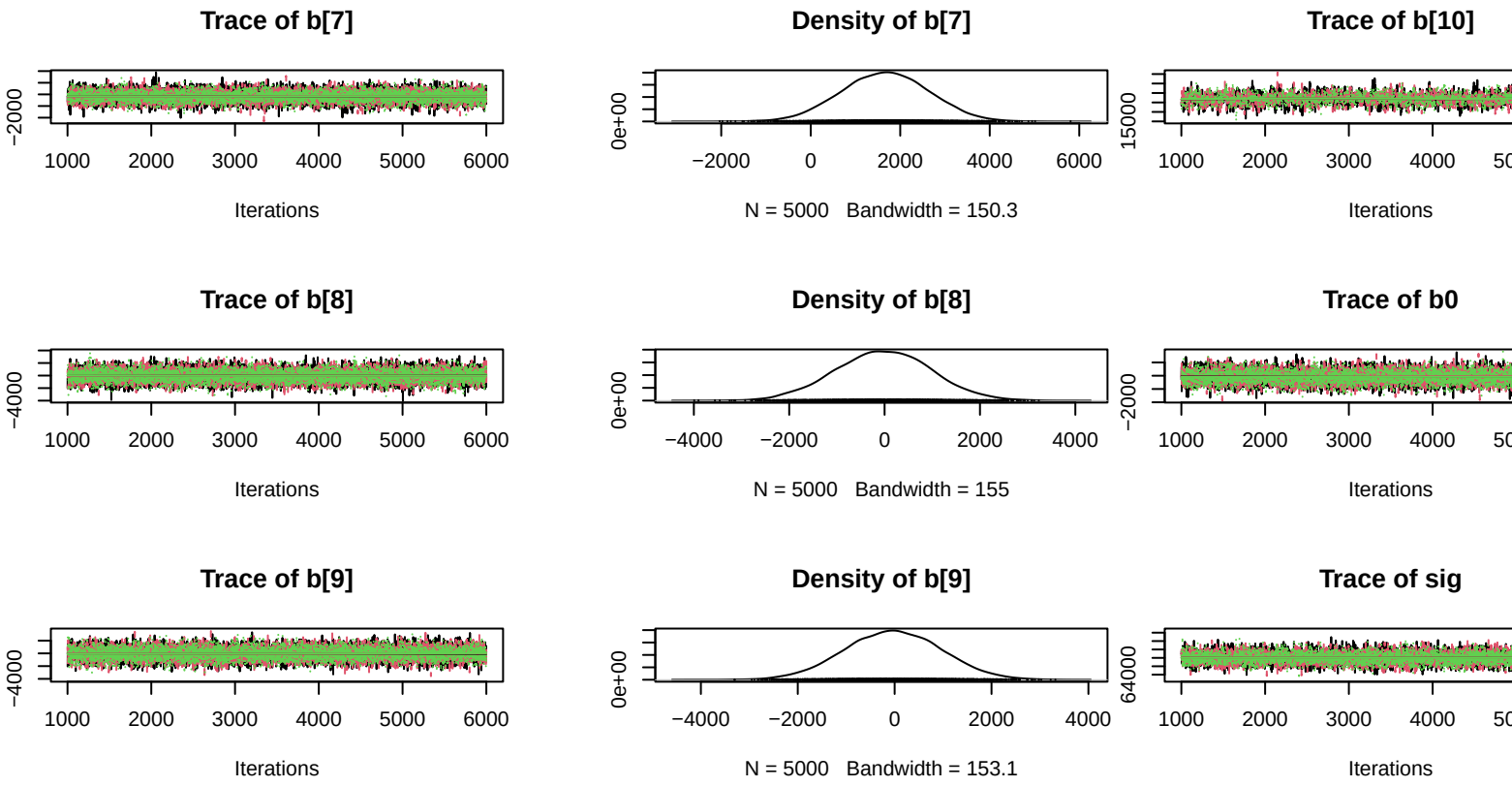
```
                n=nrow(newdat2))
params1 = c("b0","b", "sig")
inits1 = function() {
    inits = list("b0"=rnorm(1,0.0,100.0), "b"=rnorm(10,0.0,100.0), "prec"=rgamma(1,1.0,1.0))
}
mod2_jags = jags.model(textConnection(mod2_jags_string), data=data2_jags, inits=inits1, n.chains=3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1460
##    Unobserved stochastic nodes: 12
##    Total graph size: 18584
##
## Initializing model
```

```
update(mod2_jags, 1000) # burn-in

mod2_jags_sim = coda.samples(model=mod2_jags,
                         variable.names=params1,
                         n.iter=5000)

mod2_jags_csim = do.call(rbind, mod2_jags_sim) # combine multiple chains
```

```
# gelman.diag(mod2_jags_sim)
# autocorr.diag(mod2_jags_sim)
# autocorr.plot(mod2_jags_sim)
# effectiveSize(mod2_jags_sim)
plot(mod2_jags_sim)
```






```
summary(mod2_jags_sim)
```

```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean        SD  Naive SE Time-series SE
## b[1]       2.186    0.1797  0.001467        0.00252
## b[2]    6350.964  629.2909  5.138139       13.99366
## b[3]     -85.461 1000.3501  8.167825        8.17994
## b[4]    1807.089  955.5717  7.802210        8.39536
## b[5]     -65.019 1001.1059  8.173995        8.24303
## b[6]     -92.245  994.5384  8.120372        8.06186
## b[7]    1663.906  970.1813  7.921497        8.20600
## b[8]     -29.108 1000.4141  8.168347        8.00172
## b[9]     -30.738  988.4340  8.070530        7.97518
## b[10] 17397.253  565.3942  4.616425       12.91622
## b0     1849.788  983.0725  8.026753        9.76640
## sig   68018.299 1310.4262 10.699585       12.58885
##
## 2. Quantiles for each variable:
##
##             2.5%       25%       50%       75%     97.5%
## b[1]       1.835     2.064     2.186     2.308     2.535
## b[2]    5094.464  5924.712  6355.902  6774.678  7585.594
## b[3]   -2054.609  -759.354   -81.693   579.417  1889.663
## b[4]     -83.459  1159.396  1811.860  2452.689  3657.714
## b[5]   -2007.161  -738.390   -67.275   605.223  1883.109
## b[6]   -2048.844  -773.298   -91.712   573.368  1850.288
## b[7]    -219.665  1006.306  1669.946  2324.619  3546.856
## b[8]   -2005.066  -705.068   -26.241   653.775  1934.579
```

```
## b[9]  -1967.781  -710.557   -26.574   649.336  1896.568
## b[10] 16291.768 17012.837 17400.983 17776.028 18505.109
## b0      -48.731  1185.775  1838.222  2506.332  3828.529
## sig   65536.038 67120.223 67999.223 68889.031 70642.871
```

The model coefficient for `TotRmsAbvGrd`, b[10], is quite large as compared with the other coefficients, indicating that this feature is a stronger driver of `SalePrice`. As with the previous model, the trace plots show convergence for all the coefficients and normal densities.

Let's check the residuals for this model.

```
X = cbind(rep(1.0, data2_jags$n), data2_jags$LotArea,data2_jags$OverallCond,data2_jags$HouseStyle_coded1.5Unf,
          data2_jags$HouseStyle_coded1Story,data2_jags$HouseStyle_coded2.5Fin,data2_jags$HouseStyle_coded2.5Unf,
          data2_jags$HouseStyle_coded2Story,data2_jags$HouseStyle_codedSFoyer,data2_jags$HouseStyle_codedSLvl,da
head(X)
```

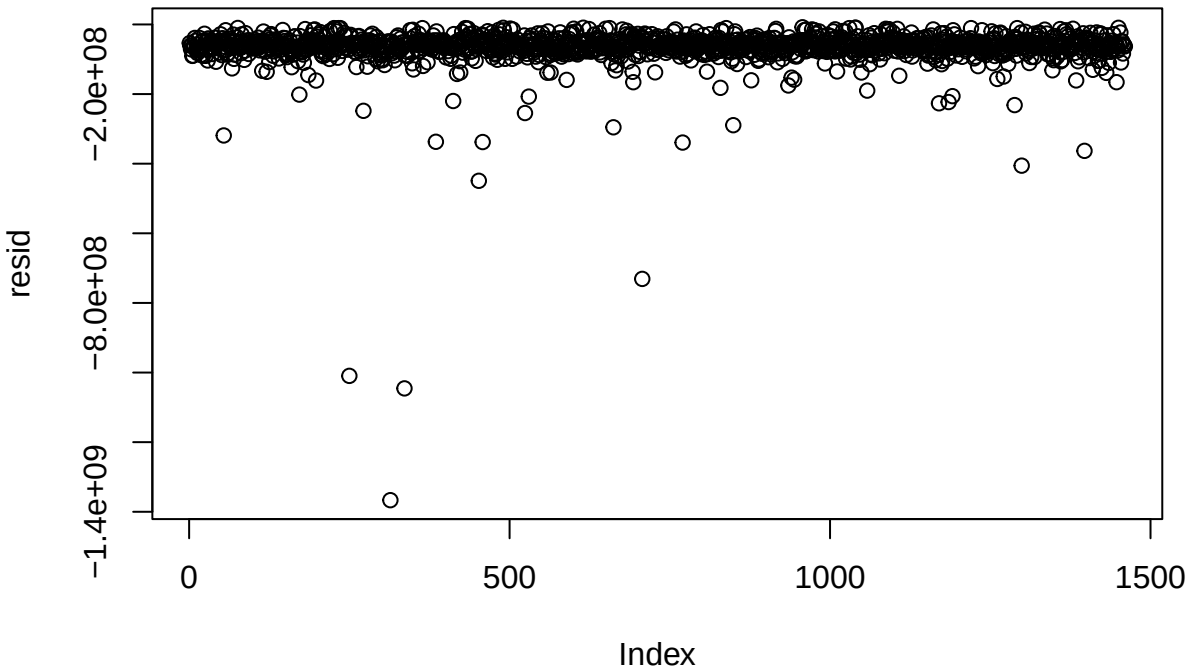
```
##      [,1]  [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,]    1  8450    5    0    0    0    0    1    0     0     8
## [2,]    1  9600    8    0    1    0    0    0    0     0     6
## [3,]    1 11250    5    0    0    0    0    1    0     0     6
## [4,]    1  9550    5    0    0    0    0    1    0     0     7
## [5,]    1 14260    5    0    0    0    0    1    0     0     9
## [6,]    1 14115    5    0    0    0    0    0    0     0     5
```

```
(pm_params = colMeans(mod2_jags_csim)) # posterior mean
```

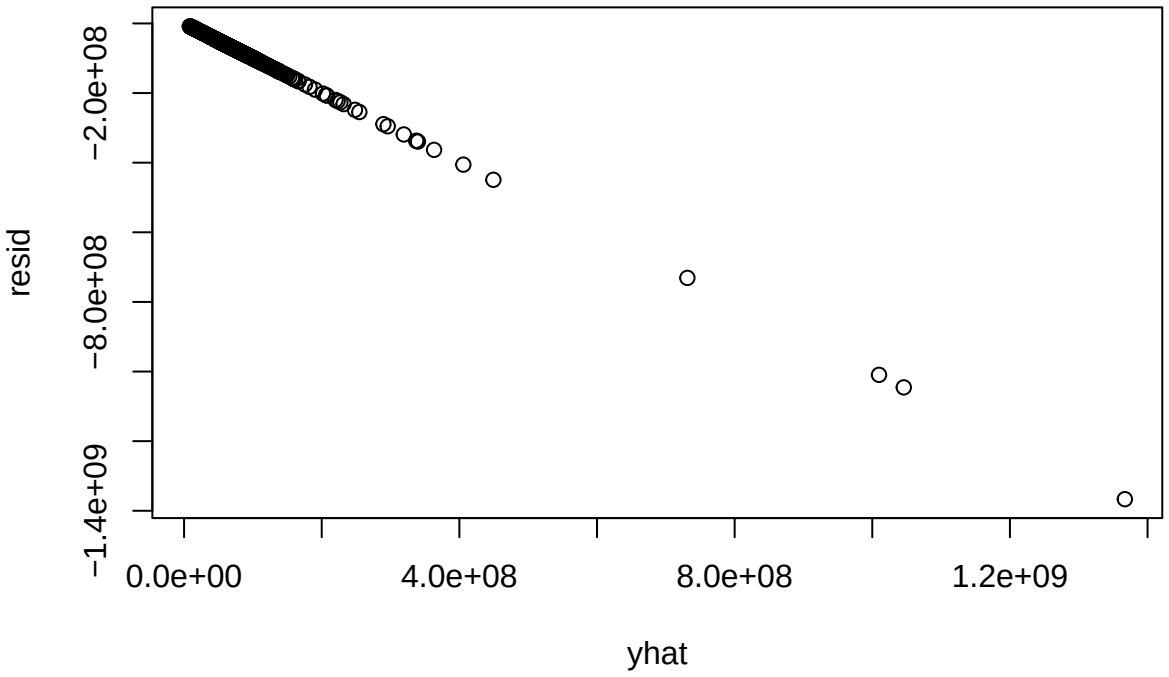
```
##         b[1]         b[2]         b[3]         b[4]         b[5]         b[6]
##     2.186195  6350.963759   -85.461073  1807.088694   -65.019211   -92.245052
##         b[7]         b[8]         b[9]        b[10]           b0          sig
##  1663.906356   -29.107525   -30.737518 17397.252686  1849.787918 68018.299072
```

Create an inference vector, $\hat{y}$, generated by the model and examine the residuals.

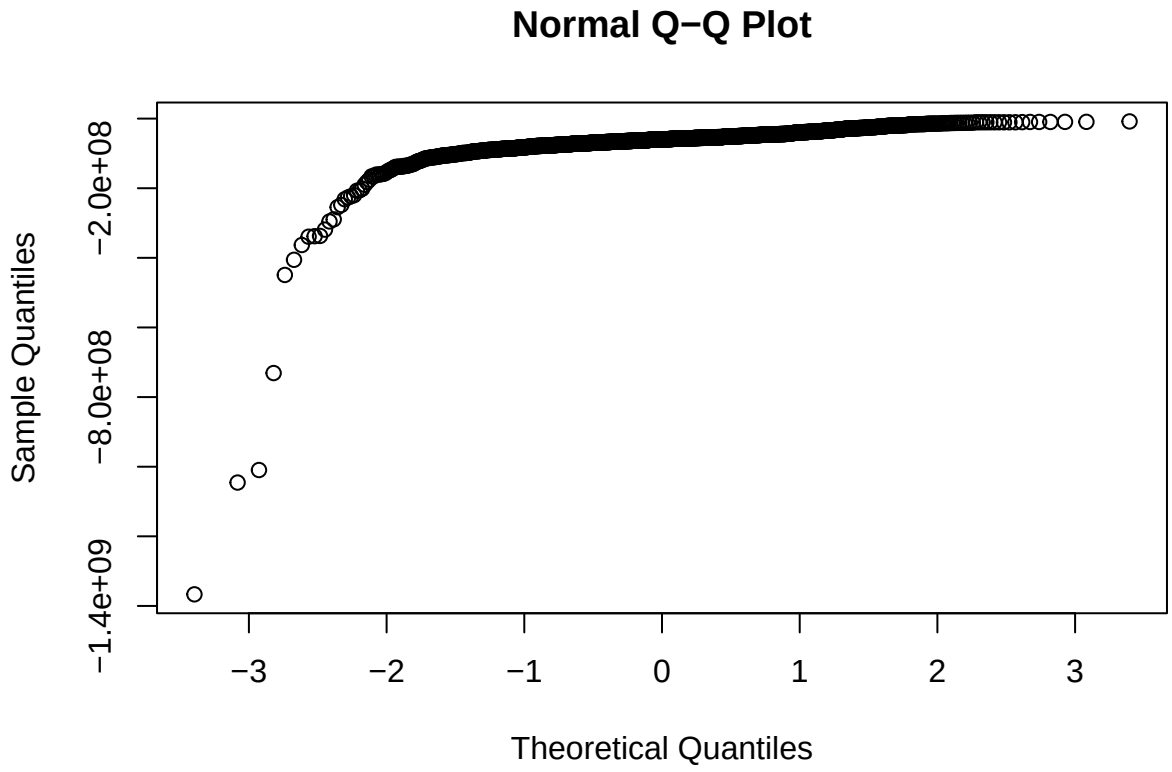
```
yhat = drop(X %*% pm_params[1:11])
resid = data2_jags$y - yhat
plot(resid) # residuals against data index
```




```
plot(yhat, resid) # residuals against predicted values
```




```
qqnorm(resid) # checking normality of residuals
```



We see that these plots are very similar to those generated by the first model, however, the magnitude of the residual outliers for the second model is much less than those generated by the first model. We can conclude that adding the additional feature has indeed improved the model's prediction capability.

Comparison of DIC metrics for both models.

Let's compare the two models using the DIC metric:

```
dic.samples(mod1_jags, n.iter=1e3)
```

```
## Mean deviance:  37363
## penalty 3.162
## Penalized deviance: 37366
```

```
dic.samples(mod2_jags, n.iter=1e3)
```

```
## Mean deviance:  36640
## penalty 3.66
## Penalized deviance: 36644
```

Upon examination of the deviance information criterion (DIC), the penalized deviance of the second model (which incorporates `TotRmsAbvGrd`) is lower than the first model. We conclude that the second model that incorporates the additional feature is a better fit (and predictor) for the data. Further confirmation of this conclusion is found in the relatively larger magnitude in the b[10] coefficient.