Week 8: Model Checking

by Michael Escobar

March 4, 2019

1 Introduction

"Applied Statisticians are statisticians who take their assumptions seriously" I.R. Savage.

Basically, good applied statisticians check ther assumptions. So, we will discuss ways of looking at the data, the model, and ther results to see if everything makes sense. Some of the objectives:

- Look for unusual observations
- Comparison of different models
- General goodness-of-fit statistics.

Also, the above statistics can form the basis of choosing a good model. These methods are also presented below.

After the different methods are discussed there are worked examples showing how to add code to an Openbug/Jags program to calculate some of these values.

Aside: Is this Bayesian? The view here is that checking model assumption is good applied statistical analysis wheather the statistical analysis is Bayesian or Frequentist. Here we look at more ad hoc methods. Next week we look at more formal Bayesian methods for model selection. There are ways to put this on a more formal footing by putting prior beliefs on "the mathematical form of the model". For example one can model general uncertainty on the distribution functions and likelihoods by using Dirichlet processes. These methods are not discussed here.

(Reference: .... Need to include some... sorry, that will be done at a later date.)

It should also be noted that this Bayesian work builds off of the extensive model checking work that was done in the non-bayesian statistical literature which was done especially in the 1980's (and with earlier roots in the '60's and '70's). As part of this work, there was an emphasis on practicing good habits like plotting residuals and other early visualization techniques.

2 Looking at Individual Observations

First, let us look at ways to identify unusual observations.

2.1 Some basic measures for individual observations

Gelfand et al (1992) suggested the following different type of measures of how unusual a particular observation might be:

1. Residuals. Defined by:
$$y_i - E(Y_i).$$
2. Standardized Residuals:
$$\frac{y_i - E(Y_i)}{\sqrt{V(Y_i)}}.$$
3. The chance of getting a more extreme observations. Define by:
$$\min[P(Y_i < y_i), P(Y_i > y_i)].$$

4. Chance of a more surprised observation. Defined by:

$$P(Y_i|f(Y_i) \leq f(y_i)).$$

5. The predictive ordinate of the observation. Defined by:

$$f(y_i).$$

In the above, note the definition of some of the terms in the above equations:

- The observed value is y_i.
- The probability function $P(\cdot)$ and the density/probability mass function $f(\cdot)$ are for the predictive distribution of the observation.
- The expectation function $E(\cdot)$, and the variance function $V(\cdot)$ are based on the perdictive distribution of the variable Y_i.

Some things to note about the above:

1. For types (1) and (5), we need to calibrate these statistics. That is, one needs to know what values would be considered good or bad values. That is, what values of the statistics would cause concerns. The other three statistics are already standardized so one would know what values would flag that observation as being unusual.
2. Statistics (3), the chance of getting a more extreme observation, is similar to a 2-sided p-value. That is, it looks at the probability of seeing values larger (or smaller) than the observed value. That is, one looks at the two tail areas (and take the smaller area). Please see figure 1.
3. The surprise statistics, statistics (4), looks at the probabiltiy of seeing a value which is less likely then the observed observation. An alternative mathematical description of this statistics is:

$$\int_{\{z|f(z)\leq f(y_i)\}} f_{Y_i}(z)dz.$$

   Basically, this statistic is similar to statistics (3) when the predictive distribution is unimodal. However, when the distribution is multimodal, then this probability integrates over the values of the random variable where the density is less than the density at the observed value. Please see figure 1.
4. The predicitive ordinate is the value of the density/probabiity mass function at the observed value. In general, this measures how rare the observation is. Please see figure 2

2.2 What to use for the predictive distribution

When using a (posterior) predicitive distribution there is an issue of what data to use to define the posterior predictive distribution to evaluate the observed data. If one uses the observation which is being checked as part of the data to create the posterior predictive distribution then one is "using the data twice". This can lead to over fitting. However, it is easer to simple "use the data twice" so that is one way to create the predictive distribution. Alternatively, there are some methods to consider to prevent this. So, first two methods of avoiding the double ues of data are described and for the methods which uses the data twice are discussed in the worked examples.

Here are some approaches:

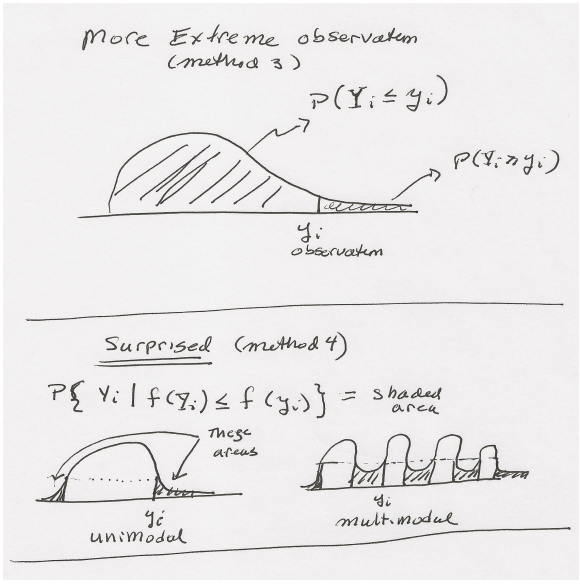

Figure 1: TOP: The "more extreme observation". This looks at the area under the curve to the right and the left of the observed value. The more extreme observation is the small of these two areas. It is sort of like a one tailed p-value. BOTTOM: The surprise statistics. This is similar to the value of "the more extreme observation". If the density is unimodal, then this statistic is like a two tailed test. When the density is multimodal, then it contains the area of "all the valleys" of the density.

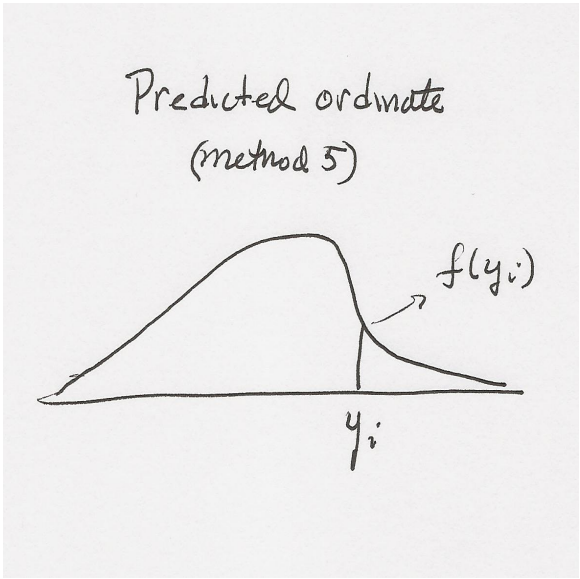

Figure 2: The predictive ordinate. Here we are looking at the value of the density at the location Y_i. We should not see many low values since a low value means that the value is rare.

- Use the data twice. This is the easiest thing to do. However, besides seeming basically wrong from a philosophic point of view, it also has a basic problem of overfitting. Still, this is the primary method used in the below examples.
- Training/Testing/Validating data sets. Here, the data is divided into two or three different data sets. Perhaps a .5/.25/.25 split. First, one does all the fitting on the training data set. Then, the other two datasets can be used to choose the best model and validate. If one has a small data set, then this can severaly limit the power of the analysis.
- Have a "hold out sample". Fit everthing on most of the data with just some hold out samples to check the model. Usually the hold out sample is a small percentage of the data. So, the reduction in power is not very much.
- k-fold cross validation. Here the data is separated into k blocks. So, if one divides the data into, say 10 blocks, then one would loop through the 10 different blocks using the the other 9 blocks to fit the model. This model would be used to predict the values of the left out block. A drawback of this method is that if one makes 10 blocks, then the analysis takes 10 times longer to run.
- Leave one out cross validation. So, this is a more extreme version of the k-fold cross validation since each block has only one observation. This might seem crazy because it seems like one might need to run the analysis n times, where n is the sample size. However, sometimes there is a mathematical trick that can be used. Such a trick exist for the predictive ordinate (method 5 above). More about this trick is discussed below in the section on conditional predictive ordinate.

3 Goodness of Fit statistics

Besides looking at the statistics for just one observation, there could also be interest in looking at methods look for an overall goodness of fit. If the obsevations are independent, then one can look at sums of the different individual observation statistics. Commons ones would be the following:

- Sum of squared errors.
$$\sum_{i=1}^{n} \left(y_i - E(Y_i)\right)^2.$$
- A type of Pearson chi-squared:
$$\sum_{i=1}^{n} \frac{\left(y_i - E(Y_i)\right)^2}{V(Y_i)}.$$
- A deviance statistics:
$$\sum_{i=1}^{n} -2 * \log(p(y_i|\theta_i)),$$
where $p(Y_i|\theta_i)$ is the likelihood of the observed value y_i given the parameter θ_i.

As a goodness of fit statistics, it would be useful to calibrate these statistics. That is, it would be helpful to know how large these statistics would be under the model.

Also, the above statistics might be useful to compare two models. This is because the above statistics tell how well the model fits the data. If one used the data twice, then a more flexible model with more parameters would usually fit better when the same data is used to build the model and to assess the goodness of fit. So, there is a real danger of overfitting in such a case. To prevent this, there are related statistics which are build off the above statistics which penalize models with more parameters.

Before we get to these other models. Let us first look at an example of how to fit the above.

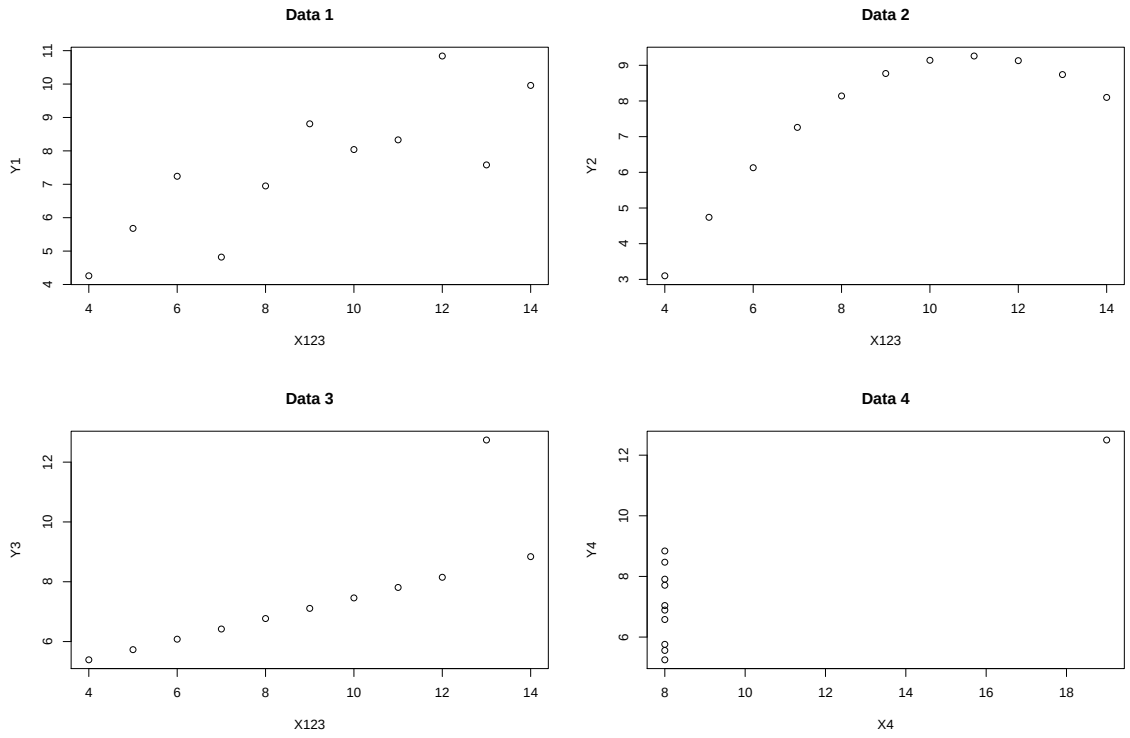

Figure 3: These data sets are from Anscombe (1973, American Statistician). All the summary statisitics for the 4 datasets are identical. However, as one can see from the plots, the relationship between X and Y appear very different. Data 1 looks like what data from a linear regression should look. In Data 2, X has a quadratic relationship with Y. Data 3 has an outliner. For Data 4, there is a very strong leverage point which entirely dictates the fitted regression line.

4 Worked example

To illustrate how to apply these methods, let's apply these methods to the data set know as Anscombe's quartet. These is a set of four data sets. The data sets were constructed so that each of the four datasets have the same summary statistics. These datasets are plotted in figure 3. Note that there are obvious differences in the four datasets. These datasets were created to show that just the statistics alone don't telll the whole story of the data.

For the computer code that runs this example and for the output from R, please refer to the files: Anscombe4ResDiagShort.txt, Anscombe4ResDiagShortOutput.txt, Anscombe4ResDiagLong.txt, Anscombe4ResDiagLongOutput.txt, and Anscombe4.csv.

Here is the data for these datasets:

```
> print(anscombe)
   X123    Y1   Y2    Y3 X4    Y4
1    10  8.04 9.14  7.46  8  6.58
2     8  6.95 8.14  6.77  8  5.76
3    13  7.58 8.74 12.74  8  7.71
4     9  8.81 8.77  7.11  8  8.84
5    11  8.33 9.26  7.81  8  8.47
6    14  9.96 8.10  8.84  8  7.04
7     6  7.24 6.13  6.08  8  5.25
8     4  4.26 3.10  5.39 19 12.50
```

```
9    12 10.84 9.13  8.15  8  5.56
10    7  4.82 7.26  6.42  8  7.91
11    5  5.68 4.74  5.73  8  6.89
```

Do model the simple linear regression for each of these datasets, the following basic model will be used:

```
model{
  for(i in 1:N){
    y[i]~dnorm(mu[i],tau)
mu[i] <- beta0 + beta1*x[i]

}

beta0 ~dnorm(0, .01)
beta1 ~dnorm(0, .01)
tau<- 1/std/std
std ~ dunif(0,20)
}
```

4.1 Predicted observations

For many of the residuals and for the overall goodness of fit statistics, there is an issue of knowing how large these values should be. One method of determining how much these values could vary is to create a predicted value for each observation. Then, one can calculate the different residuals and other statistics for both the actual observed values and for the predicted values. From the predicted values, one can obtain the posterior predictive distribution of any of the test functions. Using these generated prective values, one can find out how much any diagnostic statistic could be expected to vary or how extreme it might be.

In the above data set, this is done by inserted the below line right after modellling `y[i]` in the code. So, the following line is inserted in the code:

```
y.rep[i]~dnorm(mu[i],tau)
```

Then, when one creates one of the residual statistics such as:

```
res[i]<-(y[i]-mu[i])                #  estimate of the residuals for this model
stdres[i]<-res[i]*sqrt(tau)        #  for the standardized residuals
```

Then one can insert the lines below:

```
res.rep[i]<- y.rep[i] - mu[i]
stdres.rep[i]<- res.rep[i]*sqrt(tau)
```

These lines will create a distribution for the `res[i]` and `stdres[i]` under the fitted model. So, the credible region for `res.rep[i]` will be the expected 95% credible range for where we would expect the values for `res[i]`. If `res[i]` is outside this range, then we might question if this value is too big.

4.2 Amended Code for Diagnostics

The below code is the same linear regression model that is fitted above with additional code to obtain different diagnostic values.

```
model{
  for(i in 1:N){
    y[i]~dnorm(mu[i],tau)
mu[i] <- beta0 + beta1*x[i]

#####################################
#        model checking steps are here.........

#   getting the residuals for the observed values...
#   note: I am deviating from the bugs manual... not getting the moments.

res[i]<-(y[i]-mu[i])                 #  estimate of the residuals for this model
stdres[i]<-res[i]*sqrt(tau)        #  for the standardized residuals

dev1.obs[i]<-pow(res[i],2)
dev2.obs[i]<-pow(stdres[i],2)

#  getting a replicated sample..... This is a sample of the predictive distribution

y.rep[i]~dnorm(mu[i],tau)
p.smaller[i] <-step(y[i]-y.rep[i])       # check to see the probability of getting a more extreme value

#  residual and moments of replicated data....   this gives the predicted distribution for these values
res.rep[i]<- y.rep[i] - mu[i]
stdres.rep[i]<- res.rep[i]*sqrt(tau)

dev1.rep[i]<-pow(res.rep[i],2)
dev2.rep[i]<-pow(stdres.rep[i],2)

#  likelihood for each observed and replicated data....
#  note: need to know the density function of the probability model
loglike[i]<-  (0.5)*log(tau/6.283) + (-0.5)*tau*pow((y[i]-mu[i]),2)
loglike.rep[i]<-  (0.5)*log(tau/6.283) + (-0.5)*tau*pow((y.rep[i]-mu[i]),2)

p.inv[i]<- 1/exp(loglike[i])              #  this is to find the predictive ordinate of the observatio

}

beta0 ~dnorm(0, .01)
beta1 ~dnorm(0, .01)
#beta2 <-0
tau<- 1/std/std
std ~ dunif(0,20)

##############################
#     summing the diagnostic values
```

```
chidev1.obs <- sum(dev1.obs[])
chidev2.obs <- sum(dev2.obs[])

chidev1.rep <- sum( dev1.rep[] )
chidev2.rep <- sum( dev2.rep[] )

chidev1.pval<-step(chidev1.obs-chidev1.rep)
chidev2.pval<-step(chidev2.obs-chidev2.rep)

#   Deviance statistic
dev<-   -2*sum(loglike[])
dev.rep <-  -2*sum(loglike.rep[])
dev.pval<-step(dev-dev.rep)
}
```

In looking at what happens with the residual values, let us first work with data set 3. From figure 3, we see that this data set has an outlier value.

4.2.1 The residual values

For the residual values, the residuals are created in the model file with the commands:

```
res[i]<-(y[i]-mu[i])                 #  estimate of the residuals for this model
res.rep[i]<- y.rep[i] - mu[i]
```

The second line is to get the predicted residual values.

For dataset 3, the output for the residuals is the following:

```
> print(AnscombeLin.sim,dig=3)
Inference for Bugs model at "Anscombe2LinMod.txt",
Current: 1 chains, each with 10000 iterations (first 700 discarded)
Cumulative: n.sims = 9300 iterations saved
                  mean         sd   2.5%    25%    50%     75%    97.5%
.................  edited values  ..............
res[1]           -0.543      0.493 -1.520 -0.843 -0.543  -0.248    0.453
res[2]           -0.223      0.487 -1.177 -0.518 -0.223   0.074    0.757
res[3]            3.222      0.757  1.710  2.769  3.219   3.675    4.767
res[4]           -0.388      0.468 -1.317 -0.675 -0.384  -0.104    0.545
res[5]           -0.698      0.557 -1.805 -1.036 -0.696  -0.360    0.437
res[6]           -1.183      0.877 -2.939 -1.709 -1.188  -0.663    0.614
res[7]            0.097      0.634 -1.164 -0.293  0.095   0.492    1.379
res[8]            0.417      0.859 -1.294 -0.109  0.409   0.953    2.166
res[9]           -0.863      0.648 -2.160 -1.256 -0.869  -0.474    0.458
res[10]          -0.068      0.546 -1.150 -0.401 -0.071   0.267    1.035
res[11]           0.252      0.741 -1.219 -0.199  0.248   0.715    1.736
res.rep[1]        0.012      1.526 -2.965 -0.916  0.013   0.913    3.056
res.rep[2]        0.027      1.528 -2.952 -0.910  0.015   0.935    3.107
res.rep[3]        0.011      1.540 -3.050 -0.912  0.024   0.950    3.078
res.rep[4]       -0.012      1.541 -3.023 -0.955 -0.022   0.932    3.078
res.rep[5]        0.001      1.559 -3.107 -0.940  0.000   0.930    3.200
res.rep[6]       -0.007      1.546 -3.103 -0.940 -0.018   0.947    3.074
res.rep[7]       -0.019      1.524 -3.158 -0.948  0.000   0.916    2.911
res.rep[8]        0.006      1.518 -3.001 -0.915 -0.015   0.916    3.058
```

```
res.rep[9]       -0.023      1.558 -3.179 -0.954 -0.019   0.923    3.004
res.rep[10]       0.007      1.530 -3.087 -0.924  0.012   0.972    3.027
res.rep[11]      -0.025      1.518 -3.111 -0.963 -0.027   0.930    3.006
```

The residuals for each observations are in the variables `res`. To calibrate how large they are, look at the values for `res.rep` which give the predicted distribution for the values of `res`. The outlier is at the 3rd observation. Looking at the above line for `res[3]`, the posterior mean for that residual is 3.222. To see if this might be considered large, look at the distribution for `res.rep[3]`. There we see that the 95% credible region is between -3.050 and 3.078. So, the residual of 3.222 is somewhat large.

For the other observations, the posterior mean of these statistics (for `res[ ]`) are within the ranges for the corresponding `res.rep[ ]`). So, there are no other suspect observations.

Instead of looking at the `res` posterior means and then hunting for the corresponding `res.rep` terms, one can write some R code to put everything together. This is done in the following set of code:

```
>
> xxx<-AnscombeLin.sim
> ## getting the residuals and the calibrations:
>
> temp<-cbind(xxx$mean$res,t(apply(xxx$sims.list$res.rep,2,function(x){c(quantile(x,probs=c(0.025,.975)
> colnames(temp)=c("res","2.5%","97.5%","mean","SD");temp
              res      2.5%    97.5%         mean       SD
 [1,] -0.54291232 -2.964625 3.055575  0.012449076 1.526343
 [2,] -0.22281460 -2.952050 3.107050  0.027145958 1.528494
 [3,]  3.22193891 -3.050050 3.077575  0.011046168 1.540118
 [4,] -0.38786614 -3.022525 3.077725 -0.011701711 1.541494
 [5,] -0.69796230 -3.106675 3.200050  0.001102357 1.558963
 [6,] -1.18310908 -3.102575 3.074150 -0.006588333 1.545791
 [7,]  0.09728078 -3.157725 2.911050 -0.018771655 1.523652
 [8,]  0.41737658 -3.001100 3.058350  0.005883631 1.517781
 [9,] -0.86301560 -3.178575 3.003725 -0.023117702 1.558074
[10,] -0.06776632 -3.087050 3.026525  0.007049951 1.529625
[11,]  0.25233118 -3.111300 3.005625 -0.024951404 1.517609
>
```

4.2.2 Standardized Residuals

Using the same line of reasoning as above, one can get the standardized residual and predictive distribution of the standardized residual by adding the following lines of code to the model:

```
stdres[i]<-res[i]*sqrt(tau)        #  for the standardized residuals
stdres.rep[i]<- res.rep[i]*sqrt(tau)
```

This leads to the following output after running OpenBugs (or JAGS) through R:

```
                 mean         sd   2.5%    25%    50%     75%    97.5%
   ..............................
stdres[1]       -0.400      0.336 -1.052 -0.623 -0.402  -0.175    0.264
stdres[2]       -0.164      0.321 -0.791 -0.384 -0.164   0.052    0.457
stdres[3]        2.373      0.791  0.863  1.825  2.356   2.904    3.956
stdres[4]       -0.286      0.313 -0.895 -0.498 -0.284  -0.073    0.322
stdres[5]       -0.513      0.385 -1.265 -0.772 -0.513  -0.256    0.252
stdres[6]       -0.870      0.613 -2.042 -1.286 -0.872  -0.464    0.349
stdres[7]        0.071      0.417 -0.743 -0.214  0.069   0.356    0.878
```

```
stdres[8]          0.306      0.570 -0.828 -0.081  0.303   0.698    1.399
stdres[9]         -0.635      0.452 -1.509 -0.943 -0.639  -0.334    0.270
stdres[10]        -0.051      0.358 -0.753 -0.295 -0.052   0.195    0.639
stdres[11]         0.185      0.489 -0.776 -0.145  0.184   0.523    1.124
stdres.rep[1]      0.005      0.999 -1.942 -0.655  0.010   0.654    1.966
stdres.rep[2]      0.014      0.999 -1.954 -0.647  0.011   0.678    1.989
stdres.rep[3]      0.007      1.000 -1.986 -0.669  0.016   0.679    1.947
stdres.rep[4]     -0.006      1.005 -1.955 -0.696 -0.018   0.664    1.993
stdres.rep[5]      0.000      1.008 -1.966 -0.684  0.000   0.676    1.992
stdres.rep[6]     -0.010      1.008 -2.011 -0.695 -0.011   0.687    1.931
stdres.rep[7]     -0.010      0.993 -1.976 -0.684  0.000   0.667    1.917
stdres.rep[8]      0.004      0.996 -1.916 -0.665 -0.012   0.663    1.990
stdres.rep[9]     -0.015      1.008 -2.005 -0.693 -0.014   0.664    1.944
stdres.rep[10]     0.009      1.007 -1.972 -0.673  0.008   0.693    1.968
stdres.rep[11]    -0.014      1.000 -1.996 -0.684 -0.019   0.668    1.937
```

Again, for observation 3, the `stdres[3]` posterior mean is 2.373. This can be compared to the 95% credible region for `stdres.rep[3]` which is the interval (-1.986, 1.947). So, again the posterior mean for `stdres[3]` is outside the predictive interval which indicates that one should be concerned with this observation.

Also, in looking at the predictive intervals for all the observations, note that these 95% intervals for `stdres.rep` are about -1.9 to 1.9. That is because these are standardized and the predictive distributions is approximately standard normal.

Also, here one can use some R code to put the posterior mean of the `stdres` values along sides the intervals from the `stdres.rep` values. Here is the code:

```
> temp<-cbind(xxx$mean$stdres,t(apply(xxx$sims.list$stdres.rep,2,function(x){c(quantile(x,probs=c(0.025
> colnames(temp)=c("stdres","2.5%","97.5%","mean","SD");temp
           stdres      2.5%    97.5%          mean        SD
 [1,] -0.39950246 -1.942000 1.966100  0.0053664927 0.9986355
 [2,] -0.16439702 -1.954000 1.989050  0.0135518536 0.9985008
 [3,]  2.37282888 -1.986050 1.946525  0.0070153683 0.9995992
 [4,] -0.28563052 -1.955050 1.993000 -0.0062515037 1.0045901
 [5,] -0.51337333 -1.966050 1.992050 -0.0002715687 1.0079102
 [6,] -0.86971545 -2.010525 1.931000 -0.0101928187 1.0078091
 [7,]  0.07070758 -1.975525 1.916525 -0.0101668872 0.9934903
 [8,]  0.30581289 -1.916000 1.990000  0.0038278228 0.9956372
 [9,] -0.63460808 -2.005150 1.944000 -0.0152348066 1.0084129
[10,] -0.05052549 -1.971525 1.967525  0.0091665979 1.0067734
[11,]  0.18458021 -1.995525 1.937150 -0.0144544426 0.9998753
>
```

4.2.3 Chance of getting a more extreme observation

To look at getting a more extreme observation, one can create a variable which is one if `y[]` is bigger than `y.rep`. The posterior mean of this variable would be the probabilty that the $P(y_i \geq Y_i|\text{data})$, where y_i is the observed value and Y_i is the random variable. So, if that posterior mean was close to zero or one that would mean that this observation was a realatively rare observation. Also, in general, one would expect that posterior mean of this statistic for all the observations would be approximately uniform.

So, this would mean including the following code in the model file:

```
p.smaller[i] <-step(y[i]-y.rep[i])
```

For Data 3, the output for this statistics was:

```
                 mean         sd   2.5%    25%    50%     75%    97.5%
         ........................
p.smaller[1]    0.351      0.477  0.000  0.000  0.000   1.000    1.000
p.smaller[2]    0.434      0.496  0.000  0.000  0.000   1.000    1.000
p.smaller[3]    0.969      0.174  0.000  1.000  1.000   1.000    1.000
p.smaller[4]    0.399      0.490  0.000  0.000  0.000   1.000    1.000
p.smaller[5]    0.313      0.464  0.000  0.000  0.000   1.000    1.000
p.smaller[6]    0.231      0.422  0.000  0.000  0.000   0.000    1.000
p.smaller[7]    0.522      0.500  0.000  0.000  1.000   1.000    1.000
p.smaller[8]    0.604      0.489  0.000  0.000  1.000   1.000    1.000
p.smaller[9]    0.283      0.451  0.000  0.000  0.000   1.000    1.000
p.smaller[10]   0.478      0.500  0.000  0.000  0.000   1.000    1.000
p.smaller[11]   0.572      0.495  0.000  0.000  1.000   1.000    1.000
```

For this statistics, the value for observation 3 was 0.969. This is somewhat near 1, but that does not look too bad. Perhaps the values for the other observation are a bit closer to the middle than expected??? It is hard to tell with only 11 observations. (One might get a better visual by doing a Q-Q plot.)

4.2.4 Goodness of fit statistics

For the goodness of fit statistics are created in a similar way. Two of them are obtained by first squaring the residual and the standardize residual, respectively and then by summing up over all observations. The deviance is obtained from the log likelihood. Also, Openbug actually automatically calculates a deviance statistic. The one generated by Openbugs is called `deviance`.

To compare the goodness of fit, one can use the value of the statistics using the `y.rep` values to get the predictive distribution of the goodness of fit statistics. Also, one can use a similar method as the `p.smaller` used above to get a type of "bayes p-value" for the statistics.

So, one needs to add the following lines to the model file:

```
 ### inside the for loop
dev1.obs[i]<-pow(res[i],2)
dev2.obs[i]<-pow(stdres[i],2)

dev1.rep[i]<-pow(res.rep[i],2)
dev2.rep[i]<-pow(stdres.rep[i],2)

loglike[i]<-  (0.5)*log(tau/6.283) + (-0.5)*tau*pow((y[i]-mu[i]),2)
loglike.rep[i]<-  (0.5)*log(tau/6.283) + (-0.5)*tau*pow((y.rep[i]-mu[i]),2)

#### outside the for loop


chidev1.obs <- sum(dev1.obs[])
chidev2.obs <- sum(dev2.obs[])

chidev1.rep <- sum( dev1.rep[] )
chidev2.rep <- sum( dev2.rep[] )

chidev1.pval<-step(chidev1.obs-chidev1.rep)
chidev2.pval<-step(chidev2.obs-chidev2.rep)
```

```
#   Deviance statistic
dev<-   -2*sum(loglike[])
dev.rep <-  -2*sum(loglike.rep[])
dev.pval<-step(dev-dev.rep)
```

In the above,

- `chidev1` is the $\sum_{i=1}^{n} (y_i - E(Y_i))^2$ term.
- `chidev2` is the $\sum_{i=1}^{n} \frac{(y_i - E(Y_i))^2}{V(Y_i)}$ term.
- `dev` is the $\sum_{i=1}^{n} -2 * \log(p(y_i|\theta_i))$ term.

In the bugs output we have:

```
                 mean         sd   2.5%    25%    50%     75%    97.5%
               ........................
chidev1.pval    0.437      0.496  0.000  0.000  0.000   1.000    1.000
chidev2.pval    0.437      0.496  0.000  0.000  0.000   1.000    1.000
chidev1.obs    18.517      6.828 13.850 14.830 16.430  19.610   35.531
chidev2.obs     9.991      4.474  3.124  6.727  9.392  12.682   20.110
chidev1.rep    25.930     23.943  5.055 12.420 19.650  31.275   82.664
chidev2.rep    11.043      4.706  3.833  7.581 10.330  13.900   21.845
dev            37.722      3.315 33.950 35.290 36.820  39.220   46.370
dev.rep        38.773      7.676 25.580 33.320 38.200  43.570   55.220
dev.pval        0.437      0.496  0.000  0.000  0.000   1.000    1.000
deviance       37.722      3.315 33.950 35.290 36.820  39.220   46.370
```

Note that the statistics `dev` and `deviance` are the same. The first is created by the code and the second is from Openbugs.

Comparing these, we see that the posterior mean for `chidev2.obs` (which is 9.991) is inside the credible interval for `chidev2.rep` which has interval (3.833, 21.845). Also, the value for `chidev2.pval` is .437 which means that the "bayesian pvalue" is right in the middle of what is to be expected. Also, we can see that there are similar results for `chidev1` and `dev`.

Also, we can use R code to put the columns together. See the following:

```
 #  deviance, from model/not Openbugs intrinsic
> x1<-xxx$sims.list$dev.rep
> temp<-c(xxx$mean$dev,quantile(x1,probs=c(0.025,.975)),mean(x1),sd(x1))
> names(temp)=c("Deviance","2.5%","97.5%","mean","SD");temp
 Deviance      2.5%     97.5%      mean        SD
37.722152 25.580000 55.220500 38.773382  7.675552
>
> x1<-xxx$sims.list$chidev2.rep
> temp<-c(xxx$mean$chidev2.obs,quantile(x1,probs=c(0.025,.975)),mean(x1),sd(x1))
> names(temp)=c("ChiDev2","2.5%","97.5%","mean","SD");temp
  ChiDev2      2.5%     97.5%      mean        SD
 9.991439  3.833375 21.845250 11.042673  4.706075
>
>
> ##### non calibrated ("pval-stats")
>
> apply(xxx$sims.list$p.smaller,2,mean)
```

```
 [1] 0.3508602 0.4341935 0.9686022 0.3989247 0.3131183 0.2313978 0.5220430
 [8] 0.6036559 0.2834409 0.4780645 0.5719355
> xxx$mean$chidev2.pval
[1] 0.4374194
> xxx$mean$dev.pval
[1] 0.4374194
>
>
```

5 Some more complicated methods.

The topics covered here are:

1. The cross validated predictive density. For each observation, one can calculate this term using a "magic formula".

2. The DIC statistics. This statistics is looked at again and some curious results for hierarchical models are explored. There are still some open questions about the use of this statistics.s not discussed. So, that algorithm is discussed in more details now.

6 Conditional Predictive ordinate

When performing model diagnostics, most of the methods that were discussed in detail, were methods based on "using the data twice". That is, when calculated the posterior mean of the residual, the strategy was to generate a new $Y_{\text{new},i}$ value from the predictive distribution using all the data. Then this $Y_{\text{new},i}$ was compared to the observed Y_i value by getting $\text{res}_i = (Y_i - Y_{\text{new},i})$. Since the Y_i was part of the data for the posterior predictive distribution and in getting the residual, res_i, each data point is used twice. The purpose of doing a "cross-validated" statistic is to avoid this double use of each data point.

Now, it would be very difficult/time consuming if one was to rerun the analysis n times to account for each data point. However, this can be avoid by using a wonderful trick. This involves getting the "cross-validated predictive ordinate" which is sometimes called the CPO. The CPO of the i-th element is $f(y_i|y_{-i})$ which is the value of posterior predictive density at the value $Y = y_i$ when the predictive density is from all the data points except for the i-th point. Sometimes it is convient to use the notation CPO_i for $f(y_i|y_{-i})$. The CPO can be used by itself as a type of residual analysis or it is used to create a "pseudo-Bayes factor" so one can compare different models.

6.1 Magic formula

Using some algebraic magic, it turns out that the CPO can be estimated without running the model n times. First, please note the following mathematical fact that is given as a lemma.

Lemma: Let y be a vector of observations where each of the observations y_i comes from the distribution with density $f(y_i|\theta)$. Also, for all i, j, let y_i and y_j be conditionally independent given θ. Then

$$\frac{1}{f(y_i|y_{-i})} = E_{\theta|y}\left[\frac{1}{f(y_i|\theta)}\right].$$

Proof: First note the following:

$$f(y) = f(y_i)f(y_i|y_{-i}).$$

That is, the joint density of the vector of observations, y, is equal to the product of the density at y_i times the density of the oberervations y_{-i}. Note that the vector y_{-1} is the vector of observations without the point y_i. Note that $f(y_i|y_{-i}) \neq f(y_i)$ since y_i is only conditionally independent of y_{-i} given θ.

Now, we can rearrange terms in the last equation to get the first line in the below sequence so that:

$$\begin{aligned}
\frac{1}{f(y_i|y_{-i})} &= \frac{f(y_{-i})}{f(y)} \\
&= \int \frac{f(y_{-i}|\theta)f(\theta)}{f(y)} d\theta \\
&= \int \frac{f(y_i|\theta)}{f(y_i|\theta)} \frac{f(y_{-i}|\theta)f(\theta)}{f(y)} d\theta
\end{aligned}$$

$$
\begin{aligned}
&= \int \frac{1}{f(y_i|\theta)} \frac{f(y|\theta)f(\theta)}{f(y)} d\theta \\
&= \int \frac{1}{f(y_i|\theta)} f(\theta|y) d\theta \\
&= E_{\theta|y}\left[\frac{1}{f(y_i|\theta)}\right].
\end{aligned}
$$

In the above, the second line, note that since $f(y)$ does not the parameter θ in it, then it can pass in or out of the integration. In the third line, the first ratio is just one is setting up for the next line. In the fourth line, note that $f(y_i|\theta)f(y_{-i}|\theta) = f(y|\theta)$. This is true since y and the vector y_{-1} are conditionally independent given θ. To go from the fourth to the fifth line involves an application of Bayes Theorem. The last line completes the proof.

Therefore, with the above lemma, one can estimate $[f(y_i|y_{-i})]^{-1}$ by MCMC by calculating the creating a node which is $[f(y_i|\theta)]^{-1}$ and then calculating the posterior mean of this node. The inverse of the posterior mean can then be used as an estimate of the posterior predicitive ordinate. That is, if we run an MCMC chain with M iterations and obtain values $\theta^{(m)}$ at the m-th iteration of the chain, then one can estimate CPO_i by:

$$
\widehat{\text{CPO}}_i = \left(\frac{1}{M} \sum_{m=1}^{M} f(y_i|\theta^{(m)}) \right)^{-1}.
$$

6.2 Calculating the CPO with MCMC

Using the above formula, one can compute an estimate of the CPO. It does involve a series of steps, but each step is not very hard.

1. Create nodes for the value of $[f(y_i|\theta)]^{-1}$. Let us say that these nodes are in the vector of nodes: `p.inv[]`.
2. Run the MCMC and monitor the nodes `p.inv[]`. Find the posterior mean for these nodes.
3. The CPO_i's are estimated by the 1/posterior mean of these nodes.

6.3 pseudo-Bayes Factor

The pseudo Bayes Factor (PBF) is a way of comparing two models that was suggested by Geisser and Eddy (1979). If we are comparing model 1 versus model 2, let $f(y_i|y_{-i}, M_1)$ and $\text{CPO}_i(M_1)$ be the CPO for i-th observation from model 1 and $f(y_i|y_{-i}, M_2)$ and $\text{CPO}_i(M_2)$ be the CPO for i-th observation from model 2. Then, define the PBF_{12} comparing model 1 versus model 2 as:

$$
\text{PBF}_{12} = \prod_{i=1}^{n} \frac{f(y_i|y_{-i}, M_1)}{f(y_i|y_{-i}, M_2)} = \prod_{i=1}^{n} \frac{\text{CPO}_i(M_1)}{\text{CPO}_i(M_2)}.
$$

To understand what the pseudo Bayes factor is, let us first look at the Bayes factor. The Bayes factor is somewhat similar to the usual Bayes factor (BF) defined by:

$$
\text{BF}_{12} = \frac{f(y|M_1)}{f(y|M_2)} = \frac{\int \prod_{i=1}^{n} f(y_i|\theta, M_1) f(\theta|M_1) d\theta}{\int \prod_{i=2}^{n} f(y_i|\theta, M_2) f(\theta|M_1) d\theta}.
$$

Compare this to an expanded view of th PBF:

$$
\text{PBF}_{12} = \prod_{i=1}^{n} \frac{f(y_i|y_{-i}, M_1)}{f(y_i|y_{-i}, M_2)} = \frac{\prod_{i=1}^{n} \int f(y_i|\theta, y_{-i}, M_1) f(\theta|y_{-i}, M_1) d\theta}{\prod_{i=1}^{n} \int f(y_i|\theta, y_{-i}, M_2) f(\theta|y_{-i}, M_2) d\theta}
$$

In looking at the difference, note that the pseudo Bayes factors uses the y_{-i} values to find the cross-validated posterior predictive distribution while the Bayes factor uses the prior predictive distribution[1].

The pseudo Bayes factor is used just like the Bayes factor in comparing two models. Therefore, it is common to look at the $-2 \times \log$ of the pseudo Bayes factor. That is, one usually looks at the following:

$$-2\log(\text{PBF}) = \left(-2\sum_{i=1}^{n} \text{CPO}_i(M_1)\right) - \left(-2\sum_{i=1}^{n} \text{CPO}_i(M_2)\right).$$

So, to compare the two the models, one compares the two terms that are differed in the above. The sum $-\sum_{i=1}^{n} \text{CPO}_i(M_1)$ is called the "negative cross-validatory log-likelihood. This is a bit of an awkward sounding term and please note that this is not multiplied by two unlike the terms in the pseudo Bayes factor. Despite being awkwardly sounding, it is however fairly easy to compute given the posterior means of the $f(y_i|\theta)$'s.

6.3.1 Example of calculating CPO and pseudo-Bayes Factor

The following WinBugs model has nodes created to perform the various model diagnostics. The node `p.inv[]` is the node used to calculate the CPO and the pseudo-Bayes Factor.

To calculate the CPO, the most important line is the definition of the node `p.inv[i]`:

```
loglike[i]<-  (0.5)*log(tau/6.283) + (-0.5)*tau*pow((y[i]-mu[i]),2)
p.inv[i]<- 1/exp(loglike[i])              #  this is to find the predictive ordinate of the observatio
```

Note that in the above, the definition of the node `loglike[i]` is driven by the probability distribution of the observation. In this model it is for a normal distribution. If the distribution was different, then of course a different density function would be used there.

6.3.2 Results for Dataset 3

Running the linear model for Dataset 3, the following output was obtained for the `p.inv` variables:

```
> print(AnscombeLin.sim,dig=3)
Inference for Bugs model at "Anscombe2LinMod.txt",
Current: 1 chains, each with 10000 iterations (first 700 discarded)
Cumulative: n.sims = 9300 iterations saved
                   mean         sd   2.5%    25%    50%     75%    97.5%
      ........................
p.inv[1]          4.211      1.300  2.518  3.318  3.950   4.804    7.408
p.inv[2]          3.924      1.230  2.350  3.080  3.666   4.466    6.951
p.inv[3]       5426.946 319500.761  7.578 21.667 56.475 207.800 6168.968
p.inv[4]          4.017      1.232  2.420  3.172  3.766   4.567    7.071
p.inv[5]          4.545      1.542  2.632  3.508  4.208   5.193    8.468
p.inv[6]          7.801      9.826  2.856  4.200  5.541   8.145   26.152
p.inv[7]          4.045      1.380  2.347  3.115  3.743   4.606    7.528
p.inv[8]          4.734      2.662  2.444  3.340  4.123   5.325   10.820
p.inv[9]          5.119      2.265  2.730  3.754  4.568   5.802   10.886
p.inv[10]         3.933      1.269  2.335  3.067  3.661   4.478    7.086
p.inv[11]         4.286      1.690  2.390  3.208  3.890   4.878    8.637
```

[1]Also, there is the issue of switching the integration and the products. So, this explanation is not perfect.... Note to Mike: perhaps check Geisser and Eddy, 1979 for more explanation on the rational for this method.

However, to get the conditional predictive ordinate, we need to get invert the posterior mean of the `p.inv`. That is, we need to get:

$$f(y_i|y_{-i}) = \frac{1}{E_{\theta|y}\left[\frac{1}{f(y_i|\theta)}\right]}.$$

Also, while we are here, we can also "negative cross-validated log-likelihood". The following bit of R code will do this all automatically for us.

```
> xxx<-AnscombeLin.sim
> xxx$mean$p.inv
 [1]    4.210775    3.923597 5426.946058    4.016545    4.544849    7.801417
 [7]    4.045329    4.734316    5.118976    3.933431    4.285987
>
> # Getting predictive Ordinates and Pseudo m2LogL (aka: )
> #  Note: difference of Pseudo-m2LogL is the PsuedoBayes Factor btw models.
> #
> x1<-xxx$mean$p.inv
> PLogLI=-1*log(x1)
> temp<-cbind(x1,1/x1,PLogLI);colnames(temp)=c("p.inv","p(x)","PLogLI");temp
           p.inv         p(x)    PLogLI
 [1,]    4.210775 0.2374859765 -1.437647
 [2,]    3.923597 0.2548681879 -1.367009
 [3,] 5426.946058 0.0001842657 -8.599132
 [4,]    4.016545 0.2489701737 -1.390422
 [5,]    4.544849 0.2200292951 -1.513995
 [6,]    7.801417 0.1281818438 -2.054305
 [7,]    4.045329 0.2471986545 -1.397563
 [8,]    4.734316 0.2112237486 -1.554837
 [9,]    5.118976 0.1953515777 -1.632954
[10,]    3.933431 0.2542309845 -1.369512
[11,]    4.285987 0.2333184754 -1.455351
> Pseudom2logL= -2*sum(PLogLI);Pseudom2logL
[1] 47.54545
>
```

In the above, the means for the `p.inv` are put in the variable `x1`. Then, in the `temp` matrix above, we have the results. The first column contains the posterior mean of the `p.inv`. The second column contains our estimated values for the $f(y_i|y_{-i})$. The third column is the minus one times the log of the `p.inv`. When that third column is summed up, then we have our "negative cross-validated log-likelihood". (I call that the pseudo 2 log likelihood.... it makes more sense to me. That is because the difference between those values for two different models gives me the pseudo-bayes factor.)

Now that we have that term, we can look at it to see how well the linear model fit for dataset 3. When looking at column 2 above, note that the value of the CPO for observation 3 is .00018. This is much more extreme than we saw with the other diagnostics. To understand this, note that this is an estimate of $f(y_i|y_{-i})$ which is the predictive ordinate estimated with only the other observations and not with observation 3. When a model is fit without observation 3, the line is not pulled out by observation 3. Therefore, it then sees the Y value for observational 3 to be very far from what is expected.

7 Decision Information Criteria

Basically, it is a way models. It also contains a term for "effective number of model parameters".

Reference: Spiegelhalter, Best, Carlin, and van der Linde (2002), "Bayesian measures of model complexity and fit," J. Royal Stat. Soc. B, **64**, 583-640.

Define the following terms:

- $D(\theta, X) = -2\log(p(X|\theta))$. (Note: $p(X|\theta)$ is the pdf of the pmf.) This is the deviance function. Also note that usually X is the data from all the subjects, so this is a sum over each individual log-likelihood.
- Dbar $= E_{\theta|X}(D(\theta, X))$. This is the "Average Deviance". This term is estimated via the MCMC algorithm.
- Dhat $= D(E_{\theta|X}(\theta), X)$. That is, plug in the posterior means for θ into the deviance fromula.
- pD = Dbar − Dhat. This is sometimes referred to as the effective number of parameters in the model. Please see Spiegelhalter et al for explanation.
- The pD term can usually be approxiamated by the posterior variance of the deviance. That is, one can let $\text{pD}_{(2)} = \frac{1}{2}\text{VAR}_{\theta|X}(D(\theta, X))$. Here, $\text{VAR}_{\theta|X}(D(\theta, X))$ is the posterior variance of the deviance. This can be estimated by MCMC when sampling $D(\theta, X)$.

Then, one can define the DIC by either:

$$\text{DIC} = \text{Dbar} + \text{pD}$$

or by

$$\text{DIC} = \text{Dhat} + 2 * \text{pD}.$$

The rule for choosing a model is to pick the one with the smallest DIC. (Well, a difference of 1 or 2 units is not considered a big difference. A difference of about 10 or more is consider a big difference. More about this next week when talking about Bayes Factors.)

8 Looking at different models for a data set.

{Aside: I tried to use Anscombe's dataset 2, but the fit was "too exact" causeing issues with fit of the error parameter. So, instead, I "cooked" a data set.}

Here, I will compare a linear fit versus a quadratic fit of the same dataset. I made the data set with the following R code:

```
set.seed(304)
x<- runif(40, 0,20)
y<- rnorm(40,7-0.05*(x-12)^2,1)
```

The plot of the data is in figure 4. These data were analyzed using the code in "CookedQuadMods.txt" with output in "CookedQuadModsOutput.txt".

When using the linear model, then the output from the bugs (at the end of the print statement) gives:

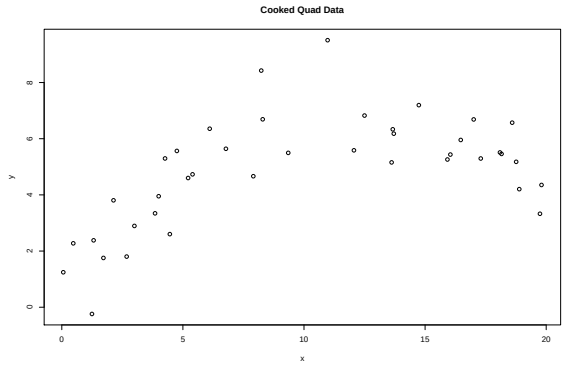

Figure 4: Plot of generated data from a quadratic relationship.

```
> print(CookedQuadLin.sim,dig=3)
Inference for Bugs model at "Anscombe2LinMod.txt",
Current: 1 chains, each with 10000 iterations (first 700 discarded)
Cumulative: n.sims = 9300 iterations saved
                 mean      sd    2.5%     25%     50%     75%   97.5%
.............................
chidev1.obs    115.239   6.430 109.200 110.800 113.200 117.500 132.452
chidev2.obs     38.993   8.900  23.534  32.570  38.315  44.680  58.555
dev            156.847   2.562 153.900 155.000 156.200 158.000 163.500
dev.rep        157.899  13.100 133.747 148.900 157.300 166.200 185.552
dev.pval         0.470   0.499   0.000   0.000   0.000   1.000   1.000
deviance       156.848   2.562 153.900 155.000 156.200 158.000 163.500

DIC info (using the rule, pD = Dbar-Dhat)
pD = 2.927 and DIC = 159.800
DIC is an estimate of expected predictive error (lower deviance is better).
>
```

Also, we can calculate the value by hand by the following R code:

```
> # comparing intrinsic and self calculated value for Deviance:
>
> x1<-xxx$sims.list$dev
> x2<-xxx$sims.list$deviance
> temp<-rbind(
+     quantile(x1,probs=c(0.025,.25,.5,.75,.975)),
+     quantile(x2,probs=c(0.025,.25,.5,.75,.975)) )
> rownames(temp)=c("SelfPrograMed:","Openbugs Made:");temp
                2.5% 25%   50% 75% 97.5%
SelfPrograMed: 153.9 155 156.2 158 163.5
Openbugs Made: 153.9 155 156.2 158 163.5
>
> c(xxx$mean$deviance,xxx$mean$dev)
[1] 156.8482 156.8470
> Dbar<-mean(x2);Dbar
```

```
[1] 156.8482
> pd2<-0.5*var(x2);pd2
[1] 3.281131
>
> devNormFunc <- function(beta0, beta1, beta2, tau, x, y){
+      mu<- beta0+beta1*x + beta2*x^2
+  return(-2*sum(log(dnorm(y,mu,1/sqrt(tau)))))}
> beta0Bar<- xxx$mean$beta0
> beta1Bar<- xxx$mean$beta1
> beta2Bar<- 0
> tauBar <- xxx$mean$tau
> Dhat <- devNormFunc(beta0Bar, beta1Bar, beta2Bar, tauBar, x,y);Dhat
[1] 153.7651
> pd1<-Dbar-Dhat;pd1
[1] 3.083101
>
> DIC1<-Dbar+pd1; DIC1
[1] 159.9313
> DIC2<-Dbar+pd2; DIC2
[1] 160.1293
>
>
```

The results comparing the linear versus a quadratic models are in the table.

	Linear	Quadratic	difference
Chidev1	115.2	47.5	
Chidev2	38.9	38.8	
deviance	156.9	121.3	
pseudoBF	156.9	121.3	
DIC1	159.93	125.6	
(pd)	(3.08)	(4.30)	
DIC2	160.13	126.8	
(pd)	(3.23)	(5.56)	
DIC (from bugs)	159.8	1254	
(pd)	(2.97)	(4.01)	