Multiple Linear Regression

Edps 590BAY

Carolyn J. Anderson

Department of Educational Psychology

ILLINOIS

Fall 2019

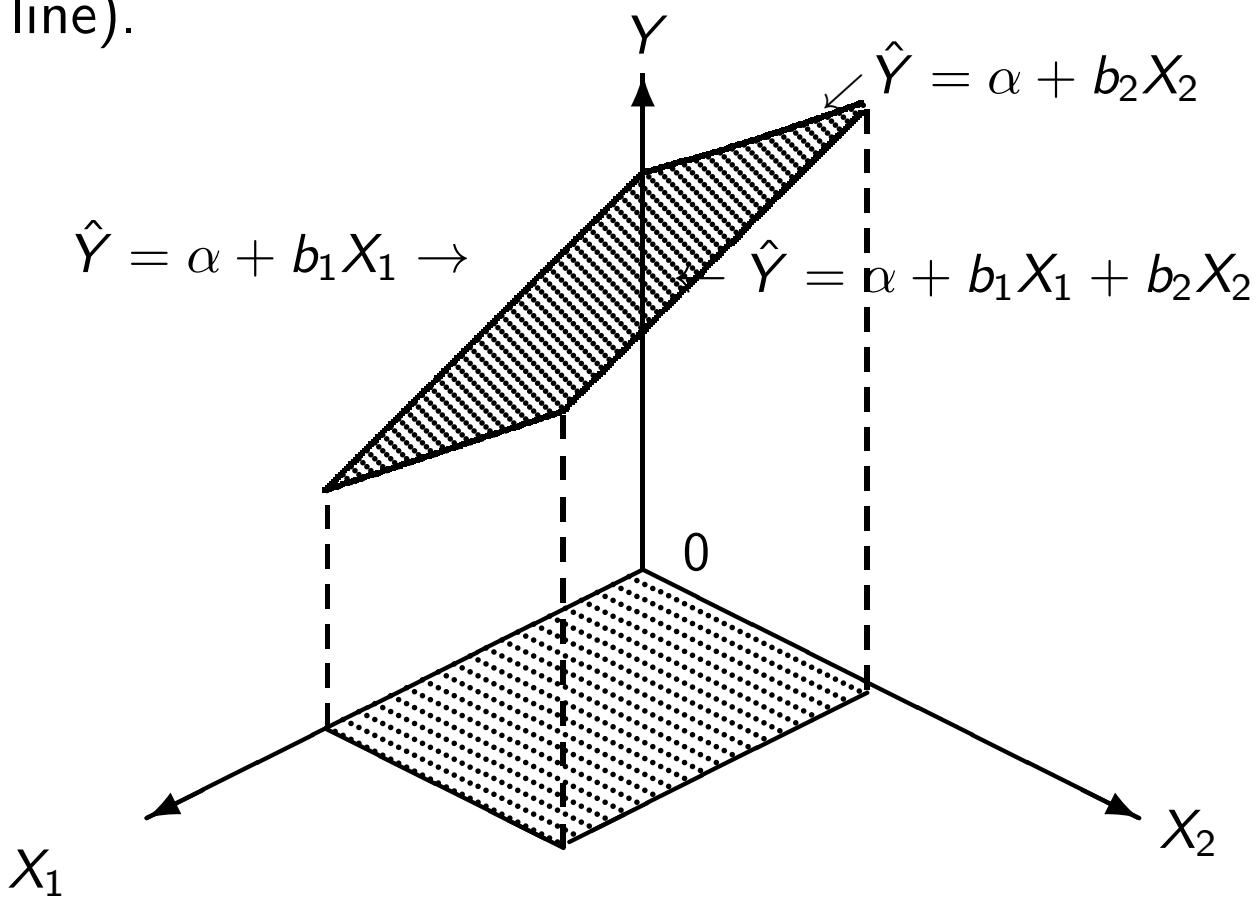
Overview

- Multiple regression
- Model evaluation
- Model comparison

Depending on the book that you select for this course, read either Gelman et al. pp xx or Kruschke Chapters chapters 13, 15 & 16 . Also I used the coda and jags, rjags, runjags and jagsUI manuals.

Multiple Regression

If we have more than one predictor, we can add them to our model. For example, for 2 predictors we try to find a plane (rather than a line).



Multiple Regression as a GLM

$$\begin{aligned} y_i &= b_0 + b_1 x_{1i} + b_2 x_{2i} + \ldots + b_k x_{ki} + \epsilon_i \\ &= \mu_i + \epsilon_i \end{aligned}$$

- Random Component: y is the response/outcome variable. We assume that $\epsilon_i \sim N(0, \sigma^2)$ so $y_i \sim N(\mu_i, \sigma^2)$.
- Linear Predictor (Systematic component) is

$$b_0 + b_1 x_{1i} + b_2 x_{2i} + \ldots + b_k x_{ki}$$

- Identity link:

$$g(E(y_i)) = \mu_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + \ldots + b_k x_{ki}$$

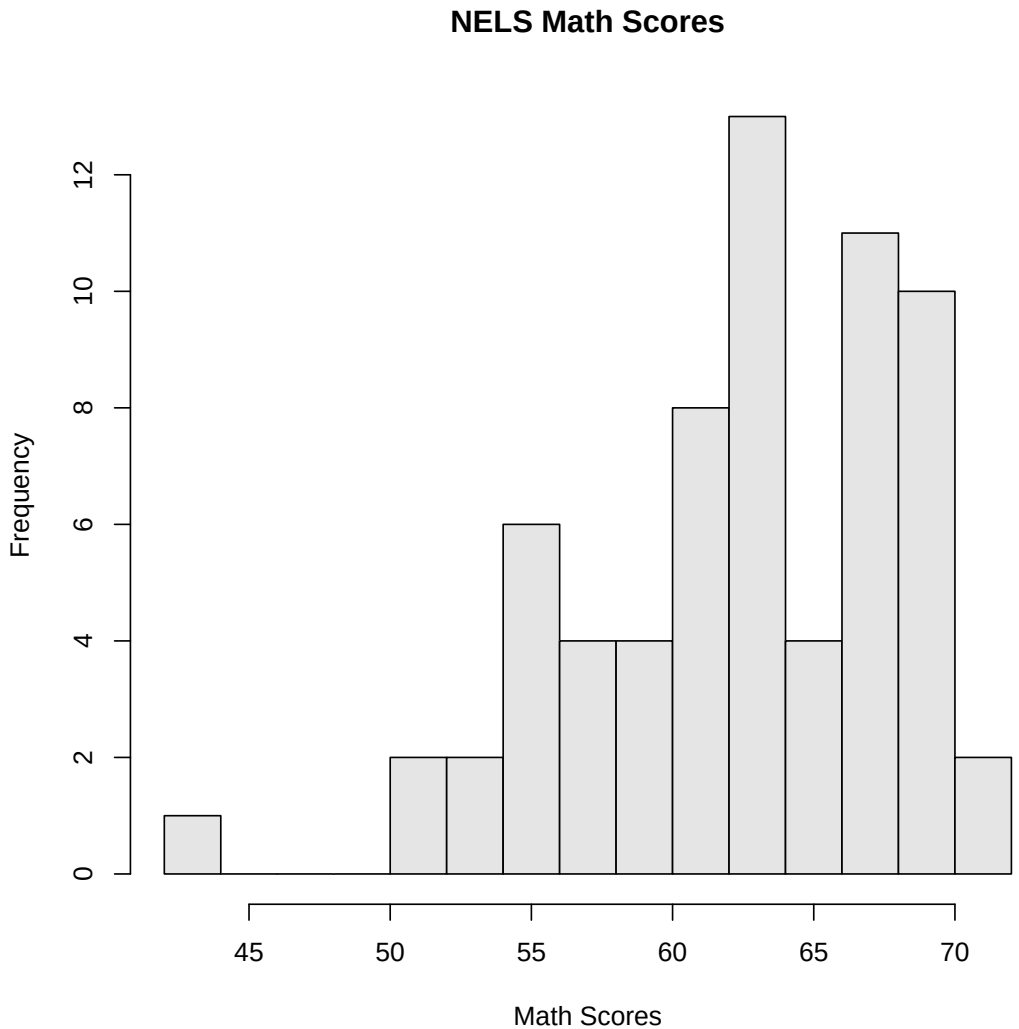
NELS: Exploratory Analysis

- We'll continue with the NELS example.
- Before modeling the data, we should do a little exploratory analysis.
- Basic descriptive statistics of math scores:

N	$\bar{y}$	sd	var	min	median	max
67	62.8209	5.6754	32.3099	43.00	63.00	71.00

- Histogram (next slide)

Distribution of Math Scores



Possible Predictor Variables

Information about variables:

- sex: 1 =male, 2 =female
- race: 1 =Asian/PI, 2 =Hispanic, 3 =Black, 4 =White. It would be best to dichotomize (white/not-white).
- Time spent doing homework: 0 =none, 1 = less then 1 hr, 2 =1 hour, 3 =2 hours, 4 = 3 hours, 5 =4 to 6 hours, 6 =7 to 9 hours, 7=more than 10 hours. This is ordinal, but we'll treat as numerical (i.e., "continuous").
- ses: I think this is composite of income, parent education, etc. We'll treat as numerical (i.e., "continuous").
- Parents education: 3 =HS (5), 4 = college grade (17), 5 =masters (24), 6 =doctorate (21). This may look odd, but this is a an urban private school in north central US. Ordinal but we may treat as numerical (i.e., "continuous").

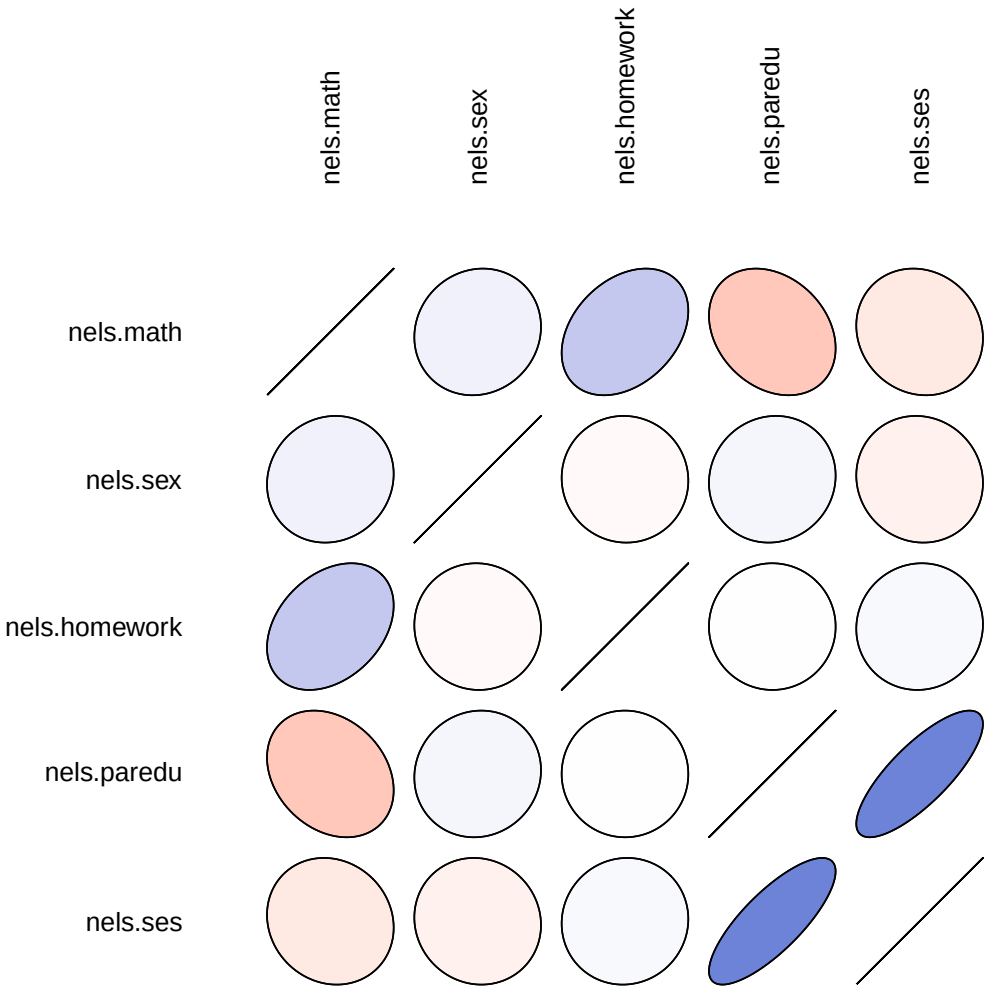
Descriptive Statistics Predictor Variables

		N	$\bar{x}$	sd(x)	min(x)	max(x)
Sex	male	36				
	female	31				
Race	non-white	7				
	white	60				
Time homework		67	3.30	1.72	0	6
ses		67	1.04	0.46	-0.35	1.85
Parent education		67	4.91	0.93	3	6

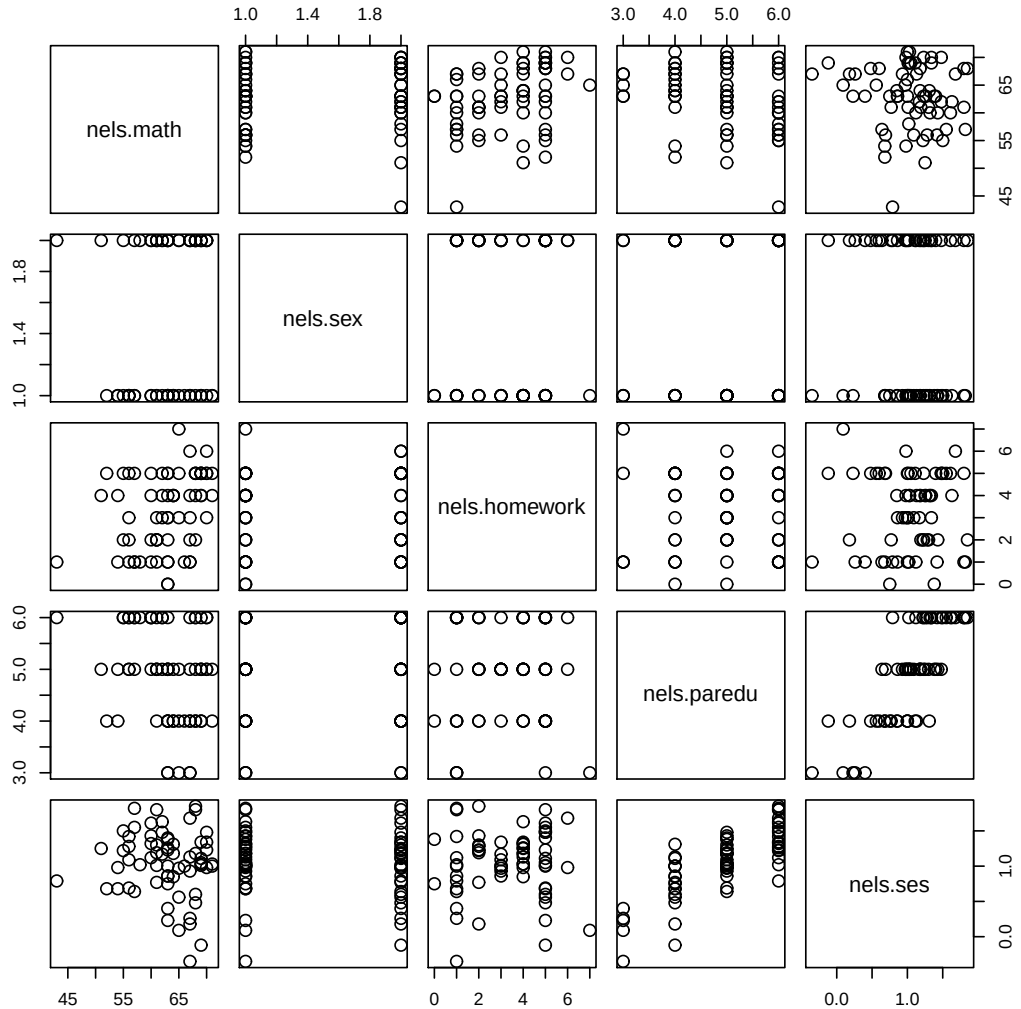
Correlations between Variables

	math	homework	paredu	ses
math	1.00	.33	-.33	-.10
homework	.33	1.00	.00	.04
paredu	-.26	.00	1.00	.79
ses	-.10	.04	.79	1.00

Look at Correlations



Another look at Bi-variate Relationships



An OLS of math

```
ols.lm <- lm(math gender + ses + paredu + homework +
white, data=nels)
```

Residuals:

Min	1Q	Median	3Q	Max
-13.8831	-2.4426	0.3711	3.4205	8.9577

Coefficients:

	Estimate	Std. Error	t value	Pr($>\|t\|$)	
(Intercept)	70.3048	5.0930	13.804	$<$ 2e-16	***
gender2	1.5486	1.3083	1.184	0.24115	
ses	3.6973	2.5384	1.457	0.15037	
paredu	-3.0370	1.2020	-2.527	0.01413	*
homework	1.0629	0.3746	2.838	0.00616	**
white1	-0.7322	2.2943	-0.319	0.75072	

—

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Residual standard error: 5.227 on 61 degrees of freedom

Multiple R-squared: 0.2161, Adjusted R-squared: 0.1518

F-statistic: 3.363 on 5 and 61 DF, p-value: 0.009523

JAGS: dataList

```
dataList  ←  list(  y =nels$math,
                    pared =nels$pared,
                    hmwk =nels$homework,
                    ses =nels$ses,
                    gender=nels$gender,
                    white = nels$white,
                    N=length(nels$math),
                    sdY = sd(nels$math)
                 )
```

JAGS: model

```
mlr1 = ''model { for (i in 1:N){
      y[i] ~ dnorm(mu[i] , precision)
      mu[i] <- b0 + b1*pared[i] + b2*hmwk[i] + b3*ses[i]
                + b4*gender[i] + b5*white[i] }
      b0 ~ dnorm(0 , 1/(100*sdY^2) )
      b1 ~ dnorm(0 , 1/(100*sdY^2) )
      b2 ~ dnorm(0 , 1/(100*sdY^2) )
      b3 ~ dnorm(0 , 1/(100*sdY^2) )
      b4 ~ dnorm(0 , 1/(100*sdY^2) )
      b5 ~ dnorm(0 , 1/(100*sdY^2) )
      sigma ~ dunif( 1E-3, 1E+30 )
      precision <- 1/sigma^2
      }
 }''
writeLines(mlr1, con=''mlr.txt'')
```

JAGS: starting values

```
initsList =
list(list("b0"=mean(nelsmath),"b1" = 0,"b2" = 0,
          "b3"=0, "b4"=0, "b5"=0,
          "sigma"=sd(nels$math)),

list("b0"=rnorm(1,50,5), "b1"=rnorm(1,-2,1),
     "b2"=rnorm(1,2,1), "b3"=rnorm(1,0,1),
     "b4"=rnorm(1,1,1), "b5"=rnorm(1,0,1),
"sigma"=sd(nels$math)),
     "sigma"=sd(nels$math)),
etc.  )
```

JAGS: runjags

```
mlr1.runjags ← run.jags(model=mlr1,
        monitor=c("b0","b1","b2","b3",
           "b4","b5","sigma","dic"),
        data=dataList,
        n.chains=4,
        inits=initsList)

plot(mlr1.runjags)
gelman.plot(mlr1.runjags)

print(mlr1.runjags)
```

Look OK?

Results

JAGS model summary statistics from 40000 samples (chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	54.566	68.759	81.282	68.524	6.7695	–
b1	-5.2689	-2.8977	-0.36232	-2.8977	1.2387	–
b2	0.32424	1.076	1.8235	1.0765	0.38455	–
b3	-1.9846	3.4508	8.336	3.4354	2.6071	–
b4	-1.1208	1.5761	4.1695	1.5537	1.3494	–
b5	-4.9743	-0.52883	4.3163	-0.47132	2.3372	–
sigma	4.4271	5.2958	6.343	5.3354	0.49596	–

Results

	MCerr	MC%ofSD	SSeff	AC.10	psrf
b0	0.46992	6.9	208	0.90431	1.0043
b1	0.083361	6.7	221	0.89339	1.006
b2	0.0062373	1.6	3801	0.17402	1.0005
b3	0.11288	4.3	533	0.7076	1.0076
b4	0.030858	2.3	1912	0.39883	1.0009
b5	0.11038	4.7	448	0.80176	1.0058
sigma	0.0043969	0.9	12723	0.030837	1.0004

Model fit assessment:
DIC = 420.2391
PED not available from the stored object
Estimated effective number of parameters: pD = 7.25924
Total time taken: 6.0 seconds

What Could We Try

- Try different starting values.
- Add more iterations using `extend.jags`.
- Use thinning as option with runjags, maybe thin=10?
- See what `autorun.jags` yields.
- Drop variables that include 0 in their high density intervals.
- Use t-distribution.

```
mlr1.extend <- extend.jags(mlr1.runjags, burnin=0,
sample=500000)
```

JAGS model summary statistics from 2040000 samples (chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	55.396	68.525	82.17	68.484	6.8368	–
b1	-5.369	-2.9114	-0.52627	-2.9094	1.2326	–
b2	0.32381	1.0766	1.8294	1.0774	0.38401	–
b3	-1.8041	3.4867	8.4596	3.4878	2.6044	–
b4	-1.0097	1.5677	4.2522	1.5725	1.3356	–
b5	-5.0723	-0.47575	4.1749	-0.4633	2.3434	–
sigma	4.4021	5.3	6.3216	5.3376	0.49789	–

More iterations

	MCerr	MC%ofSD	SSeff	AC.10	psrf
b0	0.066779	1	10481	0.90838	1.0002
b1	0.011467	0.9	11554	0.88977	1.0001
b2	0.0013947	0.4	75807	0.16714	1
b3	0.020807	0.8	15668	0.69977	1.0001
b4	0.0047707	0.4	78378	0.38238	1.0001
b5	0.018316	0.8	16369	0.80555	1.0001
sigma	0.001826	0.4	74349	0.038153	1.0001

Model fit assessment:
DIC = 420.2096
PED not available from the stored object
Estimated effective number of parameters: pD = 7.22784
Total time taken: 2.6 minutes

Better mixing but still some large auto-correlations–see figures you produced.

Thinning

```
mlr1.extend <- extend.jags(mlr1.runjags, burnin=0,
sample=500000)
```

	Lower95	Median	Upper95	Mean	SD	Mode
b0	54.666	68.547	82.022	68.523	6.9802	–
b1	-5.3965	-2.9301	-0.45199	-2.921	1.2568	–
b2	0.31793	1.0774	1.8239	1.0758	0.385	–
b3	-1.7317	3.5355	8.6108	3.5192	2.641	–
b4	-1.0702	1.5872	4.2049	1.5916	1.34	–
b5	-5.1564	-0.48398	4.1666	-0.48483	2.3717	–
sigma	4.3981	5.303	6.3228	5.3414	0.49807	–

Thinning

	MCerr	MC%ofSD	SSeff	AC.100	psrf
b0	0.1536	2.2	2065	0.3578	1.0005
b1	0.02495	2	2537	0.29614	1.0002
b2	0.0023659	0.6	26480	0.001516	1.0002
b3	0.046777	1.8	3188	0.2184	1.0002
b4	0.0093853	0.7	20386	-0.013683	1.0001
b5	0.042069	1.8	3178	0.18788	1.0003
sigma	0.0027615	0.6	32529	0.010286	1.0001

Model fit assessment:
DIC = 420.286
PED not available from the stored object
Estimated effective number of parameters: pD = 7.26381
Total time taken: 22.3 seconds

Better?

autorun.jags

See R code online

Drop gender, ses and white

Remove the corresponding b's from code.

JAGS model summary statistics from 40000 samples (chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	58.987	66.65	73.487	66.615	3.6613	–
b1	-2.8533	-1.5152	-0.14321	-1.5166	0.68987	–
b2	0.36411	1.1046	1.8597	1.1042	0.38284	–
sigma	4.4384	5.2973	6.3112	5.3332	0.4825	–

Drop gender, ses and white

	MCerr	MC%ofSD	SSeff	AC.10	psrf
b0	0.13831	3.8	701	0.70561	1.0043
b1	0.0254	3.7	738	0.68983	1.0034
b2	0.0054699	1.4	4899	0.077984	1.0012
sigma	0.0035987	0.7	17976	0.012931	1.0003

Model fit assessment:
DIC = 417.0438
PED not available from the stored object
Estimated effective number of parameters: pD = 4.14033
Total time taken: 4.7 seconds

Figure: sigma

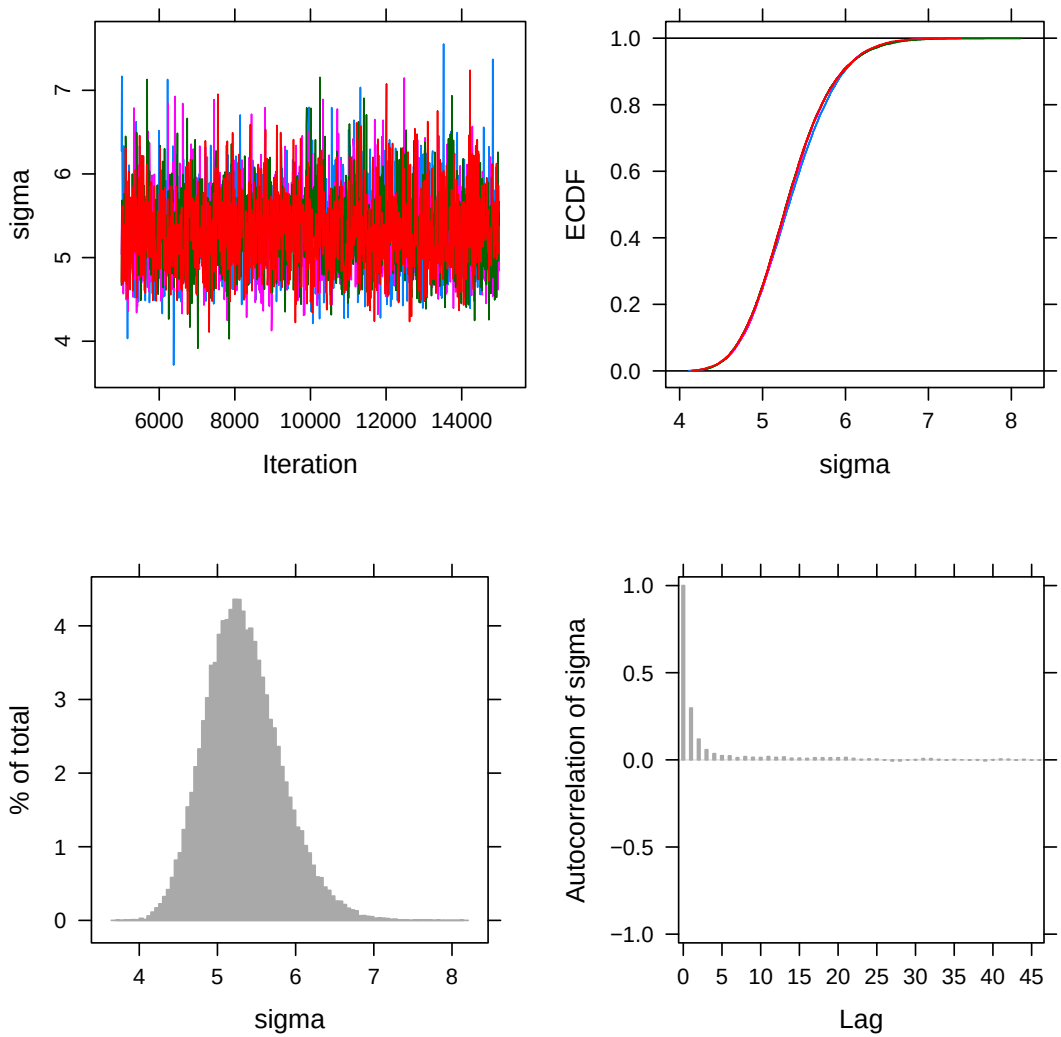


Figure: parent education

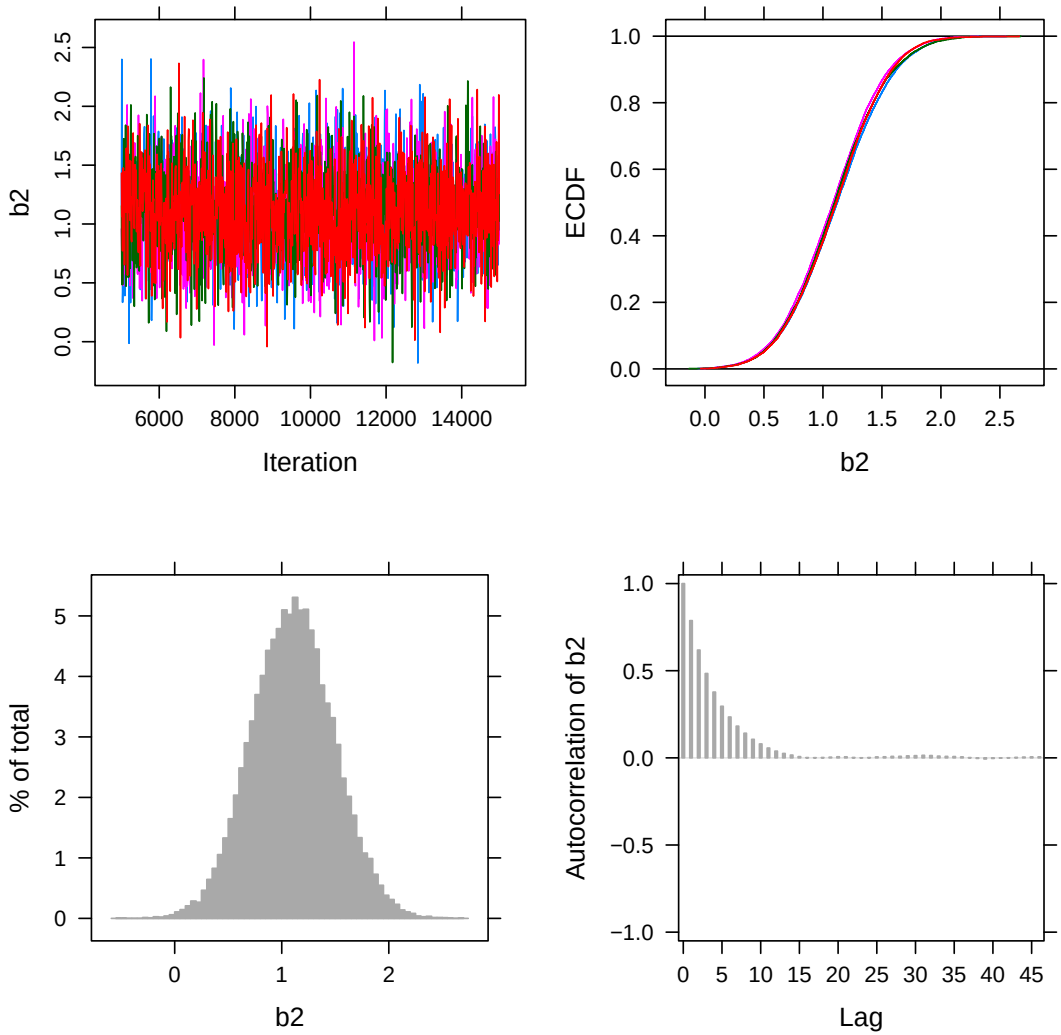


Figure: homework

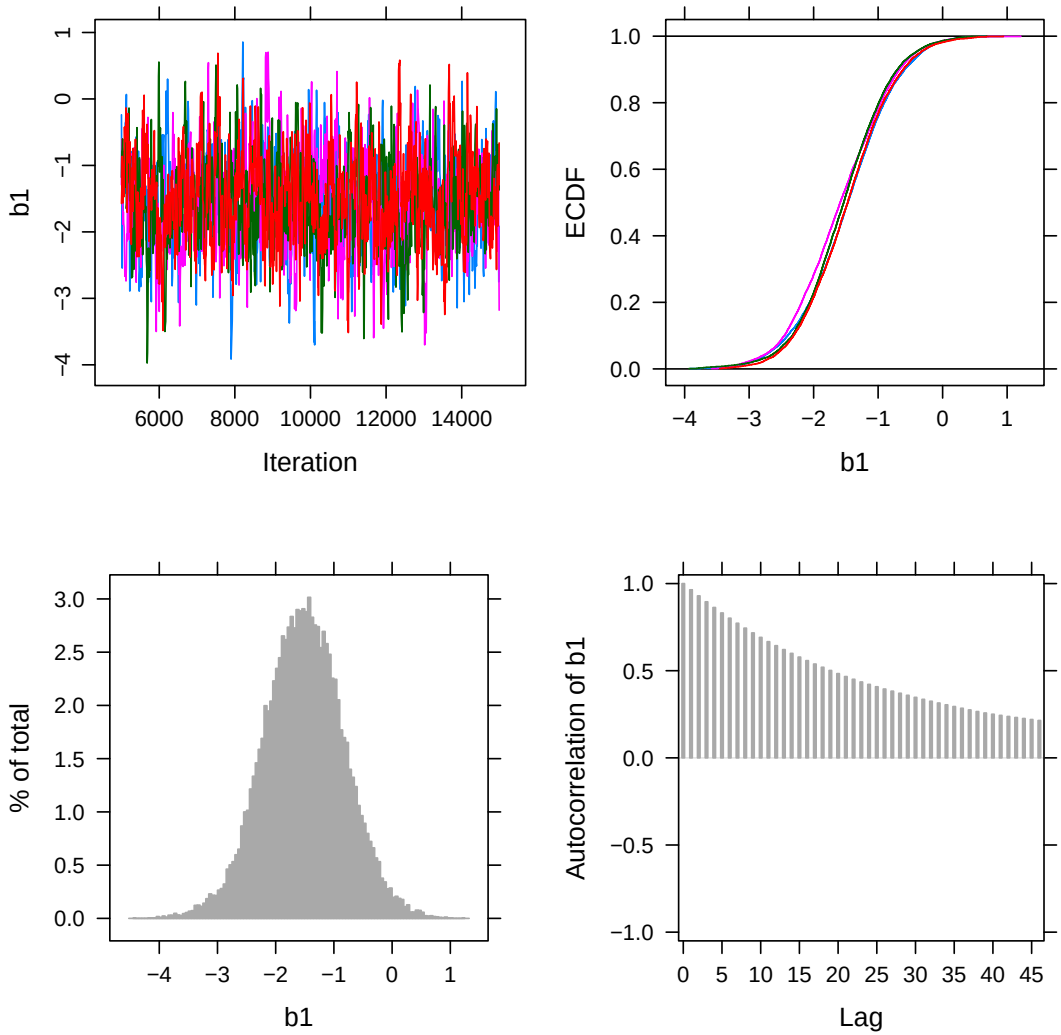
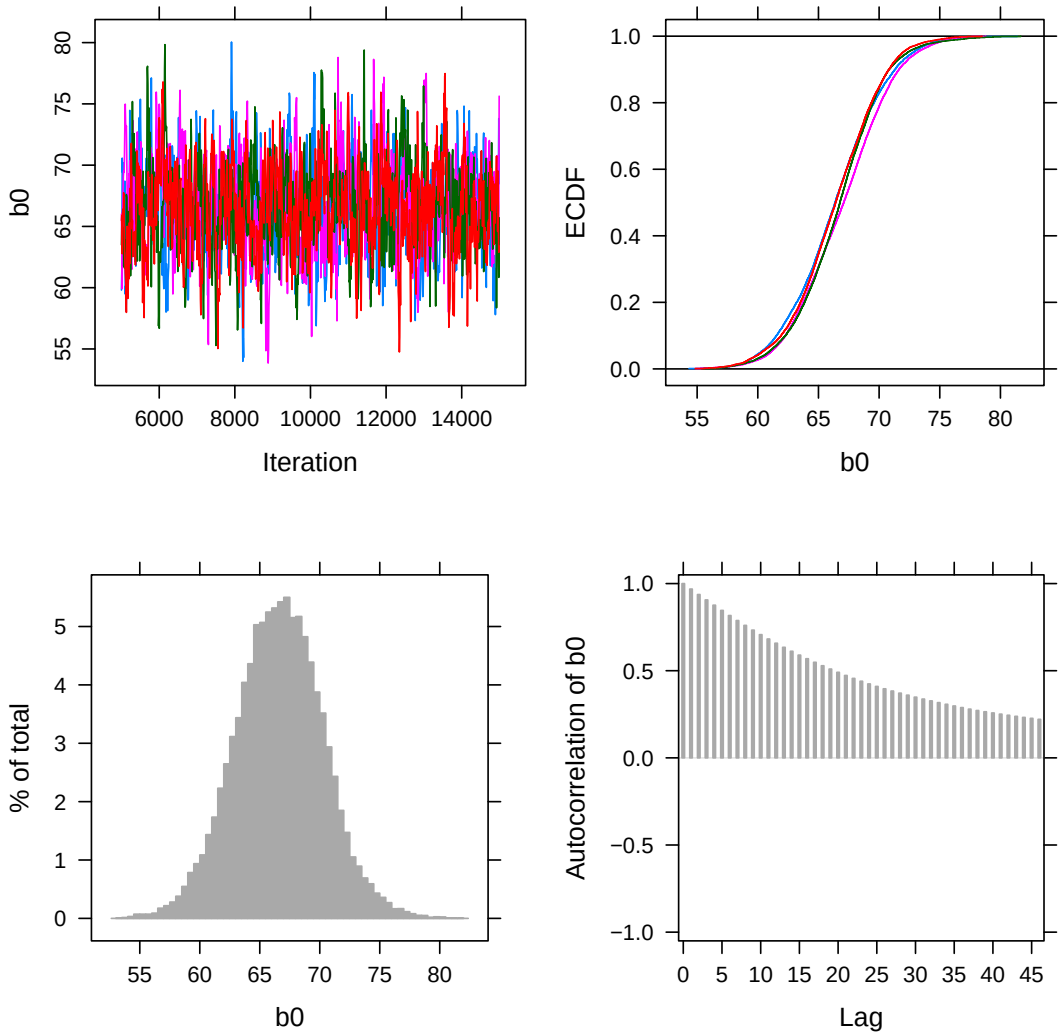


Figure: intercept



Thin Again: thin=5

JAGS model summary statistics from 40000 samples (thin = 5; chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	59.149	66.605	74.03	66.586	3.782	–
b1	-2.9105	-1.5073	-0.1238	-1.5073	0.71281	–
b2	0.35806	1.1018	1.8573	1.1009	0.38188	–
sigma	4.438	5.2982	6.3163	5.3342	0.48503	–

Thin Again: thin=5

	MCerr	MC%ofSD	SSeff	AC.50	psrf
b0	0.064769	1.7	3410	0.1771	1.0004
b1	0.012178	1.7	3426	0.17507	1.0004
b2	0.0026196	0.7	21252	0.0014437	0.99999
sigma	0.0026131	0.5	34451	0.0016297	1

Model fit assessment:
DIC = 417.1372
PED not available from the stored object
Estimated effective number of parameters: pD = 4.16808
Total time taken: 10.2 seconds

Figure: sigma

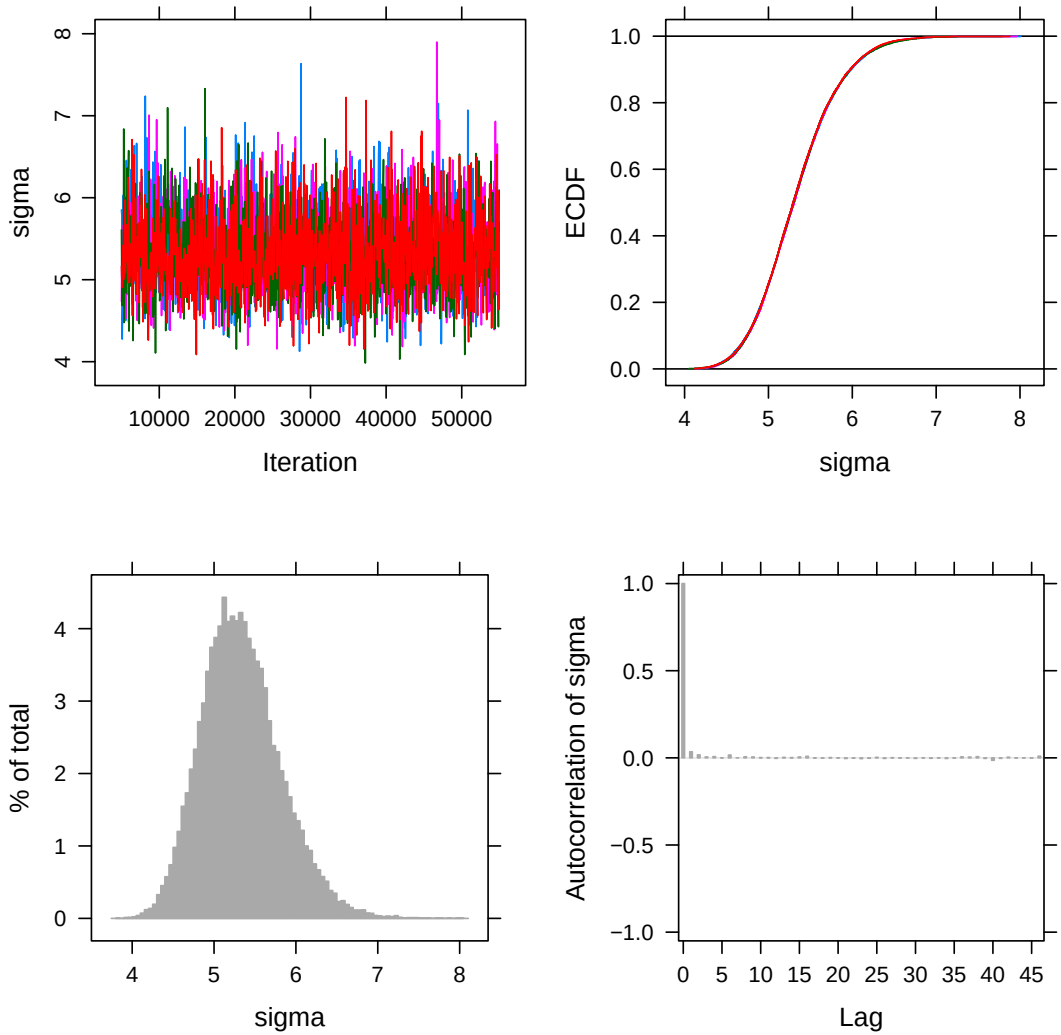


Figure: parent education

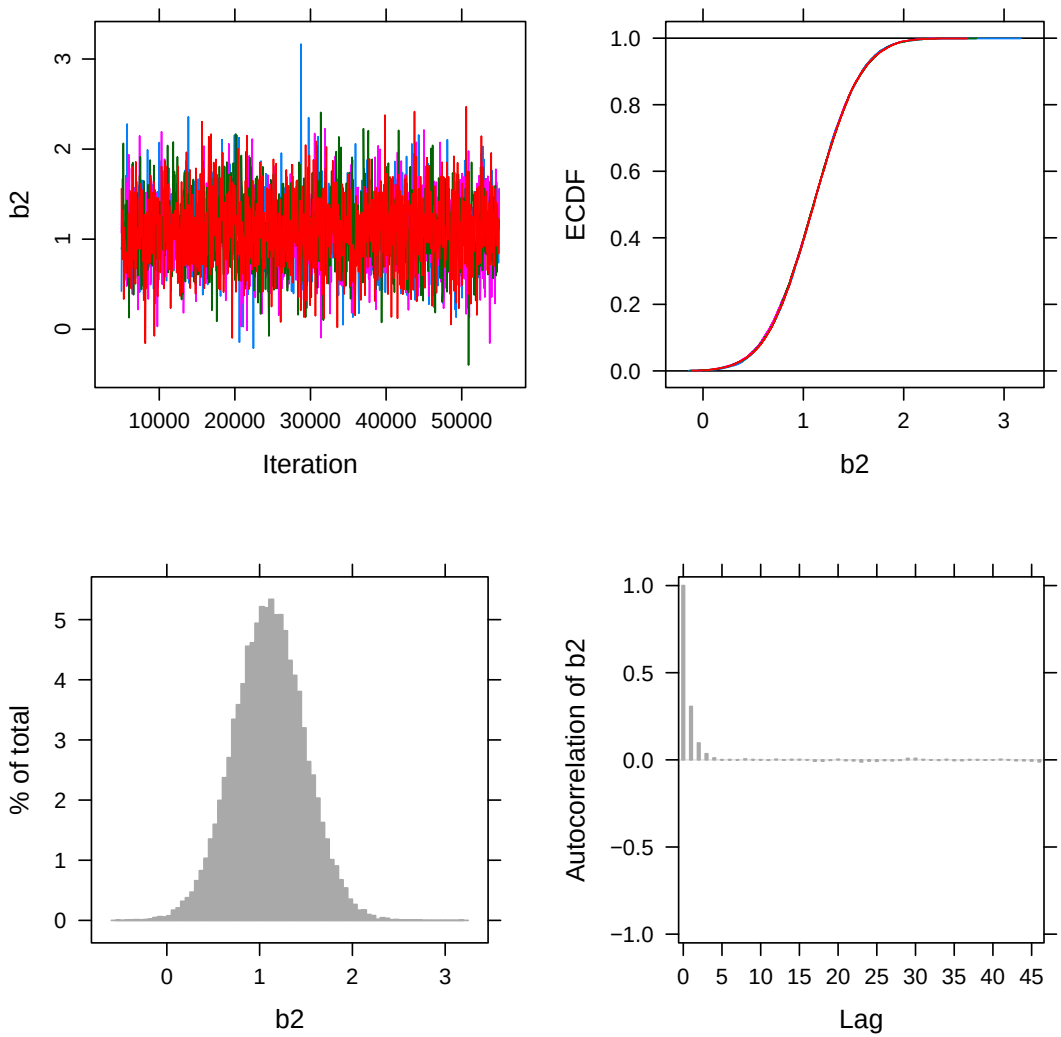


Figure: homework

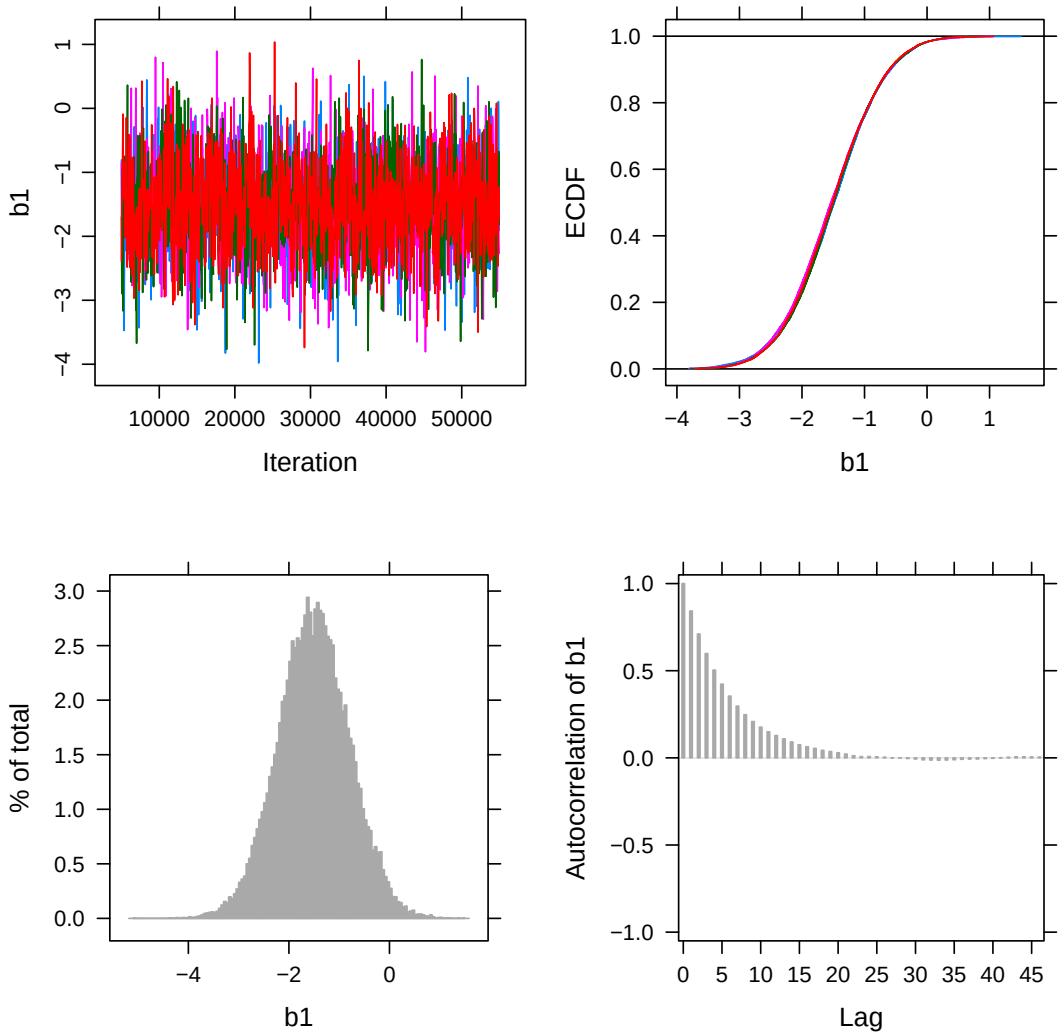
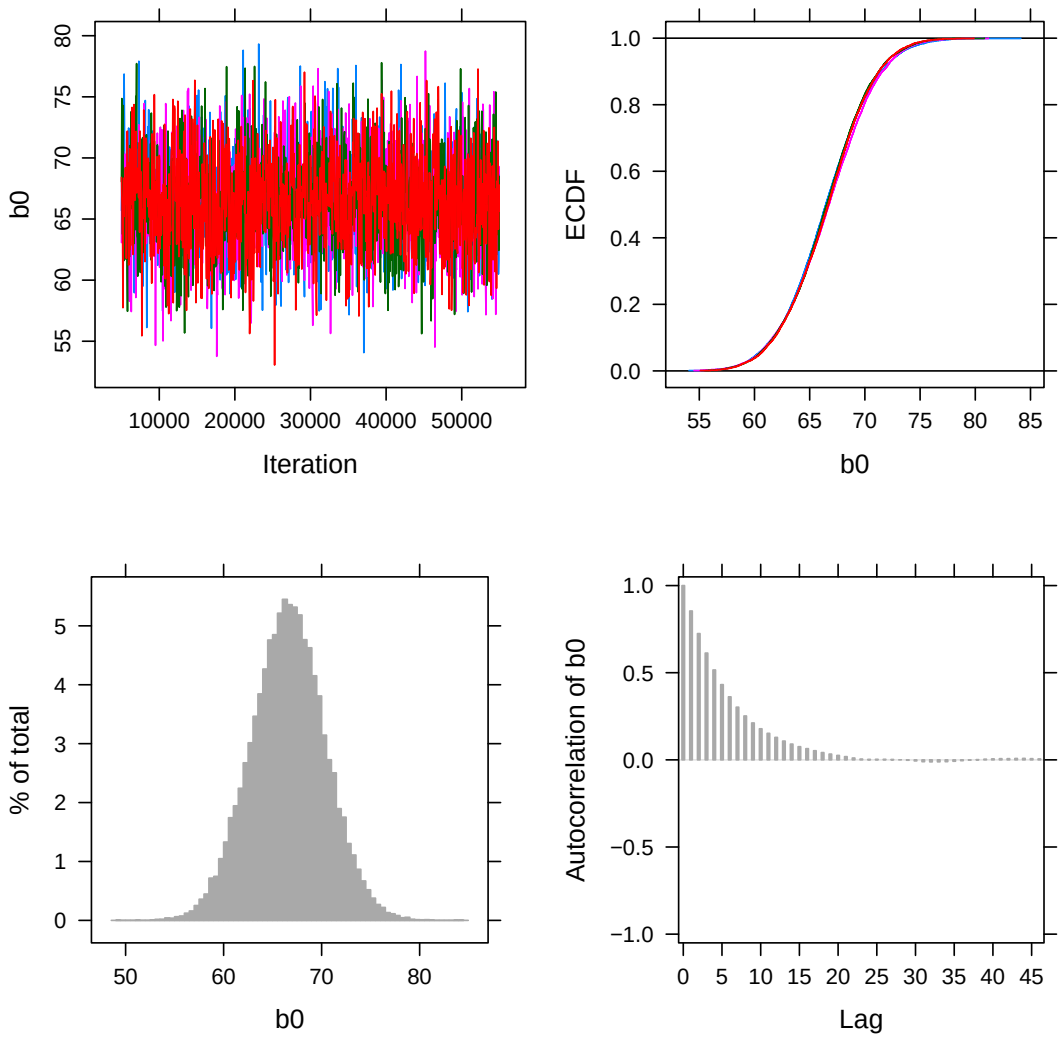


Figure: intercept



Add Interaction

Adding an interaction is just like adding another variable. I centered the variables to deal with multicolinarity so our model is now

```
model4 = "model { for (i in 1:N){
          y[i] ~ dnorm(mu[i] , precision)
          mu[i] <- b0 + b1*cpared[i] + b2*chmwk[i]
                   b3*cpared[i]*chmwk[i]
      }
      b0 ~ dnorm(0 , 1/(100*sdY^2) )
      b1 ~ dnorm(0 , 1/(100*sdY^2) )
      b2 ~ dnorm(0 , 1/(100*sdY^2) )
      sigma   dunif( 1E-3, 1E+30 )
      precision <- 1/sigma^2
}"
```

Results

The model appears to converge fine.

JAGS model summary statistics from 40000 samples (thin = 5; chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	61.529	62.826	64.061	62.821	0.64299	–
b1	-2.8596	-1.4999	-0.10376	-1.4952	0.69986	–
b2	0.4424	1.1949	1.9422	1.1964	0.38195	–
b3	-0.12834	0.55897	1.2426	0.55983	0.34877	–
sigma	4.3601	5.2295	6.2118	5.2649	0.4795	–

Model fit assessment: DIC = 416.3518 [PED not available from the stored object] Estimated effective number of parameters: pD = 5.18153

Total time taken: 11 seconds

Results

	MCerr	MC%ofSD	SSeff	AC.50	psrf
b0	0.0032042	0.5	40269	0.0044512	1.0001
b1	0.0035358	0.5	39179	0.00025875	1.0001
b2	0.0019301	0.5	39161	-0.010891	1
b3	0.0017313	0.5	40582	0.0046593	1.0001
sigma	0.0024018	0.5	39858	-0.0062027	1.0001

Model fit assessment:

DIC = 416.3518

PED not available from the stored object

Estimated effective number of parameters: pD = 5.18153

Total time taken: 11 seconds

Model Evaluation

There are many things that you can do here using the data and posterior distribution.

I will present 2 methods of getting samples from the posterior.

- Add code to your model statement so that you sample from the posterior; that is, within the loop for the likelihood add, for example

  $$\text{emp.new[i]} \sim \text{dnorm(mu[i],precision)}$$

  and add emp.new to list of parameters to monitor (output).
- Use posterior parameters and draw from posterior.

See Rmarkdown for first method and next pages for the other.

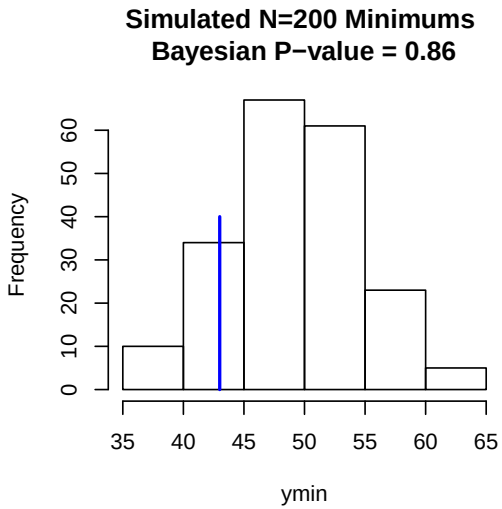
Monte Carlo of Posterior

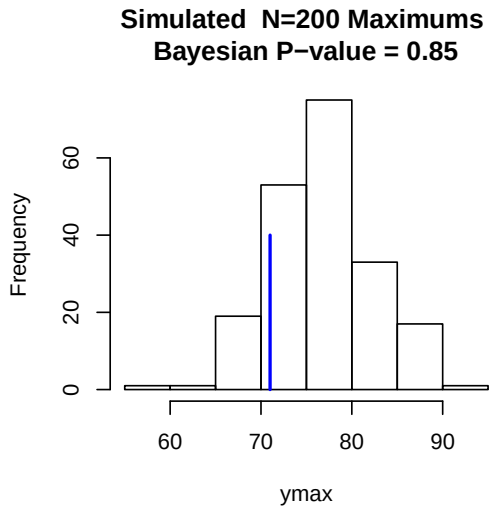
Use Monte Carlo to get posterior predictive distribution: $S = 200$ replications of "data" using draws from the posterior distribution of parameters.

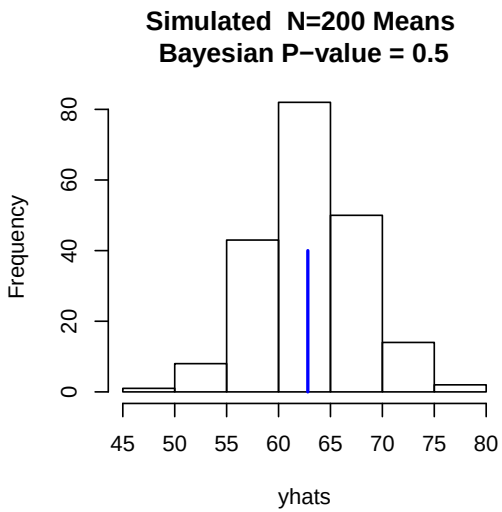
Note: The posterior parameters are a bit different, because I used a previous run when I worked up this example. The results should be about the same.

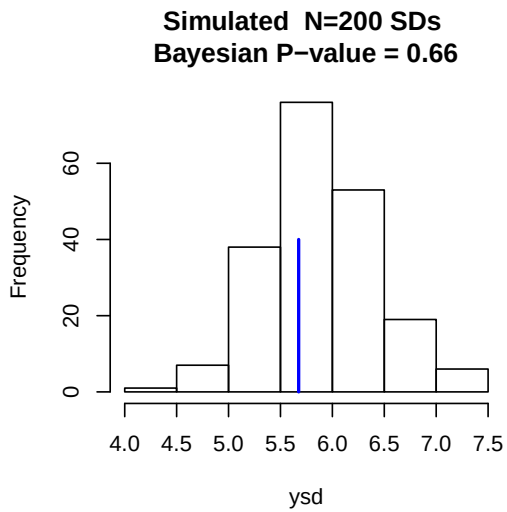
```
n ← length(nels2$math)
replications ← 200
yrep ← matrix(99,nrow=n,ncol=replications)
for (s in 1:replications){
      b0 ← rnorm(1,66.586,sd=3.7576)
      b1 ← rnorm(1,-1.517,sd=0.70876)
      b2 ← rnorm(1,1.1041,sd=0.38207)
      for (i in 1:n){
       yrep[i,s] = b0 + b1*nels$paredu[i]
          + b2*nels$homework[i] + rnorm(1,0,5.3372)
      }
}
```

Statistics on Distribution

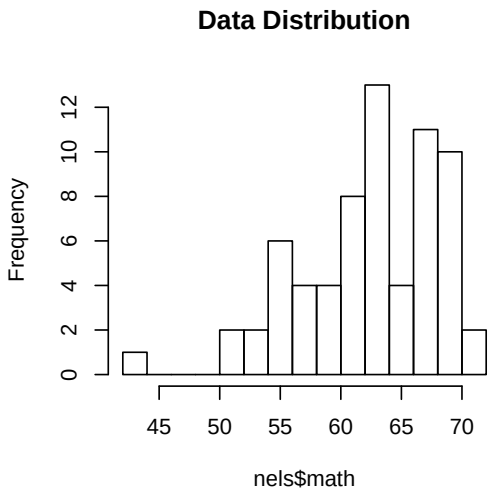


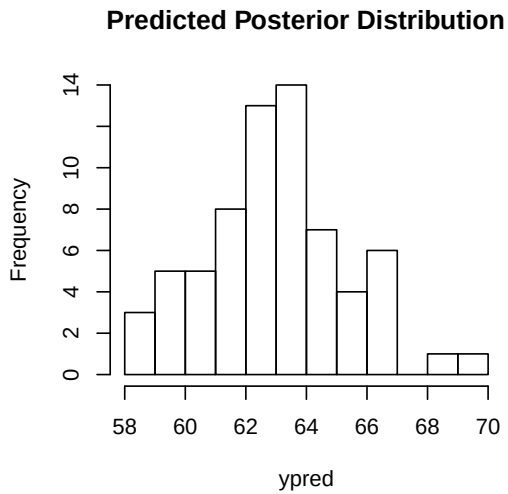






Data and Posterior Pred Distribution





Robust Multiple Linear Regression

Often we don't fit the tails of distribution very well when we use the normal distribution. An alternative is to use Students-t distribution for the data model (i.e., the likelihood).

Maybe this will further improve our model

We will need to get posterior distribution for ν, the degrees of freedom. This leads to the following model:

JAGS: model t-distribution

```
tmr = ``model { for (i in 1:N){
          y[i] ~ dt(mu[i] , precision, nu)
           mu[i] <- b0 + b1*pared[i] + b2*hmwk[i]
        }
        b0 ~ dnorm(0 , 1/(100*sdY^2) )
        b1 ~ dnorm(0 , 1/(100*sdY^2) )
        b2 ~ dnorm(0 , 1/(100*sdY^2) )
        sigma ~ dunif( 1E-3, 1E+30 )
        precision <- 1/sigma^2
        nuMinusOne ~ dexp(1/29)
        nu <- nuMinusOne+1
}
} ''
```

Results with t-distribution

JAGS model summary statistics from 40000 samples (chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	59.735	67.108	74.143	67.101	3.6502	–
b1	-2.9685	-1.5622	-0.1805	-1.5658	0.70403	–
b2	0.38532	1.1156	1.8194	1.1114	0.36607	–
sigma	3.5702	4.8388	6.0077	4.8334	0.61171	–
nu	1.564	16.637	75.416	25.083	24.539	–

Results with t-distribution

	MCerr	MC%ofSD	SSeff	AC.10	psrf
b0	0.1767	4.8	427	0.80761	1.0052
b1	0.033095	4.7	453	0.80163	1.0045
b2	0.0061505	1.7	3542	0.16689	1.0022
sigma	0.0065167	1.1	8811	0.040988	1.0004
nu	0.34048	1.4	5194	0.082768	1.0009

Model fit assessment:
DIC = 416.5414
PED not available from the stored object
Estimated effective number of parameters: pD = 4.88061
Total time taken: 2.3 minutes
From plots, we see that b_1 and b_0 are not mixing well and have large auto-correlations–Lets fix this.

Results with t-distribution with thin=5

JAGS model summary statistics from 40000 samples (thin = 5; chains = 4; adapt+burnin = 5000):

	Lower95	Median	Upper95	Mean	SD	Mode
b0	59.665	66.922	73.888	66.888	3.6247	–
b1	-2.8874	-1.5278	-0.16677	-1.5283	0.69211	–
b2	0.3915	1.1176	1.8391	1.1177	0.37009	–
sigma	3.6036	4.8466	6.0131	4.843	0.60732	–
nu	1.5402	16.895	77.435	25.816	26.091	–

Results with t-distribution with thin=5

JAGS model summary statistics from 40000 samples (thin = 5; chains = 4; adapt+burnin = 5000):

	MCerr	MC%ofSD	SSeff	AC.50	psrf
b0	0.076658	2.1	2236	0.3222	1.0008
b1	0.014479	2.1	2285	0.31784	1.001
b2	0.0028562	0.8	16790	0.008508	1
sigma	0.0036165	0.6	28201	-0.0035392	1.0001
nu	0.18429	0.7	20044	-0.013753	1.0002

Model fit assessment:
DIC = 416.543
PED not available from the stored object
Estimated effective number of parameters: pD = 4.87464
Total time taken: 5.1 minutes

Figure: b0

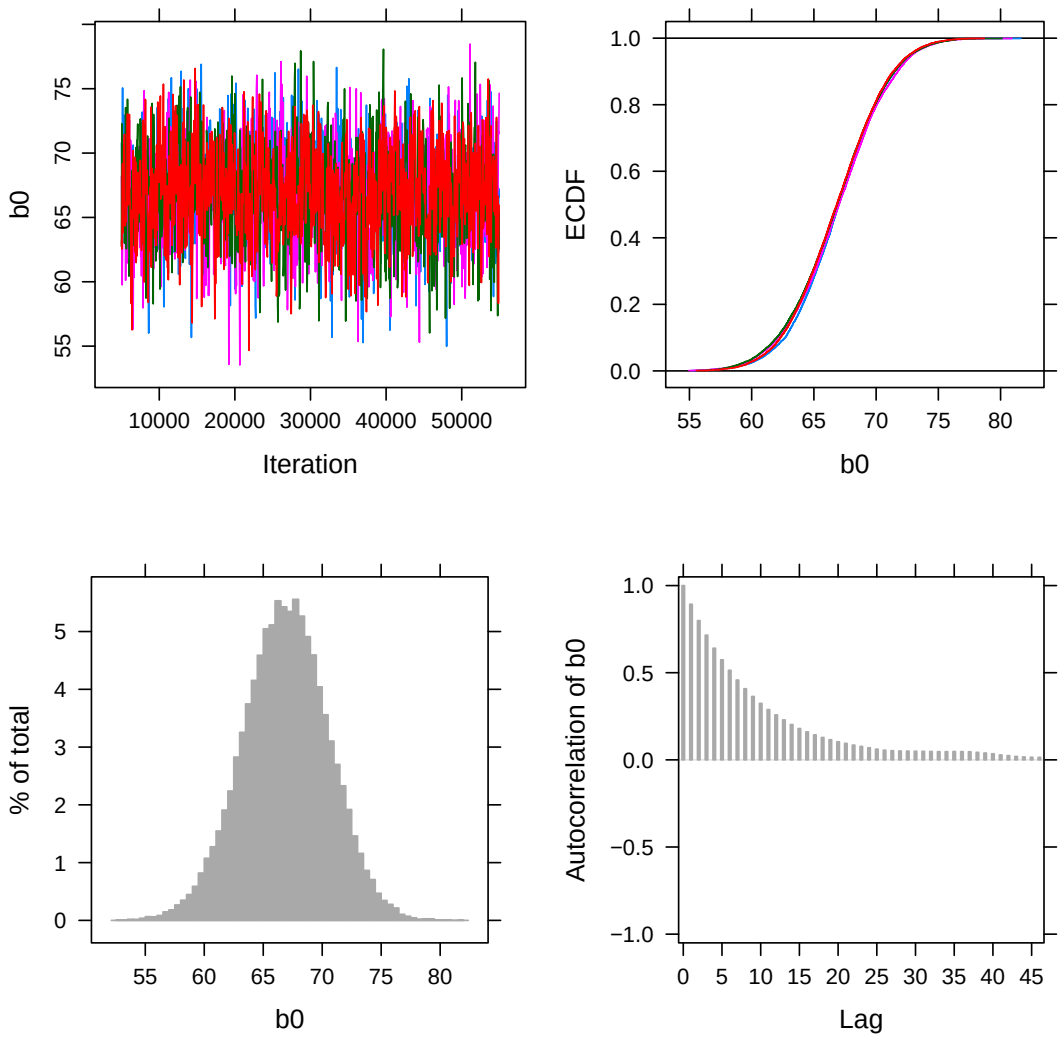


Figure: b1

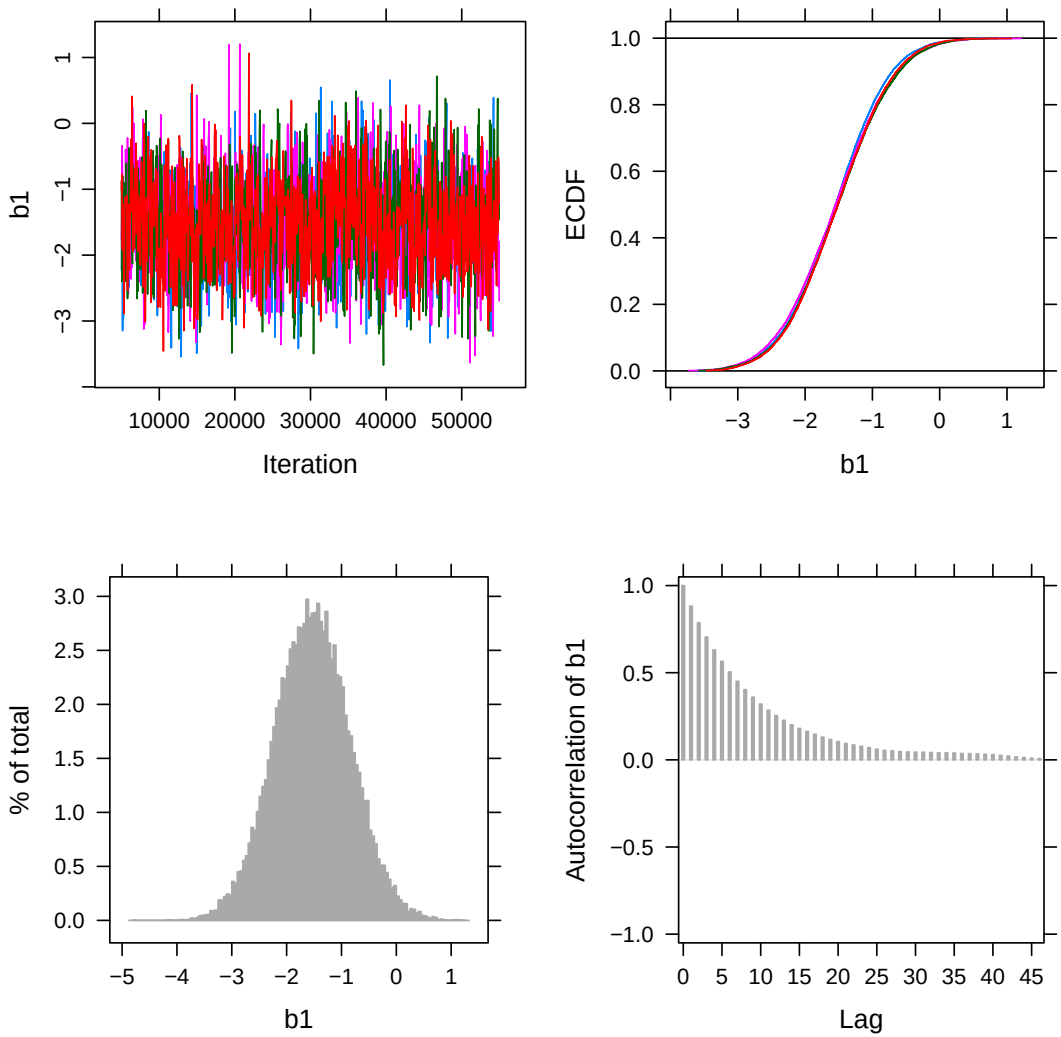

b1
Iteration
10000 20000 30000 40000 50000
ECDF
b1
% of total
b1
Autocorrelation of b1
Lag

Figure: b2

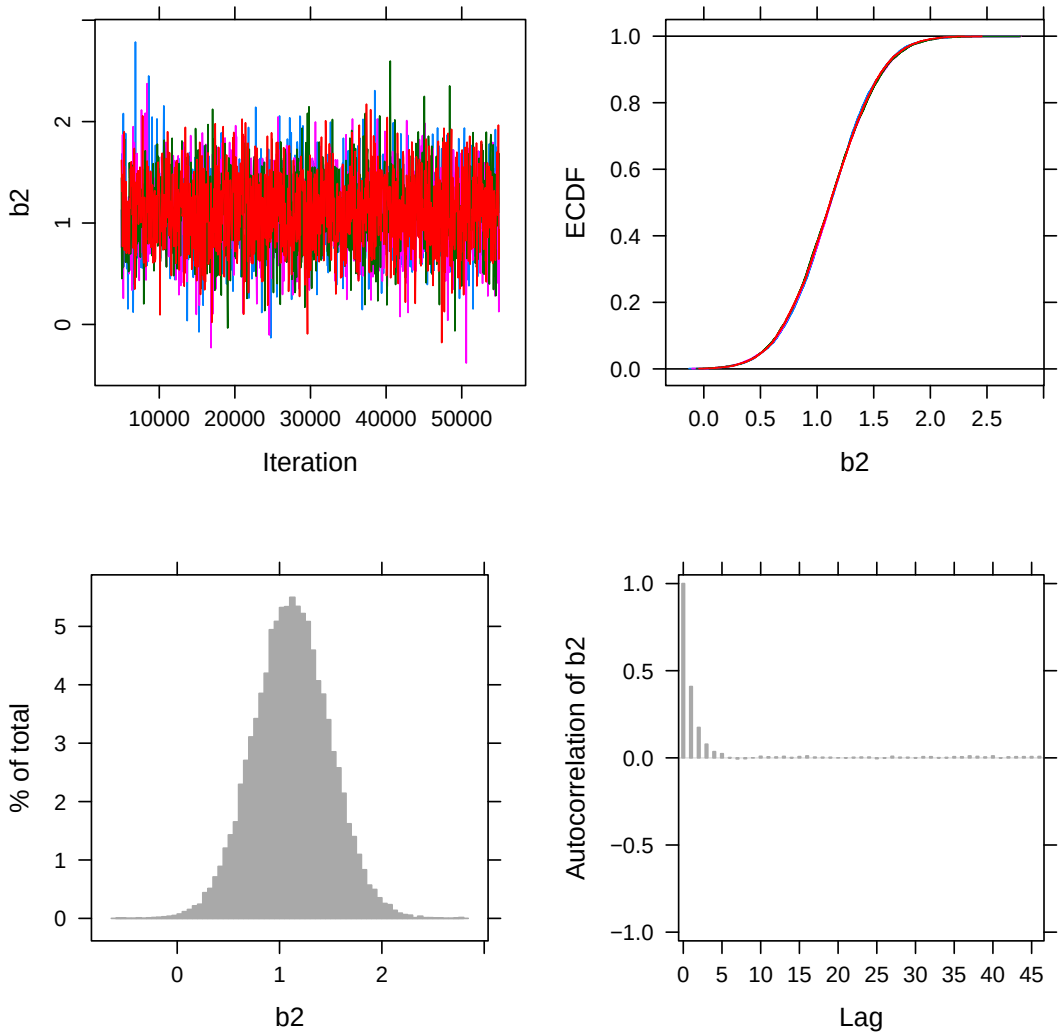

b2
Iteration
10000 20000 30000 40000 50000
0 1 2
ECDF
b2
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.5 1.0 1.5 2.0 2.5
% of total
b2
0 1 2 3 4 5
0 1 2
Autocorrelation of b2
Lag
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35 40 45

Figure: sigma

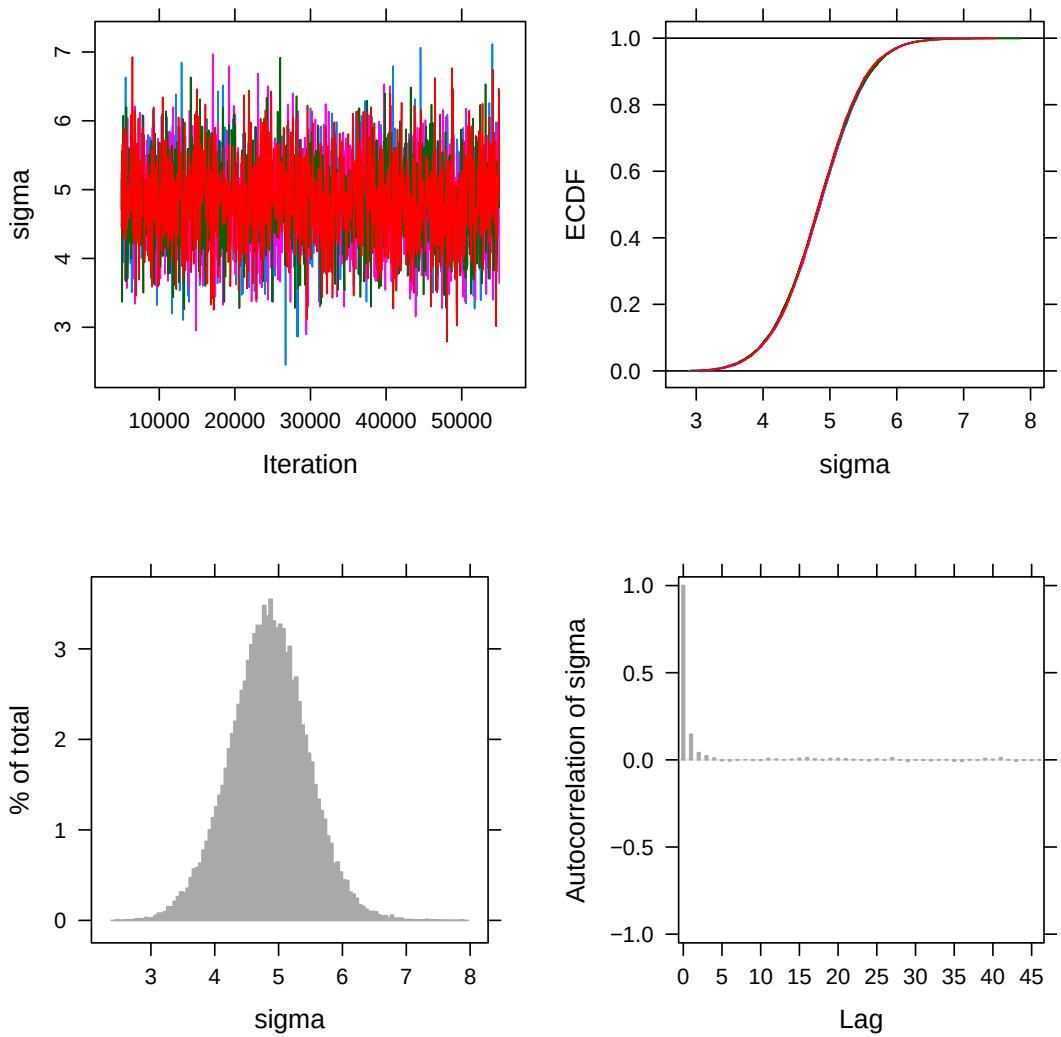


Figure: nu

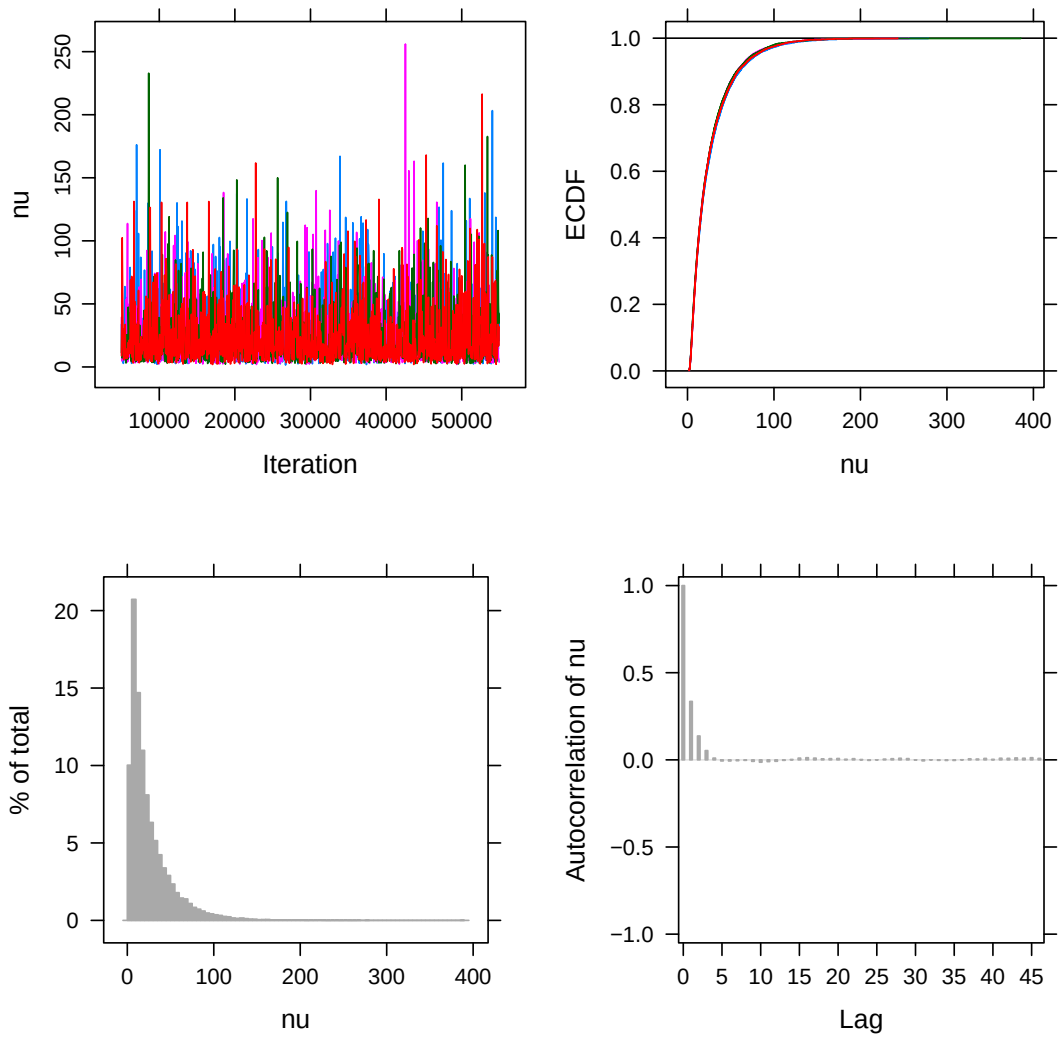


Figure: Examine posterior statistics

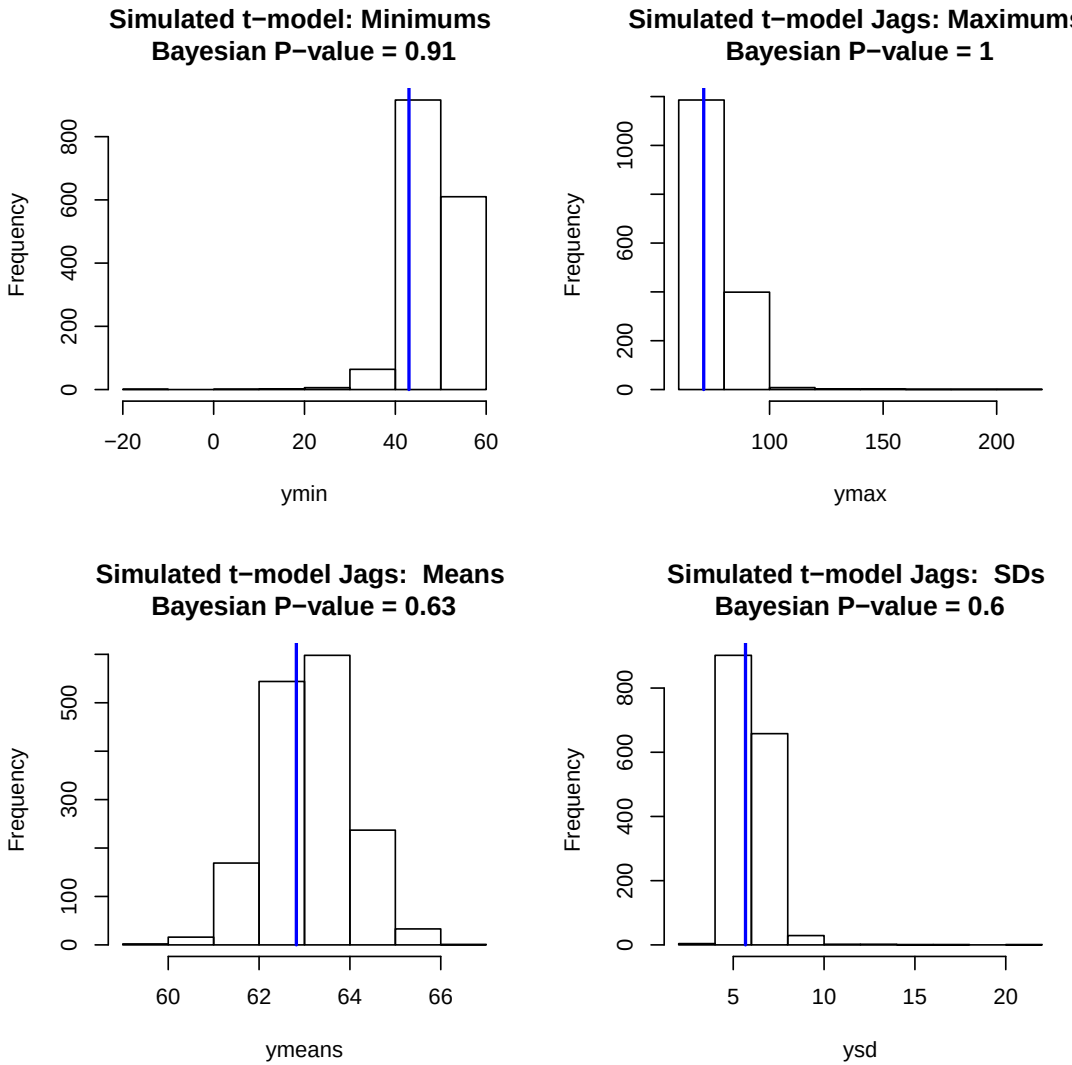
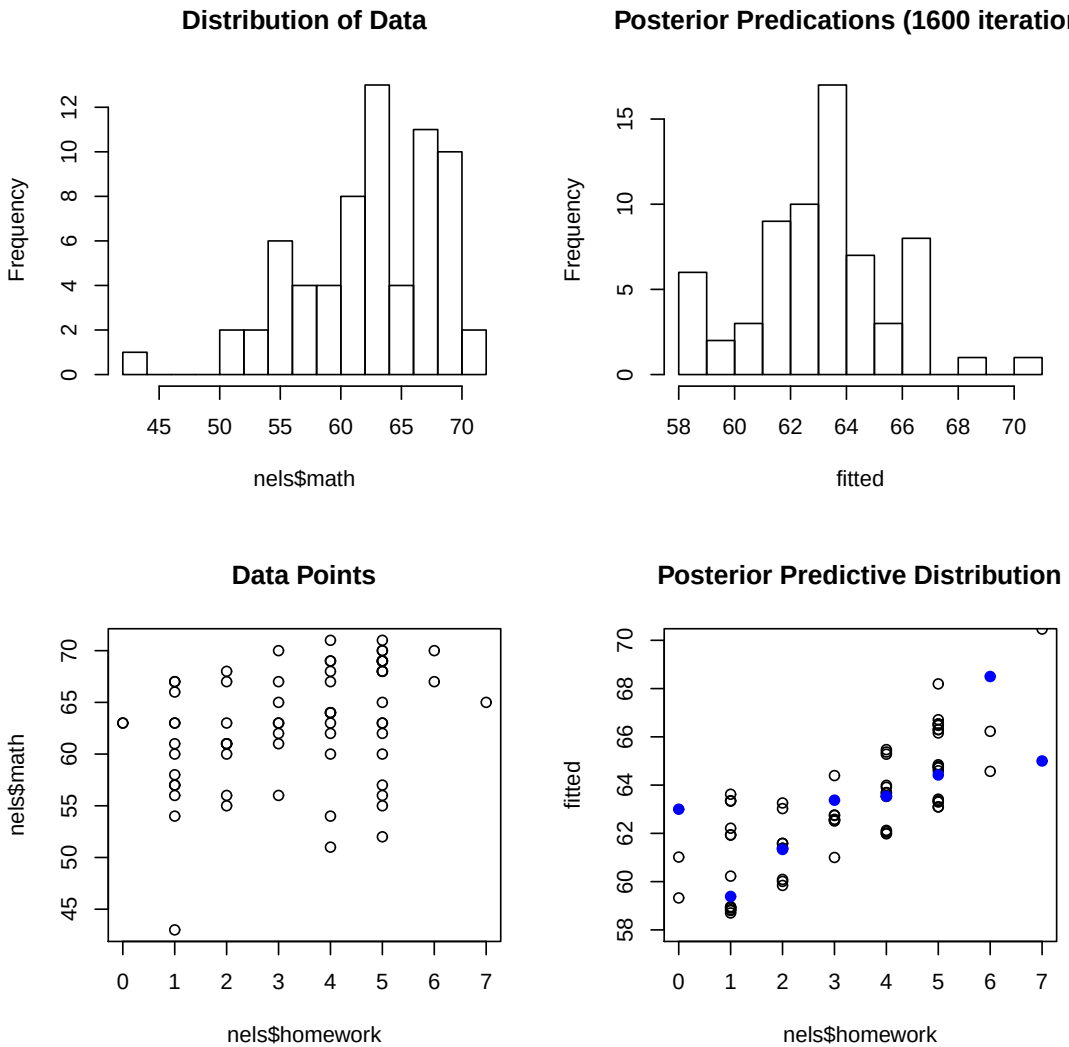


Figure: Examine posterior distribution



Summary Comments on the NELS

- In notes here I report results from raw scores.
- In the code online after doing interactions I switched back to un-centered.
- Thinning seemed to be needed to get good mixing and low auto-correlations.
- Model Evaluations:
  - Model parameter estimates seemed reasonable.
  - The normal distribution is about the same as the t-distribution; however, the *t*-produced more outlying statistics in the posterior predictive distribution.
- Improvements would not allow predicted value to be higher than the maximum on the test (i.e., deal with ceiling). Possibilities include using a different likelihood:
  - Truncated or censored distribution.
  - Beta distribution.

Model Comparison

From: Richare E. Turner "Why Gelman "hats" Bayesian model comparison" at http://www.gatsby.ucl.ac.uk/∼turner/TeaTalks/bayes-model-comp/bayes-model-comp.pdf

Conclusions

- Discrete Bayesian model comparison:
  - beware the prior
  - Uninformative priors dangerous (improper priors apocalyptic)
  - Perform a sensitivity analysis
  - Common tactic: convert model comparison into parameter estimation problem
- Philosophical inconsistency - model comparison is just (discrete) inference
- Posterior predictive tests: can tell you in what way your model is wrong without needing another to compare to another model
- Original references: Kass Greenhouse 1989, Statistical Science; Kass 1993, Journal of the Royal Statistical Society; Kass & Raftery 1995, Journal of the American Statistical Society.
- Suggestion read both Gelman's book and MacKay's book (*Information theory, inference and learning algorithms*)

If you are compelled to compare Models

We have 2 models M_1 and M_2 and data $\boldsymbol{y}$.

$$p(\theta|\boldsymbol{y}, M_k) = \frac{p(\boldsymbol{y}|\theta, M_k)p(\theta|M_k)}{p(\boldsymbol{y}|M_k)} \quad \leftarrow \text{Bayesian evidence (model likelihood)}$$

From Bayes Theorem:

$$p(M_k|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|M_k)p(M_k)}{p(\boldsymbol{y})}$$

Compute posterior odds:

$$\begin{aligned} \frac{p(M_1|\boldsymbol{y})}{p(M_2|\boldsymbol{y})} &= \frac{p(\boldsymbol{y}|M_1)}{p(\boldsymbol{y}|M_2)} \times \frac{p(M_1)}{p(M_2)} \\ &= \text{Bayes factor} \times \text{Prior Odds} \end{aligned}$$

$$\text{Bayes factor} = \frac{p(\boldsymbol{y}|M_1)}{p(\boldsymbol{y}|M_2)}$$

Bayes Factor

$$\text{Bayes factor } = BF = \frac{p(\boldsymbol{y}|M_1)}{p(\boldsymbol{y}|M_2)}$$

- Marginalized (collapsed) over parameters.
- Shows how much the prior odds change given data.
- Making a decision:
  - If $BF > 3.0$, then substantial evidence for model 1 (M_1).
  - If $BF < 1/3$, then substantial evidence for model 2 (M_2).
- BF takes into account quality of model fit to data and model complexity.
- BF favors highly predictive model and penalizes for too many unnecessary or unimportant parameters.
- Sometimes $\ln(BF)$ is reported.
- Use DIC and model parameter estimation.

Simple Method

Use the BayesFactor package in R compares all possible with model with only an intercept.
http://bayesfactorpcl.r-forge.r-project.org/

```
nels$xwhite ← as.numeric(nels$white)
bf ← regressionBF(math ∼ cparedu + chomework + ses
                  + xwhite + sex, data=nels)
bf
```

Also, the best, say 5,

```
head(bf,n=5)
```

Note: Online code I used un-centered...you get the same results.

Best 5

Bayes factor analysis

---

cparedu + chomework	:	17.83269	±0%
cparedu + chomework + ses	:	11.84489	±0%
cparedu + chomework + sex	:	8.491123	±0%
cparedu + chomework + ses + sex	:	7.718168	±0%
chomework	:	6.983524	±0%

Against denominator:

Intercept only

Bayes factor type: BFlinearModel, JZS

Alternative Comparisons

`top_compare ← head(bf)/max(bf)`

Bayes factor analysis

[1]	cparedu + chomework	:	1	±0%
	cparedu + chomework + ses	:	0.6642233	±0%
	cparedu + chomework + sex	:	0.4761549	±0%
	cparedu + chomework + ses + sex	:	0.4328101	±0%
	chomework	:	0.3916136	±0%
	cparedu + chomework + xwhite	:	0.3474389	

Against denominator:

math ∼ cparedu + chomework