Edpsy 590BAY: Linear Regression with jags

C. J. Anderson

September 20, 2019

In this document is the R code and output for Bayesian (simple) linear regression using jags.

# Setup

The libraries that I used are

library(coda)  
library(plotly)  
library(rjags)  
library(runjags)  
  
set.seed(75)

I set my working directory where the data live, where model file will be wrtten, and graphs that I save will be located at

setwd("C:/Users/cja/Dropbox/edps 590BAY/Lectures/7 Linear Regression")

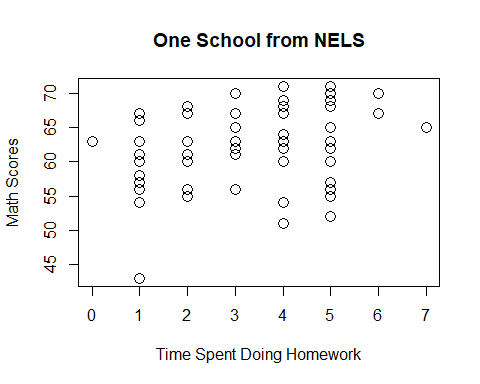
Now to read the data and look to make sure it is what I’m expecting.

nels <- read.table("nels\_school\_62821.txt",header=TRUE)  
head(nels)

## schid studid sex race homework schtyp ses paredu math clastruc size  
## 1 62821 0 1 4 4 4 1.63 6 62 3 3  
## 2 62821 1 2 4 5 4 0.48 4 68 3 3  
## 3 62821 3 1 4 5 4 0.69 5 56 3 3  
## 4 62821 4 2 4 5 4 0.60 4 68 3 3  
## 5 62821 6 2 4 2 4 0.77 4 61 3 3  
## 6 62821 7 2 4 2 4 1.19 5 61 3 3  
## urban geograph perminor rati0  
## 1 1 2 3 10  
## 2 1 2 3 10  
## 3 1 2 3 10  
## 4 1 2 3 10  
## 5 1 2 3 10  
## 6 1 2 3 10

And a graph of data that we’ll model

# Plot of data  
plot(nels$homework,nels$math,type='p',  
 main="One School from NELS",  
 xlab="Time Spent Doing Homework",  
 ylab="Math Scores",  
 cex=1.5,  
 pch=1)



# Simple Linear Regression via lm

schMathHmwk <- aggregate(nels$math, list(nels$homework), mean)

I like to add things to my figure.Below is code to add means for each value of the predictor (time spent doing homework) and a regression line.

# Adds means  
lines(schMathHmwk$Group.1,schMathHmwk$x,type='p',pch=19,col="blue")  
# Add linear regression line  
ols.lm <- lm(math~homework,data=nels)  
summary(ols.lm)  
coef(ols.lm)  
abline(ols.lm,col="blue",lwd=2)  
# Adds model and makes it big enought to read  
text(3,45,"y=59.21 + 1.09(Homework)",col="blue",cex=1.25)

# rjag For Linear Regression

## Step 1:

Create data list. Note that you should use “=” and not “<-”. Once this is set we do not need to change it (unless we change what we include as outcome or predictors/explantory variables)

dataList <- list(y=nels$math,  
 x=nels$homework,  
 N=length(nels$math),  
 sdY = sd(nels$math)  
 )

Take a quick look at this.

## Step 2:

We will start with a simple linear regression model where the data are normaly distributed and rather uniformative priors.

nelsLR1 = "model {   
 for (i in 1:N){  
 y[i] ~ dnorm(mu[i] , precision)  
 mu[i] <- b0 + b1\*x[i]   
 }  
 b0 ~ dnorm(0 , 1/(100\*sdY^2) )   
 b1 ~ dnorm(0 , 1/(100\*sdY^2) )   
 sigma ~ dunif( 1E-3, 1E+30 )  
 precision <- 1/sigma^2   
 }   
"  
writeLines(nelsLR1, con="nelsLR1.txt")

The order of things here does not matter; that is, I started with defining the likelihood/data model and then specified priors. We could have also started with the priors and then defined the likelihood.

One thing different from base R is that when using jags, dnorm want precision and not standard deviation. So 1/(100\*sd^2) is a very small number (not very prescise estimates for b0 and b1 to start), which means a very dispersed piror.  
Look for and take a look at nelsLR1.

## Step 3:

Create a list object containng intitial or starting values for the sampler.

b0Init = mean(nels$math)  
b1Init = 0  
sigmaInit = sd(nels$math)  
initsList = list(b0=b0Init, b1=b1Init, sigma=sigmaInit )

## Step 4:

Compile model and get started. I sometimes put the number of iterations very small here just to check my code from Step 2 (save a little time)

jagsNelsLR1 <- jags.model(file="nelsLR1.txt", # compiles and intializes model  
 data=dataList,  
 inits=initsList,  
 n.chains=4,  
 n.adapt=500) # iterations adaption phase

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 67  
## Unobserved stochastic nodes: 3  
## Total graph size: 166  
##   
## Initializing model

Now burn-in or warmups,

update (jagsNelsLR1, n.iter=500) # burn in of 500 iterations

And now start sampling

# gets samples from all chains for 4000 iterations  
Samples1 <- coda.samples(jagsNelsLR1, variable.names=c("b0","b1","sigma"), n.iter=4000)

## Step 5

Check whether algorithm converged. Below are things to help with this following

# To see trace and densities for each parameter  
plot(Samples1)

# Gelman statistics (Rhat or psrf)  
gelman.diag(Samples1)  
  
# Plot of Gelman statistics (Rhat or psrf)  
gelman.plot(Samples1)

# Plot of auto-correlations (each parameter for each chain)  
autocorr.plot(Samples1,auto.layout=TRUE)

# Geweke diagnonistics (each parameter for each chain)  
geweke.diag(Samples1,frac1=0.1,frac2=0.5)  
  
# Effective sample sizes  
effectiveSize(Samples1)  
  
# cumuplot (one graph per parameter per chain)  
cumuplot(Samples1,probs=c(.25,.50,.75),lwd=c(1,2),lty=c(2,1),col=c("blue","red"))

# High density intervals for each chain  
HPDinterval(Samples1)

## Step 6

If it appears the model has converged, then look at summaries of the posterior distribution.

# output summary information  
summary(Samples1)

##   
## Iterations = 1001:5000  
## Thinning interval = 1   
## Number of chains = 4   
## Sample size per chain = 4000   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE  
## b0 59.151 1.4608 0.011548 0.033527  
## b1 1.109 0.3949 0.003122 0.009075  
## sigma 5.508 0.4914 0.003885 0.005286  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## b0 56.298 58.1834 59.145 60.125 62.018  
## b1 0.325 0.8471 1.109 1.375 1.883  
## sigma 4.652 5.1647 5.468 5.809 6.579

# If you prefer runjag

You need to reset your inital values and change code that gets the sample.

Starting values are needed for each chain. For the first chain I put in smart starting values, but for the other I drew values for b0 from normal distributions with different means and standard deviations.

initsList = list(list("b0"=mean(nels$math), "b1"=sd(nels$math), "sigma"=sd(nels$math)\*\*2),   
 list("b0"=rnorm(1,-2,4), "b1"=rnorm(1,0,4) ,"sigma"=2),  
 list("b0"=rnorm(1,0,1), "b1"=rnorm(1,6,4) ,"sigma"=5),  
 list("b0"=rnorm(1,2,2), "b1"=rnorm(1,-2,4) ,"sigma"=15 )   
 )

To run runjags

out.runjags <- run.jags(model=nelsLR1, monitor=c("b0","b1","sigma","dic"),  
 data=dataList, n.chains=4, inits=initsList)

## module dic loaded

## Compiling rjags model...  
## Calling the simulation using the rjags method...  
## Adapting the model for 1000 iterations...  
## Burning in the model for 4000 iterations...  
## Running the model for 10000 iterations...  
## Simulation complete  
## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 3 variables....  
## Finished running the simulation

and some quick figures for diagnostics

plot(out.runjags)

## Generating plots...

If OK, then examine posterior statistics along with more model convergence checks.

print(out.runjags)

##   
## JAGS model summary statistics from 40000 samples (chains = 4; adapt+burnin = 5000):  
##   
## Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff  
## b0 56.364 59.182 62.08 59.183 1.4597 -- 0.021258 1.5 4715  
## b1 0.32311 1.1033 1.8605 1.1018 0.39317 -- 0.0057211 1.5 4723  
## sigma 4.5817 5.4652 6.4707 5.4979 0.48923 -- 0.0032834 0.7 22201  
##   
## AC.10 psrf  
## b0 0.10417 1.0003  
## b1 0.09833 1.0003  
## sigma -0.0034129 1  
##   
## Model fit assessment:  
## DIC = 420.0961  
## [PED not available from the stored object]  
## Estimated effective number of parameters: pD = 3.08149  
##   
## Total time taken: 4 seconds

## Robust Regression

Rather than use the normal distribution as our likelihood, we can use Student’s t-distribution. To do this, we swap out normal from the likelihood in the model defination and add in degrees of freedom as a paramter (“nu”), which means it needs a prior distribution.

tMod1 = "model {for (i in 1:N){  
 y[i] ~ dt(mu[i] , precision,nu)   
 mu[i] <- b0 + b1\*x[i]  
 }  
 b0 ~ dnorm(0 , 1/(100\*sdY^2) )   
 b1 ~ dnorm(0 , 1/(100\*sdY^2) )   
 sigma ~ dunif(0, 1E+10 )   
 precision <- 1/sigma^2  
 nuMinusOne ~ dexp(1/29)   
 nu <- nuMinusOne+1   
 } "  
  
writeLines(tMod1, con="tMod1.txt")

We actually set the proir for degrees of freedom less one, but add one back in.

initsList = list("b0"=mean(nels$math), "b1"=sd(nels$math), "sigma"=sd(nels$math)\*\*2)

Iinitalize the model

jagsModelLm2 <- jags.model(file="tMod1.txt", # compiles and intializes model  
 data=dataList,  
 inits=initsList,  
 n.chains=4,  
 n.adapt=1000) # adaptive phase for 1000 iterations

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 67  
## Unobserved stochastic nodes: 4  
## Total graph size: 169  
##   
## Initializing model

update (jagsModelLm2, n.iter=1000) # burn in of 1000 iterations

# contains samples from all chains with 2000 iterations  
tSamples2 <- coda.samples(jagsModelLm2, variable.names=c("b0","b1","sigma","nu"), n.iter=2000)

If everything look good (i.e., re-do steps in Step 5 above)

# output summary information  
summary(tSamples2)

##   
## Iterations = 2001:4000  
## Thinning interval = 1   
## Number of chains = 4   
## Sample size per chain = 2000   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE  
## b0 59.460 1.4888 0.016646 0.06378  
## b1 1.095 0.4018 0.004492 0.01645  
## nu 28.055 25.1648 0.281351 0.68773  
## sigma 5.101 0.5830 0.006518 0.01218  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## b0 56.421 58.5086 59.506 60.452 62.326  
## b1 0.329 0.8262 1.093 1.353 1.908  
## nu 3.946 10.6412 19.757 37.447 96.872  
## sigma 3.973 4.7178 5.097 5.470 6.267

### Serious Model Checking

## Removing Outlier

Note that there is an outlier in that math scores (<45) and is only does 1 hours of homework. We will remove the outlier and re-run model to see if there is an impact on the results.

Remove the student with the smallest math score:

nels2 <- subset(nels, math>min(nels$math))  
min(nels2$math)

## [1] 51

We need to re-do data list

dataList <- list(y=nels2$math,  
 x=nels2$homework,  
 N=length(nels2$math),  
 sdY = sd(nels2$math)  
 )

We can use out initial model again, “nelsLR1.txt” but out initial starting values are dependent on the data set so this needs to be re-done

b0Init = mean(nels2$math)  
b1Init = 0  
sigmaInit = sd(nels2$math)  
initsList = list(b0=b0Init, b1=b1Init, sigma=sigmaInit )

jagsModelLm1x <- jags.model(file="ModelLR1.txt",   
 data=dataList,  
 inits=initsList,  
 n.chains=4,  
 n.adapt=500) # adaptive phase for 500 iterations

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 66  
## Unobserved stochastic nodes: 3  
## Total graph size: 164  
##   
## Initializing model

update (jagsModelLm1x, n.iter=1000) # burn in of 1000 iterations

And re-run and get samples

update (jagsModelLm1x, n.iter=1000) # burn in of 1000 iterations  
  
# gets samples from all chains for 2000 iterations  
Samples1x <- coda.samples(jagsModelLm1x, variable.names=c("b0","b1","sigma"), n.iter=2000)

Check convergence (not done here, see lecture notes or re-do Step 5 above) and if OK take a look at results.

# output summary information  
summary(Samples1x)

##   
## Iterations = 2501:4500  
## Thinning interval = 1   
## Number of chains = 4   
## Sample size per chain = 2000   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE  
## b0 60.1265 1.4103 0.015768 0.048067  
## b1 0.8968 0.3787 0.004234 0.012946  
## sigma 5.0724 0.4637 0.005184 0.007405  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## b0 57.441 59.1389 60.1265 61.073 62.918  
## b1 0.145 0.6455 0.9007 1.154 1.617  
## sigma 4.250 4.7442 5.0430 5.365 6.077

Should we keep the outlier in data or not? For the rest of this set of notes, I left it in

### Posterior Checks

We now have a posterior distribution from which we can draw values. I will draw from this posterior and compute Bayesian p-values for the parameter estimates and examine these relative to the posteriors for the parameters.

There are two ways to proceed. I’ll do one here and for the other (which put computations within the model definition) is in the rode-code for the nels example.

I will take 200 draws from the posterior

n <- length(nels2$math)  
replications <- 200  
yrep <- matrix(99,nrow=n,ncol=replications)  
for (s in 1:replications){  
 b0 <- rnorm(1,60.131,sd=1.3714)  
 b1 <- rnorm(1,0.89476,sd=0.36696)  
 for (i in 1:n){  
 yrep[i,s] = b0 + b1\*nels2$homework[i] + rnorm(1,0,5.0667)  
 }  
}

Using my new data set, I will first computer descriptive statistic (check to see if results are reasonable)

# descriptive statistics  
(yhats <- apply(yrep,2,"mean"))

## [1] 62.61260 63.82591 59.67689 61.17312 63.64983 61.26399 63.73619  
## [8] 64.04615 64.50819 59.74149 61.78996 62.47586 62.08108 60.02087  
## [15] 62.66097 64.67421 61.19023 63.20546 63.39218 63.40103 57.94605  
## [22] 62.78650 64.00300 63.55406 68.80652 61.54913 64.35765 63.24897  
## [29] 63.35316 62.61541 61.56495 61.20796 63.16484 64.69646 62.18958  
## [36] 62.75815 63.48968 63.21899 61.40461 60.22854 62.10982 59.33963  
## [43] 62.81267 60.55843 63.19844 63.52826 66.18125 63.14107 62.79103  
## [50] 65.20724 63.40608 63.97457 62.75656 64.14513 62.96169 64.44578  
## [57] 61.73521 63.56658 67.47703 62.77801 65.52872 65.02461 63.80816  
## [64] 65.02033 62.77310 62.83222 61.93551 63.80171 61.66090 64.25563  
## [71] 62.10987 65.49292 62.48380 64.21227 67.86128 62.13526 63.57892  
## [78] 62.51884 59.80450 61.75064 59.27396 67.04433 63.44386 65.03843  
## [85] 65.07846 61.12446 59.76711 59.64352 66.95809 62.00520 63.91325  
## [92] 63.88933 64.53638 62.96113 64.80805 60.95147 62.85692 62.19346  
## [99] 63.48772 63.03964 63.28183 65.00957 62.58553 61.25042 60.27611  
## [106] 62.24266 65.83815 64.37275 61.50778 63.53158 63.89622 62.72883  
## [113] 63.30210 62.43731 61.97227 63.01923 65.19068 61.44617 64.01223  
## [120] 60.51267 62.55460 65.20308 60.51672 62.48156 59.55635 62.29600  
## [127] 60.96311 60.63113 61.97609 62.53570 61.10824 66.54775 64.60565  
## [134] 62.65641 64.32819 62.18067 64.09478 59.88073 65.49589 62.49622  
## [141] 63.54664 59.10924 63.73997 62.02371 66.36321 58.93314 59.53765  
## [148] 58.47095 63.65038 63.12966 63.89922 67.90585 61.45653 60.84514  
## [155] 65.32521 63.22330 61.97881 60.48072 66.57812 61.55398 60.98479  
## [162] 63.47946 64.73928 62.92600 62.48898 63.44641 65.10251 61.55683  
## [169] 65.60727 63.55367 64.18558 64.33469 61.69203 63.12176 59.96818  
## [176] 62.04742 59.96854 61.64531 64.13950 68.37682 61.56783 62.43113  
## [183] 63.85709 65.10466 62.41739 63.09244 62.44403 63.44146 62.43898  
## [190] 62.23597 62.56211 63.51835 60.30868 66.87034 62.86589 62.70707  
## [197] 63.31269 60.53345 64.77172 58.33021

(ymin <- apply(yrep,2,"min"))

## [1] 54.27060 50.59387 44.21664 45.58766 49.38106 49.64534 50.28527  
## [8] 44.60370 52.40173 46.27385 48.80073 50.45895 48.83315 46.58701  
## [15] 46.98641 54.66459 47.53438 50.18483 49.87115 53.76003 45.06983  
## [22] 48.34286 51.20071 52.85529 59.56521 47.18070 50.70222 50.54762  
## [29] 53.11847 51.98894 48.72638 46.93585 47.57535 51.14510 46.07249  
## [36] 47.80931 54.11221 52.02614 43.50901 48.39205 48.51271 46.76048  
## [43] 48.33482 48.94310 51.33673 50.85769 55.73274 54.24298 50.50160  
## [50] 52.97882 49.33322 52.61549 52.23418 52.63243 49.77553 47.10664  
## [57] 50.18151 52.93299 52.29525 44.24106 53.59420 50.73026 53.17176  
## [64] 52.64425 51.23164 51.11557 53.00968 52.66429 49.19378 56.18062  
## [71] 49.67533 53.42703 47.36476 46.86868 51.01660 53.35030 50.72040  
## [78] 46.19318 40.25886 47.10239 49.47217 51.18663 50.38417 54.86462  
## [85] 53.48901 47.88054 43.59997 47.35003 55.27846 53.27016 52.51349  
## [92] 49.01776 54.74983 53.27322 55.61611 51.58019 50.43313 52.53685  
## [99] 47.98761 39.57817 52.63379 51.42034 49.64203 48.97890 47.18939  
## [106] 48.65440 51.99352 49.58836 44.80809 52.62432 49.58754 47.90185  
## [113] 52.35606 47.75126 50.14501 52.72752 53.13716 49.19497 49.70469  
## [120] 50.43776 51.77663 51.72352 44.09785 45.23805 47.81256 51.33719  
## [127] 50.76845 46.58494 50.18771 50.72655 49.30862 52.22127 50.34030  
## [134] 49.87947 52.72141 53.91770 53.02701 47.91507 51.58755 48.41557  
## [141] 49.38360 46.87836 52.56754 52.75075 56.97255 43.57916 49.34980  
## [148] 47.40445 52.44411 52.16003 54.31520 54.60621 51.14756 47.11478  
## [155] 50.88617 50.13531 43.58872 49.40368 52.20674 46.95002 46.31430  
## [162] 50.63496 52.91385 48.22756 48.37369 51.25872 51.57804 49.91697  
## [169] 51.44452 45.61891 50.11193 55.50835 47.94999 48.64455 48.03810  
## [176] 51.49909 48.80862 52.35033 53.37358 54.24737 51.49057 49.74885  
## [183] 52.42459 50.73944 51.55253 51.85417 45.39774 49.93389 50.61138  
## [190] 48.80931 49.25437 47.15546 49.03048 52.61122 52.75573 50.01946  
## [197] 53.02288 43.93916 54.54211 43.12556

(ymax <- apply(yrep,2,"max"))

## [1] 74.49226 71.00713 73.55363 69.29342 74.12289 74.78821 75.87283  
## [8] 74.63091 78.44231 72.87570 73.73582 77.86237 74.09006 70.88554  
## [15] 74.93028 77.62617 71.83553 74.73470 74.45828 74.13052 68.43432  
## [22] 75.18879 77.47134 76.66052 78.86354 71.49039 77.98835 74.94827  
## [29] 73.86357 74.63570 77.17518 72.66765 74.63237 76.92580 79.78247  
## [36] 72.96128 74.20669 74.74894 75.80470 69.72862 72.12606 72.74738  
## [43] 77.79152 74.90893 75.07461 78.81730 79.54104 75.80193 78.37240  
## [50] 79.44146 74.99082 73.10673 76.34777 76.80983 74.02488 75.45444  
## [57] 77.70127 72.93088 78.61823 78.49871 79.16354 75.41297 75.60417  
## [64] 73.69701 77.83408 75.67996 73.60944 83.27332 73.94545 79.60991  
## [71] 77.00260 77.99521 73.56487 76.75091 82.63206 72.85815 75.92422  
## [78] 76.13135 68.90408 77.35203 74.32363 82.84671 81.07187 74.34204  
## [85] 78.77030 71.07347 72.53508 72.13135 79.99132 72.73274 75.29856  
## [92] 76.70662 78.31676 76.20492 81.48721 75.20898 72.65473 72.03915  
## [99] 74.71135 73.50120 76.49727 77.52956 71.47676 78.73565 71.78177  
## [106] 77.29804 82.89999 78.25491 74.55068 74.48506 78.00223 74.06662  
## [113] 74.06575 80.19893 75.43478 75.96783 76.28333 71.15052 76.36806  
## [120] 70.28830 75.87450 74.17817 74.31354 75.95337 72.76423 77.38265  
## [127] 73.20114 72.08692 71.55657 71.92931 72.49655 82.87952 78.85580  
## [134] 77.84111 72.39426 76.70563 72.15030 71.87737 78.60816 75.12574  
## [141] 78.31807 69.04163 72.77884 74.26233 75.89508 71.34251 71.95545  
## [148] 69.18259 77.10767 74.69941 72.26075 80.37049 73.23003 70.06427  
## [155] 77.88431 72.68696 74.63986 70.70674 77.37809 70.41411 71.19237  
## [162] 81.72286 75.39601 75.32821 73.71982 73.66796 79.36361 71.77358  
## [169] 83.61587 77.67464 76.88704 76.19320 74.72574 76.10752 72.22036  
## [176] 71.29569 70.38770 74.85752 78.65137 83.51382 70.12315 72.44999  
## [183] 74.11905 80.62185 73.63443 74.62383 76.57776 78.49893 75.27010  
## [190] 78.01861 73.88525 77.71224 69.23090 78.23307 77.89041 74.74604  
## [197] 74.03435 73.44574 79.51025 74.88559

(ysd <- apply(yrep,2,"sd"))

## [1] 4.763442 5.117990 4.954485 4.736275 5.643220 5.181529 6.056715  
## [8] 5.537291 4.960785 5.754902 5.074959 5.294975 5.510320 4.888613  
## [15] 4.954679 4.658550 4.827613 5.249844 5.690083 5.419633 5.165827  
## [22] 5.570355 5.308058 5.746821 4.804305 5.754741 6.148586 5.791271  
## [29] 4.578258 4.960173 5.634079 5.394252 5.223740 5.905053 6.252849  
## [36] 6.041594 4.719391 4.899367 6.433288 3.994564 5.215042 5.095282  
## [43] 5.332637 5.694287 5.147173 5.656708 5.030179 4.729378 5.409662  
## [50] 5.349496 5.424268 5.288451 5.019116 5.683502 5.138450 5.642280  
## [57] 5.590887 4.787745 5.184271 5.764359 6.098822 5.290713 4.895856  
## [64] 4.451780 5.096762 5.486961 4.659844 5.179808 5.306042 4.895814  
## [71] 6.022870 5.447957 4.818881 5.877354 6.621596 5.011793 5.522822  
## [78] 5.214273 5.226208 4.870112 5.742397 5.851535 5.699981 4.665178  
## [85] 5.048454 4.773196 5.504102 4.662965 5.106278 5.067469 5.324336  
## [92] 6.365350 5.082026 4.961233 5.100348 5.062531 4.664520 4.557201  
## [99] 5.512809 6.449884 5.164906 5.424880 5.109129 5.471132 5.263152  
## [106] 5.341911 6.439572 6.179466 6.200348 4.654691 5.280144 5.327094  
## [113] 5.476767 6.438476 5.411108 4.959105 5.610909 5.320274 5.576431  
## [120] 5.285586 5.503111 4.524425 5.549379 5.864654 5.525852 4.984259  
## [127] 5.540824 5.261889 4.648762 4.697077 5.450085 6.284494 6.220931  
## [134] 5.911713 4.892313 4.813579 4.799412 5.131141 5.678421 5.192840  
## [141] 5.982086 5.725126 4.540060 4.706909 4.176360 5.735104 4.860258  
## [148] 4.826644 5.683884 5.058533 4.039171 6.146660 5.006124 5.131071  
## [155] 7.012625 5.200815 5.746166 4.922467 5.186647 4.593285 5.823200  
## [162] 6.213421 5.683452 6.486133 5.146608 4.825547 6.059671 5.201264  
## [169] 6.882230 5.393951 5.703069 4.697515 5.317041 5.958598 5.734507  
## [176] 4.959088 5.038907 5.206196 4.690179 5.634525 4.454244 5.065870  
## [183] 4.884484 5.420994 5.163493 5.365021 5.864830 5.418291 4.794544  
## [190] 5.646218 5.341446 5.786792 5.008361 5.370532 6.028620 5.707879  
## [197] 5.094081 5.665250 5.147194 5.648895

Look OK?

To compute Bayesian p-values,

# Bayesian p-values  
(pmean <- mean(mean(nels2$math) < yhats))

## [1] 0.465

(pmin <- mean(min(nels2$math) < ymin))

## [1] 0.42

(pmax <- mean(max(nels2$math) < ymax))

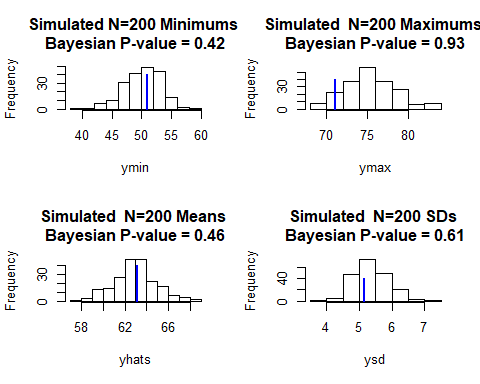
## [1] 0.93

(psd <- mean(sd(nels2$math) < ysd))

## [1] 0.61

Lets take a look at these p-values relative to distributions of each statistic.

par(mfrow=c(2,2))  
# Graphs of descriptive statistics from posterior predictive distribution   
   
par(mfrow=c(2,2))  
hist(ymin,breaks=10,main=paste("Simulated N=200 Minimums", "\nBayesian P-value =", round(pmin, 2)))  
lines(c(min(nels2$math),min(nels2$math)),c(0,40),col="blue",lwd=2)  
  
hist(ymax,breaks=10,main=paste("Simulated N=200 Maximums", "\nBayesian P-value =", round(pmax, 2)))  
lines(c(max(nels2$math),max(nels2$math)),c(0,40),col="blue",lwd=2)  
  
hist(yhats,breaks=10,main=paste("Simulated N=200 Means", "\nBayesian P-value =", round(pmean, 2)))  
lines(c(mean(nels2$math),mean(nels2$math)),c(0,40),col="blue",lwd=2)  
  
hist(ysd,breaks=10,main=paste("Simulated N=200 SDs", "\nBayesian P-value =", round(psd, 2)))  
lines(c(sd(nels2$math),sd(nels2$math)),c(0,40),col="blue",lwd=2)

 And one more set of figures to this lecture.

# Graphs of data and posterior predictive distribution  
  
ypred <- apply(yrep,1,"mean")  
par(mfrow=c(2,2))  
hist(nels2$math,main="Data Distribution",breaks=10)  
hist(ypred,main="Predicted Posterior Distribution",breaks=10)  
plot(nels2$homework,nels2$math,main="Data: math x homework")  
plot(nels2$homework, ypred,type='p',main='Predictions: math x homework',ylim=c(59,80))  
schMathHmwk <- aggregate(nels2$math, list(nels2$homework), "mean")  
lines(schMathHmwk$Group.1,schMathHmwk$x,type='p',pch=19,col="blue")

