

BRIGHAM YOUNG UNIVERSITY

Applied Bayesian Statistics

STAT 451

ADAM JACKMAN

Semester
WINTER 2013

Teacher
DR. GIL FELLINGHAM

April 11, 2013, 00000

Contents

1	Class Introduction	1
2	Bayesian Methodology	1
2.1	Metropolis Sampler	1
2.1.1	Metropolis Sampler Steps	1
2.1.2	Metropolis Code	2
3	Two Sample t test	3
4	Diagnostics	6
5	Posterior Predictives	10
6	Linear Regression	12
7	Multiple Linear Regression	16
8	ANOVA	20
8.1	Compare Means	24
8.2	ANOVA Marginal Means	25
8.3	ANOVA Interactions	26
8.4	Linear Algebra Solution	26
9	Analysis of Covariance ANCOVA	27
10	Multiple Sources of Variation	32
11	Random Coefficients Model	35
12	Hierarchical Models	39
13	Exam Debriefing	42
14	Logistic Regression	43
14.1	Logistic Regression In SAS	48
14.2	Random Effects	50
15	Poisson Likelihood	53
15.1	Log Poisson	56
15.2	Complex Model Posterior Predictive	57
15.3	Non Diagonal Covariance	59
15.4	Exam 2 Debriefing	60
16	section name	60
17	Censored Data	63

∞ January 8, 2013 ∞

1 Class Introduction

Office Hours Monday

Keep a copy of the code

Preliminary assignment email now.

2 Bayesian Methodology

Parameterize state of knowledge

Estimate parameters

$$f(\underline{\theta} | \underline{x}) = \frac{L(\underline{x} | \underline{\theta}) \prod(\underline{\theta})}{\int \int_{\underline{\theta}} L(\underline{x} | \underline{\theta}) \prod(\underline{\theta}) d\underline{\theta}} \propto L(\underline{x} | \underline{\theta}) \prod(\underline{\theta})$$

2.1 Metropolis Sampler

$$\begin{aligned} & \pi \left[\frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} \exp\left(-\frac{x}{\beta}\right) \right] + (\pi - 1) \left[\frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} \exp\left(-\frac{x}{\beta}\right) \right] \\ & \quad \frac{1}{4} \left[\frac{1}{\Gamma(2)3^2} x^{2-1} \exp\left(-\frac{x}{3}\right) \right] + \frac{3}{4} \left[\frac{1}{\Gamma(4)3^4} x^{4-1} \exp\left(-\frac{x}{3}\right) \right] \\ & \quad k = 4\Gamma(4)3^3 \end{aligned}$$

$$g(x) = k \cdot f(x) = 18x \exp\left(-\frac{x}{3}\right) + x^3 \exp\left(-\frac{x}{3}\right)$$

2.1.1 Metropolis Sampler Steps

- Initial $x = x_0$
- for $i = 1$ to n
 - generate candidate c
 - $\frac{g(c)}{g(x_{i-1})} = r$
 - let $x_i = c$ with $P = \min(1, r)$

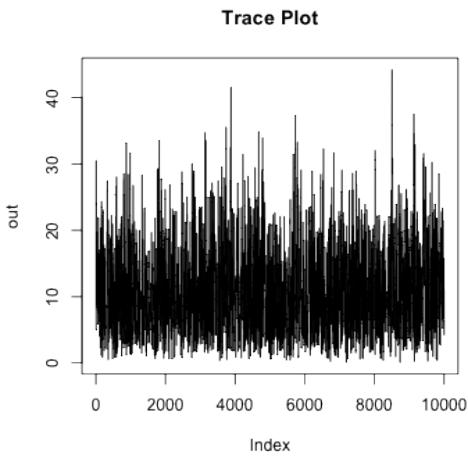
2.1.2 Metropolis Code

```
# Define g function
g <- function(x) {
  18 * x * exp(-x/3) + x^3 * exp(-x/3)
}

# Write Sampler
out <- NULL
out[1] <- 5
candidate.sigma <- 20
counter <- 0
samples <- 10^4
for (i in 2:samples) {
  out[i] <- out[i - 1]
  candidate <- rnorm(1, out[i - 1], candidate.sigma)
  if (candidate > 0) {
    r <- g(candidate)/g(out[i - 1])
    if (r > runif(1, 0, 1)) {
      out[i] <- candidate
      counter <- counter + 1
    }
  }
}
message(counter/samples) # should be about 24-40

## 0.3285

plot(out, type = "l", main = "Trace Plot")
```



Trace Plot looks good

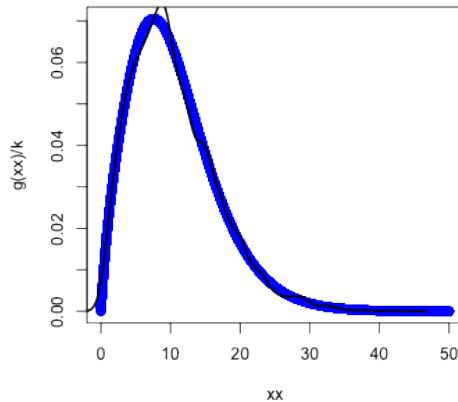
```
xx <- seq(0, 50, length = 1e+05)
k <- 4 * gamma(4) * 3^3
# Plot the exact distribution
plot(xx, g(xx)/k, lwd = 2, col = "blue")
# curve(.25*dgamma(x, shape=2, scale=3)+.75*dgamma(x, shape=4, scale=3),
# add=TRUE, col='blue') Both of these should produce the same line

lines(density(out), lwd = 2)

# Exact Mean
0.25 * (2 * 3) + 0.75 * (4 * 3)
## [1] 10.5

# Approximate Mean
```

```
mean(out)
## [1] 10.35
```



 January 10, 2013

Paul Sabin — Stat 451 TA
 Tuesday 2:30-4:00
 Friday: 9:30-11:00
 Office: 233
 Email:r.paul.sabin@gmail.com

```
# For Winbugs
install.packages("ARM", "R2WinBUGS")

# For jags
install.packages("R2jags")
```

3 Two Sample t test

$$y \sim \mathcal{N}(\mu, \sigma^2) \quad i = 1, 2$$

$$f(\mu_1) = \mathcal{N}(\mu = 0, \sigma^2 = 1000) f(\sigma^2) = \text{Unif}(0, 5000)$$

PROC MCMC two group t test

```
1 data examp2;
infile dataset;
input tmt response;
run;
5
/*
nmc: Number of Marcov Chains
nbi: Number Burn in
```

```

10 dic: information criteria
*/
proc mcmc data=examp2 outpost=post2 seed=1234 nmc=100000 nbi=5000
            statistics (alpha=.05)=(summary interval) thin=10
            moniter=( parms mudif varratio ) propcov=quannew
            diagnostics=(all) dic ;
15 array mu[2] mu1-mu2;
array sig2 [2] sig21-sig22;
parms: mu: sig2 :;
prior mu: ~ normal(0, sd=10000);
20 prior sig21 ~ unif(9, 5000); /* gamma(shape=30, scale=50) */
prior sig22 ~ unif(9, 5000);
mudif = mu1 - mu2;
varratio = sig21/sig22;
mm = mu[tmt];
25 vv = sig2[tmt];
model response ~ normal(mm, var=vv);
run;

proc export data=post2 outfile='twogroups.csv' dbms=csv replace;
30 run;

```

HPD — Highest Posterior Density the smallest interval

Fischer variance problem — a two sample t test with unequal variances

Check the posterior autocorrelations

∞ January 15, 2013 ∞

```

library(rjags)
library(R2jags)

```

JAGS uses precisions, not variances. Precisions are $\frac{1}{\sigma^2}$

```

1 library(R2jags)
mdl <- "
  model {
    for (i in 1:33) {
      y[i] ~ dnorm(mu[tmt[i]], prec[tmt[i]])
    }

    for (i in 1:2) {
      mu[i] ~ dnorm(0, 0.000001)
      vr[i] ~ dunif(0, 5000)
      prec[i] <- 1/vr[i]
    }
  }
",
15 mdl <- "
  model {
    for (i in 1:33) {
      y[i] ~ dnorm(mu[tmt[i]], prec[tmt[i]])
    }
"
20

```

```

    for (i in 1:2) {
      mu[i] ~ dnorm(100,0.01)
      vr[i] ~ dgamma(20,.05)
      prec[i] <- 1/vr[i]
25   }
}
,"mdl <- "
  model {
30    for (i in 1:33) {
      y[i] ~ dnorm(mu[tmt[i]], prec[tmt[i]])
    }

    for (i in 1:2) {
      mu[i] ~ dnorm(100,0.01)
      vr[i] ~ dgamma(10,200)
      prec[i] <- 1/vr[i]
    }
}
40 "
groups <- read.table("../data/01twogroups.dat", col.names=c("tmt", "y"))
writeLines(mdl, "twogroups.jags")
tmt <- groups$tmt
y <- groups$y
45 # Data to go into jags
data.jags <- c('tmt','y')
# what parameters to keep track of
parms <- c('mu','vr')
# Initial Values
50 innts <- function() {list('mu' = rnorm(2,125,5), 'vr' = runif(2,0,5000))}
twogroups.sim <- jags(data=data.jags, parameters.to.save=parms, inits=innts,
  model.file="twogroups.jags",
  n.iter=11000, n.burnin=1000, n.chains=1, n.thin=1)

twogroups.sim
55 sims <- as.mcmc(twogroups.sim)
plot(sims)

plot(sims[,2], type='l')

```

∞
January 17, 2013
∞

You need to think about priors

4 Diagnostics

```

library(R2jags)
mdl <- "
  model {
    for (i in 1:33) {
      y[i] ~ dnorm(mu[tmt[i]], prec[tmt[i]])

```

```

}

for (i in 1:2) {
  mu[i] ~ dnorm(0,0.000001)
  vr[i] ~ dunif(0,5000)
  prec[i] <- 1/vr[i]
}
}

groups <- read.table("data/01twogroups.dat", col.names=c("tmt", "y"))
writeLines(mdl, "code/twogroups.jags")
tmt <- groups$tmt
y <- groups$y
# Data to go into jags
data.jags <- c('tmt', 'y')
# what parameters to keep track of
parms <- c('mu', 'vr')
# Initial Values
innts <- function() {list('mu' = rnorm(2,125,5), 'vr' = runif(2,0,5000))}
twogroups.sim <- jags(data=data.jags, parameters.to.save=parms, inits=innts, model.file="code/twogroups.jags",
n.iter=6000, n.burnin=1000, n.chains=1, n.thin=1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 76
##
## Initializing model

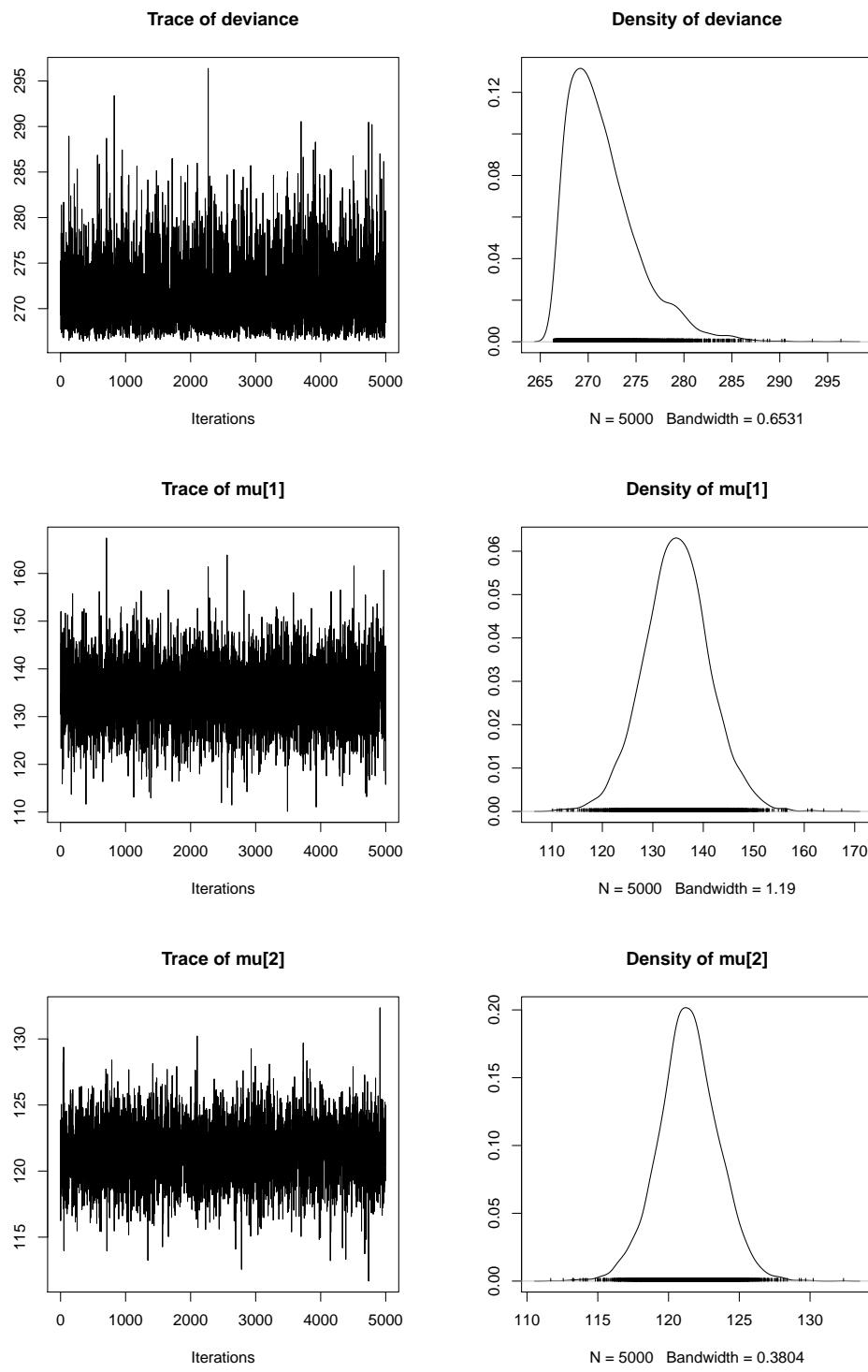
twogroups.sim

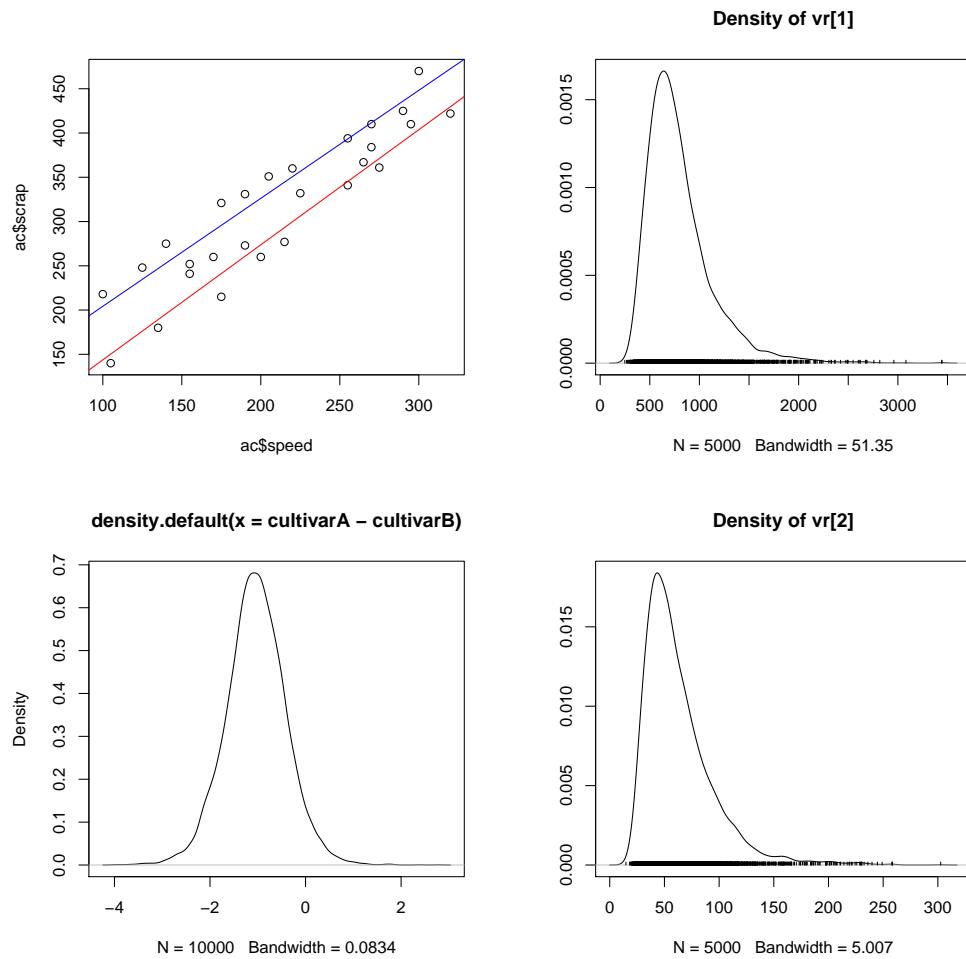
## Inference for Bugs model at "code/twogroups.jags", fit using jags,
## 1 chains, each with 6000 iterations (first 1000 discarded)
## n.sims = 5000 iterations saved
##      mu.vect sd.vect   2.5%    25%    50%    75%  97.5%
## mu[1]     134.75  6.499 122.11 130.61 134.70 138.87 147.8
## mu[2]     121.40  2.110 117.16 120.09 121.38 122.73 125.6
## vr[1]     797.45 333.228 381.36 573.92 723.20 930.48 1659.5
## vr[2]      63.12 31.741 26.22 41.34 55.22 76.11 146.2
## deviance  271.55  3.652 266.96 268.84 270.74 273.37 280.5
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 6.7 and DIC = 278.2
## DIC is an estimate of expected predictive error (lower deviance is better).

sims <- as.mcmc(twogroups.sim)
plot(sims, auto.layout=FALSE, ask=FALSE)

# plot(sims[,2], type='l')

```





First diagnostic: Look at your trace plot

Second diagnostic: look at auto corilation

```
autocorr(sims)
```

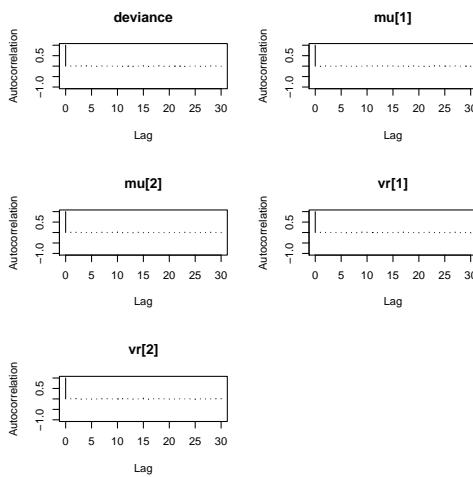
```
## , , deviance
##
##      deviance     mu[1]     mu[2]      vr[1]      vr[2]
## Lag 0  1.000000  0.0397216 -0.025591  0.5294713  0.5567647
## Lag 1  0.009287  0.0178228 -0.001470 -0.0079258  0.0045190
## Lag 5  0.008346  0.0222125 -0.004006  0.0258781  0.0005049
## Lag 10 -0.003025  0.0082836 -0.023008  0.0244412 -0.0188622
## Lag 50 -0.004687 -0.0007478 -0.009660 -0.0006706  0.0165176
##
## , , mu[1]
##
##      deviance     mu[1]     mu[2]      vr[1]      vr[2]
## Lag 0  0.039722  1.0000000 -0.019048  0.0242092  0.013122
## Lag 1  0.010297 -0.0176032 -0.018843 -0.0115282 -0.001713
## Lag 5  0.009260  0.0203083 -0.010197  0.0290402 -0.001748
## Lag 10 0.002206  0.0183513  0.048567 -0.0005999  0.015041
## Lag 50 0.001726 -0.0001966  0.002256  0.0190355 -0.014390
##
## , , mu[2]
##
##      deviance     mu[1]     mu[2]      vr[1]      vr[2]
## Lag 0 -0.025591 -0.0190482  1.000000 -0.002025 -0.017025
```

```

## Lag 1 -0.007725 -0.0083044 0.016805 -0.021768 0.005684
## Lag 5  0.030497 -0.0280916 -0.008886 0.010412 0.008702
## Lag 10 0.015048 -0.0139730 0.026550 0.019316 -0.003001
## Lag 50 0.010831 0.0004535 -0.001931 -0.006375 0.012813
##
## , , vr[1]
##
##      deviance    mu[1]    mu[2]    vr[1]    vr[2]
## Lag 0  0.529471 0.024209 -0.002025 1.000000 0.006924
## Lag 1  0.012432 0.008644 0.003187 0.013754 0.002647
## Lag 5  -0.006325 -0.007802 -0.013119 -0.001014 0.001037
## Lag 10 -0.001276 -0.022398 -0.016429 0.018451 -0.021251
## Lag 50  0.020901 -0.010891 -0.008754 0.011124 0.031264
##
## , , vr[2]
##
##      deviance    mu[1]    mu[2]    vr[1]    vr[2]
## Lag 0  0.556765 0.013122 -0.01702 0.006924 1.0000000
## Lag 1  -0.002083 0.002571 -0.00108 0.004606 0.0085805
## Lag 5  0.007769 0.018526 -0.01204 0.033278 -0.0131159
## Lag 10 0.020122 0.035722 -0.01813 0.019459 -0.0003303
## Lag 50 -0.019064 0.023849 -0.01956 0.001463 -0.0069104

autocorr.plot(sims)

```



A good autocorrelation plot shows the first bar being tall, but the rest being relatively small.

Fourth diagnostic: Effective Sample size

```
effectiveSize(sims)
```

```

## deviance    mu[1]    mu[2]    vr[1]    vr[2]
##      5000     5000     5000     5000     4840

```

these should be relatively close to the actual sample size

fifth diagnostic: Raftery Lewis Diagnostic

```
raftery.diag(sims)
```

```

##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##      Burn-in Total Lower bound Dependence
##      (M)      (N)  (Nmin)   factor (I)

```

```
## deviance 2      3741  3746      0.999
## mu[1]    2      3741  3746      0.999
## mu[2]    2      3930  3746      1.050
## vr[1]    2      3803  3746      1.020
## vr[2]    2      3561  3746      0.951
```

Check if you are within .005 (in the tails) with high probability

Dependence Factor should be less than 5

These five are the most commonly used, but there are others

```
geweke.diag(sims)

##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
## deviance   mu[1]   mu[2]   vr[1]   vr[2]
## -0.6146 -1.7786 -0.8627  0.7567  0.1023
```

Tests for drift in the samples. does something like a t test on the first 10% and the last 50% of the chain.

Heild

```
heidel.diag(sims)

##
##           Stationarity start     p-value
##           test          iteration
## deviance passed       1        0.822
## mu[1]    passed       1        0.140
## mu[2]    passed       1        0.398
## vr[1]    passed       1        0.684
## vr[2]    passed       1        0.636
##
##           Halfwidth Mean  Halfwidth
##           test
## deviance passed    271.5 0.1012
## mu[1]    passed    134.7 0.1802
## mu[2]    passed    121.4 0.0585
## vr[1]    passed    797.5 9.2366
## vr[2]    passed     63.1 0.8942
```

test of stationarity, or convergence

both the stationarity and half width mean test should pass.

Gelman needs chains started at different points

∞ January 22, 2013 ∞

5 Posterior Predictives

Likelihood $g_{i1} \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $g_{i2} \sim \mathcal{N}(\mu_2, \sigma_2^2)$

Get a new y by plugging the sampled values of the parameters into a random sampler for the likelihood

```
1 ynew_1 <- rnorm(1 ,mu1[1] ,sigma1[1])
```

```

 $\mu_1 \quad \sigma_1^2 \quad \mu_2 \quad \sigma_2^2 \quad y_{\text{new1}}$ 
 $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ 

ratios <- read.table("../Homework/01/monkeyratio.dat")[,1]
library(R2jags)
N <- length(ratios)

mdl <- "
model {
  for (i in 1:N) {
    ratios[i] ~ dbeta(alpha, beta)
  }

  alpha ~ dgamma(5, .025)
  beta ~ dgamma(5, .025)
}
"
writeLines(mdl, "code/hw1.jags")

jags.data <- c("ratios", "N")
jags.params <- c("alpha", "beta")
jags.inits <- function() {
  list('alpha' <- rgamma(5,.025), 'beta' <- rgamma(5,.025))
}
set.seed(5487)

ratios.sim <- jags(data=jags.data,
                     parameters.to.save=jags.params,
                     # inits=jags.inits,
                     model.file="code/hw1.jags",
                     n.iter=6000, n.burnin=1000, n.chains=1, n.thin=1)

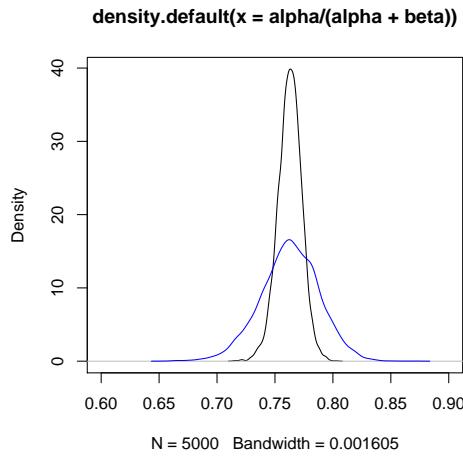
## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 10
##
## Initializing model

sims <- as.mcmc(ratios.sim)

# autocorr.plot(sims)
alpha <- sims[,1]
beta <- sims[,2]
ynew <- rbeta(10000,alpha,beta)

# Plot of the density of the mean and the posterior predictive
plot(density(alpha/(alpha+beta)), xlim=c(.6,.9))
lines(density(ynew), col='blue') # Plot of the posterior predictive
```



We don't make up data we get it from "Pedagogically flexible data acquisition."—Dr Tolly.
You can use other priors, they just may not be as easy for example:

6 Linear Regression

Likelihood

$$y_i \sim \mathcal{N}(\mu_i, \sigma^2)$$

The mean is different for every data point, but the variance is constant.

$$\begin{aligned}\mu_1 &\leftarrow \beta_0 + \beta_1 x \\ \beta_0 &\sim \mathcal{N}(, \tau = .001) \\ \beta_1 &\sim \mathcal{N}(\mu = 0, \tau = .001) \\ \sigma^2 &\sim\end{aligned}$$

January 24, 2013

```
jags.modelfile <- "code/linreg.jags"
mdl <- "
model {
for (i in 1:N) {y[i] ~ dnorm(mu[i], prec)
mu[i] <- beta_0 + beta_1 * x[i]
}
invisible(BeGiN_TiDy_IdEr)
}

linregdata <- read.table("data/02linreg.dat", col.names = c("lotsize", "manhours"))
N <- nrow(linregdata)
y <- linregdata[, 2]
x <- linregdata[, 1]
jags.data <- c("y", "x", "N")
jags.params <- c("beta_0", "beta_1", "sigma2")

set.seed(5487)
linreg.sim <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/linreg.jags",
n.iter = 10000, n.burnin = 5000, n.chains = 1, n.thin = 1)
```

```
## module glm loaded
## Error:
## Error parsing model file:
## syntax error on line 7 near """"

linreg.sim

## Error: object 'linreg.sim' not found
```

Let's check for convergence

```
sims <- as.mcmc(linreg.sim)

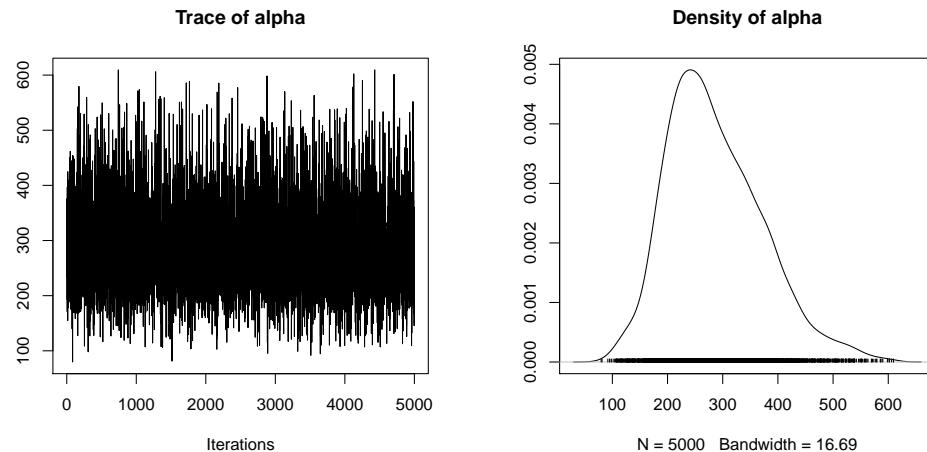
## Error: object 'linreg.sim' not found

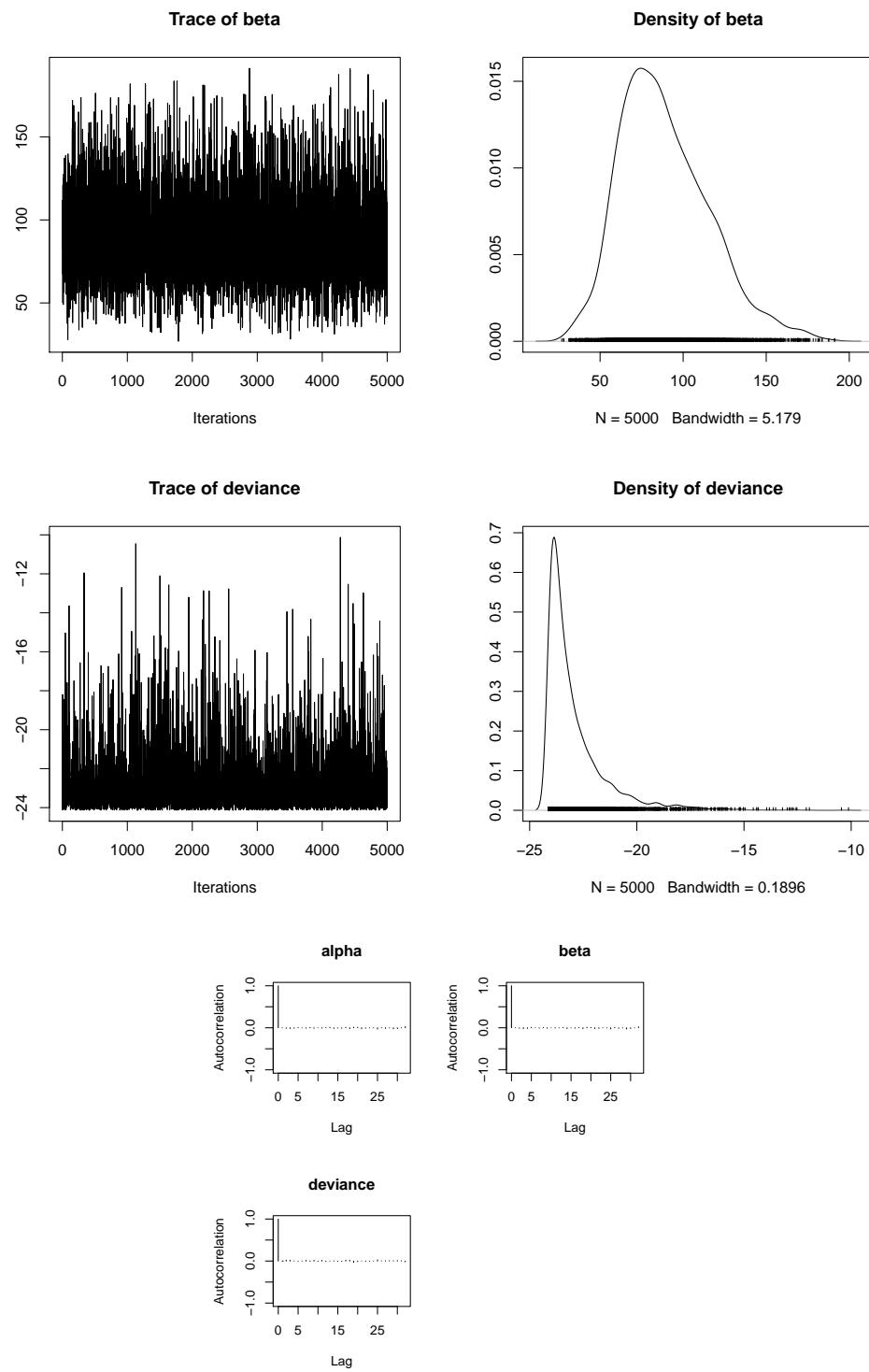
# Trace Plots
plot(sims, auto.layout = FALSE, ask = FALSE)
# Autocorrelation plots
autocorr.plot(sims, ask = FALSE)
effectiveSize(sims)

##      alpha      beta deviance
##      5000      5000      5000

raftery.diag(sims)

##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##      Burn-in Total Lower bound Dependence
##            (M)     (N)   (Nmin)       factor (I)
##  alpha    2      3741  3746      0.999
##  beta     2      3680  3746      0.982
##  deviance 2      3680  3746      0.982
```





Here's the frequentest method for an ANOVA

```
fit.1 <- lm(y ~ x)
anova(fit.1)

## Analysis of Variance Table
##
## Response: y
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## x          1 13600   13600    1813 1e-10 ***
## Residuals  8      60       7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

to check the value of the slope

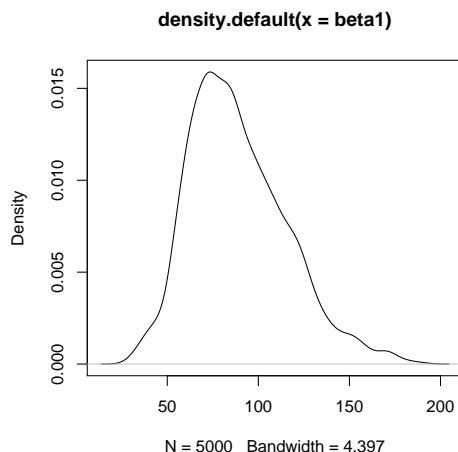
```
beta0 <- sims[, 1]
beta1 <- sims[, 2]
sigma2 <- sims[, 4]

## Error: subscript out of bounds

# Probability slope is greater than 0
mean(beta1 > 0)

## [1] 1

# Plot the density of the slope of the line
plot(density(beta1))
```



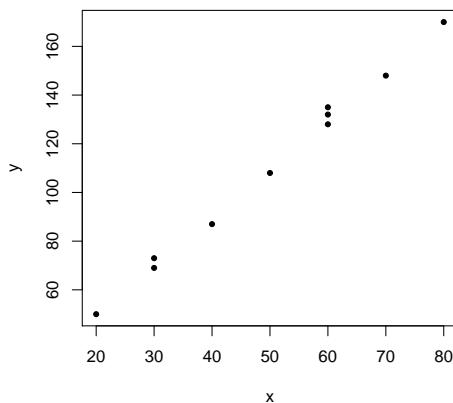
Plot of regression

```
plot(x, y, pch = 20)
abline(mean(beta0), mean(beta1))
xx <- seq(min(x), max(x), by = 0.01)

ans <- beta0 + outer(beta1, xx, "*")
lines(xx, apply(ans, 2, quantile, 0.025), col = "blue")
lines(xx, apply(ans, 2, quantile, 0.975), col = "blue")

# generate a new data set with the first set of
points(xx, beta0[1] + beta1[1] * xx + rnorm(length(xx), 0, sqrt(sigma2[1])), pch = 20)

## Error: object 'sigma2' not found
```



Homework Plot the prediction interval from the problem in class

generate a normal using ans each ans and sigma

————— January 29, 2013 —————

7 Multiple Linear Regression

```

1 %let datadir = Z:\Dropbox\Active\STAT451\Notes\data;
  data vo2;
    infile "&datadir.\03vo2.dat" firstobs=2;
    input id gender age bmi mph hr rpe maxvo2;
5 run;

ods _all_ close;
ods html;

10 proc mcmc data=vo2 outpost=vo2out seed=1234
    nmc=300000
    nbi=60000
    thin=30
    statistics=(summary interval)
15   diagnostics=(rl ess)
    dic
    propcov=quane
    monitor=(-parms_);
20 parms b1 50 b2 10 b3 0 b4 0 b5 0 b6 0 s2f 20 s2m 30;
25   prior b1 ~ normal(50, var=1000);
    prior b2 ~ normal(10, var=1000);
    prior b3 ~ normal(0, var=1000);
    prior b4 ~ normal(0, var=1000);
    prior b5 ~ normal(0, var=1000);
    prior b6 ~ normal(0, var=1000);

```

```

prior s2f ~ gamma(2, scale=20);
prior s2m ~ gamma(2, scale=30);

30
mu= b1 + b2*gender + b3*bmi + b4*mph + b5*hr + b6*rpe;
if gender=0 then vv=s2f;
else vv=s2m;

35
model maxvo2 ~ normal(mu, var=vv);
run;

```

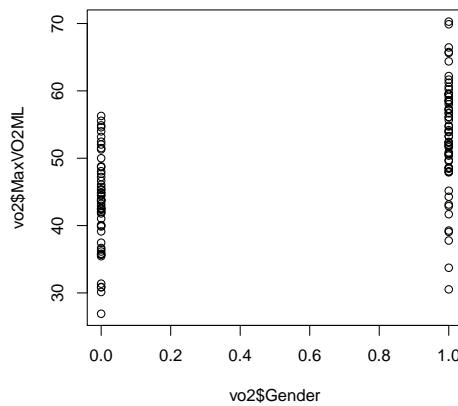
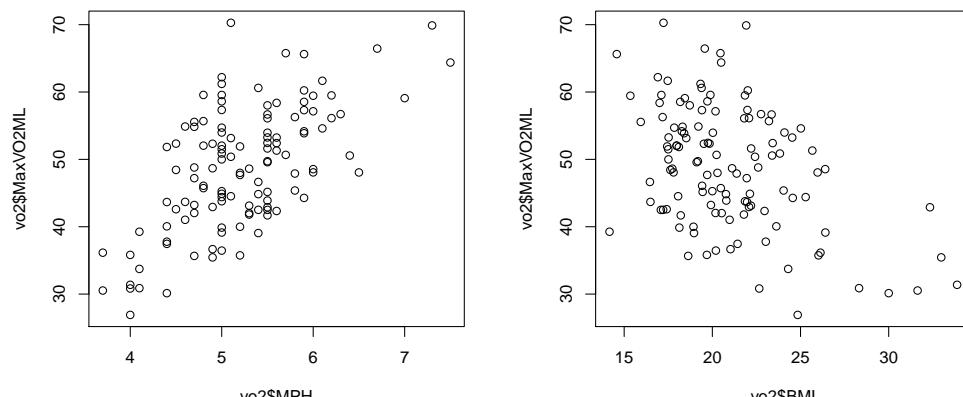
Raftery-lewis diagnostics should be less than 5.

Let the variance increase proportionaly with the *xs*

```

vo2 <- read.table("data/03vo2.dat", skip = 1, col.names = c("ID", "Gender", "Age1",
  "BMI", "MPH", "HR", "RPE", "MaxVO2ML"))
plot(vo2$MPH, vo2$MaxVO2ML)
plot(vo2$BMI, vo2$MaxVO2ML)
plot(vo2$Gender, vo2$MaxVO2ML)

```



DIC Deviance information criteria (calculated in formula 1)

$$\text{deviance} = -2 \log(\mathcal{L}) \quad (1)$$

—————  January 31, 2013  —————

```
# Using the VO2 Data
N <- nrow(vo2)
mdl <- '
model {
  for( i in 1:N){
    y[i] ~ dnorm(mu[i], prec)
    mu[i] <- b_0 +
      b_gen*gender[i] +
      b_bmi*bmi[i] +
      b_mph*mph[i] +
      b_hr*hr[i] +
      b_rpe*rpe[i] +
      # b_gen_bmi*gen_bmi[i] +
      # b_gen_mph*gen_mph[i] +
      # b_gen_hr*gen_hr[i] +
      # b_gen_rpe*gen_rpe[i] +
      # b_bmi_mph*bmi_mph[i] +
      # b_bmi_hr*bmi_hr[i] +
      # b_bmi_rpe*bmi_rpe[i] +
      b_mph_hr*mph_hr[i] # +
      # b_mph_rpe*mph_rpe[i] +
      # b_hr_rpe*hr_rpe[i]
  }
  b_0 ~ dnorm(0, .00001);
  b_gen ~ dnorm(0, .001);
  b_bmi ~ dnorm(0, .001);
  b_mph ~ dnorm(0, .001);
  b_hr ~ dnorm(0, .001);
  b_rpe ~ dnorm(0, .001);

  # b_gen_bmi ~ dnorm(0, .001);
  # b_gen_mph ~ dnorm(0, .001);
  # b_gen_hr ~ dnorm(0, .001);
  # b_gen_rpe ~ dnorm(0, .001);
  # b_bmi_mph ~ dnorm(0, .001);
  # b_bmi_hr ~ dnorm(0, .001);
  # b_bmi_rpe ~ dnorm(0, .001);
  b_mph_hr ~ dnorm(0, .001);
  # b_mph_rpe ~ dnorm(0, .001);
  # b_hr_rpe ~ dnorm(0, .001);

  vr ~ dgamma(2, .05)
  prec <- 1/vr
}
'
writeLines(mdl, "code/multipleRegression.jags")

y <- vo2[,8]
gender <- vo2[,2]
bmi <- vo2[,4]
hr <- vo2[,6]
mph <- vo2[,5]
rpe <- vo2[,7]

# gen_bmi <- gender*bmi
# gen_hr <- gender*hr
# gen_mph <- gender*mph
# gen_rpe <- gender*rpe
# bmi_mph <- bmi*mph
# bmi_hr <- bmi*hr
# bmi_rpe <- bmi*rpe
mph_hr <- mph*hr
# mph_rpe <- mph*rpe
# hr_rpe <- hr*rpe
```

```

jags.data <- c(
  "N",
  "y",
  "gender",
  "bmi",
  "hr",
  "mph",
  "rpe",
  # "gen_bmi",
  # "gen_hr",
  # "gen_mph",
  # "gen_rpe",
  # "bmi_mph",
  # "bmi_hr" #,
  # "bmi_rpe",
  # "mph_hr" #,
  # "mph_rpe",
  # "hr_rpe",
)
jags.params <- c(
  "b_0",
  "b_gen",
  "b_bmi",
  "b_hr",
  "b_mph",
  "b_rpe",
  # "b_gen_bmi",
  # "b_gen_hr",
  # "b_gen_mph",
  # "b_gen_rpe",
  # "b_bmi_mph",
  # "b_bmi_hr",
  # "b_bmi_rpe",
  "b_mph_hr",
  # "b_mph_rpe",
  "vr"
)

set.seed(1234)
multr.sim <- jags(data=jags.data,
                     parameters.to.save=jags.params,
                     model.file="code/multipleRegression.jags",
                     n.iter=10000, n.burnin=5000, n.chains=1, n.thin=1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 1294
##
## Initializing model

```

Full	756.4
-hr	764.4
-rpe	759.2
+ all two way interactions	766.7

$$\begin{aligned}
 IC &= -2 \log(\mathcal{L}) + \text{penalty(parameters)} \\
 \text{Deviance} &= -2 \log(\mathcal{L}) \\
 DIC &= -2 \log(\mathcal{L}) + pD \\
 pD &= \frac{\text{VAR(deviance)}}{2}
 \end{aligned}$$

8 ANOVA

Cell means model of ANOVA

Treatment 1	Treatment 2	Treatment 3	Treatment 4
y_{11}	y_{21}	y_{31}	y_{41}
y_{12}	y_{22}	y_{32}	y_{42}
y_{13}	y_{23}	y_{33}	y_{43}
y_{14}	y_{24}	y_{34}	y_{44}

so given this data we say that $y_i \sim N(\mu, \sigma^2)$

```

anova <- read.table("data/04anova.dat", col.names=c("tmt", "response", "tmt1"))
N <- nrow(anova)
mdl <- 'model {
  # Likelihood
  for (i in 1:N){
    response[i] ~ dnorm(mu[tmt[i]], prec)
  }

  # Create priors for each treatment
  for(i in 1:4){
    mu[i] ~ dnorm(15,.0001)
  }
  prec <- 1/vv
  vv ~ dgamma(1.1,.1)
}

writeLines(mdl, "code/ANOVAmodel.jags")
response <- anova$response
tmt <- anova$tmt

jags.data <- c("N", "response", "tmt")
jags.params <- c("mu", "vv")
anova.sim <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/ANOVAmodel.jags",
n.iter = 12000, n.chains = 1, n.thin = 1)

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 68
##
## Initializing model

sims <- as.mcmc(anova.sim)
plot(sims, auto.layout = FALSE, ask = FALSE)
autocorr.plot(sims)
raftery.diag(sims)

##
## Quantile (q) = 0.025

```

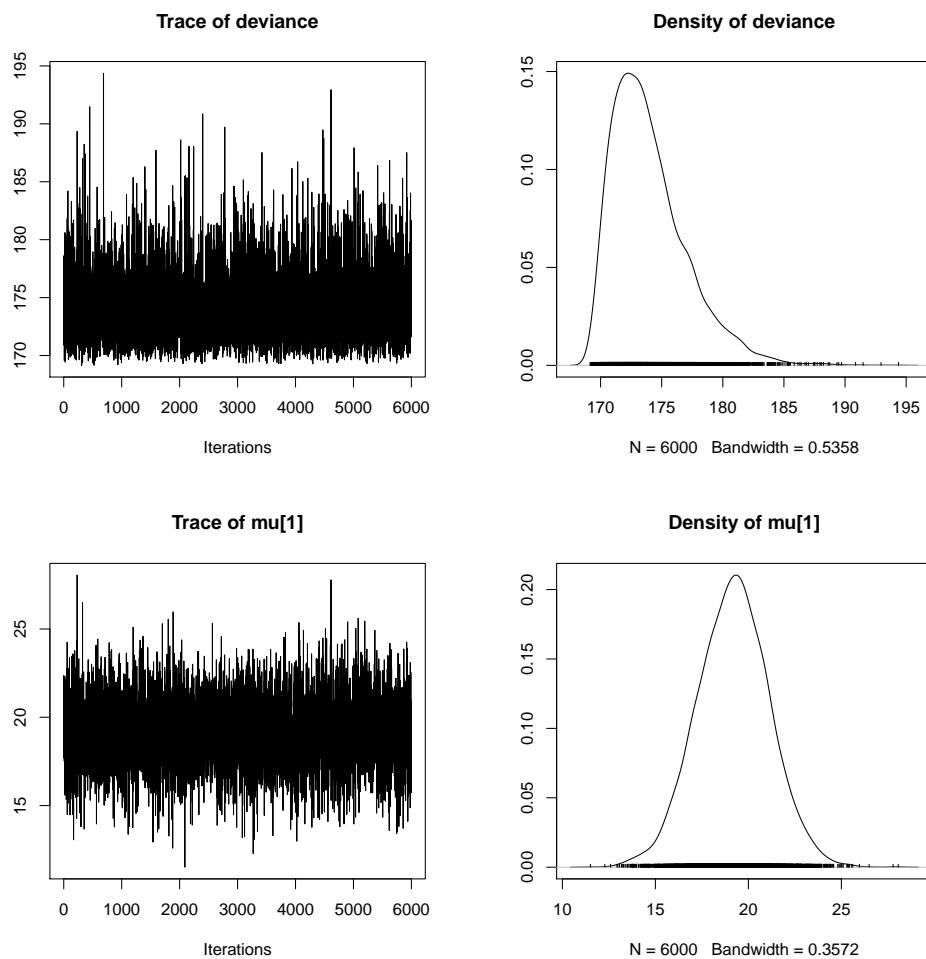
```

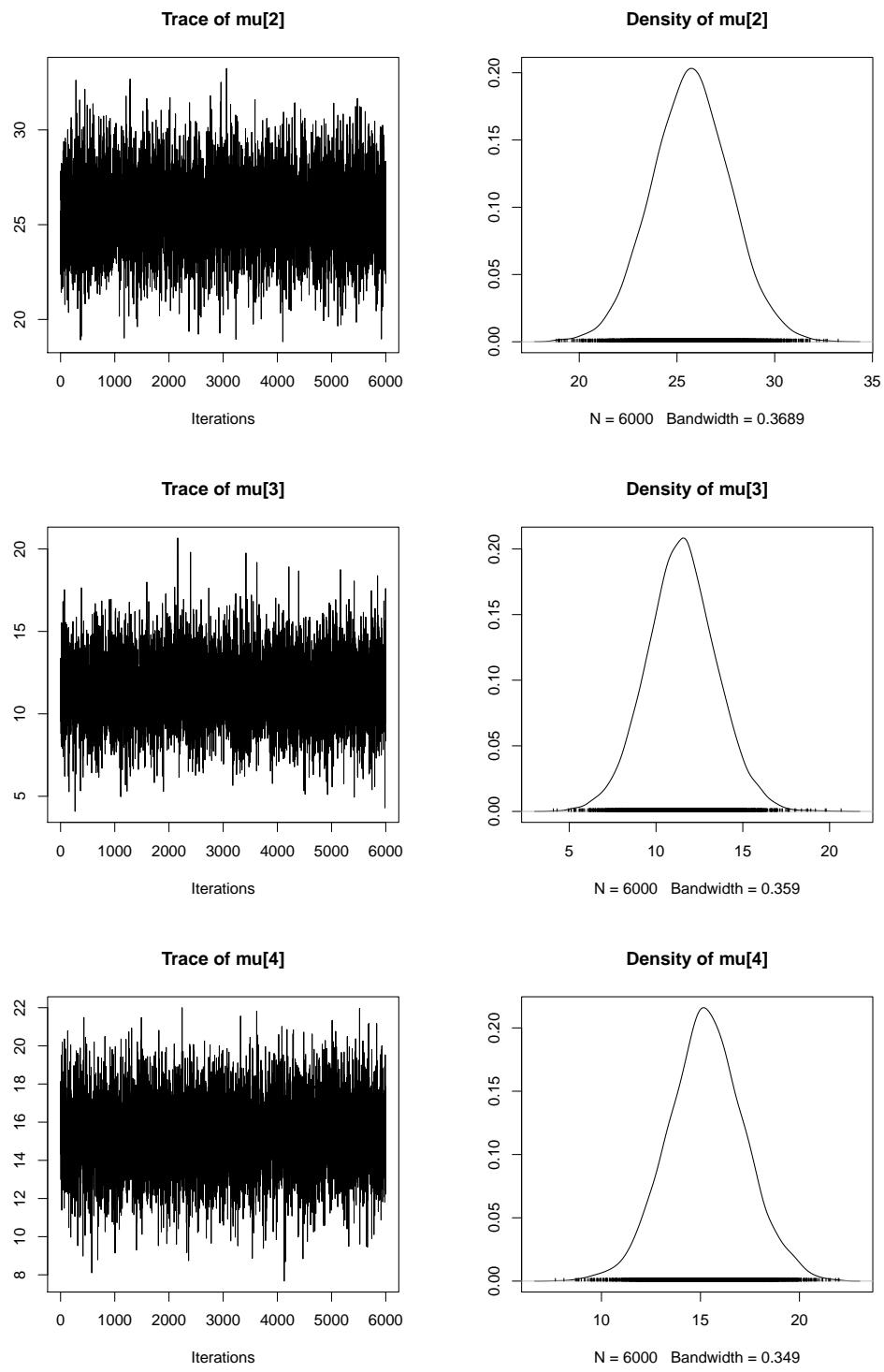
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##          Burn-in   Total Lower bound Dependence
##          (M)      (N)    (Nmin)   factor (I)
## deviance 2       3761   3746      1.00
## mu[1]    2       3710   3746      0.99
## mu[2]    2       3761   3746      1.00
## mu[3]    2       3761   3746      1.00
## mu[4]    2       3813   3746      1.02
## vv       3       4028   3746      1.08

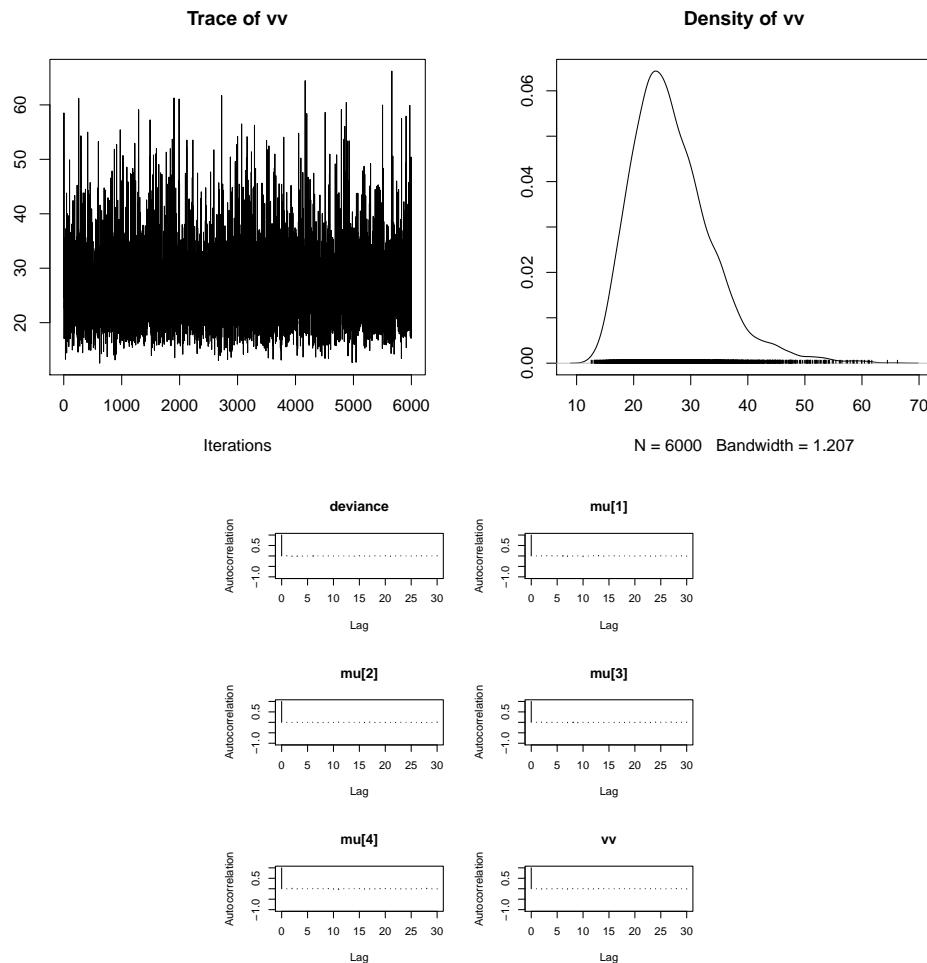
effectiveSize(sims)

## deviance   mu[1]     mu[2]     mu[3]     mu[4]       vv
## 6448       6000     6000     6000     5800     6000

```







```

mdl <- 'model {
  # Likelihood
  for (i in 1:N){
    response[i] ~ dnorm(mu[tmt[i]], prec[tmt[i]])
  }

  # Create priors for each treatment
  for(i in 1:4){
    mu[i] ~ dnorm(15,.0001)
    prec[i] <- 1/vv[i]
    vv[i] ~ dgamma(1.1,.1)
  }
}

writeLines(mdl, "code/ANOVAmodelNCvariance.jags")
si

## Error: object 'si' not found'

```

Devience is the $-2\log$ likelihood

8.1 Compare Means

We don't care about multiple test because we are not under the constraint of a null hypothesis

```
diff12 <- sims[, 2] - sims[, 3]
mean(dif12 > 0)

## Error: object 'dif12' not found

dif13 <- sims[, 2] - sims[, 4]
dif14 <- sims[, 2] - sims[, 5]
dif23 <- sims[, 3] - sims[, 4]
dif24 <- sims[, 3] - sims[, 5]
dif34 <- sims[, 4] - sims[, 5]
mean(dif13 > 0)

## [1] 0.9958

mean(dif14 > 0)

## [1] 0.9277

mean(dif23 > 0)

## [1] 1

mean(dif24 > 0)

## [1] 0.9995

mean(dif34 > 0)

## [1] 0.08267
```

Two by three factorial design.

```
twoway <- read.table("data/05twoway.dat", col.names = c("block", "seedType", "inoculate",
"yield"))
```

Using cell means method

		a	b	
	dea	μ_{11}	μ_{12}	$\mu_{1\cdot}$
Table should be rotated	con	μ_{21}	μ_{22}	$\mu_{2\cdot}$
	liv	μ_{31}	μ_{22}	$\mu_{3\cdot}$
		$\mu_{\cdot 1}$	$\mu_{\cdot 2}$	

Degrees of freedom is 5, one for type, two for inoculate, and two for interaction

```
y <- as.numeric(twoway$seedType)

mdl <- '
model {
  for (i in 1:24) {
    yield[i] ~ dnorm(mu[ncult[i],ninoc[i]], 1/vv)
  }

  for (i in 1:2) {
    for (j in 1:3) {
      mu[i, j] ~ dnorm(0, .000001)
    }
  }

  vv ~ dgamma(1.5, .1)
}'
```

```

writeLines(mdl, 'code/CultModel.jags')

yield <- twoway$yield
ncult <- as.numeric(twoway$seedType)
ninoc <- as.numeric(twoway$inoculate)

jags.data <- c('yield', 'ncult', 'ninoc')
jags.params <- c('mu', 'vv')

innits <- function(){list('mu'= matrix(0,2,3), 'vv', 15)}

cult1.jags <- jags( data=jags.data,
                     # inits=innits,
                     parameters.to.save=jags.params,
                     model.file='code/CultModel.jags',
                     n.ITER=12000, n.burnin=2000,
                     n.chains=1, n.thin=1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 85
##
## Initializing model

cult1.sim <- as.mcmc(cult1.jags)

```

Effects model and cell means model are two different ways to do analysis of variance. BYU was a pioneer of the cell means model.

8.2 ANOVA Marginal Means

marginals are calculated as $\mu_{1\cdot} = \frac{\mu_{11} + \mu_{12} + \mu_{13}}{3}$ its just a mean of the means

```

mu1dot <- (cult1.sim[, 2] + cult1.sim[, 4] + cult1.sim[, 6])/3
mu2dot <- (cult1.sim[, 3] + cult1.sim[, 5] + cult1.sim[, 7])/3

plot(mu1dot - mu2dot)
mean(mu2dot - mu1dot > 0)

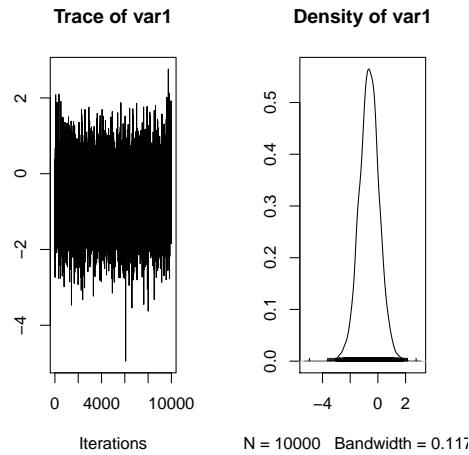
## [1] 0.8171

mudot1 <- (cult1.sim[, 2] + cult1.sim[, 3])/2
mudot2 <- (cult1.sim[, 4] + cult1.sim[, 5])/2
mudot3 <- (cult1.sim[, 6] + cult1.sim[, 7])/2

mean(mudot1 - mudot2 < 0) # Probability mu_dot2 is bigger than mu_dot1
## [1] 0.9858

mean(mudot3 - mudot2 > 0) # Probability mu_dot3 is bigger than mu_dot2
## [1] 0.9991

```



8.3 ANOVA Interactions

interactions are tested in four cell groups the formula is

$$\begin{aligned}\mu_{11} - \mu_{21} &= \mu_{12} - \mu_{22} \\ \mu_{11} - \mu_{21} - \mu_{12} + \mu_{22} &= 0\end{aligned}$$

```
int1 <- cult1.sim[, 2] - cult1.sim[, 4] - cult1.sim[, 3] - cult1.sim[, 4]
int2 <- cult1.sim[, 4] - cult1.sim[, 6] - cult1.sim[, 5] + cult1.sim[, 7]
```

Finally, compare to seed types that with the best inoculate, does it make sense to pay for the better seed or does the $\mu_{11} - \mu_{21} = \mu_{12} - \mu_{22}$ come from the inoculate

```
se1 <- cult1.sim[, 6] - cult1.sim[, 7]
mean(se1 < 0)
## [1] 0.6194
```

8.4 Linear Algebra Solution

$$\hat{\mu} = (W'W)^{-1}W'y$$

Least Squares Solution

```
w <- rbind(diag(1, 6), diag(1, 6), diag(1, 6), diag(1, 6))
muhat <- solve(t(w) %*% w) %*% t(w) %*% twoway$yield
```

$$\begin{aligned}y &= W\mu \\ y &= WI\mu \\ y &= \underbrace{WA^{-1}}_X \underbrace{A\mu}_\beta\end{aligned}$$

$$\begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & -1 & 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_{11} \\ \mu_{12} \\ \mu_{13} \\ \mu_{21} \\ \mu_{22} \\ \mu_{23} \end{bmatrix} = \begin{bmatrix} \mu_{..} \\ \mu_{1.} - \mu_{2.} \\ \mu_{.1} - \mu_{.2} \\ \mu_{.2} - \mu_{.3} \\ \mu_{11} - \mu_{12} - \mu_{21} + \mu_{22} \\ \mu_{12} - \mu_{13} - \mu_{22} + \mu_{23} \end{bmatrix}$$

~~~~~ February 12, 2013 ~~~~

```
dev <- function(x) {
  fp <- length(y) * log(2*pi)
  sp <-
}
```

~~~~~ February 14, 2013 ~~~~

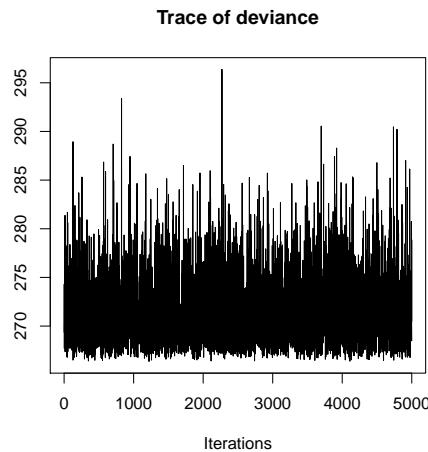
9 Analysis of Covariance ANCOVA

Join Me 777-129-200

```
ac <- read.table("data/06ancova.dat", header = TRUE)
```

First thing you do for analysis is plot the data

```
plot(ac$speed, ac$scrap, pch = ac$lines)
```



Let's try and come up with a model

$$y_{ij} = \beta_0 + \beta_1 \cdot \text{Line} + \beta_2 \cdot \text{Speed} + \beta_3 \cdot \text{Line} \cdot \text{Speed} \quad (2)$$

Or we could do this

$$y_{ij} = \beta_{0i} + \beta_{1i} \cdot \text{Speed}$$

```

mdl <- 'model {
  for(i in 1:27) {
    scrap[i] ~ dnorm(mu[i], prec);
    mu[i] <- b_0[line[i]] + b_1[line[i]]*speed[i];
  }

  for (i in 1:2) {
    b_0[i] ~ dnorm(30, .001);
    b_1[i] ~ dnorm(0, .01);
  }

  vr ~ dgamma(1.5, .0125);
  prec <- 1/vr;
}

writeLines(mdl, "code/ANCOVAModel.jags")

```

prior for beta not covers 30×100

```

speed <- ac$speed
line <- ac$line
scrap <- ac$scrap
jags.data <- c('speed', 'line', 'scrap')
jags.params <- c('b_0', 'b_1', 'vr')
ancova.jags <- jags(
  data=jags.data,
  # inits=innits,
  parameters.to.save=jags.params,
  model.file='code/ANCOVAModel.jags',
  n.iter=12000, n.burnin=2000,
  n.chains=1, n.thin=1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 148
##
## Initializing model

ancova.jags

## Inference for Bugs model at "code/ANCOVAModel.jags", fit using jags,
## 1 chains, each with 12000 iterations (first 2000 discarded)
## n.sims = 10000 iterations saved
##      mu.vect sd.vect   2.5%    25%    50%    75%   97.5%
## b_0[1]    81.738 15.570 50.441 71.518 81.938 92.185 111.585
## b_0[2]    13.553 16.417 -18.292  2.490 13.616 24.485 45.839
## b_1[1]     1.218  0.074  1.079  1.168  1.217  1.266  1.365
## b_1[2]     1.297  0.074  1.149  1.248  1.297  1.347  1.440
## vr       363.557 85.792 229.893 302.708 351.820 411.422 569.708
## deviance 241.252  3.179 236.996 238.870 240.626 242.949 249.002
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.1 and DIC = 246.3
## DIC is an estimate of expected predictive error (lower deviance is better).

```

Diagnostics

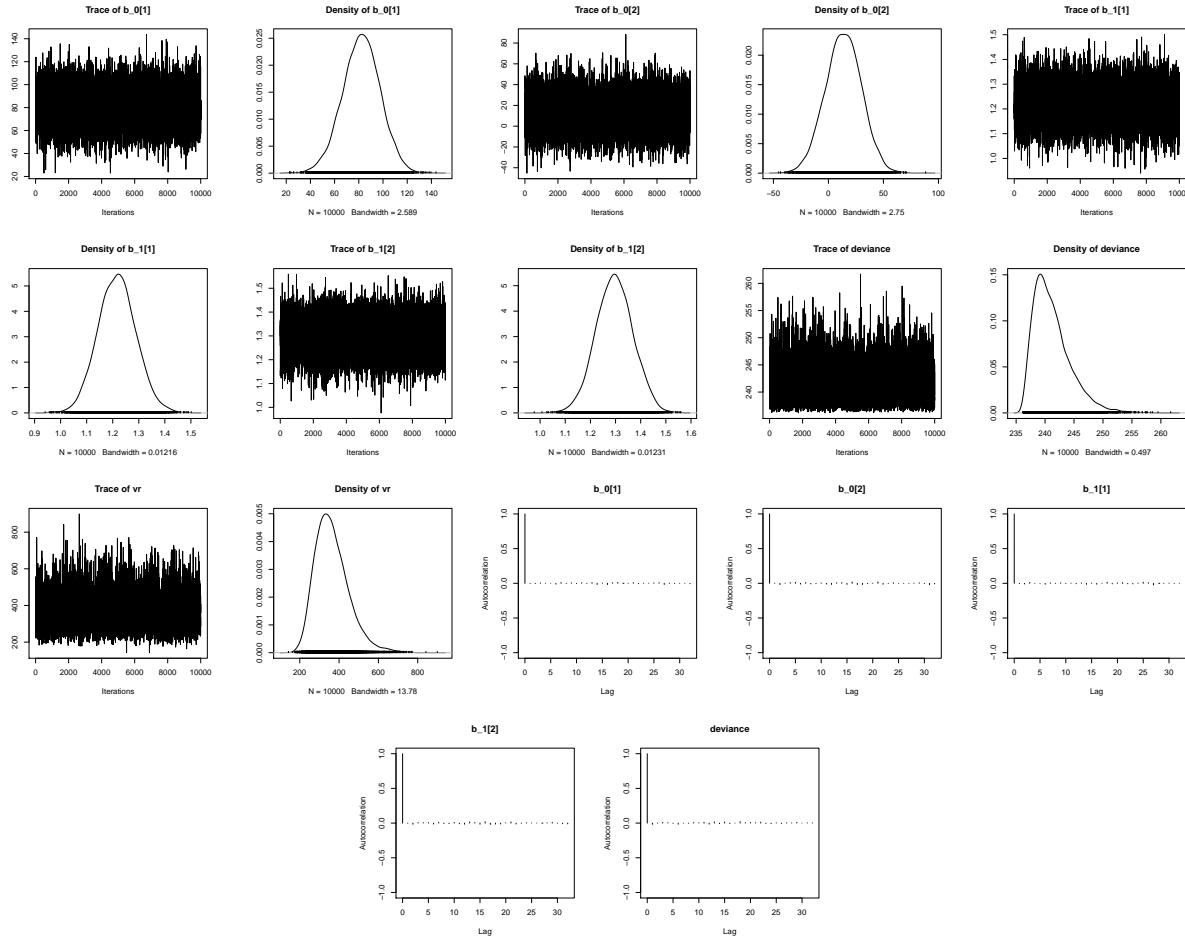
```

ancova.sim <- as.mcmc(ancova.jags)
plot(ancova.sim, auto.layout = FALSE, ask = FALSE)
autocorr.plot(ancova.sim, auto.layout = FALSE, ask = FALSE)
rafthery.diag(ancova.sim)

##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##

```

```
##          Burn-in Total Lower bound Dependence
##          (M)      (N)  (Nmin)   factor (I)
## b_0[1]    2      3650  3746     0.974
## b_0[2]    2      3680  3746     0.982
## b_1[1]    2      3680  3746     0.982
## b_1[2]    2      3834  3746     1.020
## deviance  2      3680  3746     0.982
## vr        2      3620  3746     0.966
```



Test the probability that the β_1 are different

```
b11 <- ancova.sim[, 3]
b12 <- ancova.sim[, 4]
slopdif <- b11 - b12
mean(slopdif < 0)

## [1] 0.7814

mean(slopdif > 0)

## [1] 0.2186
```

Try a new model without the different β_1 then compare the DIC

```
mdl <- 'model {
  for(i in 1:27) {
    scrap[i] ~ dnorm(mu[i], prec);
    mu[i] <- b_0[line[i]] + b_1*speed[i];
  }
}'
```

```

b_1 ~ dnorm(0, .01);
for (i in 1:2) {
  b_0[i] ~ dnorm(30, .001);
}

vr ~ dgamma(1.5, .0125);
prec <- 1/vr;
}
'
writeLines(mdl, 'code/ANCOVAModel_1.jags')

```

prior for beta not covers

```

speed <- ac$speed
line <- ac$line
scrap <- ac$scrap
jags.data <- c('speed', 'line', 'scrap')
jags.params <- c('b_0', 'b_1', 'vr')
ancova1.jags <- jags( data=jags.data,
  # inits=innits,
  parameters.to.save=jags.params,
  model.file='code/ANCOVAModel_1.jags',
  n.iter=12000, n.burnin=2000,
  n.chains=1, n.thin=1)

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 142
##
## Initializing model

ancova1.jags

## Inference for Bugs model at "code/ANCOVAModel_1.jags", fit using jags,
## 1 chains, each with 12000 iterations (first 2000 discarded)
## n.sims = 10000 iterations saved
##      mu.vect sd.vect   2.5%    25%    50%    75% 97.5%
## b_0[1]    74.154 11.580 50.987 66.643 74.236 81.949 96.90
## b_0[2]    21.933 12.213 -2.044 13.621 22.005 30.055 45.95
## b_1       1.257  0.052  1.156  1.222  1.256  1.291  1.36
## vr       369.910 85.910 233.121 308.243 359.554 419.347 564.88
## deviance 242.053  2.761 238.636 240.016 241.464 243.406 248.97
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 3.8 and DIC = 245.9
## DIC is an estimate of expected predictive error (lower deviance is better).

```

Diagnostics

```

ancova1.sim <- as.mcmc(ancova1.jags)
# plot(ancova.sim, auto.layout=FALSE, ask=FALSE) autocorr.plot(ancova.sim,
# auto.layout=FALSE, ask=FALSE) raftery.diag(ancova.sim)

```

Lets test differences in line by testing difference in intercept

```

b_0_1 <- ancova1.sim[, 1]
b_0_2 <- ancova1.sim[, 2]
line_diff <- b_0_1 - b_0_2
mean(line_diff < 0)

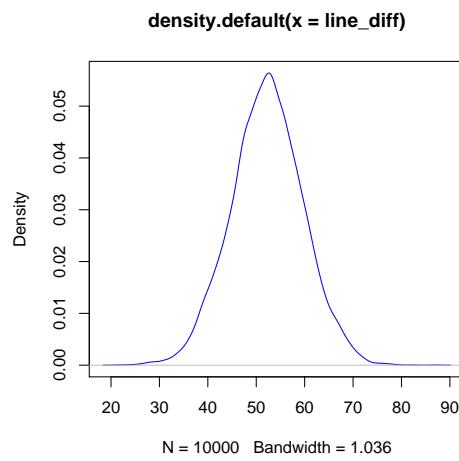
## [1] 0

mean(line_diff > 0)

## [1] 1

plot(density(line_diff), col = "blue")

```



```

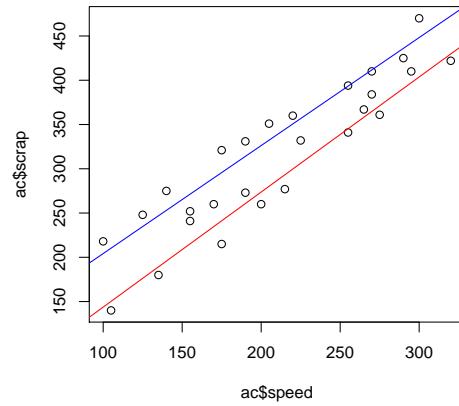
plot(ac$speed, ac$scrap, pch = ac$lines)
abline(82.2, 1.22, col = "blue")
abline(13.7, 1.3, col = "red")

b_0_1 <- ancova.sim[, 1]
b_0_2 <- ancova.sim[, 2]
b_1_1 <- ancova.sim[, 3]
b_1_2 <- ancova.sim[, 4]

y1_300 <- 300 * b_1_1 + b_0_1
y2_300 <- 300 * b_1_2 + b_0_2
diff_300 <- y1_300 - y2_300
mean(diff_300 > 0)

## [1] 1

```



∞ February 21, 2013 ∞

| | | B | | | | |
|---|---|------------|------------|------------|------------|------------------|
| | | 1 | 2 | 3 | 4 | |
| A | 1 | μ_{11} | μ_{12} | μ_{13} | μ_{14} | $\mu_{1\bullet}$ |
| | 2 | μ_{21} | μ_{22} | μ_{23} | μ_{24} | $\mu_{2\bullet}$ |
| | 3 | μ_{31} | μ_{32} | μ_{33} | μ_{34} | $\mu_{3\bullet}$ |
| | 4 | μ_{41} | μ_{42} | μ_{43} | μ_{44} | $\mu_{4\bullet}$ |

10 Multiple Sources of Variation

$$\begin{aligned}\underline{y} &= X\underline{\beta} + \varepsilon \\ \varepsilon &\sim \mathcal{N}(\underline{0}, \sigma^{2T}) \\ \underline{y} &= X\underline{\beta} + Z\underline{u} + e \\ e &\sim \mathcal{N}(\underline{0}, R) \\ \underline{u} &\sim \mathcal{N}(\underline{0}, G)\end{aligned}$$

$$\begin{aligned}\text{E}(Y) &= \text{E}(X\beta + Zu + e) = \text{E}(X\beta) + \text{E}(Zu) + \text{E}(e) = X\beta \\ \text{Var}(Y) &= \text{Var}(X\beta + Zu + e) = \text{Var}(Zu + e)\end{aligned}$$

If we assume that u and e are independent then we can move forward easily

$V(A\underline{z}) = AV(z)A'$ $AV(z)A'$ is a $p \times p$ matrix

$$\text{Var}(Zu + e) = V(Z\underline{u}) + \text{Var}(\underline{e}) = Z\text{Var}(\underline{u})Z' + V(\underline{e} = ZGZ' + R$$

————— \bowtie February 26, 2013 \bowtie —————

| | Block 1 | Block 2 |
|-------------|-----------------|-----------------|
| Treatments: | I (y_{11}) | I (y_{12}) |
| | II (y_{21}) | II (y_{22}) |

Model in matrix form (Equation 3)

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \begin{bmatrix} e_{11} \\ e_{12} \\ e_{21} \\ e_{22} \end{bmatrix} \quad (3)$$

$$\begin{aligned}e_{ij} &\sim \text{iid } N(0, \sigma_e^2) \\ u_j &\sim \text{iid } N(0, \sigma_b^2)\end{aligned}$$

$$\begin{aligned}R &= \begin{bmatrix} \sigma_e^2 & 0 & 0 & 0 \\ 0 & \sigma_e^2 & 0 & 0 \\ 0 & 0 & \sigma_e^2 & 0 \\ 0 & 0 & 0 & \sigma_e^2 \end{bmatrix} \\ G &= \begin{pmatrix} \sigma_b^2 & 0 \\ 0 & \sigma_b^2 \end{pmatrix}\end{aligned}$$

$$\begin{aligned} & \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \sigma_b^2 & 0 \\ 0 & \sigma_b^2 \end{pmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \sigma_e^2 & 0 & 0 & 0 \\ 0 & \sigma_e^2 & 0 & 0 \\ 0 & 0 & \sigma_e^2 & 0 \\ 0 & 0 & 0 & \sigma_e^2 \end{bmatrix} \\ ZGZ' + R = & \begin{bmatrix} \sigma_e^2 + \sigma_b^2 & 0 & \sigma_b^2 & 0 \\ 0 & \sigma_e^2 + \sigma_b^2 & 0 & \sigma_b^2 \\ \sigma_b^2 & 0 & \sigma_e^2 + \sigma_b^2 & 0 \\ 0 & \sigma_b^2 & 0 & \sigma_e^2 + \sigma_b^2 \end{bmatrix} \end{aligned}$$

$$\rho_{y_{11}y_{21}} = \frac{\sigma_b^2}{\sqrt{\sigma_3^2 + \sigma_b^2} \sqrt{\sigma_3^2 + \sigma_b^2}} = \frac{\sigma_b^2}{\sigma_3^2 + \sigma_b^2}$$

```
twoway <- read.table("data/05twoway.dat", col.names = c("block", "seedType", "inoculate",
"yield"))
```

First model with multiple sources of variance, s2blk is a hyper-prior, or a prior on a prior

```
mdl <- 'model {
  for (i in 1:21) {
    yield[i] ~ dnorm(mu[i], 1/s2e)
    mu[i] <- aaron[cultn[i], inocn[i]] + u[block[i]]
  }

  for (i in 1:2) {
    for (j in 1:3) {
      aaron[i,j] ~ dnorm(30, .001);
    }
  }

  for (i in 1:4) {
    u[i] ~ dnorm(0, 1/s2blk)
  }

  s2e ~ dgamma(1.5, .1);
  s2blk ~ dgamma(1.5, .1);
}

writeLines(mdl, 'code/MixedModels.jags')

yield <- twoway[, 4]
cultn <- as.numeric(twoway$seedType)
inocn <- as.numeric(twoway$inoculate)
block <- twoway[, 1]

jags.data <- c("yield", "cultn", "inocn", "block")
jags.params <- c("aaron", "s2e", "s2blk")
set.seed(343)
mixedm.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/MixedModels.jags",
n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

## Compiling model graph
##   Resolving undeclared variables
##   Allocating nodes
##   Graph Size: 137
##
## Initializing model'
```

Difference between 1 and 2

```

mixedm.sim <- as.mcmc(mixedm.jags)
cultivarA <- (mixedm.sim[, 1] + mixedm.sim[, 3] + mixedm.sim[, 5])/3
cultivarB <- (mixedm.sim[, 2] + mixedm.sim[, 4] + mixedm.sim[, 6])/3

mean(cultivarA)

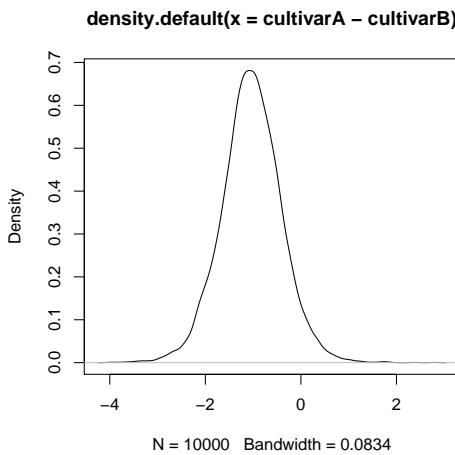
## [1] 30.09

mean(cultivarB)

## [1] 31.13

plot(density(cultivarA - cultivarB))

```



∞ February 28, 2013 ∞

Simpler Mixed Model

```

bond <- read.table("data/07mixedmods.dat", header = TRUE)
metn <- rep(1:3, 21/3)
bond <- cbind(bond, metn)

mdl <- " model{\nfor (i in 1:21){\npressure[i] ~ dnorm(mu[i], 1/s2error);\nmu[i] <- gamma[metn[i]] + u[ingot[i]]\n}\ninvisible(\nwriteLines(mdl, \"code/SimpleMixedModel.jags\")

metn <- bond[, 4]
ingot <- bond[, 1]
pressure <- bond[, 3]

jags.data <- c("metn", "ingot", "pressure")
jags.params <- c("gamma", "s2error", "s2ing")

bond.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/SimpleMixedModel.jags",
n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

## Error:
## Error parsing model file:
## syntax error on line 6 near ""

bond.sim <- as.mcmc(bond.jags)

## Error: object 'bond.jags' not found

```

ICC or interclass corelation is a measure of reliability of measurements

$$ICC = \frac{\sigma_i^2}{\sigma_e^2 + \sigma_i^2} \quad (4)$$

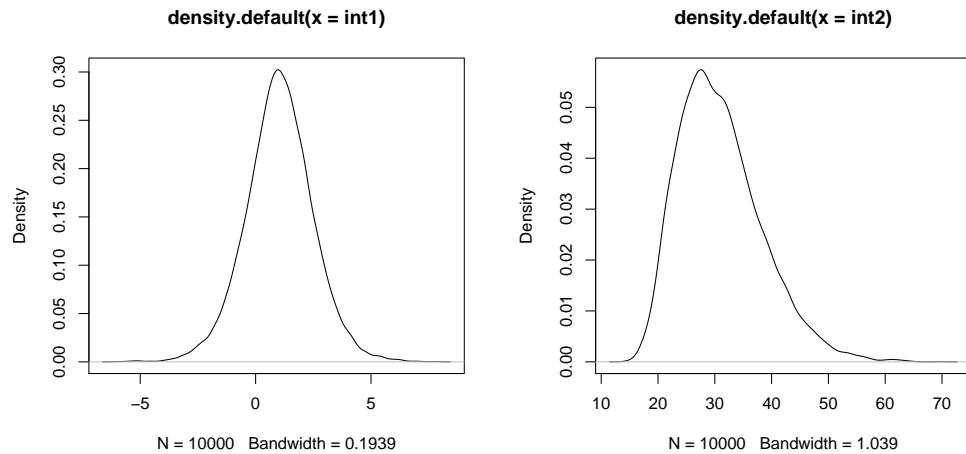
```
icc <- bond.sim[, 5]/(bond.sim[, 6] + bond.sim[, 5])
## Error: object 'bond.sim' not found
plot(density(icc))
## Error: object 'icc' not found

plot something
indif <- bond.sim[, 3] - bond.sim[, 2]
## Error: object 'bond.sim' not found
icdif <- bond.sim[, 3] - bond.sim[, 4]
## Error: object 'bond.sim' not found
```

Back to Complex

| | | | | | |
|------|---|---------------|---------------|---------------|--|
| | | | inoc | | |
| | | con | dea | liv | |
| cult | a | α_{11} | α_{12} | α_{13} | |
| | b | α_{21} | α_{22} | α_{23} | |

```
int1 <- mixedm.sim[, 2] - mixedm.sim[, 4] - mixedm.sim[, 3] + mixedm.sim[, 5]
int2 <- mixedm.sim[, 4] - mixedm.sim[, 6] - mixedm.sim[, 5] + mixedm.sim[, 7]
plot(density(int1))
plot(density(int2))
```



Let's say that there's no interaction, because

 March 5, 2013

11 Random Coefficients Model

Genetically identical seeds we treat as subjects

Sixty observations, but only ten wheat types, so there are really somewhere in the middle

$$\text{yield}_{ij} = \beta_0 + \beta_1 \cdot \text{moisture}_j + u_{1i} + u_{2i} \cdot \text{moisture}_j + \text{error}_{ij}$$

so the deviation u has both a slope and an intercept.

$$y = \begin{bmatrix} 1 & 10 \\ 1 & 17 \\ 1 & \vdots \\ \vdots & \ddots \end{bmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \begin{bmatrix} 1 & 10 & 0 & 0 & 0 & 0 & \dots \\ 1 & 57 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots \\ 1 & 40 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 16 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots \\ 0 & 0 & 0 & 0 & 1 & 39 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{pmatrix} u_{01} \\ u_{11} \\ u_{02} \\ u_{12} \\ u_{03} \\ u_{13} \\ u_{04} \\ u_{14} \\ u_{05} \\ u_{15} \\ u_{06} \\ u_{16} \end{pmatrix}$$

$$G = \begin{bmatrix} \sigma_{int}^2 & 0 & 0 & 0 & \dots \\ 0 & \sigma_{slp}^2 & 0 & 0 & \dots \\ 0 & 0 & \sigma_{int}^2 & 0 & \dots \\ 0 & 0 & 0 & \sigma_{slp}^2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

```
wheat <- read.table("data/10randomcoef.dat", header=TRUE)
yield <- wheat$yield
variety <- wheat$variety
moisture <- wheat$moisture
mdl <- ' model {
  for (i in 1:60) {
    yield[i] ~ dnorm(mu[i], 1/s2err)
    mu[i] <- b0 + b1 * moisture[i] + u0[variety[i]] + u1[variety[i]] * moisture[i]
  }
  b0 ~ dnorm(30, 0.001)
  b1 ~ dnorm(0, 0.1)

  for (i in 1:10) {
    u0[i] ~ dnorm(0, 1/s2int)
    u1[i] ~ dnorm(0, 1/s2slp)
  }

  s2err ~ dgamma(1.1, .5)
  s2int ~ dgamma(1.1, .1)
  s2slp ~ dgamma(1.1, 2)
}
'
writeLines(mdl, 'code/RandomCoeff.jags')
```

```

jags.data <- c("yield", "variety", "moisture")
jags.params <- c("b0", "b1", "u0", "u1", "s2err", "s2int", "s2slp")

rc1.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/RandomCoeff.jags",
  n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

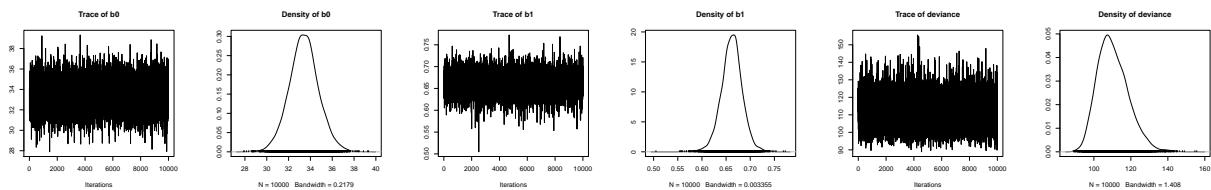
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 366
##
## Initializing model

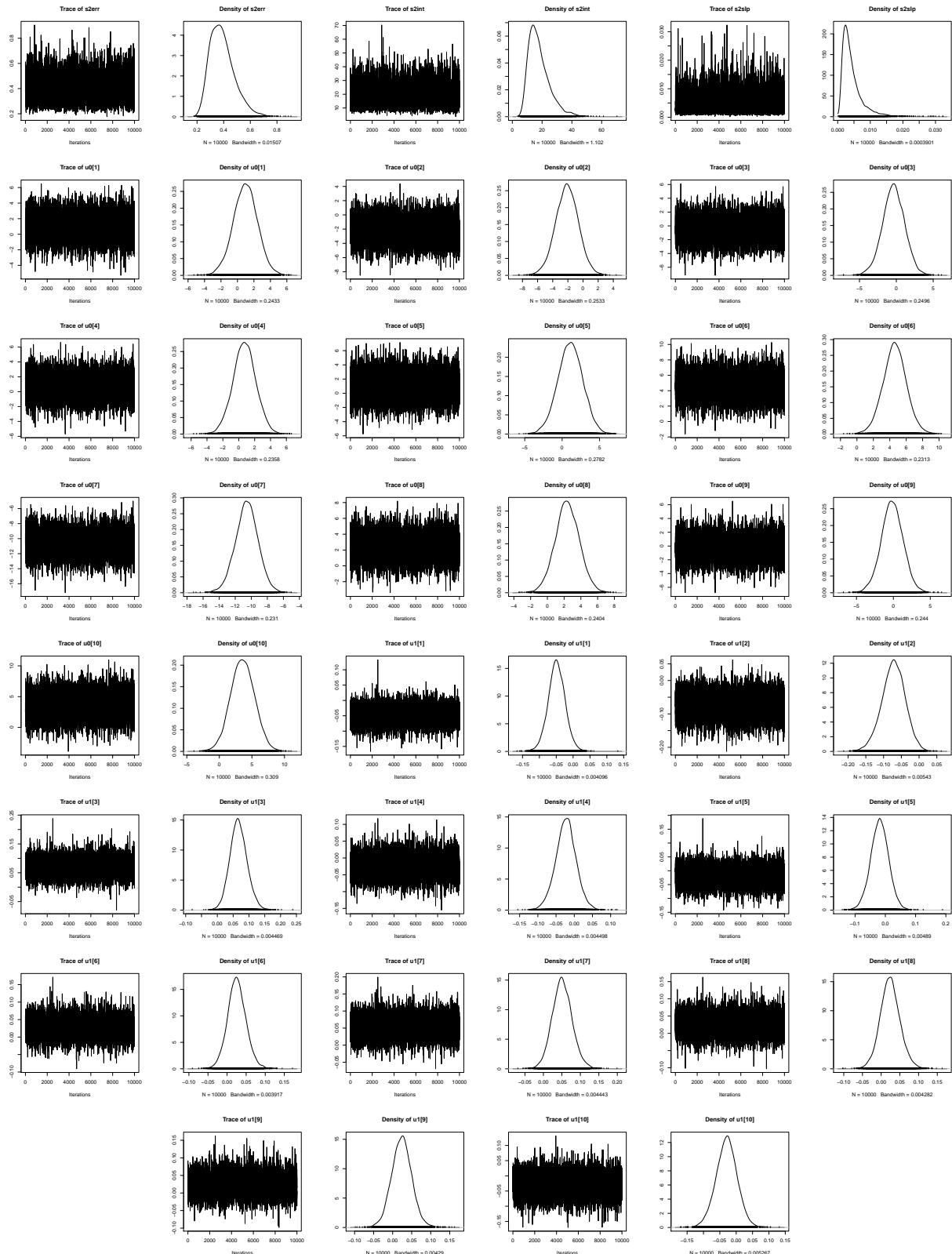
rc1.jags

## Inference for Bugs model at "code/RandomCoeff.jags", fit using jags,
## 1 chains, each with 12000 iterations (first 2000 discarded)
## n.sims = 10000 iterations saved
##      mu.vect sd.vect   2.5%    25%    50%    75%   97.5%
## b0      33.406  1.355 30.710 32.525 33.397 34.263 36.139
## b1       0.662  0.022  0.617  0.648  0.662  0.675  0.704
## s2err     0.389  0.092  0.248  0.322  0.378  0.442  0.603
## s2int    17.953  7.124  8.213 12.823 16.548 21.613 36.010
## s2slp     0.004  0.003  0.001  0.002  0.003  0.005  0.013
## u0[1]     0.958  1.472 -1.995 -0.016  0.964  1.925  3.839
## u0[2]    -2.114  1.518 -5.174 -3.128 -2.109 -1.107  0.894
## u0[3]    -0.371  1.501 -3.410 -1.352 -0.376  0.639  2.534
## u0[4]     0.716  1.447 -2.145 -0.225  0.713  1.656  3.551
## u0[5]     1.097  1.664 -2.225 -0.001  1.102  2.217  4.370
## u0[6]     4.602  1.434  1.760  3.684  4.604  5.529  7.451
## u0[7]   -10.563  1.420 -13.406 -11.479 -10.560 -9.637 -7.808
## u0[8]     2.380  1.449 -0.466  1.433  2.365  3.350  5.277
## u0[9]    -0.162  1.492 -3.148 -1.131 -0.170  0.815  2.783
## u0[10]    3.573  1.840 -0.041  2.339  3.551  4.809  7.153
## u1[1]    -0.049  0.026 -0.099 -0.065 -0.049 -0.032  0.002
## u1[2]    -0.072  0.033 -0.141 -0.093 -0.071 -0.050 -0.009
## u1[3]     0.067  0.028  0.016  0.049  0.066  0.085  0.124
## u1[4]    -0.023  0.028 -0.080 -0.041 -0.023 -0.005  0.033
## u1[5]    -0.019  0.030 -0.079 -0.038 -0.019  0.001  0.040
## u1[6]     0.026  0.025 -0.022  0.010  0.025  0.041  0.075
## u1[7]     0.051  0.027 -0.001  0.033  0.050  0.068  0.107
## u1[8]     0.024  0.027 -0.028  0.007  0.023  0.041  0.079
## u1[9]     0.024  0.027 -0.028  0.006  0.024  0.041  0.077
## u1[10]    -0.029  0.033 -0.095 -0.050 -0.029 -0.008  0.034
## deviance 110.389  8.401 96.576 104.372 109.459 115.604 128.815
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 35.3 and DIC = 145.7
## DIC is an estimate of expected predictive error (lower deviance is better).

rc1.sim <- as.mcmc(rc1.jags)
plot(rc1.sim, auto.layout = FALSE, ask = FALSE)

```





Find the variance of the y

```

g <- rbind(cbind(18.645, 0), cbind(0, 0.005))
r <- diag(0.392, 60)
z <- model.matrix(~-1 + as.factor(variety) + as.factor(variety):moisture, wheat)
Z <- z[, c(1, 11, 2, 12, 3, 13, 4, 14, 5, 15, 6, 16, 7, 17, 8, 18, 9, 19, 10, 20)] # need to rearrange so that the intercepts are at the top
G <- kronecker(diag(1, 10), g)
V <- Z %*% G %*% t(Z) + r

```

∞ March 7, 2013 ∞

To get the slope or intercept for the particular variety, add the fixed effect and the random effect for the variety.

12 Hierarchical Models

$$y = \mu_0 + \mu_s \text{moisture} + e$$

$$\mu_0 \sim \mathcal{N}(\mu_0, \sigma_0^2), \mu_s \sim \mathcal{N}(\mu, \sigma_s^2), e \sim (0, \sigma_e^2)$$

```

wheat <- read.table("data/10randomcoef.dat", header=TRUE)
yield <- wheat$yield
variety <- wheat$variety
moisture <- wheat$moisture
mdl <- 'model {
    for (i in 1:60) {
        yield[i] ~ dnorm(mu[i], 1/s2err)
        mu[i] <- u0[variety[i]] + u1[variety[i]] * moisture[i]
    }
    b0 ~ dnorm(30, 0.001)
    b1 ~ dnorm(0, 0.1)

    for (i in 1:10) {
        u0[i] ~ dnorm(beta0, 1/s2int)
        u1[i] ~ dnorm(beta1, 1/s2slp)
    }

    beta0 ~ dnorm(30, 0.001)
    beta1 ~ dnorm(0, 0.1)

    s2err ~ dgamma(1.1, .5)
    s2int ~ dgamma(1.1, .1)
    s2slp ~ dgamma(1.1, 2)
}
'

writeLines(mdl, 'code/Hierarchical.jags')

jags.data <- c("yield", "variety", "moisture")
jags.params <- c("beta0", "beta1", "u0", "u1", "s2err", "s2int", "s2slp")

hier.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/Hierarchical.jags",
n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 334
##
## Initializing model

hier.jags

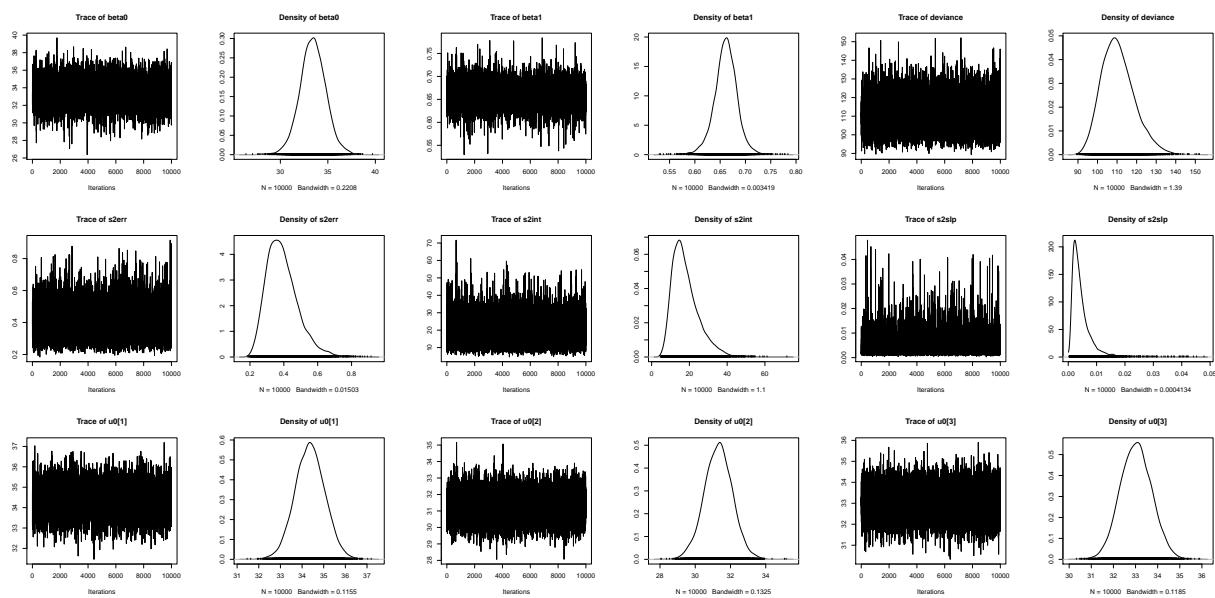
```

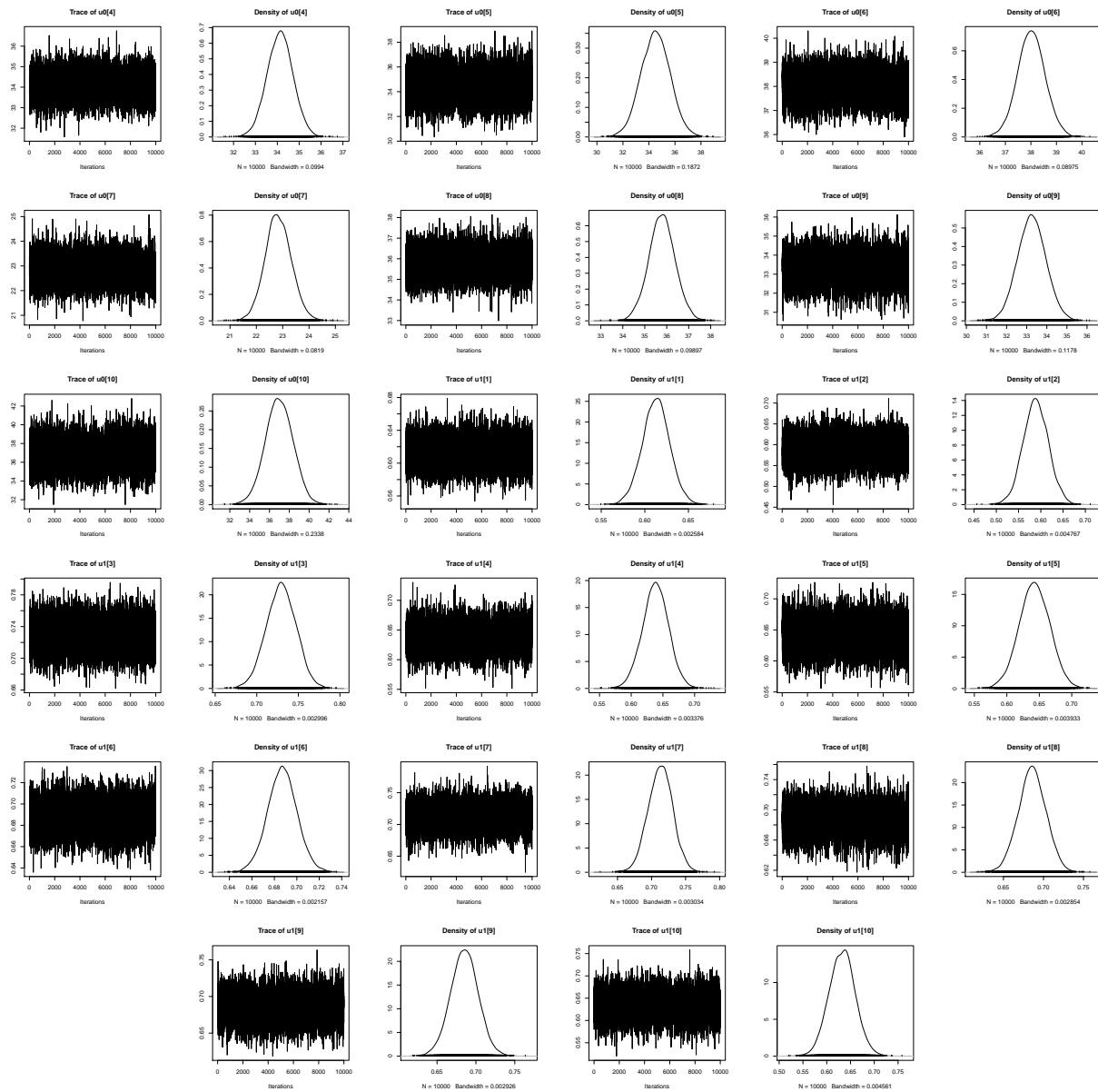
```

## Inference for Bugs model at "code/Hierarchical.jags", fit using jags,
## 1 chains, each with 12000 iterations (first 2000 discarded)
## n.sims = 10000 iterations saved
##          mu.vect sd.vect 2.5%   25%   50%   75% 97.5%
## beta0    33.422  1.366 30.708 32.548 33.423 34.308 36.162
## beta1     0.661  0.023  0.613  0.648  0.662  0.675  0.706
## s2err     0.391  0.094  0.247  0.324  0.379  0.444  0.613
## s2int    18.019  7.236  8.118 12.861 16.541 21.633 35.702
## s2slp     0.005  0.004  0.001  0.002  0.004  0.006  0.015
## u0[1]    34.367  0.689 33.034 33.905 34.362 34.826 35.724
## u0[2]    31.309  0.794 29.711 30.783 31.322 31.840 32.870
## u0[3]    33.022  0.706 31.659 32.538 33.019 33.497 34.405
## u0[4]    34.126  0.592 32.972 33.725 34.129 34.518 35.286
## u0[5]    34.508  1.114 32.321 33.749 34.511 35.256 36.712
## u0[6]    38.019  0.539 36.958 37.661 38.017 38.376 39.098
## u0[7]    22.834  0.499 21.876 22.506 22.821 23.160 23.828
## u0[8]    35.783  0.599 34.601 35.388 35.780 36.178 36.983
## u0[9]    33.254  0.704 31.901 32.778 33.251 33.717 34.645
## u0[10]   36.978  1.391 34.238 36.047 36.969 37.916 39.672
## u1[1]    0.613  0.016  0.581  0.603  0.613  0.623  0.644
## u1[2]    0.589  0.028  0.533  0.570  0.589  0.608  0.645
## u1[3]    0.729  0.018  0.694  0.717  0.729  0.741  0.764
## u1[4]    0.638  0.021  0.598  0.625  0.638  0.652  0.679
## u1[5]    0.643  0.023  0.596  0.627  0.642  0.659  0.688
## u1[6]    0.687  0.013  0.661  0.678  0.687  0.696  0.713
## u1[7]    0.713  0.018  0.678  0.701  0.713  0.725  0.748
## u1[8]    0.686  0.017  0.652  0.675  0.686  0.697  0.719
## u1[9]    0.685  0.018  0.650  0.674  0.685  0.697  0.720
## u1[10]   0.632  0.027  0.579  0.614  0.633  0.650  0.686
## deviance 110.551  8.430 96.364 104.516 109.746 115.607 129.397
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 35.5 and DIC = 146.1
## DIC is an estimate of expected predictive error (lower deviance is better).

hier.sim <- as.mcmc(hier.jags)
plot(hier.sim, auto.layout = inter_plots, ask = inter_plots)

```





Hierarchical models are virtually identical as random coefficients models, but are more pure Bayesian in thought process and methodology.

```

1 %let notedir = Z:\Dropbox\Active\STAT451\Notes\data;
filename data "&notedir./data/10randomcoef.dat";
filename outfile "&notedir./output/2013-03-07.html";
ods listing close;
5
data monkeyRatio;
  infile data;
  input obs variety yield moisture;
run;
10 ODS GRAPHICS on / imagename="2013-03-07-Plots";
  ods html body=outfile (url=none)

```

```

GPATH="&notedir.\figures\>;

15 proc mcmc data=wheat output=wheat_chains nmc=100000 nbi=20000 thin=10
   seed=1234 monitor=(_parms_);
   diag(ess autocorr r1) propcov=quanew;
   parms mu0 30 mul 0 s2 1;
   parms s2int 30 s2slp .1;
   random int ~ normal(mu0, var=s2int) subject=variety monitor=(int_1 int_2);
   random slp ~ normal(mul, var=s2slp) subject=variety;
   prior mu0 ~ normal(30, var=100);
   prior mul ~ normal(0, var=0.1);
   prior s2 ~ dunif(0,3);
   prior s2int ~ dunif(0,50);
   prior s2slp ~ dunif(0,.2);
   mu = int + slp*moisture;
   model yield ~ normal(mu, var=s2);
run;
ods html close;

```

—————  March 12, 2013  —————

13 Exam Debriefing

Covergence Diagnostics should have included:

- Traceplots
- Autocorrelation
- Effective sample size
- Raftery-lewis

Model should have been a model with six means and six variances

| | Gain | | | |
|----------|-------------------------|-------------------------|-------------------------|-------------------------------|
| | 1 | 2 | 3 | |
| Duration | μ_1 | μ_2 | μ_3 | $\frac{\mu_1+\mu_2+\mu_3}{3}$ |
| | μ_4 | μ_5 | μ_6 | $\frac{\mu_4+\mu_5+\mu_6}{3}$ |
| | $\frac{\mu_1+\mu_4}{2}$ | $\frac{\mu_2+\mu_5}{2}$ | $\frac{\mu_3+\mu_6}{2}$ | |

Next exam will probably have

- a random effects and or hierachal model
- Non-normal likelihood

14 Logistic Regression

What we have done so far:

- ANOVA
- Regression
- ANCOVA — Both Categorical and Continuous predictors
- Mixed — More than one source of variability

| | | Response | |
|------------|-------------|-------------|------------|
| | | Categorical | Continuous |
| Predictors | Categorical | ANOVA | |
| | Continuous | Regression | |

$y = \text{Binomial}(n, \pi)$, with $0 \leq \pi \leq 1$

Odds: $\frac{\pi}{1-\pi}$

Sports “3 to 1” is 3 failures for every one success or $\pi = .25$

$Odds > 0$, 1 should be the midpoint

odds have a skewed distribution

| probability | Odds | log(Odds) |
|---------------|-----------------|-------------|
| π | Success-Failure | $\log(S/F)$ |
| $\frac{1}{5}$ | 1 - 1 | 0 |
| $\frac{1}{5}$ | 1 - 4 | -1.39 |
| $\frac{4}{5}$ | 4 - 1 | 1.39 |

```
chd <- rbind(c(17, 274, 0, 0, 0), c(15, 122, 0, 1, 0), c(7, 59, 0, 0, 1), c(5, 32,
  0, 1, 1), c(1, 8, 1, 9, 9), c(9, 39, 1, 1, 0), c(3, 17, 1, 0, 1), c(14, 58, 1,
  1, 1))
colnames(chd) <- c("CHD", "nRisk", "Cat", "agegrp", "abECG")
tmt <- seq(8)
chd <- cbind(chd, tmt)
CHD <- chd[, 1]
nRisk <- chd[, 2]
tmt <- chd[, 6]
```

Cell means model

```
mdl <- ' model {
  for (i in 1:8) {
    CHD[i] ~ dbin(p[i], nRisk[i]);
    logit(p[i]) <- b[tmt[i]];
    b[i] ~ dnorm(0, .1);
  }
}
writeLines(mdl, 'code/LogitBinomial.jags')

jags.data <- c("CHD", "nRisk", "tmt")
jags.params <- c("b")

# innits <- list(list('p' <- runif(8,0,1)))

logit.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/LogitBinomial.jags",
```

```

n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1

## module glm loaded

## Compiling model graph
##   Resolving undeclared variables
##   Allocating nodes
##   Graph Size: 42
##
## Initializing model

```

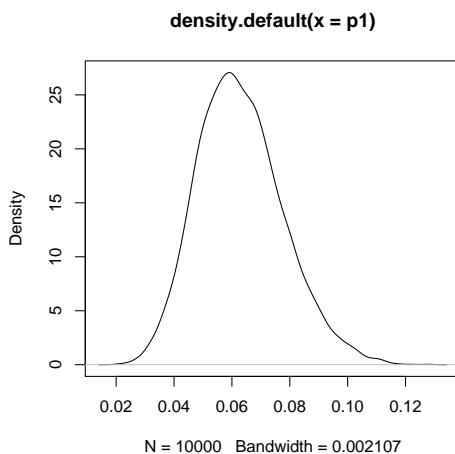
∞ March 14, 2013 ∞

$$\begin{aligned}
\text{logit}(p_i) &= b_i \\
\log\left(\frac{p_i}{1 - p_i}\right) &= b_i \\
\frac{p_i}{1 - p_i} &= e^{b_i} \\
p_i &= e^{b_i} - pe^{b_i} \\
p_i + p_i e^{b_i} &= e^{b_i} \\
p_i &= \frac{e^{b_i}}{1 + e^{b_i}} \frac{e^{-b_i}}{e^{-b_i}} \\
p_i &= \frac{1}{e^{-b_i} + 1}
\end{aligned}$$

```

logit.sim <- as.mcmc(logit.jags)
p1 <- 1/(exp(-logit.sim[, 1]) + 1)
plot(density(p1))

```



```

jags.params <- c("b", "p")
logit.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/LogitBinomial.jags",
n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## Compiling model graph

```

```
##      Resolving undeclared variables
##      Allocating nodes
##      Graph Size: 42
##
## Initializing model

logit.sim <- as.mcmc(logit.jags)
```

factorial design

| | | abEUG | |
|-------|-------|-------|---------|
| Cat-N | abECG | Y | O |
| | | N | μ_1 |
| Cat-Y | abECG | Y | μ_3 |
| | | N | μ_2 |
| | | abEUG | |
| Cat-Y | abECG | Y | O |
| | | N | μ_5 |
| Cat-Y | abECG | Y | μ_7 |
| | | N | μ_6 |
| Cat-Y | abECG | Y | μ_8 |

To find the marginals for cat

$$\text{Cat-n} = \frac{\mu_1 + \mu_2 + \mu_3 + \mu_4}{4} \quad \text{Cat-y} = \frac{\mu_1 + \mu_2 + \mu_3 + \mu_4}{4} \quad (5)$$

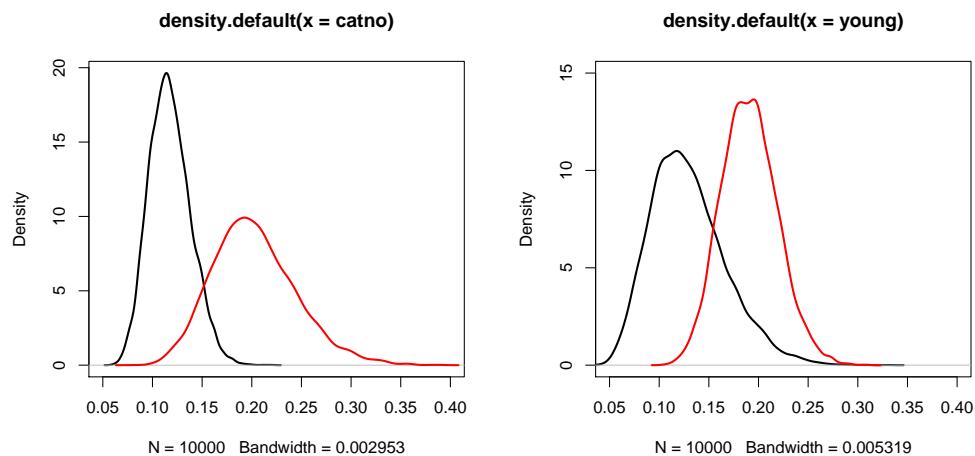
```
p1 <- logit.sim[, 10]
p2 <- logit.sim[, 11]
p3 <- logit.sim[, 12]
p4 <- logit.sim[, 13]
p5 <- logit.sim[, 14]
p6 <- logit.sim[, 15]
p7 <- logit.sim[, 16]
p8 <- logit.sim[, 17]
```

Find the marginals

```
catno <- (p1 + p2 + p3 + p4)/4
catyes <- (p5 + p6 + p7 + p8)/4
plot(density(catno), xlim = c(0.05, 0.4), lwd = 2)
lines(density(catyes), lwd = 2, col = "red")

young <- (p1 + p3 + p5 + p7)/4
old <- (p2 + p4 + p6 + p8)/4
plot(density(young), xlim = c(0.05, 0.4), ylim = c(0, 15), lwd = 2)
lines(density(old), lwd = 2, col = "red")

abNo <- (p1 + p2 + p5 + p6)/4
abYes <- (p3 + p4 + p7 + p8)/4
```



To compare old to young we use an odds ratio or a risk ratio

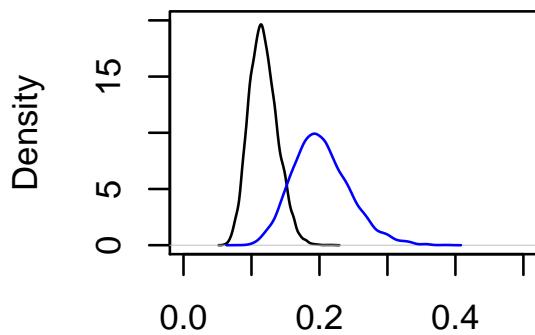
```
ORAge <- (old/(1 - old))/(young/(1 - young))
mean(ORAge > 1)
## [1] 0.9027

RRage <- old/young
mean(RRage > 1)
## [1] 0.9027

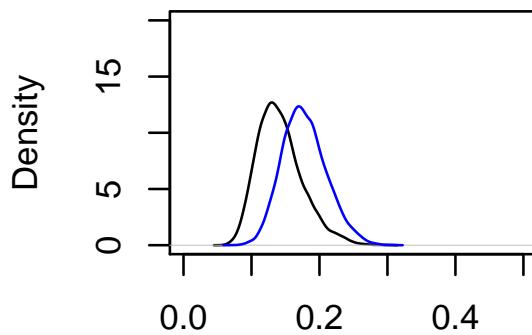
ORab <- (abYes/(1 - abYes))/(abNo/(1 - abNo))
ORCat <- (catyes/(1 - catyes))/(catno/(1 - catno))

mean(ORab)
## [1] 1.402
mean(ORCat)
## [1] 2.001

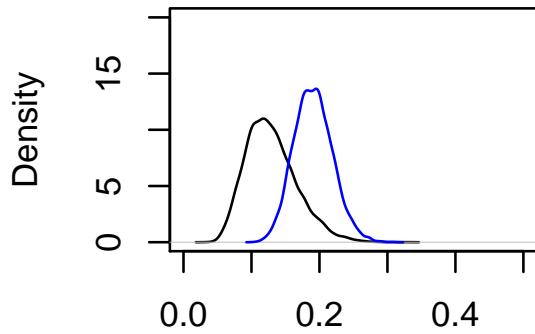
par(mfrow = c(2, 2))
xlims <- c(0, 0.5)
ylims <- c(0, 20)
plot(density(catno), xlim = xlims, ylim = ylims)
lines(density(catyces), col = "blue")
plot(density(abNo), xlim = xlims, ylim = ylims)
lines(density(abYes), col = "blue")
plot(density(young), xlim = xlims, ylim = ylims)
lines(density(old), col = "blue")
par(mfrow = c(1, 1))
```

density.default(x = catno)

N = 10000 Bandwidth = 0.002953

density.default(x = abNo)

N = 10000 Bandwidth = 0.004593

density.default(x = young)

N = 10000 Bandwidth = 0.005319

two way interaction for abnormal ECG and age averaged over catacolamines

$$\mu_1 + \mu_5 - \mu_2 - \mu_6 - \mu_3 - \mu_7 + \mu_4 + \mu_8 = 0$$

to average across abnormal ECG, rewrite the tables abECG-N

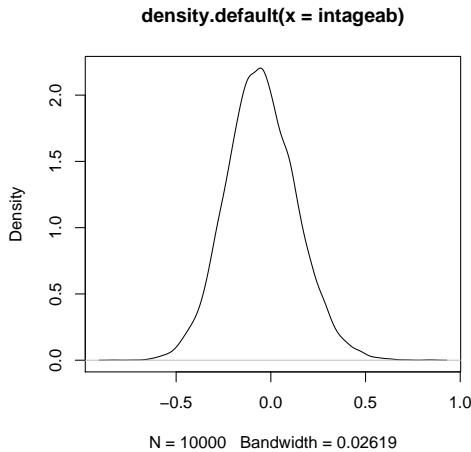
| | abEUG | |
|-----|---------|---------|
| | Y | O |
| Cat | μ_1 | μ_2 |
| Y | μ_5 | μ_6 |

$$\begin{array}{c}
 \text{abEUG} \\
 \text{abECG-Y} \quad \begin{array}{ccccc}
 & Y & & O & \\
 \text{Cat} \quad N & \mu_3 & & \mu_4 & \\
 & Y & \mu_7 & & \mu_8
 \end{array} \\
 \mu_1 + \mu_3 - \mu_2 - \mu_4 - \mu_5 - \mu_7 + \mu_6 + \mu_8 = 0
 \end{array}$$

```

intageab <- p1 + p5 - p2 - p6 - p3 - p7 + p4 + p8
plot(density(intageab))

```



∞ March 19, 2013 ∞

14.1 Logistic Regression In SAS

JAGS freefloating, SAS constrained

| | | | |
|-------|-------|-----------------|-----------------|
| | Cat-N | Age | |
| | | Y | O |
| ECG | N | $p_1, .07, .06$ | $p_2, .12, .12$ |
| | Y | $p_3, .10, .12$ | $p_4, .16, .16$ |
| <hr/> | | | |
| | Cat-Y | Age | |
| | | Y | O |
| ECG | N | $p_5, .12, .15$ | $p_6, .20, .23$ |
| | Y | $p_7, .17, .18$ | $p_8, .26, .24$ |

SAS:

$$\begin{aligned}
 \log \left(\frac{p}{1-p} \right) &= \beta_0 + \beta_{\text{cat}} + \beta_{\text{age}} + \beta_{\text{ecg}} \\
 p &= \frac{1}{1 + e^{-x\beta}} \\
 p_5 &= \frac{1}{1 + e^{-\beta_0 - \beta_{\text{cat}}}}
 \end{aligned}$$

This model has 4 degrees of freedom

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

```
1/(1 + exp(2.6263 - 0.6199))
## [1] 0.1185
```

JAGS Model

$$\log\left(\frac{p_i}{1-p_i}\right) = b_i \quad i = \{1, \dots, 8\}$$

this model has 8 degrees of freedom

```
mdl <- 'model {
  for (i in 1:8) {
    CHD[i] ~ dbin(p[i], nRisk[i]);
    logit(p[i]) <- bint + bcat*cat[i] + bage*age[i] + becg*ecg[i];
  }

  bint ~ dnorm(0, .1);
  bcat ~ dnorm(0, .1);
  bage ~ dnorm(0, .1);
  becg ~ dnorm(0, .1);
}

writeLines(mdl, "code/LogitReducedModel.jags")'

CHD <- chd[, 1]
nRisk <- chd[, 2]
tmt <- chd[, 6]
cat <- chd[, 3]
age <- chd[, 4]
ecg <- chd[, 5]

jags.data <- c("CHD", "nRisk", "cat", "age", "ecg")
jags.params <- c("bint", "bcat", "bage", "becg", "p")

set.seed(12)
logit.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/LogitReducedModel.jags",
  n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 70
##
## Initializing model
```

SAS's Deviance is calculate as Dbar – Dmean

computing $pD = \frac{\text{Var}(\text{deviance})}{2}$ is a better way to calculate it in general

14.2 Random Effects

```
seeds <- read.table("data/seednew.dat", stringsAsFactors = FALSE, skip = 1, col.names = c("r",
  "n", "tmt", "nseed", "ntype"))
sum(seeds$n[seeds$tmt == 1])
## [1] 272
sum(seeds$r[seeds$tmt == 1])
## [1] 99
var(seeds$r[seeds$tmt == 1])
## [1] 40.7
```

Is the variability reasonable?

```
mdl <- 'model {
  for (i in 1:21) {
    r[i] ~ dbin(p[i], n[i])
    logit(p[i]) <- b[tmt[i]] + e[i]
  }

  for (i in 1:4) {
    b[i] ~ dnorm(0, 1)
  }

  for (i in 1:21) {
    e[i] ~ dnorm(0, prec)
  }
  s2 ~ dunif(0,2);
  prec <- 1/s2
}

writeLines(mdl, 'code/LogitMixedModel.jags')
mdl <- 'model {
  for (i in 1:21) {
    r[i] ~ dbin(p[i], n[i])
    logit(p[i]) <- b[tmt[i]]
  }

  for (i in 1:4) {
    b[i] ~ dnorm(0, 1)
  }
}
writeLines(mdl, 'code/Logit09Simplified.jags')

r <- seeds[, 1]
n <- seeds[, 2]
tmt <- seeds[, 3]
jags.data <- c("r", "n", "tmt")
jags.params <- c("b", "e")

seed.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/LogitMixedModel.jags",
  n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

## Compiling model graph
##   Resolving undeclared variables
##   Allocating nodes
##   Graph Size: 135
```

```

##  

## Initializing model  
  

jags.params <- c("b")  

seedsimple.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/Logit09Simplified.jags",  

n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)  
  

## Compiling model graph  

##   Resolving undeclared variables  

##   Allocating nodes  

##   Graph Size: 73  

##  

## Initializing model  
  

seed.jags  
  

## Inference for Bugs model at "code/LogitMixedModel.jags", fit using jags,  

## 1 chains, each with 12000 iterations (first 2000 discarded)  

## n.sims = 10000 iterations saved  

##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  

## b[1]      -0.522  0.221 -0.948 -0.665 -0.525 -0.385 -0.068  

## b[2]       0.816  0.221  0.384  0.676  0.808  0.953  1.270  

## b[3]      -0.457  0.268 -1.008 -0.625 -0.446 -0.281  0.041  

## b[4]       0.017  0.254 -0.507 -0.143  0.028  0.186  0.503  

## e[1]      -0.287  0.298 -0.937 -0.466 -0.264 -0.082  0.232  

## e[2]      -0.016  0.265 -0.559 -0.177 -0.010  0.149  0.506  

## e[3]      -0.271  0.268 -0.846 -0.436 -0.252 -0.086  0.207  

## e[4]       0.338  0.289 -0.174  0.136  0.318  0.518  0.946  

## e[5]       0.144  0.282 -0.401 -0.035  0.134  0.314  0.728  

## e[6]       0.113  0.381 -0.585 -0.128  0.086  0.329  0.948  

## e[7]       0.068  0.261 -0.442 -0.099  0.061  0.230  0.605  

## e[8]       0.233  0.276 -0.264  0.045  0.213  0.403  0.820  

## e[9]      -0.171  0.275 -0.748 -0.347 -0.157  0.012  0.335  

## e[10]     -0.217  0.264 -0.767 -0.383 -0.204 -0.040  0.278  

## e[11]     0.128  0.350 -0.517 -0.093  0.103  0.326  0.919  

## e[12]     0.170  0.338 -0.451 -0.050  0.144  0.373  0.913  

## e[13]     -0.120  0.310 -0.766 -0.312 -0.108  0.073  0.494  

## e[14]     -0.210  0.319 -0.904 -0.406 -0.187 -0.002  0.375  

## e[15]     0.292  0.311 -0.258  0.079  0.265  0.480  0.978  

## e[16]     -0.221  0.413 -1.168 -0.433 -0.172  0.042  0.459  

## e[17]     -0.314  0.368 -1.148 -0.526 -0.275 -0.068  0.315  

## e[18]     0.088  0.292 -0.465 -0.098  0.076  0.266  0.706  

## e[19]     -0.010  0.296 -0.595 -0.192 -0.016  0.171  0.601  

## e[20]     0.302  0.301 -0.237  0.097  0.282  0.484  0.956  

## e[21]     -0.073  0.363 -0.828 -0.284 -0.062  0.148  0.641  

## deviance  98.860  5.637 88.953 94.822 98.517 102.524 110.858  

##  

## DIC info (using the rule, pD = var(deviance)/2)  

## pD = 15.9 and DIC = 114.7  

## DIC is an estimate of expected predictive error (lower deviance is better).  
  

seedsimple.jags  
  

## Inference for Bugs model at "code/Logit09Simplified.jags", fit using jags,  

## 1 chains, each with 12000 iterations (first 2000 discarded)  

## n.sims = 10000 iterations saved  

##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  

## b[1]      -0.552  0.126 -0.800 -0.637 -0.551 -0.467 -0.308  

## b[2]       0.796  0.125  0.556  0.712  0.795  0.879  1.043  

## b[3]      -0.406  0.183 -0.766 -0.529 -0.404 -0.283 -0.056  

## b[4]       0.125  0.168 -0.207  0.010  0.126  0.238  0.454  

## deviance 111.575  2.873 108.077 109.515 110.875 112.916 118.965  

##  

## DIC info (using the rule, pD = var(deviance)/2)  

## pD = 4.1 and DIC = 115.7  

## DIC is an estimate of expected predictive error (lower deviance is better).

```

 March 21, 2013

```

1 %let hwdir=Z:/Dropbox/Active/STAT451/Notes;
  filename data "&hwdir./pbib.dat";
  filename chains "&hwdir./MCMCChains.csv";
  filename html "&hwdir./MCMC_Output.html";
5   ods html close;
  ODS GRAPHICS on / imangename="MCMCPlots";
  ods html body=html (url=none)
    GPATH=&hwdir/figure/";
10  data seeds;
    infile '&hwdir./data/seeds.dat' firstobs=2;
    input seed $ type $ r n;
    if seed='a75' and type='bean' then tmt=1;
15    if seed='a75' and type='cuc' then tmt=2;
    if seed='a73' and type='bean' then tmt=3;
    if seed='a73' and type='cuc' then tmt=4;
    plate=_N_;
run;
20 /* Our attempt */
proc mcmc data=seeds seed=12 nmc=200000 nbi=2000 thin=20 propcov=quanew
            statistics=(summary interval) diag=(rl ess) moniter=
              -parms_ p)
            outpost=chdout dic;
25   array b[4];
   array e[21];
   array p[4];
   parms b p e s2;
30   prior b: ~ normal(0, prec=1);
   random e ~ normal(0, var=s2) subject=plate;
   prior s2 ~ uniform(0,2);
35   p[tmt] = logistic(b[tmt]);
   model chd ~ binomial(nrisk, p[tmt]);
run;

/* His Code */
40 proc mcmc data=seeds seed=12 nmc=200000 nbi=2000 thin=20 outpost=chdout
            statistics=(summary interval) diag=(rl autocorr ess)
            monitor=( p alpha s2plate )
            propcov=quanew dic;
45   array alpha[4] alpha1-alpha4;
   array p[4] p1-p4;
   parms alpha: 0;
   parms s2plate 1;
   prior alpha: ~ normal(0, var=10); /* Fixed Effect */
   random b~ normal(0, var=s2plate) subject=plate monitor=(b); /* Random
      Effect: has hyper-prior */
   prior s2plate ~ gamma(1.5, scale=2);
50

```

```

p[tmt] = logistic(alpha[tmt]+b) ;
model r ~ binomial(n, p[tmt]) ;
run ;

```

15 Poisson Likelihood

11epi.dat Epilepsy

- “pid” — Patient ID
- “scount” — Seizure Count
- “tmt” — Treatment
- “base” — Base Effect
- “age” — Age
- “visit”

S is the number of seizures (scount)

```

epi <- read.table("data/11epi.dat", col.names = c("pid", "scount", "tmt", "base",
                                                 "age", "visit"))
N <- nrow(epi)
pid <- epi$pid
N_pid <- length(unique(pid))
scount <- epi$scount
tmt <- epi$tmt
base <- epi$base
age <- epi$age
visit <- epi$visit
epi$indic4 <- ifelse(visit == 4, 1, 0)
indic4 <- epi$indic4

```

$$S_i \sim \text{Poisson}(\lambda)$$

$$\lambda = a_0 + a_{\text{age}} * \text{age}_i + a_{\text{tmt}} * \text{tmt}_i + a_{\text{base}} * \text{base}_i$$

∞ March 26, 2013 ∞

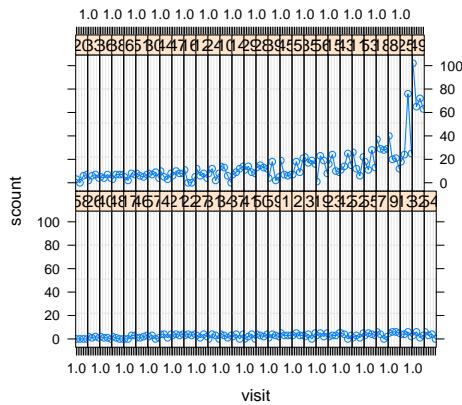
Plotting data

```

library(nlme)
g_epi <- groupedData(scount ~ visit | pid, epi)
plot(g_epi)
plot()

## Error: argument "x" is missing, with no default

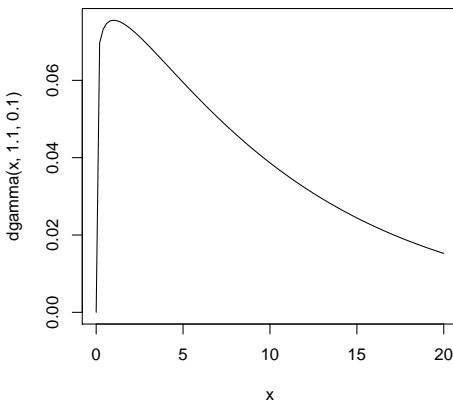
```



Things learned from this plot:

- There doesn't seem to be an effect due to time
- Subjects matter, so we need a random

```
curve(dgamma(x, 1.1, 0.1), from = 0, to = 20)
```



```
mdl <- "model {\nfor (i in 1:N) {\\nscount[i] ~ dpois(lambda[i])\\nlambda[i] <- a_0 + a_age*age[i] + a_base*base[i] + a_tmt*tmt[i]\nwriteLines(mdl, \"code/EPI_Interaction.jags\")\n\njags.data <- c(\"N\", \"pid\", \"scount\", \"age\", \"tmt\", \"base\")\njags.params <- c(\"a_0\", \"a_age\", \"a_base\", \"a_tmt\", \"int_age_base\", \"int_age_tmt\", \"int_base_tmt\", \"ll\")\nset.seed(3245)\nepi.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = \"code/EPI_Interaction.jags\", n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)\n\n## module glm loaded\n\n## Error:\n## Error parsing model file:\n## syntax error on line 6 near \"\"\n\nepi.jags\n\n## Error: object 'epi.jags' not found"
```

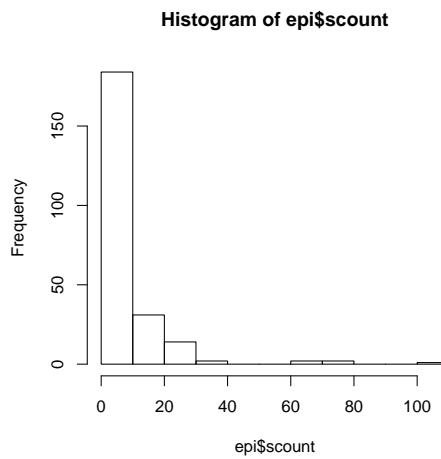
```

epi.sim <- as.mcmc(epi.jags)
## Error: object 'epi.jags' not found
plot(epi.sim, auto.layout = inter, ask = inter)
## Error: object 'epi.sim' not found
autocorr.plot(epi.sim, auto.layout = inter, ask = inter)
## Error: object 'inter' not found
rafthery.diag(epi.sim)
## Error: object 'epi.sim' not found

```

Posterior Predictive Plots

```
hist(epi$scount)
```



We can't do general posterior predictives so let's look at particular values

```

hist(epi$scount[epi$tmt == 1 & epi$age >= 19 & epi$age <= 23 & 30 <= epi$base & epi$base <=
60])
plot(density(epi$age))
curve(dnorm(x, 30, 8), add = TRUE, col = "red")
plot(density(epi$base))
curve(dgamma(1.1, scale = 30), add = T, color = "red")

## Error: 'expr' must be a function, or a call or an expression containing 'x'

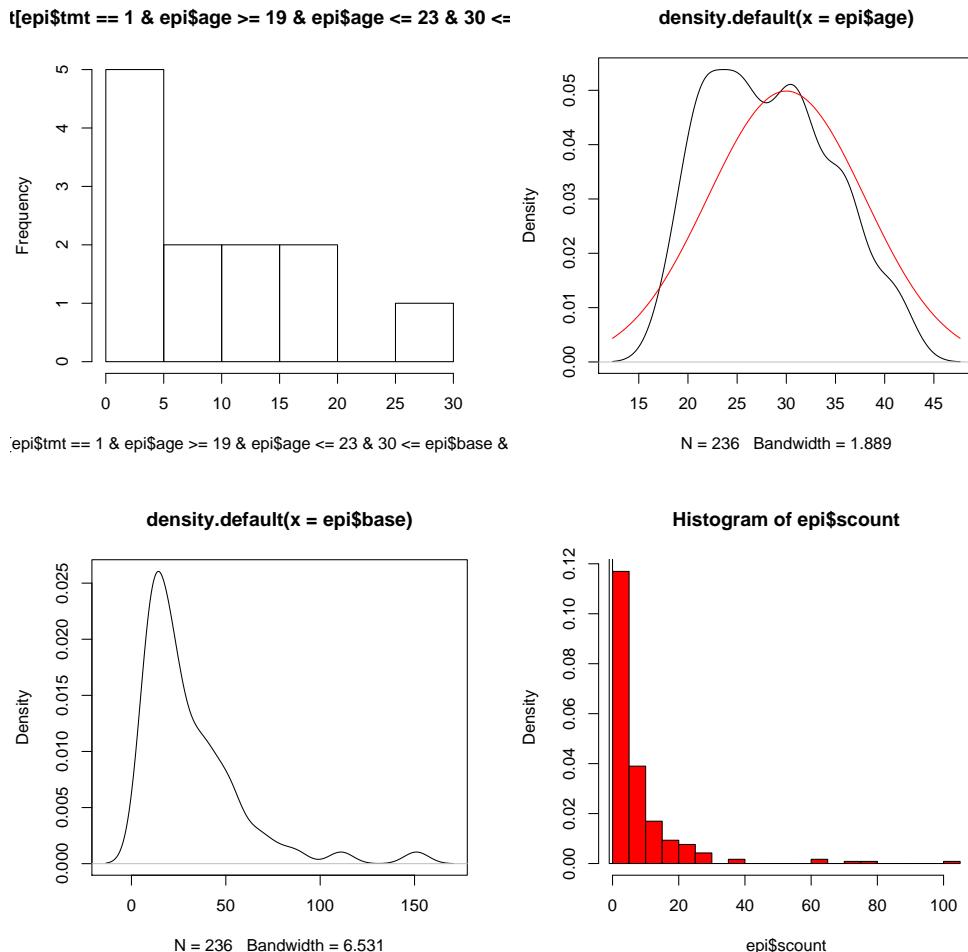
people <- cbind(age = rnorm(iter, 30, 8), )

## Error: object 'iter' not found

iter <- 10^4
pp <- numeric(iter)
for (i in 1:iter) {
  rage <- rnorm(1, 30, 8)
  rbase <- rgamma(1, 1.1, scale = 30)
  rtmt <- rbinom(1, 1, 0.5)
  x1 <- epi.sim[i, 1] + epi.sim[i, 2] * rage + epi.sim[, 3] * rbase + epi.sim[, 4] * rtmt +
    epi.sim[i, 6] * rage * rbase + epi.sim[i, 7] * rage * rtmt +
    epi.sim[i, 8] * rbase * rtmt + rexp(1, epi.sim[i, 9])
  pp[i] <- rpois(1, x1)
}
## Error: object 'epi.sim' not found

```

```
hist(epi$scount, breaks = 20, freq = F, col = "red")
hist(pp, breaks = 20, freq = F, add = T)
```



15.1 Log Poisson

 March 28, 2013

```
1 count[i] <- dpois(lambda[i])
  log(lambda[i]) <- a_0 + a_age*log(age[i]) + a_base*log(base[i]/4) +
    a_tmt*tmt[i] +
    a_base_tmt*log(base[i]/4)+tmt[i] + a_4*indic4[i] +
    u[pid[i]] + aobs[i]
5 aobs[i] ~ dnorm(0,1/s2obs)
```

```
lbase4 <- log(base/4)
lage <- log(age)
mdl <- 'model {
  for (i in 1:N) {
    scount[i] ~ dpois(lambda[i])
    log(lambda[i]) <- a_0 + a_age*lage[i] + a_base*lbase4[i] + a_tmt*tmt[i] +
      a_base_tmt*lbase4[i]*tmt[i] + a_4*indic4[i] +
```

```

        u_sub[pid[i]] + u_obs[i]
    u_obs[i] ~ dnorm(0, 1/s2obs)
}
for (i in 1:N_pid) {
    u_sub[i] ~ dnorm(0, 1/s2sub)
}

a_0      ~ dnorm(0, 0.0001)
a_base   ~ dnorm(0, 0.0001)
a_tmt    ~ dnorm(0, 0.0001)
a_base_tmt ~ dnorm(0, 0.0001)
a_age    ~ dnorm(0, 0.0001)
a_4      ~ dnorm(0, 0.0001)
s2obs    ~ dgamma(1.1, 1)
s2sub    ~ dgamma(1.1, 1)
}
'
writeLines(mdl, 'code/EPIPAPER.jags')

jags.data <- c("N", "N_pid", "pid", "scount", "lage", "tmt", "lbase4", "indic4")
jags.params <- c("a_0", "a_age", "a_base", "a_tmt", "a_base_tmt", "a_4", "s2obs",
                 "s2sub")

set.seed(3245)
epiP.jags <- jags(data = jags.data, parameters.to.save = jags.params, model.file = "code/EPIPAPER.jags",
                     n.iter = 12000, n.burnin = 2000, n.chains = 1, n.thin = 1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 2306
##
## Initializing model

```

If you have correlated X 's then you shift the X

∞ April 2, 2013 ∞

15.2 Complex Model Posterior Predictive

```

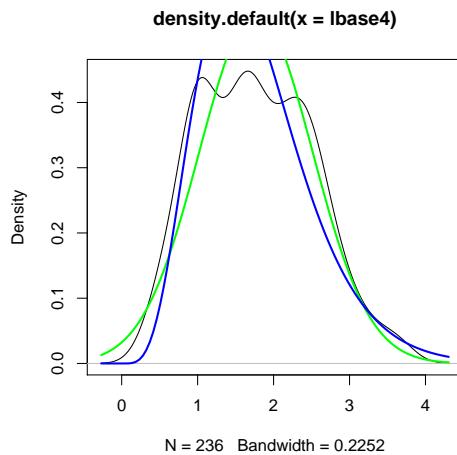
epiP.sim <- as.mcmc(epiP.jags)
mCol <- function(name, data = epiP.sim) {
  col <- which(attr(data, "dimnames")[[2]] == name)
  data[, col]
}
a_0 <- mCol("a_0")
a_age <- mCol("a_age")
a_base <- mCol("a_base")
a_tmt <- mCol("a_tmt")
a_base_tmt <- mCol("a_base_tmt")
a_4 <- mCol("a_4")
s2sub <- mCol("s2sub")
s2obs <- mCol("s2obs")

```

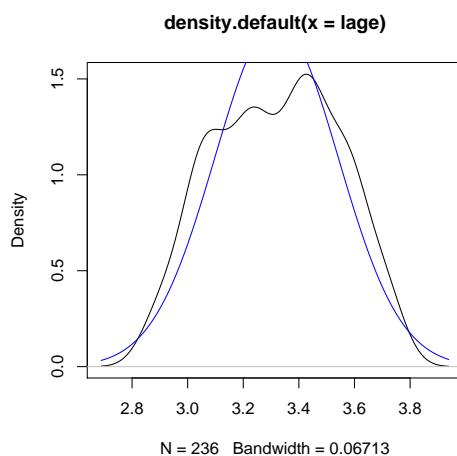
We might be able to sample better from age with a gamma

$$\begin{aligned}\mu &= \alpha\beta \\ v &= \alpha\beta^2 \\ \beta &= \frac{v}{\mu} = .31\end{aligned}$$

```
b <- var(lbase4)/mean(lbase4)
a <- mean(lbase4)/b
plot(density(lbase4))
curve(dnorm(x, mean(lbase4), sd(lbase4)), add = TRUE, col = "green", lwd = 2)
curve(dgamma(x, a, scale = b), add = TRUE, col = "blue", lwd = 2)
```



```
plot(density(lage))
curve(dnorm(x, mean(lage), sd(lage)), add = TRUE, col = "blue")
```

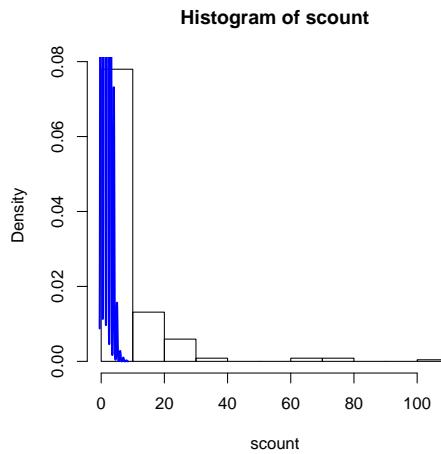


```
lmu <- numeric(10000)
mean_lage <- mean(lage)
sd_lage <- sd(lage)
for (i in 1:10000) {
  lb <- rgamma(1, a, scale = b)
```

```

tt <- rbinom(1, 1, 0.5)
la <- rnorm(1, mean_lage, sd_lage)
i4 <- rnorm(1, 1, 0.25)
lmu <- a_0[i] + a_age[i] * la + a_base[i] * lb + a_tmt[i] * tt + a_base_tmt[i] *
    lb * tt + a_4[i] * i4 + rnorm(1, 0, sqrt(s2sub[i])) + rnorm(1, 0, sqrt(s2obs[i]))
}
pp <- rpois(10000, exp(lmu))
hist(scount, freq = FALSE)
lines(density(pp), col = "blue", lwd = 2)

```



15.3 Non Diagonal Covariance

$$\begin{aligned}
 y &= x\beta + Zu + e \\
 e &\sim \mathcal{N}(0, R) \\
 u &\sim \mathcal{N}(0, G) \\
 V(y) &= ZGZ + R
 \end{aligned}$$

Compound Symmetric Structure

$$\begin{bmatrix}
 \sigma_e^2 + \sigma_u^2 & \sigma_u^2 & \sigma_u^2 & \sigma_u^2 \\
 \sigma_u^2 & \sigma_e^2 + \sigma_u^2 & \sigma_u^2 & \sigma_u^2 \\
 \sigma_u^2 & \sigma_u^2 & \sigma_e^2 + \sigma_u^2 & \sigma_u^2 \\
 \sigma_u^2 & \sigma_u^2 & \sigma_u^2 & \sigma_e^2 + \sigma_u^2
 \end{bmatrix}$$

$$\rho = \frac{\text{Cov}(1, 2)}{\sqrt{\text{Var}(1)\text{Var}(2)}} = \frac{\sigma_u^2}{\sqrt{\sigma_e^2\sigma_u^2}}$$

Three Groups in the experiment

- Control

- Reps
- Weight

`ind1 = (50,50, 40, 60, 55, 60, 45)`

Auto Regressive (AR(1))

$$V(Sub) = \begin{pmatrix} 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 & \rho^6 \\ \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 \\ \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 \\ \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 \\ \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 \\ \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho \\ \rho^6 & \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 \end{pmatrix}$$

```
weight <- read.table("data/weight.dat", header = TRUE)
weight$tz <- weight$time - 1
weight$st <- paste0(weight$tmt, "/", weight$subject)
weight$subj <- c(rep(1:20, times = 7), rep(21:36, times = 7), rep(37:57, times = 7))
weight$subj <- as.numeric(as.factor(paste(weight$tmt, sprintf("%02s", weight$subject))))
```

∞ April 4, 2013 ∞

15.4 Exam 2 Debriefing

The first problem could be done with a simple beta binomial conjugate prior model.

You can't add an error term where you have only a single observation per subject.

To make it random, go back to the basic beta-binomial and then put priors and alpha and beta.

When you say $b_i + e_i$, and $e_i \sim \mathcal{N}(0, \sigma^2)$ is a noise term, we say that it is centered at 0, because it shouldn't be biased.

16 section name

Auto Regressive (AR(1)) covariance structure for a single person:

$$V(\mathbf{y}_i) = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 & \rho^6 \\ \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 \\ \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 \\ \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 \\ \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 \\ \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho \\ \rho^6 & \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 \end{pmatrix}$$

With this multivariate correlation structure, we have a multivariate normal distribution

Multivariate Normal PDF

$$V = \begin{pmatrix} \sigma^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sigma^2 \end{pmatrix}$$

$$(\mathbf{y} - \mu) = \begin{pmatrix} y_1 - \mu \\ y_2 - \mu \\ \vdots \\ y_7 - \mu \end{pmatrix}$$

$$\begin{pmatrix} y_1 - \mu & y_2 - \mu & \dots & y_7 - \mu \end{pmatrix} \begin{pmatrix} \frac{1}{\sigma^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma^2} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sigma^2} \end{pmatrix} \begin{pmatrix} y_1 - \mu \\ y_2 - \mu \\ \vdots \\ y_7 - \mu \end{pmatrix}$$

$$\mathbf{y}_i \sim \text{MVN}(\mu_i, \mathbf{V})$$

$$\mu_{ij} = \beta_0 + \beta_1 * t_i$$

$$\beta_0 \sim \mathcal{N}(85, 1000)$$

$$\beta_1 \sim \mathcal{N}(0, 100)$$

$$\rho \sim \text{Beta}(1, 1)$$

$$\sigma^2 \sim \text{Gamma}(1.1, 0.1)$$

```
mdl <- 'model{
  for (i in 1:57) {
    y[i, 1:7] ~ dnorm(mu[i, 1:7], tau[1:7, 1:7]);
    for (j in 1:7) {
      mu[i, j] <- b0[tmt[i, j]] + b1[tmt[i, j]]*tz[i, j]
    }
  }

  for (i in 1:3){
    b0[i] ~ dnorm(70, .001);
  }
}'
```

```

    b1[i] ~ dnorm(0, 001);
}

rho ~ dbeta(1,1);
s2err ~ dunif(0,50);

for(i in 1:7){
  for(j in 1:7){
    vv[i, j] <- s2err*rho^abs(i-j);
  }
}
tau[1:7, 1:7] <- inverse(vv[1:7, 1:7])
}

writeLines(mdl, 'code/MultivariateNormal.jags')

# Straight R
y <- matrix(weight[order(weight$subj), ]$response, 57, 7, byrow = TRUE)
# Reshape
library(reshape2)
y <- acast(weight[, c("subj", "time", "response")], subj ~ time, value.var = "response")
tz <- acast(weight[, c("subj", "time", "tz")], subj ~ time, value.var = "tz")
tmt <- acast(weight[, c("subj", "time", "tmt")], subj ~ time, value.var = "tmt")
jags.data <- c("y", "tz", "tmt")
jags.params <- c("b0", "b1", "s2err", "rho")
ininit <- function() {
  list(b0 = rep(70, 3), b1 = rep(0, 3), s2err = runif(1, 0, 50), rho = runif(1,
    0, 1))
}
ar1.jags <- jags(data = jags.data, parameters.to.save = jags.params, inits = ininit,
  model.file = "code/MultivariateNormal.jags", n.iter = 12000, n.burnin = 2000,
  n.chains = 1, n.thin = 1)

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 1051
##
## Initializing model

ar1.jags

## Inference for Bugs model at "code/MultivariateNormal.jags", fit using jags,
## 1 chains, each with 12000 iterations (first 2000 discarded)
## n.sims = 10000 iterations saved
##      mu.vect sd.vect   2.5%    25%    50%    75%   97.5%
## b0[1]    79.750  0.746  78.318  79.251  79.749  80.251  81.215
## b0[2]    79.748  0.832  78.126  79.192  79.758  80.311  81.360
## b0[3]    81.074  0.730  79.601  80.589  81.079  81.569  82.510
## b1[1]   -0.024  0.088  -0.197  -0.082  -0.024  0.035  0.150
## b1[2]    0.237  0.099  0.044  0.171  0.238  0.304  0.430
## b1[3]    0.338  0.086  0.169  0.279  0.337  0.397  0.504
## rho     0.951  0.009  0.933  0.946  0.952  0.958  0.967
## s2err   11.192  1.974  7.973  9.792  10.994  12.344  15.661
## deviance 543.302  4.037 537.353 540.373 542.653 545.541 552.814
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 8.1 and DIC = 551.4
## DIC is an estimate of expected predictive error (lower deviance is better).

```

17 Censored Data

Mice Survival data

Time is time to death

Right censoring happens if you can't record data

Left censoring happens if you start recording data on someone after your main data

| | |
|----------------------|----------------------------|
| Left Censored | CDF |
| Frequentest do this: | Known Data likelihood |
| | Right Censored SDF |

to do this we need to write our own likelihood.

```

1  data mice;
   infile 'micesurvival.dat' firstobs=2;
   input mid tmt time censored;
run;

5  proc mcmc data=mice outpost=mice1 nmc=1000000 nbi=2000 thin=100 seed=1234
   monitor=(r mu) diag=(ess autocorr r1) dic propcov=quanew;

10 array mu[4] mu1-mu4;
   parms mu:;
   parms r;
   prior mu: ~ gamma(1.5, scale=3);
   prior r ~ gamma(1.5, scale=4);
   mm = mu[tmt];
   if censored = 0 then
      ll = logpdf('gamma', time, r, mm);
   else
      ll = logpdf('gamma', censored, r, mm);
   model general(ll);
20 run;
```

SAS Needs zeros for censored times

```

mice <- read.table("data/micesurvival.dat", header = TRUE)
### JAGS uses an indicator variable that is 1 if censored and 0 if not
mice$isCensored <- ifelse(is.na(mice$time), 1, 0)

mice$timecen <- ifelse(is.na(mice$time), NA, mice$time)

### JAGS expects the outcome to be missing if censored recode tiem so that if
### it is censored, it has a missing value
mice$cenLimit <- ifelse(is.na(mice$time), mice$censored, 40)

# Model Statement
model <- 'model {
  for (i in 1:n){
    isCensored[i] ~ dinterval(time[i], cenLimit[i]);
    time[i] ~ dgamma(r, test[tmt[i]])
  }
  r ~ dgamma(1.5, .25)
  for (j in 1:numtx) {
    mu[j] ~ dgamma(1.5, .33)
  }
}'
```

```

        test[j] <- 1/mu[j]
    }
}

writeLines(model, 'code/Censored.jags')

### Consolidating the Data to pass from R to JAGS
n <- length(mice$time)
isCensored <- mice$isCensored
time <- mice$timecen
cenLimit <- mice$cenLimit
numtx <- max(mice$tmt)
tmt <- mice$tmt

### named list of data
jags.data <- list(n = n, isCensored = isCensored, time = time, cenLimit = cenLimit,
                  tmt = tmt, numtx = numtx)

### The time variable has missing values and it needs initialized values for
### the missing data I'm setting the initial value to the observed value if no
### censoring
timeInit <- ifelse(is.na(time), 42, NA)

innts <- list(list(time = timeInit, r = rgamma(1, shape = 1.5, rate = 0.25), mu = rgamma(numtx,
                                             shape = 1.55, rate = 0.33)))

### Params to monitor
parms <- c("mu", "r")

survcen1.sim <- jags(data = jags.data, inits = innts, parameters.to.save = parms,
                      model.file = "code/Censored.jags", n.iter = 12000, n.burnin = 2000, n.chains = 1,
                      n.thin = 1)

## module glm loaded

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph Size: 335
##
## Initializing model

```

WinBUGS code is will probably not give the correct answers, verify with SAS