# Stats 102C - Lecture 9-2

Miles Chen

Week 9 Wednesday

# The Hierarchical model for multiple players

We can use the same hierarchical model to explain the data (hits and outs) of multiple players.

Each player has their own batting average. $\theta_s$

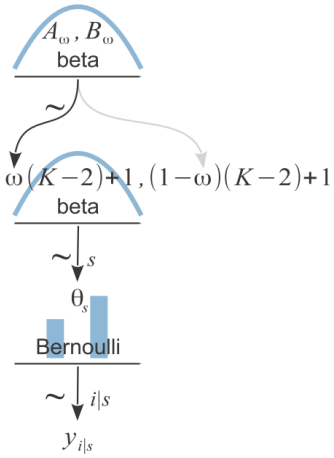
- Player 1 has batting average $\theta_1$
- Player 2 has batting average $\theta_2$
- Player s has batting average $\theta_s$

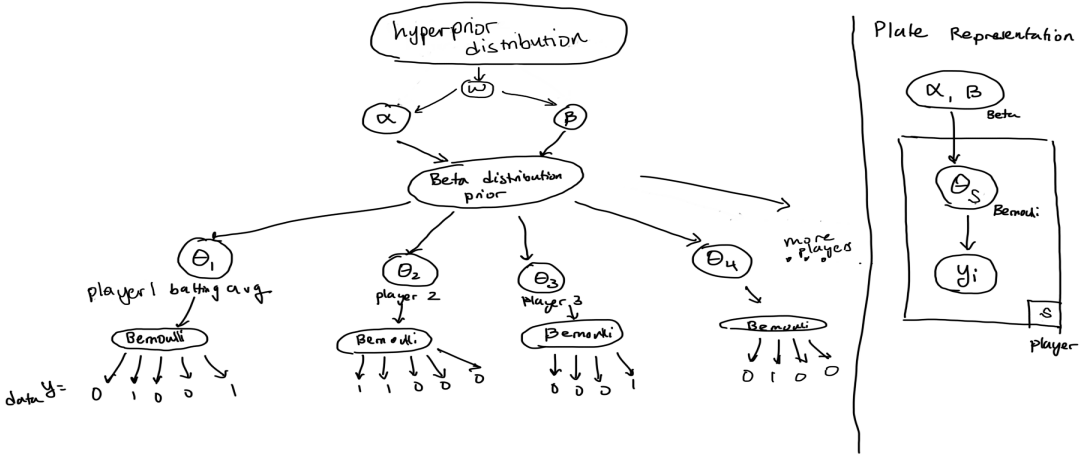
# The Hierarchical model for a Baseball Player

- $K = 300$
  - $K$ is an arbitrarily selected constant
- $\omega \sim \text{Beta}(A_\omega, B_\omega)$
  - The mode $\omega$ is a random value drawn from the hyperprior distribution whose parameters are arbitrarily selected.
- $\alpha = \omega(K-2)+1, \beta = (1-\omega)(K-2)+1$
  - Shape parameters $\alpha, \beta$ are determined by $\omega$ and $K$
  - They determine the beta distribution used to generate $\theta$
- $\theta_s \sim \text{Beta}(\alpha, \beta)$
  - Each player's batting average $\theta_s$ is randomly drawn from this beta distribution.
- $y_i|s \sim \text{Bernoulli}(\theta_s)$
  - Whether we see a hit or out is a random outcome of the Bernoulli distribution that depends on the batting average $\theta_s$ of player $s$

# Hierarchical Diagram

From Doing Bayesian Data Analysis, Chapter 9





Section 1

# JAGS

# JAGS

JAGS is a program that can perform Gibbs sampling quickly for complex hierarchical models.

JAGS stands for "Just another Gibbs Sampler". It is developed using the BUGS modeling language (BUGS stands for Bayesian inference Using Gibbs Sampling).

You must download and install JAGS from http://mcmc-jags.sourceforge.net/

Once you have JAGS installed, you can interact with it using R. There are a few interfaces to JAGS from R and the one I will use is package `R2jags`.

```
library(R2jags)
```

# The BUGS model specification language

Using JAGS does not require us to figure out the conditional distributions of each parameter.

Instead, we describe the hierarchical model using the BUGS language.

The BUGS language allows us to state that certain parameters come from certain distributions. It also allows us to specify how the distributions and parameters are related.

The BUGS language is reminiscent of R but there are distinctions and you will want to consult the documentation for what parameters are expected.

http://www.openbugs.net/Manuals/ModelSpecification.html

# Recap of our Hierarchical model

Revisit Slides 12 - 14 to see our hierarchical model

# Our model in the BUGS Language

```
bayes.mod <- function() {
  for(s in 1:n_players){
    theta[s] ~ dbeta(alpha, beta)
    for(j in 1:n_atbats[s]){
      Y[s, j] ~ dbern(theta[s])
    }
  }
  omega ~ dbeta(Aomega, Bomega)
  alpha <- omega * (K - 2) + 1
  beta <- (1 - omega) * (K - 2) + 1
  Aomega <- 3
  Bomega <- 7
  K <- 300
}
```

# Line by line

```
bayes.mod <- function() {
# ...
}
```

When you specify the model in R2jags, you wrap it in a function. This is not a function that will every be run in R, but this is how the R2jags interface expects a model to be specified.

It is also possible to write a separate text file with the model specification.

# Line by line

```
for(s in 1:n_players){
  theta[s] ~ dbeta(alpha, beta)
# ...
}
```

In our model, we write that for each player `s` has a value of `theta[s]` which is drawn from a beta distribution with parameters `alpha` and `beta`.

We haven't defined `alpha` and `beta` yet, but as long we define them later in the model, it will work.

# Line by line

```
for(j in 1:n_atbats[s]){
  Y[s, j] ~ dbern(theta[s])
}
```

We reference the actual data `Y` a matrix with where each row represents a player. Each value in that row is a 0 or 1 representing a hit or miss for each at bat. Each of the values in the row are random draws from a Bernouli distribution, where the probability is `theta[s]` (which was specified earlier)

```
omega ~ dbeta(Aomega, Bomega)
```

omega comes from a beta distribution. This is our hyperprior distribution and it depends on the hyperhyperparameters Aomega and Bomega.

```
alpha <- omega * (K - 2) + 1
beta <- (1 - omega) * (K - 2) + 1
Aomega <- 3
Bomega <- 7
K <- 300
```

These lines define values that are determined by other values. Note these values have <- to define them, while the values that came from random distributions used ~.

`alpha` and `beta` are defined using `K` and the random value of `omega`.

`Aomega`, `Bomega`, and `K` are arbitrarily chosen values and defined here.

# Other requirements for JAGS

```
data_jags <- list("Y", "n_atbats", "n_players")
bayes_mod_params <- c("theta", "omega")
```

Using JAGS requires a few lists and other functions to be passed.

You must specify the data that will be used. This is done by providing a list of character names of the data objects in the global environment.

You must also specify the parameters that must be estimated by the Gibbs sampler. In this case, `theta` is a vector of length 10, and `omega` is a single value.

```
bayes_mod_inits <- function() {
  list("theta" = rbeta(n_players, 3, 7), "omega" = rbeta(1, 3, 7))
}
```

We must also provide a function that will initialize values of the parameter list.

This function is written in R.

You can use any of R's random generating functions. You must make sure that the random function will provide a valid number of the correct length. For example, the random function specifying `theta` must provide 10 numbers because there are 10 values of theta to estimate.

Also using `rnorm(1)` to specify `omega` could result in an error because `rnorm` might produce a negative value and that would cause problems in the model.

```
bayes_mod_fit <- jags(data = data_jags, inits = bayes_mod_inits,
                     parameters.to.save = bayes_mod_params,
                     n.chains = 3, n.iter = 15000,
                     n.burnin = 1000, model.file = bayes.mod)
```

# Simulated data:

```
summary_table <- rbind(y, n_atbats, round(y/n, 4))
rownames(summary_table) <- c("hits", "at_bats", "BA")
print(summary_table)
```

```
##             [,1]     [,2]    [,3]     [,4]    [,5]     [,6]     [,7]     [,8]     [,9]    [,10]
## hits    118.0000 139.0000 119.000 148.0000 137.000 137.0000 167.0000 149.0000 163.0000 152.0000
## at_bats 486.0000 493.0000 498.000 499.0000 521.000 518.0000 513.0000 520.0000 517.0000 501.0000
## BA        0.2428   0.2819   0.239   0.2966   0.263   0.2645   0.3255   0.2865   0.3153   0.3034
```

# Running JAGS

```
bayes.mod <- function() {
  for(s in 1:n_players){
    theta[s] ~ dbeta(alpha, beta)
    for(j in 1:n_atbats[s]){
      Y[s, j] ~ dbern(theta[s])
    }
  }
  omega ~ dbeta(Aomega, Bomega)
  alpha <- omega * (K - 2) + 1
  beta <- (1 - omega) * (K - 2) + 1
  Aomega <- 3
  Bomega <- 7
  K <- 300
}

data_jags <- list("Y", "n_atbats", "n_players")
bayes_mod_params <- c("theta", "omega")
bayes_mod_inits <- function() {
  list("theta" = rbeta(n_players, 3, 7), "omega" = rbeta(1, 3, 7))
}
```

# Running JAGS

```
set.seed(123)
bayes_mod_fit <- jags(data = data_jags, inits = bayes_mod_inits,
                      parameters.to.save = bayes_mod_params,
                      n.chains = 3, n.iter = 15000,
                      n.burnin = 1000, model.file = bayes.mod)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5066
##    Unobserved stochastic nodes: 11
##    Total graph size: 5099
##
## Initializing model
```

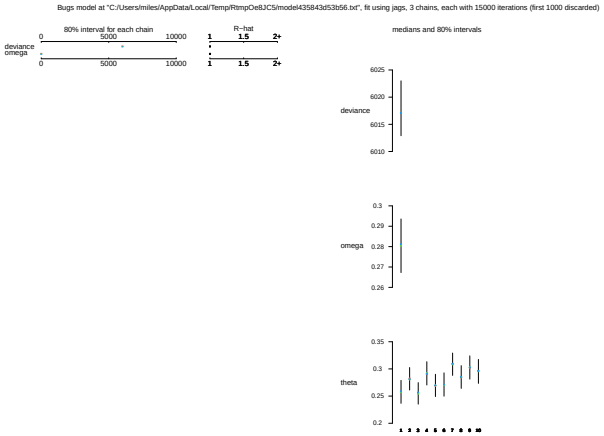
# JAGS Results

```
print(bayes_mod_fit)
```

```
## Inference for Bugs model at "C:/Users/miles/AppData/Local/Temp/RtmpOe8JC5/model435843d53b56.txt", fit using jags,
##  3 chains, each with 15000 iterations (first 1000 discarded), n.thin = 14
##  n.sims = 3000 iterations saved
##            mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat n.eff
## omega        0.281   0.010    0.261    0.273    0.281    0.287    0.301 1.002  1400
## theta[1]     0.258   0.016    0.225    0.247    0.258    0.268    0.289 1.003   900
## theta[2]     0.282   0.016    0.252    0.271    0.281    0.292    0.315 1.001  2100
## theta[3]     0.255   0.016    0.224    0.245    0.256    0.266    0.287 1.002  1200
## theta[4]     0.291   0.017    0.260    0.280    0.291    0.303    0.324 1.001  2800
## theta[5]     0.270   0.016    0.239    0.259    0.269    0.280    0.301 1.001  2400
## theta[6]     0.271   0.016    0.239    0.260    0.271    0.282    0.303 1.001  2600
## theta[7]     0.309   0.016    0.276    0.298    0.309    0.320    0.341 1.002  1700
## theta[8]     0.285   0.016    0.254    0.274    0.285    0.296    0.316 1.001  3000
## theta[9]     0.303   0.016    0.271    0.292    0.303    0.314    0.336 1.001  3000
## theta[10]    0.296   0.017    0.263    0.284    0.296    0.308    0.329 1.001  3000
## deviance  6017.638   4.050 6011.514 6014.721 6017.113 6020.007 6027.084 1.001  3000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 8.2 and DIC = 6025.8
## DIC is an estimate of expected predictive error (lower deviance is better).
```

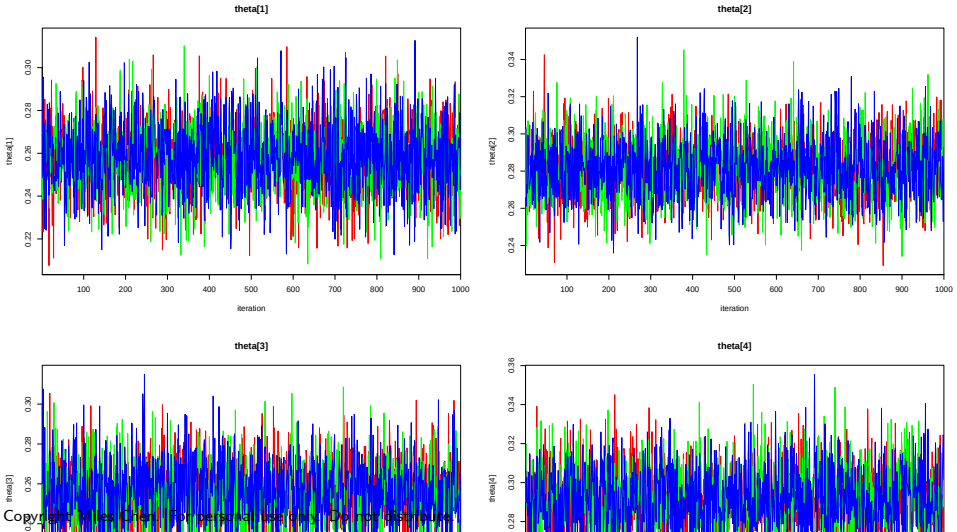
# Results Plots

```
plot(bayes_mod_fit)
```



# Diagnostic Plots

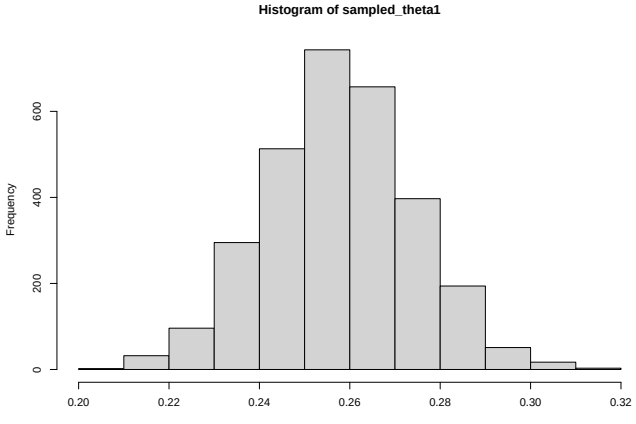
```
traceplot(bayes_mod_fit, varname = "theta", ask = FALSE)
```



# Using the sampled values

You can access the sampled values in the `$BUGSoutput$sims.list` of the jags output.

```
sampled_theta1 <- bayes_mod_fit$BUGSoutput$sims.list$theta[ , 1]
hist(sampled_theta1)
```



# Using the sampled values

What is the probability that Player 1 will get 2 or 3 hits in the next 10 at bats?

```
h <- function(theta) { dbinom(2, 10, theta) + dbinom(3, 10, theta) }

# expected value using monte carlo estimation
mean(h(sampled_theta1))
```

```
## [1] 0.5273955
```

Section 2

# Jags model 2

# Jags model 2

Here we model the total number of hits rather than each individual hit. (Compare to slide 10)

```
bayes.mod <- function() {
  for(s in 1:n_players){
    y[s] ~ dbinom(theta[s], n[s])
    theta[s] ~ dbeta(alpha, beta)
  }
  omega ~ dbeta(Aomega, Bomega)
  alpha <- omega * (K - 2) + 1
  beta <- (1 - omega) * (K - 2) + 1
  Aomega <- 3
  Bomega <- 7
  K <- 300
}
```

# Relevant Changes

```
for(s in 1:n_players){
  y[s] ~ dbinom(theta[s], n_atbats[s])
  theta[s] ~ dbeta(alpha, beta)
}
```

This time, the data `y` represents the total number of hits, rather than individual outcome of hits. For each player `s`, the number of hits `y[s]` is modeled with a binomial distribution with parameters `theta[s]` and `n_atbats[s]`. Both the batting average `theta` and the number of at-bats `n_atbats` are subset to be player specific.

The value `theta[s]` is drawn from a beta distribution with parameters `alpha` and `beta`.

The ret of the model is the same as before. `alpha` and `beta` are determined by `omega` and `K`. `omega` comes from another beta distribution, while `K`, `Aomega`, and `Bomega` are arbitrarily chosen.

Section 3

# Jags model 3

# Jags model 3

Model the value `K` as a random variable. Everything else is the same as Model 2.

```
K ~ dgamma(2, .008)
```

This gamma distribution was arbitrarily chosen because it is very broad and can allow for K to be almost any value from 0 to 1000+, with higher probabilities around 200.

Yes, these values are arbitrary, but they influence a hyperhyperparameters, so the influence on the theta parameters that are most interesting to us is fairly limited.