Week 15 Activity

For this activity we will use NBA free throw shooting data.

player	position	FTA	FTM
Aldridge	F	55	42
Durant	F	103	92
Favors	F	23	11
Green	F	67	46
Horford	F	29	22
James	F	162	113
Jordan	F	56	22
Love	F	75	63
Olynyk	F	30	22
Beal	G	61	50
Curry	G	114	103
Harden	G	115	101
Irving	G	84	76
Leonard	G	102	95
Oladipo	G	6	6
Parker	G	14	14
Paul	G	33	29
Wall	G	93	78

```
# Model - see page 250 in text
modelString <- 'model {
  for ( s in 1:Nsubj ) {
    z[s] ~ dbin( theta[s], N[s] )
    theta[s] ~ dbeta( omega*(kappa-2)+1, (1-omega)*(kappa-2)+1)
  }
  omega ~ dbeta(2.55, .45) # 85%
  kappa <- kappaMinusTwo + 2
  kappaMinusTwo ~ dgamma(.01, .01)
}'
writeLines( modelString, con='HierModel.txt')


# RUN JAGS
jags.hier <- jags.model( file = "HierModel.txt",
                         data = dataList,
                         n.chains = 3,
                         n.adapt = 50000)
```

JAGS Code for Hierarchical Model

```
## Compiling model graph
##    Resolving undeclared variables
```

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 9
##    Unobserved stochastic nodes: 11
##    Total graph size: 42
##
## Initializing model
```

```
update(jags.hier, 10000)

num.mcmc <- 10000
codaSamples <- coda.samples( jags.hier,
                             variable.names = c('omega', 'kappa','theta'),
                             n.iter = num.mcmc)
```

	player	position	FTA	FTM	FT.PCT	post.mean	lower	upper
theta[1]	Beal	G	61	50	0.82	0.85	0.77	0.92
theta[2]	Curry	G	114	103	0.90	0.90	0.85	0.94
theta[3]	Harden	G	115	101	0.88	0.88	0.83	0.93
theta[4]	Irving	G	84	76	0.90	0.90	0.84	0.95
theta[5]	Leonard	G	102	95	0.93	0.92	0.87	0.96
theta[6]	Oladipo	G	6	6	1.00	0.90	0.82	0.99
theta[7]	Parker	G	14	14	1.00	0.91	0.84	1.00
theta[8]	Paul	G	33	29	0.88	0.88	0.81	0.95
theta[9]	Wall	G	93	78	0.84	0.86	0.80	0.91

The mean and HPD values for omega and kappa are

```
HPDinterval(combine.mcmc(codaSamples))[1:2,]
```

```
##           lower       upper
## kappa 3.9234378 202.9491017
## omega 0.8667158   0.9999999
```

```
colMeans((combine.mcmc(codaSamples))[,c(1:2)])
```

```
##      kappa      omega
## 72.1518278  0.9100631
```

Exercises

1. Specify independent models using a uniform prior for θ for the players Curry, Beal, and Oladipo. Write the the sampling model and priors for these models. Recall you can find the posterior here analytically without using MCMC.

2. Specify independent models using an informative prior, of your choice, for θ for the players Curry, Beal, and Oladipo. Write the the sampling model and priors for these models. Recall you can find the posterior here analytically without using MCMC. Defend your prior choice.

3. Compare the posterior HDI from these models with those found using the hierarchical models. Note the HPDinterval function can be applied directly to samples from a beta distribution as `HPDinterval(mcmc(data  = rbeta(n = 5000, shape1 = a.star, shape2 = b.star)))`

Uniform Priors

- Beal (50 / 61 = 0.82)
- Curry (103/ 114 = 0.9)
- Oladipo (6/ 6 = 1)

Informative Priors

- Beal (50 / 61 = 0.82)
- Curry (103/ 114 = 0.9)
- Oladipo (6/ 6 = 1)

4. Reflect on the differences/similarities in the credible intervals between the two different priors as well as the hierarchical model. If you were going to bet on the players shooting percentages for the next season, which would you prefer?

5. Now fit a frequentist model to estimate θ for the players Curry, Beal, and Oladipo. Describe the sampling model you've assumed with your code (hint: `glm()` is one possible route). Discuss the differences in your intervals with what you've computed in part 3. If you had to choose one analysis: HM, Bayes with uniform priors, Bayes with informative priors, or the frequentist approach which would you choose and why?

Optional Exercise

Use a dataset containing baseball batting averages and fit a hierarchical model with the groups as the player positions. Your goal is to model batting averages of the players and assess the differences between the player positions in this dataset.

```
BattingAverage <- read.csv('http://math.montana.edu/ahoegh/teaching/stat491/data/BattingAverage.csv')
head(BattingAverage)
```

```
##          Player     PriPos Hits AtBats PlayerNumber PriPosNumber
## 1 Fernando Abad    Pitcher    1      7            1            1
## 2   Bobby Abreu Left Field   53    219            2            7
## 3    Tony Abreu   2nd Base   18     70            3            4
## 4 Dustin Ackley   2nd Base  137    607            4            4
## 5    Matt Adams   1st Base   21     86            5            3
## 6 Nathan Adcock    Pitcher    0      1            6            1
```

Assume this is an analysis you have been asked to perform as part of a job interview. Write a 1-2 page summary of your analysis using R Markdown. This should include an introduction and conclusion.

```
# Data
z <- BattingAverage$Hits
N <- BattingAverage$AtBats
s <- BattingAverage$PlayerNumber
c <- BattingAverage$PriPosNumber
Nsubj = length(unique(s))
Ncat =  length(unique(c))

dataList = list(
    z = z ,
    N = N ,
    c = as.numeric(c) , # c in JAGS is numeric, in R is possibly factor
    Nsubj = Nsubj ,
    Ncat = Ncat
  )

# Model
modelString = "
  model {
    for ( sIdx in 1:Nsubj ) {
      z[sIdx] ~ dbin( theta[sIdx] , N[sIdx] )
      theta[sIdx] ~ dbeta( omega[c[sIdx]]*(kappa[c[sIdx]]-2)+1 ,
                           (1-omega[c[sIdx]])*(kappa[c[sIdx]]-2)+1 )
    }
    for ( cIdx in 1:Ncat ) {
      omega[cIdx] ~ dbeta( omega0*(kappa0-2)+1 ,
                           (1-omega0)*(kappa0-2)+1 )
      kappa[cIdx] <- kappaMinusTwo[cIdx] + 2
      kappaMinusTwo[cIdx] ~ dgamma( 0.01 , 0.01 ) # mean=1 , sd=10 (generic vague)
    }
    omega0 ~ dbeta( 1.0 , 1.0 )
    kappa0 <- kappaMinusTwo0 + 2
    kappaMinusTwo0 ~ dgamma( 0.01 , 0.01 )  # mean=1 , sd=10 (generic vague)
  }
  "
writeLines( modelString, con='HierModelComb.txt')
```

```
# Initialize
initsList = function() {
    thetaInit = rep(NA,Nsubj)
    for ( sIdx in 1:Nsubj ) { # for each subject
      resampledZ = rbinom(1, size=N[sIdx] , prob=z[sIdx]/N[sIdx] )
      thetaInit[sIdx] = resampledZ/N[sIdx]
    }
    thetaInit = 0.001+0.998*thetaInit # keep away from 0,1
    kappaInit = 100 # lazy, start high and let burn-in find better value
    return( list( theta=thetaInit ,
                  omega=aggregate(thetaInit,by=list(c),FUN=mean)$x ,
                  omega0=mean(thetaInit) ,
                  kappaMinusTwo=rep(kappaInit-2,Ncat) ,
                  kappaMinusTwo0=kappaInit-2 ) )
}

 # RUN THE CHAINS
  parameters = c( "theta","omega","kappa","omega0","kappa0")
  adaptSteps = 10000              # Number of steps to adapt the samplers
  burnInSteps = 10000             # Number of steps to burn-in the chains
  nChains = 2
  num.mcmc = 50000

# MCMC
jagsModel = jags.model( "HierModelComb.txt" , data=dataList , inits=initsList ,
                            n.chains=nChains , n.adapt=adaptSteps )
    # Burn-in:
    cat( "Burning in the MCMC chain...\n" )
    update( jagsModel , n.iter=burnInSteps )
    # The saved MCMC chain:
    cat( "Sampling final MCMC chain...\n" )
    codaSamples = coda.samples( jagsModel , variable.names=parameters ,
                                n.iter=num.mcmc)

HPDinterval(codaSamples)
```