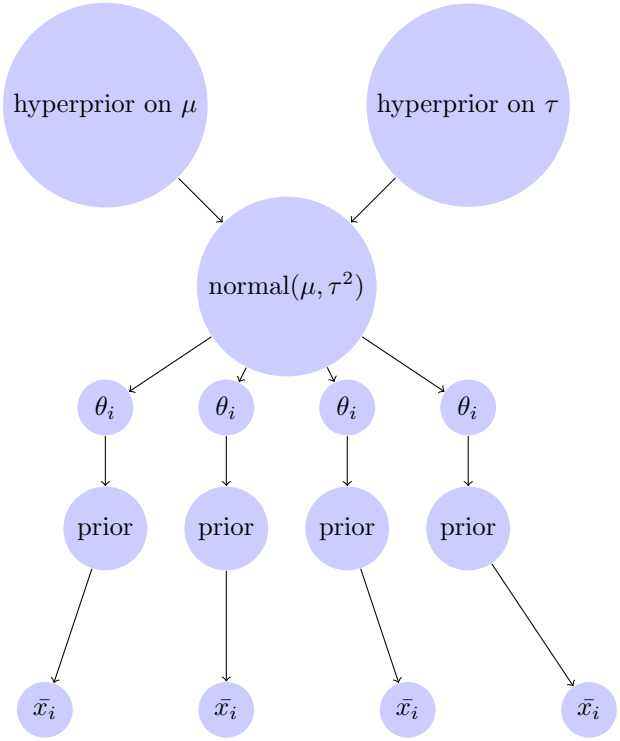
Lab 5

Peter Brody-Moore and Harry Bendekgey
Math 153: Bayesian Statistics

April 24, 2018

1 Normal Data

a) We have a hyper-prior on μ, a hyper-prior on τ, a prior on each of the θs, and then a distribution on each of the $\bar{x}$ values. We want to draw a graph with edges that connect vertices where the behavior of one depends on the other:



b) This graph depends on so many parameters that we won't recognize the posterior distribution. However, say we take θ as given. Then, the distribution of the xbars is independent of the other parameters, so we can sample from the conditional. Therefore, a Gibbs Sampler implementation is appealing because if we pretend we know everything else and sample from the conditionals, then it's much simpler to take draws from the posterior.

```
chainsAverage <- colMeans(colMeans(sims[,1:5,]))
# Bayesian Estimates
chainsAverage
```

```
## theta[1] theta[2] theta[3] theta[4] theta[5] theta[6] theta[7] theta[8]
##       NA       NA       NA       NA       NA       NA       NA       NA
##       mu      tau
##      NaN       NA
```

c) The frequentist estimates would either assume that all the groups are unique or assume that all the groups are estimating the same quantity. If they assume that the groups are different then the frequentists fall into Stein's paradox. Since there are more than 3 paramters being estimated (8 schools), setting $\hat{\theta}_i = y_i$ a.k.a making the estimate the sample mean is an inadmissible estimator because there exists an estimator with a lower expected squared error loss. If you assume all the groups are estimating the same quanitity then the frequentists would use overall mean. This still won't be good because in reality the groups probably aren't estimating the same quantity so grouping them together won't lead to accurate predictions.

The bayesian estimate we are producing is likely better than these two because it combines characteristics from both. It takes information from all the schools, but doesn't assume that the θ_i's are equal for all the schools.

Looking at a few specific estimates, the bayesian model generated a prediction of 10.84 for θ_1 and 5.254 for θ_3. Meanwhile, the separate estimate frequentist model would predict 28 for θ_1 and -3 for θ_2, while the pooled estimate frequentist model would predict 8.75 for both θ_1 and θ_2. Although we don't know the true θ_i values, the bayesian estimates show the desirable property of shrinkage without assuming every school is the same (and thus using the overall mean as the estimate).

```
#Simulate Data
mu <- 5
tau <- 10
J <- 8
thetas <- rnorm(J,mu,tau)
sds <- c(5,7,10,3,15,12,10,5) # assume we have access to
xbars <- c()
for(i in 1:J){
  xbars[i] <- rnorm(1,thetas[i],sds[i])
}


theta.update <- function (){
  theta.hat <- (mu/tau^2 + xbars/sds^2)/(1/tau^2 + 1/sds^2)
  V.theta <- 1/(1/tau^2 + 1/sds^2)
  rnorm(J, theta.hat, sqrt(V.theta))
}
mu.update <- function (){
  rnorm(1, mean(theta), tau/sqrt(J))
}
tau.update <- function (){
  sqrt(sum((theta-mu)^2)/rchisq(1,J-1))
}
n.chains <- 5
n.iter <- 1000
sims <- array (NA, c(n.iter, n.chains, J+2))
dimnames(sims) <- list (NULL, NULL, c(paste("theta[", 1:8, "]", sep=""), "mu", "tau"))
for (m in 1:n.chains){
```

```
  mu <- rnorm(1,mean(xbars),sd(xbars))
  tau <- runif(1,0,sd(xbars))
  for (t in 1:n.iter){
    theta <- theta.update()
    mu <- mu.update()
    tau <- tau.update()
    sims[t,m,] <- c(theta, mu, tau)
  }
}

chainsAverage <- colMeans(colMeans(sims[,1:5,]))
# Comparisons
quantile(sort(sims[,1,1]),c(.025, .975)) # Credible Interval

##  2.5% 97.5%
## -5.61 13.15

chainsAverage[1] # Posterior Mean Estimate

## theta[1]
##    3.731

thetas[1] # Actual Value

## [1] -0.7192

quantile(sort(sims[,1,4]),c(.025, .975)) # Credible Interval

##   2.5%  97.5%
##  6.139 17.535

chainsAverage[4] # Posterior Mean Estimate

## theta[4]
##     11.8

thetas[4] # Actual Value

## [1] 8.493
```

d) Here we use hyperiors $\mu = 5$ and $\tau = 8$ to generate θ_i's. Then we use these θ_i's and inputted standard deviations to generate $\bar{x_i}$'s. Finally, take the $\bar{x_i}$'s and standard deviations as known in our model and use the model to estimate the θ_i's. We look at the comparison between two of the known θ_i's, estimated θ_i's, and credible intervals for the two of estiamtes:

θ_1 has a estimated value of 3.73, a credible interval of [-5.61,13.15], and an actual value of -0.719.
θ_4 has a estimated value of 11.80, a credible interval of [6.14,17.54], and an actual value of 8.49.
Our simultation performs decently well. The estimated values are relatively close to the actual values, and the credible intervals cover the actual values.

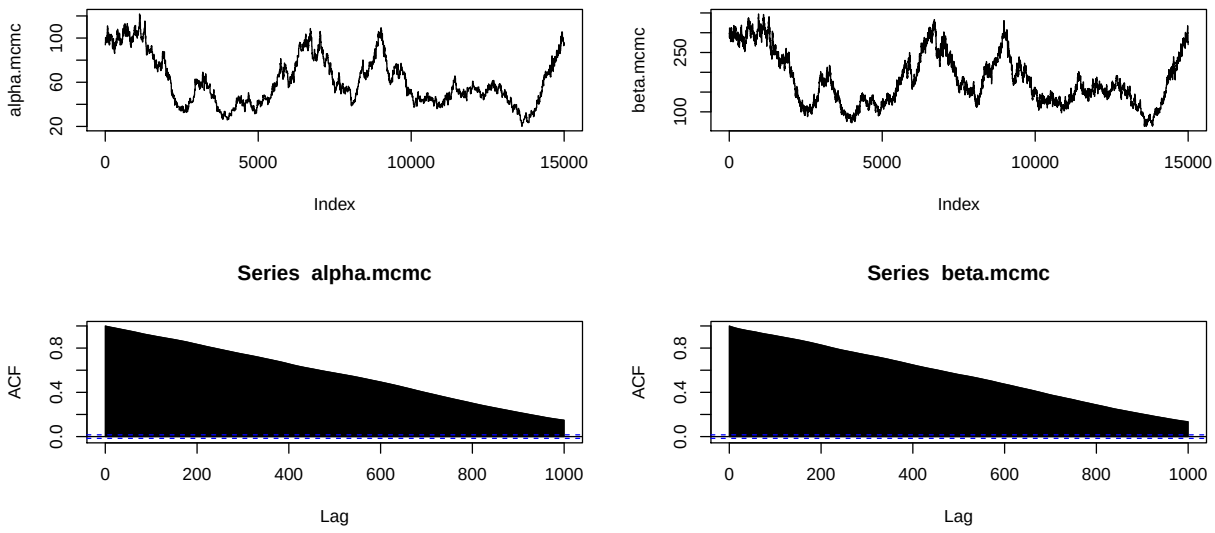
2 Binomial Data

We are 2 weeks into the 2018 MLB season, so hitters have fairly unstable batting averages currently. Their batting averages at the end of last season were much more stable (based on very large sample sizes). Let's try to guess their last season batting averages (which we will treat as their "true" batting average) based on this year's results.

We will treat these batting averages as outcome of several non-identical but conditionally independent binomial experiments, so we will observe x_i and n_i coming from a $\text{Bin}(n_i, p_i)$ experiment for $i = 1, \ldots, k$. We will assume that these p_i all came from a common $\text{Beta}(\alpha, \beta)$ distribution, and our task is to learn the value of these hyper-parameters (rather than just fixing them as we have done early on).

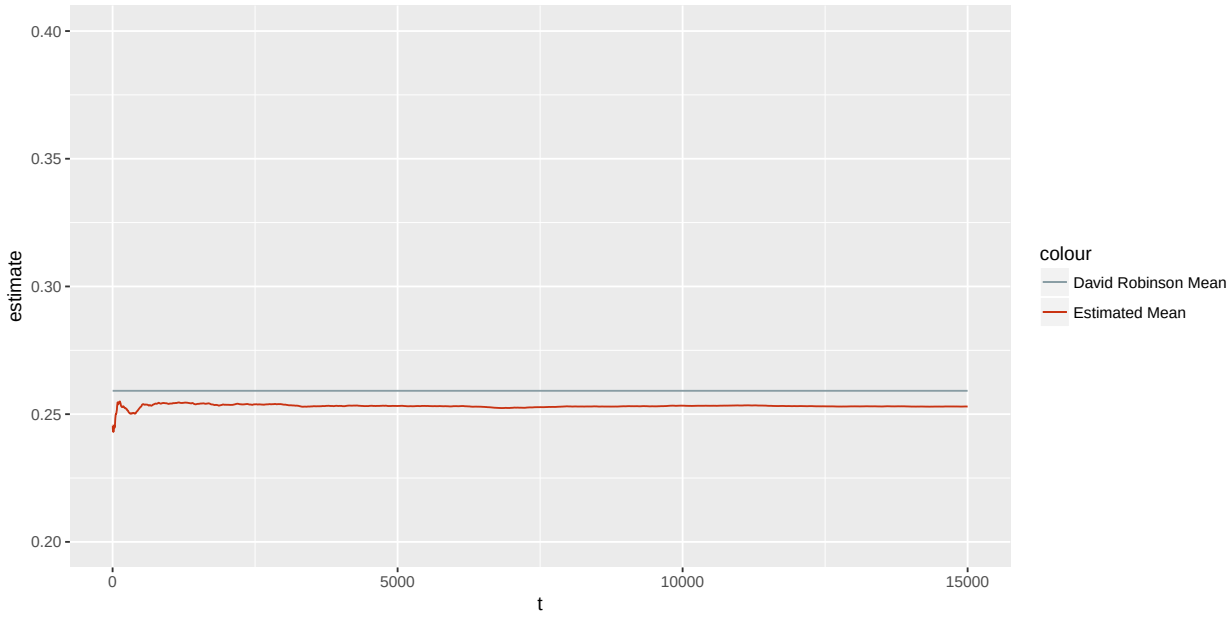
The posterior conditionals for the Gibbs sampler are all easy (standard calculations) except for the conditional on α and β, the hyper-parameters. These are not of recognizable form, and the code we will use relies on a Metropolis-Hastings step to generate these as we move through the Gibbs sampler. We simulate:

```
par(mfrow=c(2,2))
plot(alpha.mcmc,type="l")
plot(beta.mcmc,type="l")
acf(alpha.mcmc,1000)
acf(beta.mcmc,1000)
```

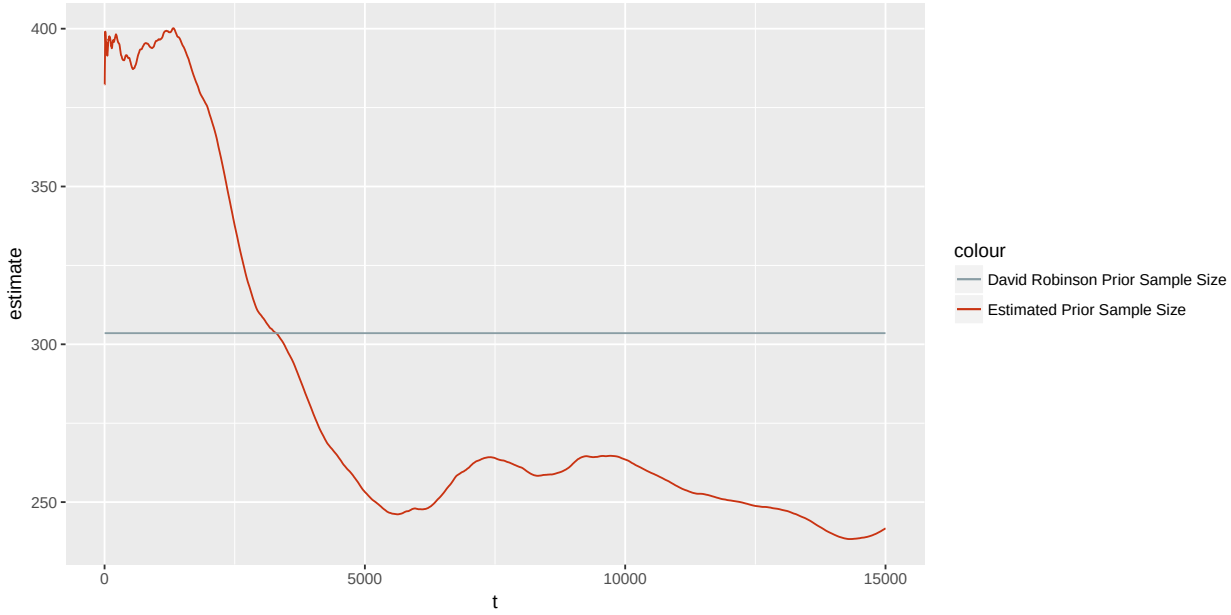


This doesn't particularly look like a converging chain. There are a few problems here, mainly with the convergence properties of the Metropolis-Hastings Chain. To examine these problems closer, let's look at the convergence of $\frac{\alpha}{\alpha+\beta}$, which we think of as the "prior mean", and the convergence of $\alpha + \beta$, which we think of as the "prior sample size". David Robinson also estimated these values using a considerably larger data set, and we've included his estimates for reference.

```
ggplot(df, aes(x=c(1:M),y=estimate, color="Estimated Mean")) + geom_line() +
  geom_line(aes(x=x, y=y,color="David Robinson Mean"), cutoff) +
  scale_color_manual(values=wes_palette(n=2, name="Royal1")) +
  scale_y_continuous(limits = c(0.2, 0.4)) + labs(x="t")
```




```
ggplot(df, aes(x=c(1:M),y=estimate, color="Estimated Prior Sample Size")) + geom_line() +
  geom_line(aes(x=x, y=y,color="David Robinson Prior Sample Size"), cutoff) +
  scale_color_manual(values=wes_palette(n=2, name="Royal1")) + labs(x="t")
```



This gives us some insight into what's going on. The chain employs independent normal random walks for α and β. As we can see here, though, the chain is pretty confident in its value for the prior mean, which tells us that the high-density area of the joint distribution of α and β lives on a diagnoal line. The random walk is having a lot of difficulty navigating that diagonal. Perhaps if we used a random walk that was more condusive to the structure of this posterior, we would see much faster convergence.

Our estimates corroborate David Robinson's estimate for the true mean of batting averages, which we

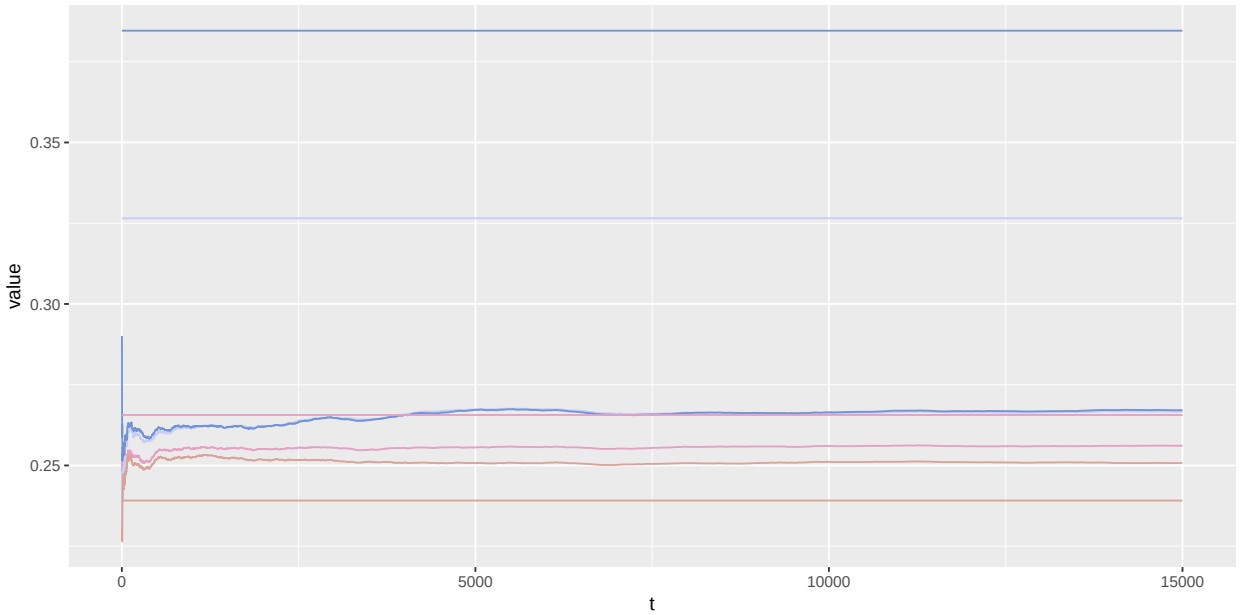
expect. We would not expect our data to have the same confidence level, encoded as prior sample size, as David Robinson, considering he was working on a much larger data set. Considering our estimator doesn't know what its prior sample size should be, it's hard to make a statement about how reasonable David Robinson's estimate is.

Now let's see how the Bayesian estimators compare to the two frequentist estimators: the one that assumes all data comes from a single binomial process, and estimate everyone's batting average to be that, or the one that assumes everyone's batting average will remain constant throughout the season.

```
mean(data$freq.unique.SE)

## [1] 0.004134

mean(data$bayes.SE)

## [1] 0.001148

mean(data$freq.same.SE)

## [1] 0.001256
```

Interestingly, there is very little difference between the Bayesian estimator and the estimator in which all players are assumed to be the same. I sampled a few players to see how our estimate for their batting averages compare to what they've been hitting so far this season.

```
ggplot() + geom_line(data=df_theta, aes(x = t, y = value, group=variable,col=variable)) +
  geom_line(data=df_true, aes(x=x, y=y, group=variable, col=variable)) +
  scale_color_manual(values=wes_palette(n=4, name="GrandBudapest2")) +
  guides(color=FALSE)
```



This tells an interesting story: we are looking at four hitters with wildly different batting averages so far this season, and yet we are estimating almost the exact same batting average for all of them. This explains why we are seeing very little difference between the Bayesian estimator and the all-the-same frequentist

estimator: the Bayesian estimator is looking at the data, and doesn't trust any of it. With so few at-bats, our posterior is consistent with all players having the same true batting average, and the variability being explained entirely by random noise. We can visualize this shrinkage by looking at the spread of the estimators compared to the spread of the data, and of last year's batting averages:

```
ggplot() +
  geom_line(data=df_est,
            aes(x=fct_inorder(x, ordered=TRUE), y=value, group=variable, col=variable)) +
  guides(colour=FALSE) +
  theme(axis.title.x=element_blank(), axis.text.x = element_text(size=12))
```