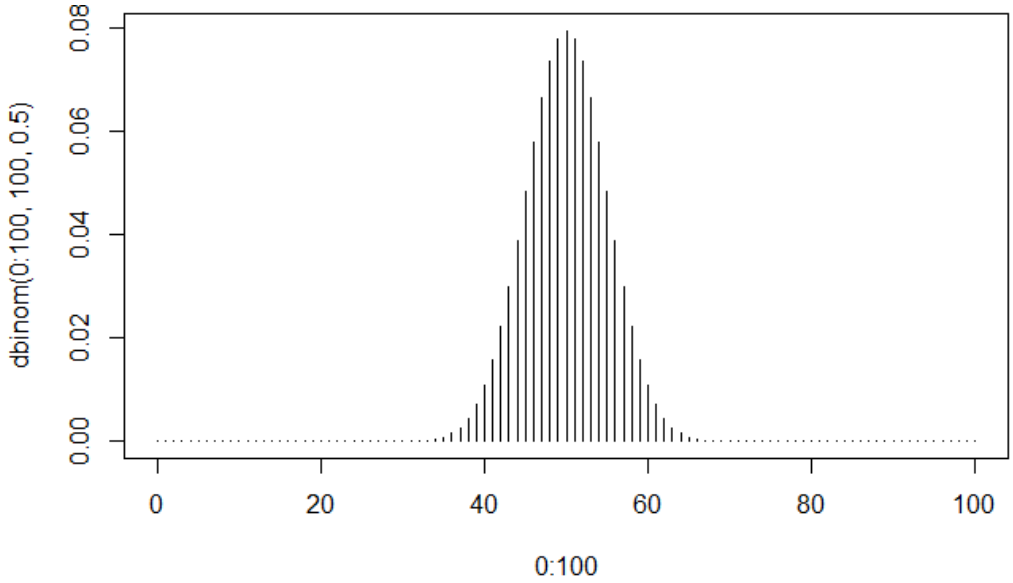
1. *In this session we saw a "coin toss" prior, where each covariate is included in the model with prior probability ½. With 3 covariates, does this prior favor models that include 0,1,2 or 3 covariates? What sized models does this approach favor with 100 covariates?*

   With 3 covariates, we have 8 equally-likely outcomes. Writing "N" for not included and "Y" for included variables, these are NNN, NNY, NYN, NYY, YNN, YNY, YYN, YYY. Counting up we see that the total numbers of "Y"s can be 0/1/2/3 with probability 1/8, 3/8, 3/8, 1/8 respectively.

   Formally, the prior for the number of covariates included follows a Binomial distribution; Binomial(3, 0.5). For 100 covariates it would be Binomial (100, 0.5). The prior support is as follows:



   Note that support is quite tightly focused on values between around 35 to 65, so very sparse and very complex models have very low prior support – which might not be immediately intuitive.

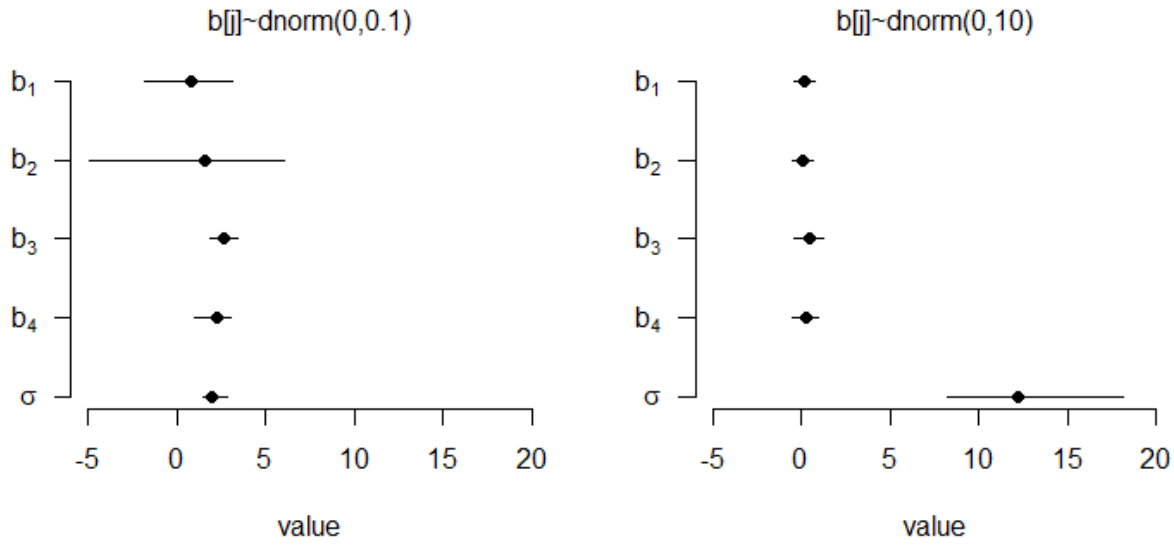
2. *Using the script on the class site, install the rjags package and run the FTO model-averaging example from class. Make sure you are using R version 4.3 or higher.*

   As in Session 4, if you have trouble with this software please reach out in person.

3. *Stan and JAGS both use a version of the "BUGS" language to specify models, but they specify the Normal distribution in different ways. As we've seen, Stan uses* `y~normal(mean,sd)` *for a Normal distribution with a given mean and standard deviation (SD). JAGS instead uses* `y~dnorm(mean, precision)`*, where precision = 1/variance = 1/standard.deviation*^{2}*. And the usual written form is* $Y \sim N(mean, variance)$*, which is different again. This can all be very confusing! What happens in the model in Q2 if the dnorm(0,0.1) is accidentally given as dnorm(0,10)?*

   Changing one number in one line of code makes a big difference, as these summaries (posterior

95% credible intervals and medians) show:



Note that in the second version, the prior on each slope coefficient is much tighter around zero – which doesn't agree well with the data, which clearly show some age and genotype effects. To compensate, the degree of noise, described by σ, is much larger in the second version than in the first.

Code

```
# Question 1

par(mar=c(4,4,0.3,0.3))
plot(x=0:100, y=dbinom(0:100, 100, 0.5), type="h")

# Question 2 - mostly code from slides

setwd("C:/Users/kenrice/Desktop/SISGBayes")
load("yX_FTO.Rdata")
y <- yX$y
X <- yX$X
n <- nrow(X)
p <- ncol(X)

setwd("C:/Users/kenrice/Desktop/SISGBayes/exercises")
library("rjags")
# first, write the model as to a text file
cat(file="linearprog2.txt", "model{
   for(j in 1:p){
      b[j]~dnorm(0, 0.1)   } # make this value 0.1 or 10 in what
follows
   z[1] <- 1  # fix the intercept to be in the model
   for(j in 2:p){
      z[j] ~ dbern(0.5)   }
     inv.sigma2 ~ dgamma( 0.5, 1.839 )
   sigma <- sqrt(1/inv.sigma2)
   for(i in 1:n){
     mu[i] <- x[i,1]*b[1]*z[1] + x[i,2]*b[2]*z[2] + x[i,3]*b[3]*z[3] +
x[i,4]*b[4]*z[4]
     y[i] ~ dnorm(mu[i], inv.sigma2) }
}")
# compile code based on model and data, then run chain
jags1 <- jags.model("linearprog2.txt", data=list(y=y,x=X, n=nrow(X),
p=ncol(X)) )
update(jags1, 50000) # initial iteraions

# version with dnorm(0,0.1)
jags1.out <- coda.samples(jags1, c("b","inv.sigma2", "z"),
n.iter=100000)[[1]]
summary(jags1.out)

# version with dnorm(0,10)
jags2.out <- coda.samples(jags1, c("b","inv.sigma2", "z"),
n.iter=100000)[[1]]
summary(jags2.out)
```

```
# code for a plot
do.one <- function(mmm){
     plot(0,0,xlim=c(-5,20), ylim=c(1, 5), axes=FALSE, xlab="value",
ylab="")
     segments(mmm[,2], 5:1, mmm[,3], 5:1)
     points(y=5:1, x=mmm[,1], pch=19)
     axis(side=1)
     axis(side=2, las=1, at=5:1, c(
          expression(b[1]),
          expression(b[2]),
          expression(b[3]),
          expression(b[4]),
          expression(sigma))
     )
     }

# quantiles
tab1 <- summary(jags1.out)$quant[1:5,c(3,1,5)]
tab1[5,] <- 1/sqrt(tab1[5,]) # switch to SD scale, not precision
tab2 <- summary(jags2.out)$quant[1:5,c(3,1,5)]
tab2[5,] <- 1/sqrt(tab2[5,]) # switch to SD scale, not precision

par(mfrow=c(1,2))
par(mar=c(4,4,2,0)+0.2)
do.one(tab1)
mtext(side=3, line=1, "b[j]~dnorm(0,0.1)")
do.one(tab2)
mtext(side=3, line=1, "b[j]~dnorm(0,10)")
```