Multiple linear regression prediction

Let Y_i be the percent increase in GOP support from 2012 to 2016 in county $i = 1, \ldots, n$. We model

$$Y_i|\beta, \sigma^2 \sim \text{Normal}(\alpha + X_{1i}\beta_1 + \ldots + X_{pi}\beta_p, \sigma^2)$$

where X_{ji} s the j^{th} covariate for county i. All variables are centered and scaled. We select prior $\sigma^2 \sim \text{InvGamma}(0.01, 0.01)$ and $\alpha \sim \text{Normal}(0, 100)$ for the error variance and intercept, and compare different priors for the regression coefficients.

Load and standardize the election data

```
# Load the data
 junk <- is.na(Y+rowSums(X))
 Y    <- Y[!junk]
 X    <- X[!junk,]
 n    <- length(Y)
 p    <- ncol(X)
 n
```

```
## [1] 3111
```

```
 p
```

```
## [1] 10
```

```
 X    <- scale(X)
# Fit the model to a training set of size 100 and make prediction for the remaining observations
 set.seed(0820)
 test  <- order(runif(n))>100
 table(test)
```

```
## test
## FALSE  TRUE
##   100  3011
```

```
 Yo    <- Y[!test]    # Observed data
 Xo    <- X[!test,]

 Yp    <- Y[test]     # Counties set aside for prediction
 Xp    <- X[test,]

 no    <- length(Yo)
 np    <- length(Yp)
 p     <- ncol(Xo)
```

Fit the linear regression model with Gaussian priors

```r
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```r
model_string <- "model{

  # Likelihood
  for(i in 1:no){
    Yo[i]   ~ dnorm(muo[i],inv.var)
    muo[i] <- alpha + inprod(Xo[i,],beta[])
  }

  # Prediction
  for(i in 1:np){
    Yp[i]  ~ dnorm(mup[i],inv.var)
    mup[i] <- alpha + inprod(Xp[i,],beta[])
  }

  # Priors
  for(j in 1:p){
    beta[j] ~ dnorm(0,0.0001)
  }
  alpha     ~ dnorm(0, 0.01)
  inv.var   ~ dgamma(0.01, 0.01)
  sigma     <- 1/sqrt(inv.var)
}"
```

Compile the model in JAGS

```r
# NOTE: Yp is not sent to JAGS!
model <- jags.model(textConnection(model_string),
                    data = list(Yo=Yo,no=no,np=np,p=p,Xo=Xo,Xp=Xp))
```

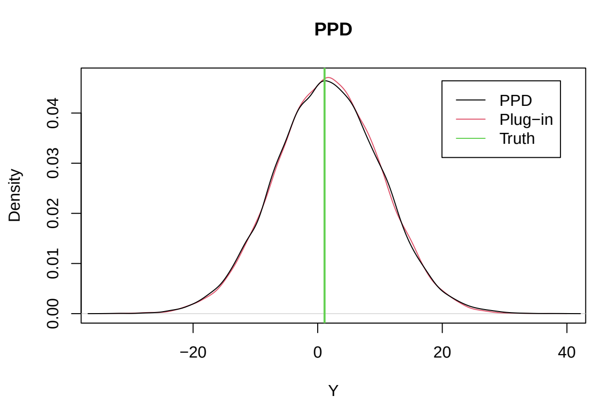
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 100
##    Unobserved stochastic nodes: 3023
##    Total graph size: 43576
##
## Initializing model
```

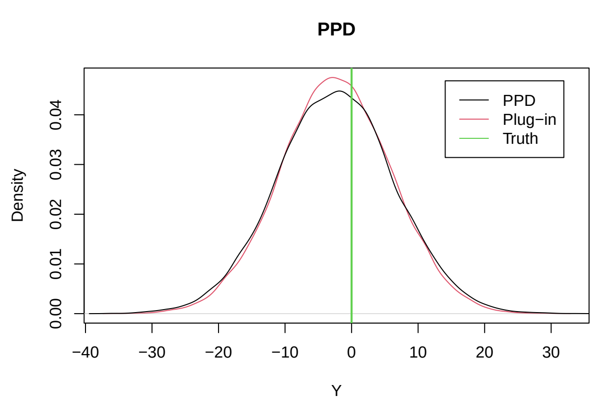
```r
update(model, 10000, progress.bar="none")

samp <- coda.samples(model,
        variable.names=c("beta","sigma","Yp","alpha"),
        n.iter=20000, progress.bar="none")

summary(samp[,-c(1:np)])
```

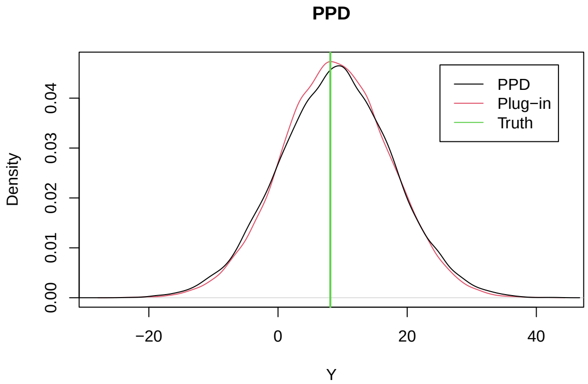
```
##
## Iterations = 10001:30000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean     SD Naive SE Time-series SE
## alpha     6.568 0.8631 0.006103       0.006558
## beta[1]  -1.552 1.2378 0.008753       0.015493
## beta[2]   4.315 1.1442 0.008090       0.018583
## beta[3]  -3.364 1.1713 0.008282       0.015001
## beta[4]  -4.720 1.1626 0.008221       0.018854
## beta[5]  -2.336 1.8878 0.013349       0.041516
## beta[6]  -5.930 1.8161 0.012842       0.038166
## beta[7]  -2.575 1.2083 0.008544       0.019242
## beta[8]  -4.386 1.8193 0.012864       0.040856
## beta[9]   7.930 2.0975 0.014831       0.053792
## beta[10]  4.388 1.8172 0.012849       0.039292
## sigma     8.334 0.6312 0.004464       0.005189
##
## 2. Quantiles for each variable:
##
##              2.5%    25%    50%     75%   97.5%
## alpha     4.8708  5.996  6.568  7.1460  8.2549
## beta[1]  -3.9784 -2.370 -1.558 -0.7283  0.8802
## beta[2]   2.0378  3.555  4.323  5.0897  6.5269
## beta[3]  -5.6508 -4.165 -3.373 -2.5879 -1.0395
## beta[4]  -6.9716 -5.494 -4.721 -3.9402 -2.4123
## beta[5]  -6.0313 -3.584 -2.352 -1.0780  1.4343
## beta[6]  -9.4501 -7.170 -5.920 -4.7026 -2.3721
## beta[7]  -4.9689 -3.381 -2.563 -1.7733 -0.1997
## beta[8]  -7.9196 -5.618 -4.391 -3.1598 -0.8407
## beta[9]   3.7221  6.535  7.918  9.3312 12.0615
## beta[10]  0.7876  3.174  4.398  5.6022  7.9427
## sigma     7.2250  7.885  8.293  8.7320  9.6976
```

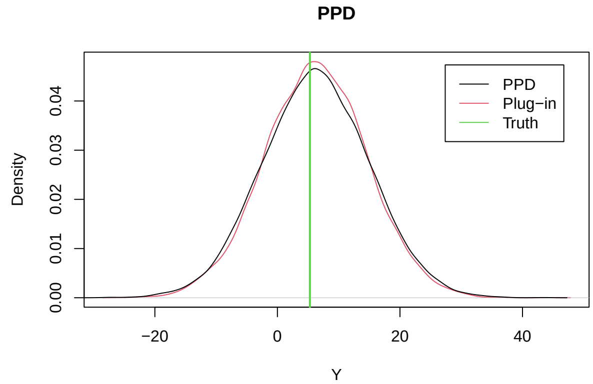
Plot samples from the posterior preditive distribution (PPD) and plug-in distribution

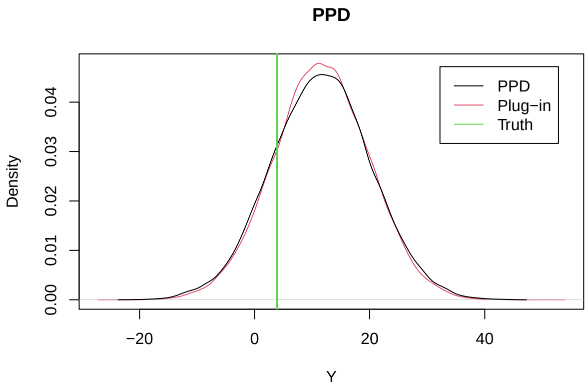
```
#Extract the samples for each parameter

 samps        <- samp[[1]]
 Yp.samps     <- samps[,1:np]
 alpha.samps <- samps[,np+1]
 beta.samps   <- samps[,np+1+1:p]
 sigma.samps <- samps[,ncol(samps)]

# Compute the posterior mean for the plug-in predictions

 beta.mn  <- colMeans(beta.samps)
 sigma.mn <- mean(sigma.samps)
 alpha.mn <- mean(alpha.samps)


# Plot the PPD and plug-in

 for(j in 1:5){

    # Plug-in
    mu <- alpha.mn+sum(Xp[j,]*beta.mn)
    y  <- rnorm(20000,mu,sigma.mn)
    plot(density(y),col=2,xlab="Y",main="PPD")

    # PPD
    lines(density(Yp.samps[,j]))

    # Truth
    abline(v=Yp[j],col=3,lwd=2)

    legend("topright",c("PPD","Plug-in","Truth"),col=1:3,lty=1,inset=0.05)
 }
```












```
# plug-in 95% intervals
low1   <- alpha.mn+Xp%*%beta.mn - 1.96*sigma.mn
high1  <- alpha.mn+Xp%*%beta.mn + 1.96*sigma.mn
cover1 <- mean(Yp>low1 & Yp<high1)
mean(cover1)
```

```
## [1] 0.9448688
```

```
# PPD 95% intervals
low2   <- apply(Yp.samps,2,quantile,0.025)
high2  <- apply(Yp.samps,2,quantile,0.975)
cover2 <- mean(Yp>low2 & Yp<high2)
mean(cover2)
```

```
## [1] 0.9545002
```

Notice how the PPD densities are slightly wider than the plug-in densities. This is the effect of accounting for uncertainty in β and σ, and it explains the slightly lower covarage for the plug-in predictions. However, for these data coverage is still OK for the plug-in predictions.