PROBIT MODEL ON 105th HOUSE

WINBUGS MODEL

```
model
{
#
#  X[,1] = DW-NOMINATE 1st Dimension
#  X[,2] = DW-NOMINATE 2nd Dimension
#  X[,3] = 1 if Republican, 0 otherwise
#  X[,4] = 1 if South (CQ def.), 0 otherwise
#
#  PRIORS
#
         for (k in 1 : 3) { beta[k] ~ dnorm(0,0.001)} # vague priors
#
# LIKELIHOOD
#
    for (i in 1 : 434)  # loop over congressional districts
    {
#
         X[i,3] ~ dbern(p[i]);
         probit(p[i]) <- mu[i];
         mu[i] <- beta[1]+X[i,2]*beta[2]+X[i,4]*beta[3]
#
#  Borrowed From Simon Jackman
#
         llh[i] <- X[i,3]*log(p[i]) + (1-X[i,3])*log(1-p[i]);
    }
    sumllh <- sum(llh[]);
#
}
```

WINBUGS INITS

```
# starting values
list(beta=c(1,1,1))
```

STATA OUTPUT (For Reference)

```
. probit partydum southdum x2

Iteration 0:   log likelihood = -306.65663
Iteration 1:   log likelihood = -298.38522
Iteration 2:   log likelihood = -298.37588

Probit regression                                 Number of obs   =        443
                                                  LR chi2(2)      =      16.56
                                                  Prob > chi2     =     0.0003
Log likelihood = -298.37588                       Pseudo R2       =     0.0270

------------------------------------------------------------------------------
    partydum |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    southdum |   .4938932   .1455113     3.39   0.001     .2086964      .77909
          x2 |  -.4628059   .1362891    -3.40   0.001    -.7299277   -.1956841
       _cons |  -.1111626   .0757394    -1.47   0.142    -.2596092    .0372839
------------------------------------------------------------------------------
```

WINBUGS OUTPUT

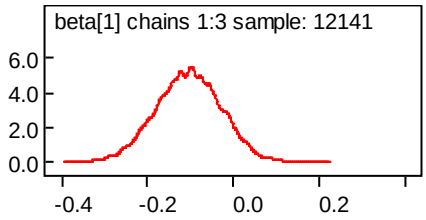
Model “Blew Up” at 15,141!! With 3 chains

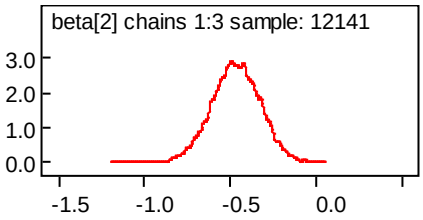
	node	mean	sd	MC error	2.5%	median	97.5%	start	sample
const	beta[1]	-0.1057	0.07703	9.577E-4	-0.2595	-0.1046	0.04287	1	15141
south	beta[3]	0.4921	0.1467	0.001851	0.207	0.4919	0.7781	1	15141
x2	beta[2]	-0.4698	0.1394	0.001418	-0.7483	-0.4688	-0.2006	1	15141
	sumllh	-293.8	1.25	0.0121	-297.0	-293.4	-292.4	1	15141

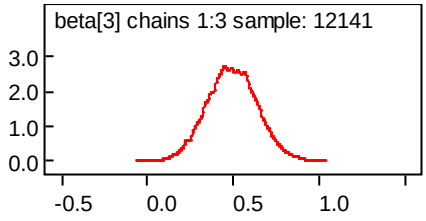
Discarding the first 1000 trials

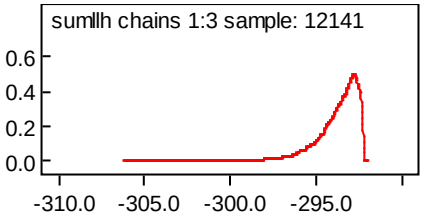
	node	mean	sd	MC error	2.5%	median	97.5%	start	sample
const	beta[1]	-0.1052	0.07665	0.001071	-0.2577	-0.1041	0.04136	1001	12141
south	beta[3]	0.4913	0.1463	0.001917	0.2095	0.4902	0.7796	1001	12141
x2	beta[2]	-0.4694	0.1394	0.001581	-0.7497	-0.4686	-0.2016	1001	12141
	sumllh	-293.8	1.233	0.01363	-297.0	-293.4	-292.4	1001	12141

Density Plots

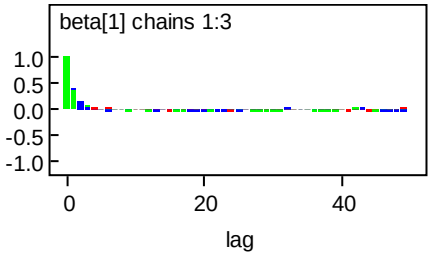


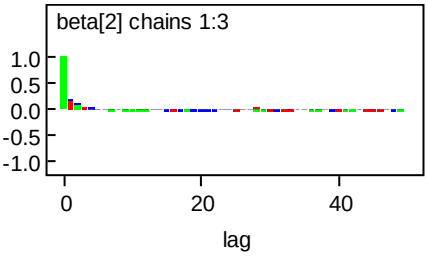


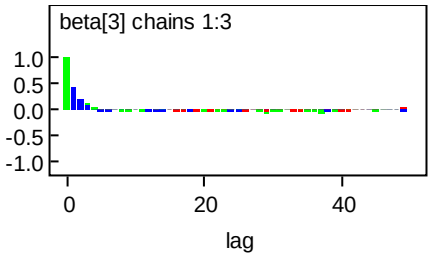


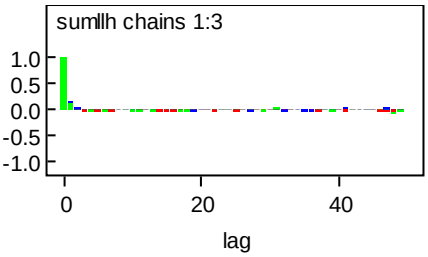


AutoCorrelation Plots

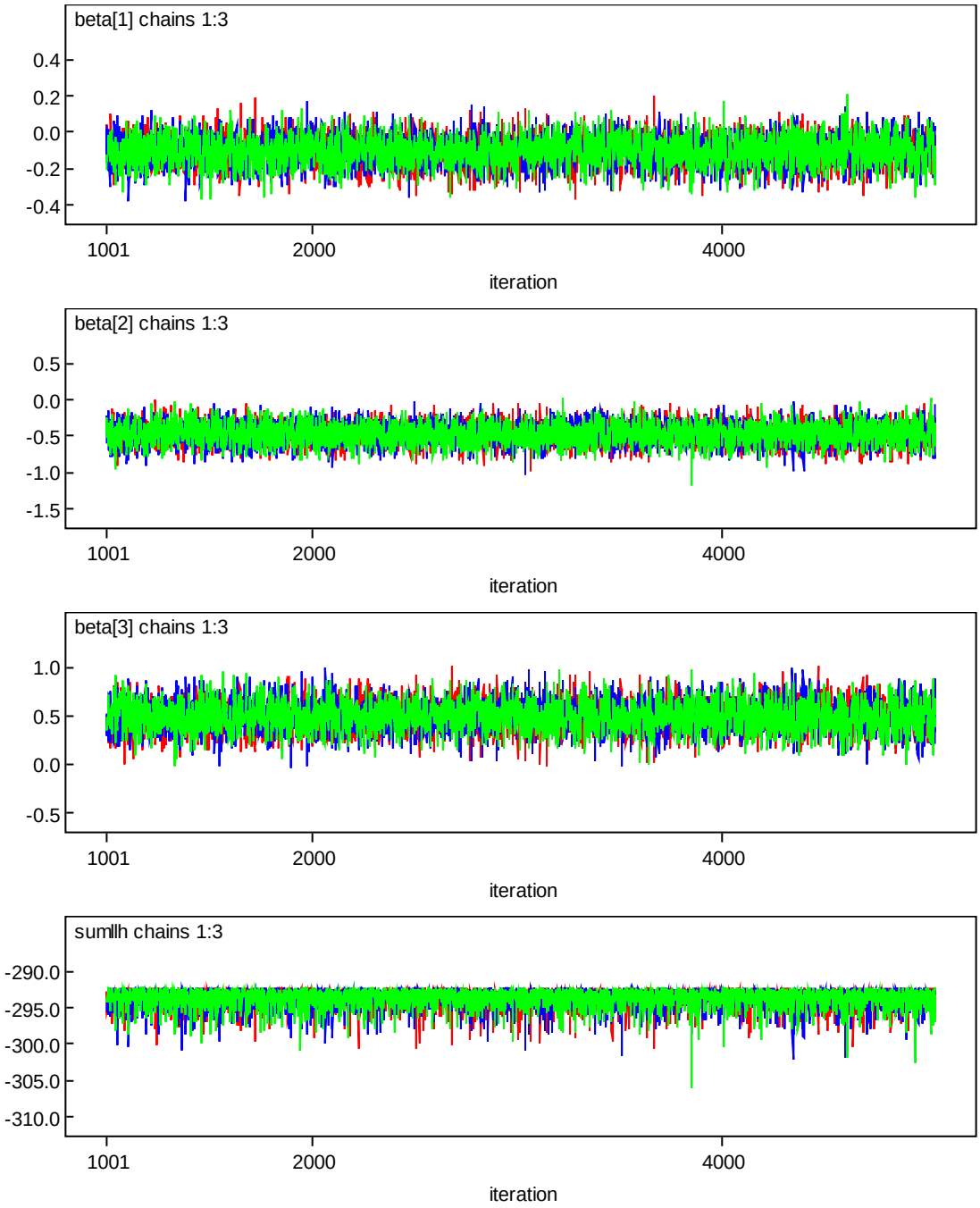








History Plots



Brooks, Gelman, and Rubin Diagnostic

The B-G-R Diagnostic requires running multiple chains and it is based upon a comparison of between and within variance of the multiple chains. Note that the chains must start from very different initial values! Let m=# of chains and n=# of iterations. The Within and Between formulas are:

$$\text{Within chain variance } W = \frac{1}{m(n-1)} \sum_{j=1}^{m} \sum_{i=1}^{n} \left(\theta_j^i - \bar{\theta}_j \right)^2$$

$$\text{Between chain variance } B = \frac{n}{m-1} \sum_{j=1}^{m} \left(\theta_j - \bar{\theta} \right)^2$$

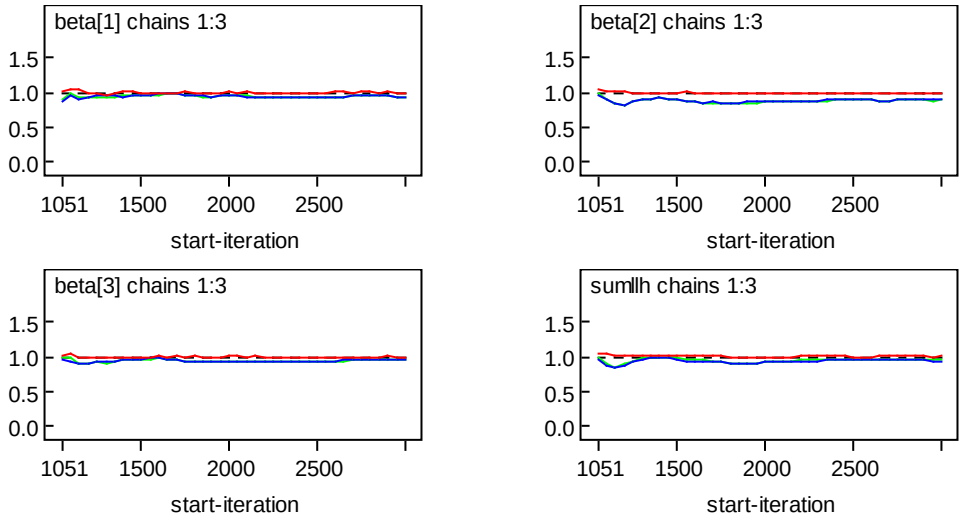
And the overall variance is:

$$\text{Estimated variance } \hat{V}(\theta) = \left(1 - \frac{1}{n}\right) W + \frac{1}{n} B$$

And the Gelman-Rubin Statistic is:

$$\text{The Gelman-Rubin Statistic } \sqrt{R} = \sqrt{\frac{\hat{V}(\theta)}{W}}$$

In the graphs below the Green Line is the width of the central interval constructed from the pooled runs (all widths are normalized so that the maximum value is 1). The Blue line is the average width of the 80% intervals constructed from each run. The Red line is R.



Once convergence is reached, W and V(θ) (within and overall) should be about equal because variation within the chains and variations between the chains should coincide, so R should be about equal to 1.

R General Linear Model and Optimizer Function

```
#
#
# House_105_example.r -- GLM and OPTIM Examples
#
#
library(MASS)
#
#
#
# ****************************
#  fr is called by optim
# ****************************
#
fr <- function(beta){
lambda <- NULL
vaguevariance <- 1.0
lambda[1] <- beta[1]
lambda[2] <- beta[2]
lambda[3] <- beta[3]
#
i <- 1
logL <- 0.0
while (i <= nrow) {
#
#  Calculate "1" and "0" probabilities
#
       sum <- lambda[1] + lambda[2]*TT[i,4]+lambda[3]*TT[i,2]
```

```
       pyes <- pnorm(sum)
       pno <- 1.0 - pyes
       logL <- logL + TT[i,3]*log(pyes) + (1 - TT[i,3])*log(pno)

   i <- i + 1
   }
#return(-logL)
priorbetas <- -(lambda[1]*lambda[1])/(2*vaguevariance) -
(lambda[2]*lambda[2])/(2*vaguevariance)- (lambda[3]*lambda[3])/(2*vaguevariance)
#
#  Multiply Likelihood Distribution by Priors.  Note that
#  the priors on the betas are
#  normals with variance = 1000 so they wash out!
#
return(-logL-priorbetas)
}
#
#
#  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#  Read in data from STATA output
#  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#
rcx.file <- "c:/docs_bayesian_statistics/h105_example.raw"
#
# Standard fields and their widths
#
rcx.fields <- c("dwnom1n","dwnom2n","partydum","southdum")
rcx.fieldWidths <- c(9,11,11,11)
#
# Input File
#
TT <- read.fwf(file=rcx.file,widths=rcx.fieldWidths,as.is=TRUE,col.names=rcx.fields)
dim(TT)
nrow <- length(TT[,1])
ncol <- length(TT[1,])
nparam <- ncol - 1
#
#                 STATA OUTPUT FOR REFERENCE
#  &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
#  TT[,1]          x1 |       443    .0674718    .4179452       -.69       1.44
#  TT[,2]          x2 |       443   -.0308804    .4962939      -1.38       1.49
#  TT[,3]    partydum |       443    .5214447    .5001047          0          1
#  TT[,4]    southdum |       443    .3092551    .4627091          0          1
#
#  . probit partydum southdum x2
#
#  Iteration 0:   log likelihood = -306.65663
#  Iteration 1:   log likelihood = -298.38522
#  Iteration 2:   log likelihood = -298.37588
#
#  Probit regression                                 Number of obs   =        443
#                                                    LR chi2(2)      =      16.56
#                                                    Prob > chi2     =     0.0003
#  Log likelihood = -298.37588                       Pseudo R2       =     0.0270
#
#  ------------------------------------------------------------------------------
#      partydum |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
#  -------------+----------------------------------------------------------------
#      southdum |   .4938932   .1455113     3.39   0.001     .2086964      .77909
#            x2 |  -.4628059   .1362891    -3.40   0.001    -.7299277   -.1956841
#         _cons |  -.1111626   .0757394    -1.47   0.142    -.2596092    .0372839
#  ------------------------------------------------------------------------------
```

```
#
probit105 <-glm(TT[,3] ~ TT[,4]+TT[,2],family=binomial(link=probit))
sumprobit105 <- summary(probit105)
#
#  ---- Useful Commands To See What is in an Object
#
#> length(sumprobit105)
#[1] 17
#> class(sumprobit105)
#[1] "summary.glm"
#> names(sumprobit105)
# [1] "call"           "terms"          "family"         "deviance"
# [5] "aic"            "contrasts"      "df.residual"    "null.deviance"
# [9] "df.null"        "iter"           "deviance.resid" "coefficients"
#[13] "aliased"        "dispersion"     "df"             "cov.unscaled"
#[17] "cov.scaled"
#
#  &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
#     USE OPTIM TO GET PARAMETERS OF PROBIT
#
# *** Calculate Log-Likelihood ***
#
beta <- NULL
beta[1] <- 1.0
beta[2] <- 1.0
beta[3] <- 1.0
#
# ****************************************
#  DO MAXIMUM LIKELIHOOD MAXIMIZATION HERE
# ****************************************'
#
model <- optim(beta,fr,hessian=TRUE)
#model <- optim(c(1.00,1.00,1.0),fr,hessian=TRUE)
#
#  Log-Likelihood (inverse -- optim minimizes!!)
#
logLmax <- model$value
#
#  Parameter Estimates
#
betamax <- model$par
#
# convergence an integer code.
#     0 indicates successful convergence.
#
nconverge <- model$convergence
#
# counts
#  A two-element integer vector giving the number of calls to
#    fn (function) and gr (gradients -- optional) respectively.
ncounts <- model$counts
#
xhessian <- model$hessian
#
#  Perform Eigenvalue-Eigenvector Decomposition of Hessian Matrix
#
ev <- eigen(xhessian)
#
#  The Two Lines Below Put the Eigenvalues in a
#    Diagonal Matrix -- The first one creates an
#    identity matrix and the second command puts
#    the singular values on the diagonal
```

```
#
Lambda <- diag(nparam)
diag(Lambda) <- ev$val
#
#  Compute U*LAMBDA*U' for check below
#
XX <- ev$vec %*% Lambda %*% t(ev$vec)
#
# Compute Fit of decomposition -- This is just the sum of squared
#  error -- Note that ssesvd should be zero!
#
i <- 0
j <- 0
sseeig <- 0
while (i < nparam) {
  i <- i + 1
  j <- 0
  while (j < nparam) {
     j <- j + 1
     sseeig <- sseeig + (xhessian[i,j] - XX[i,j])**2
  }
}
#
LambdaInv <- diag(nparam)
diag(LambdaInv) <- 1/ev$val
#
#  Compute U*[(LAMBDA)-1]*U' for check below
#
XXInv <- ev$vec %*% LambdaInv %*% t(ev$vec)
#
results <- rep(0,nparam*4)
dim(results) <- c(nparam,4)
#
results[,1] <- betamax
results[,2] <- sqrt(diag(XXInv))
results[,3] <- betamax/sqrt(diag(XXInv))
results[,4] <- pt(-abs(results[,3]),nrow-nparam-1)*2
#
```

R OUTPUT: HERE IS THE OUTPUT OF THE GLM FUNCTION:

```
> summary(model)
> sumprobit105
```

THIS JUST GIVES YOU THE FORMULA

```
Call:
glm(formula = TT[, 3] ~ TT[, 4] + TT[, 2], family = binomial(link = probit))
```

THIS SUMMARIZES the "DEVIANCE" – corresponds to the sum of squares in linear normal models

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6650  -1.1863   0.8572   1.1174   1.5439
```

THE COEFFICIENTS IN THE USUAL FORMAT (THESE ARE THE SAME AS STATA):

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.11116    0.07584  -1.466 0.142737
TT[, 4]      0.49390    0.14483   3.410 0.000649 ***
TT[, 2]     -0.46281    0.13528  -3.421 0.000623 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
```

Null Deviance is the deviation of a model that contains only the intercept term, that is, a fixed probability for all observations. In this instance:

-2*231*[LOG(231/443)]-2*212*[LOG(212/443)] = 300.83 + 312.48 = 613.31

Residual Deviance corresponds to the residual sum of squares in OLS which is used to estimate the standard deviation around the regression line. Here is is simply:
-2*[LOG LIKELIHOOD] = -2*(-298.37588)=596.75

```
    Null deviance: 613.31  on 442  degrees of freedom
Residual deviance: 596.75  on 440  degrees of freedom
AIC: 602.75
```

The Akaike Information Criterion (AIC) is:
602.75 = -2*(LOG LIKELIHOOD)+ 2*K = -2*(-298.37588)+2*3

Where k=# of betas and the LOG LIKELIHOOD is given by STATA above.

```
Number of Fisher Scoring iterations: 3
```

This is simply the number of iterations to estimate the model. Note that it is the same as the number of iterations in STATA.

```
>
```

R OUTPUT: HERE IS THE OUTPUT OF THE SUMMARY OF OPTIM:

```
> summary(model)
            Length Class  Mode
par         3      -none- numeric
value       1      -none- numeric
counts      2      -none- numeric
convergence 1      -none- numeric
message     0      -none- NULL
hessian     9      -none- numeric
>
> model$par
[1] -0.1060596  0.4791577 -0.4500236
> model$value
[1] 298.6042
> model$counts
function gradient
     128       NA
> model$convergence
[1] 0
> model$message
NULL
> model$hessian
           [,1]     [,2]      [,3]
[1,] 276.819442 83.19293 -8.827006
[2,]  83.192933 84.19293 24.061936
[3,]  -8.827006 24.06194 67.535112
>
```

Solution From OPTIM:

```
> results
            [,1]       [,2]      [,3]         [,4]
Constant [1,] -0.1060596 0.07514533 -1.411393 0.1588371538
Southdum [2,]  0.4791577 0.14347464  3.339668 0.0009103595
X2       [3,] -0.4500236 0.13459983 -3.343419 0.0008984467
>
```

Check on the Accuracy of the Inverse of the Hessian:

```
> sseeig
[1] 1.746854e-26
```