Applied Bayesian Analysis : NCSU ST 540

Homework 6

Bruce Campbell

For this problem we use the 2016 election data described at https://www4.stat.ncsu.edu/~reich/ABA/code/election2016data with data provided in the R workspace https://www4.stat.ncsu.edu/~reich/ABA/code/election_2008_2016.RData

We restrict our analysis to the counties in North and South Carolina. In JAGS, we fit the logistic regression model

$$P(Z_i = 1) = \frac{1}{1 + e^{\beta_0 + \sum_{j=1}^{p} \beta_j X_{i\,j}}}$$

where Z_i is the binary indicator that GOP support in county i increased by at least 5% from 2012 to 2016 $Z_i = 1 : Yi > 5$ and $Z_i = 0$ otherwise, where Y is the change variable in the R workspace. X_ij are the covariates in the R workspace. We standardize each covariate to have mean zero and variance one before fitting the model. The priors are $\beta_j \sim N(0, \tau^2)$

(1) Fit the model with $\tau = 1$ and $\tau = 100$

First we make some notes on the data and the preprocessing steps.

```
https://www.kaggle.com/benhamner/2016-us-election/data
demographic data on counties from US census
3195 rows and 54 columns.
```

The metadata in the county facts table is located in a dictionary. We might need this for interpretation of the coefficients.

```
county_facts_dictionary.csv
description of the columns in county_facts
https://www.kaggle.com/benhamner/2016-us-election/data
```

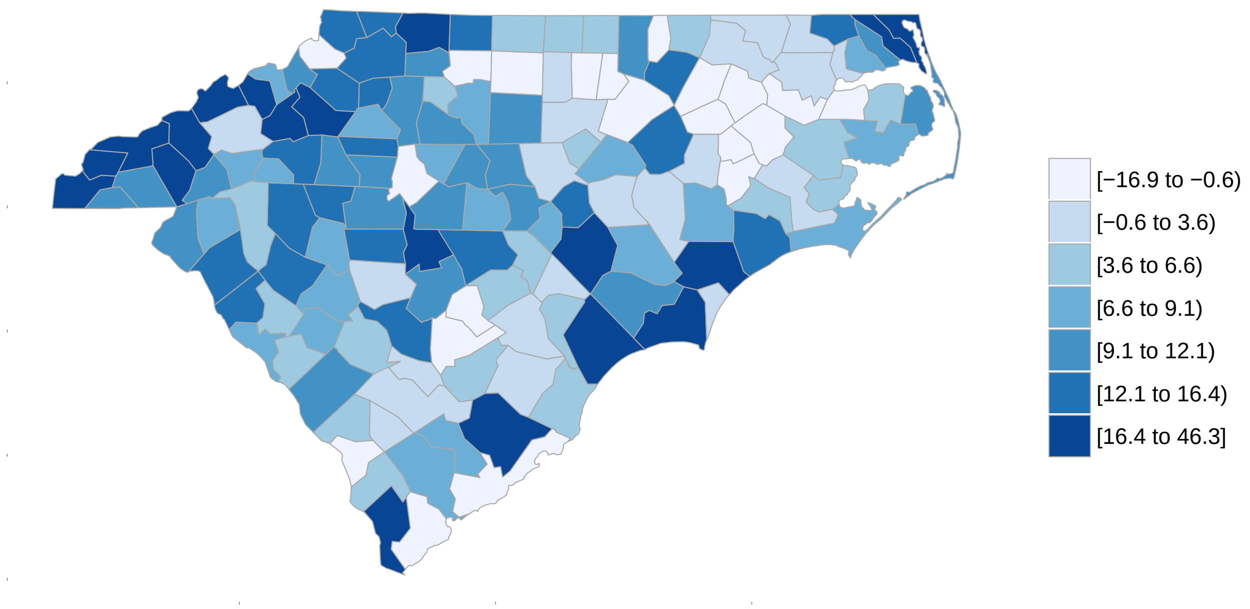
The election data

```
County_Election_08_16.csv
https://www4.stat.ncsu.edu/~reich/ABA/code/election2016data
county-level voting patterns in the 2016 Presidential elections
3112 rows and 14 columns
```

The county data is joined to the election data via the key fips_code. We load the processed data below and start our analysis with the joined dataframe all_dat.

```r
library(rjags)
library(coda)
library(choroplethr)
library(modeest)
load("election_2008_2016.RData")
carolinas <- all_dat[all_dat$state_abbreviation ==
    "NC" | all_dat$state_abbreviation ==
    "SC", ]
gop.percent.increase <- 100 * (carolinas$gop_2016 -
    carolinas$gop_2012)/carolinas$gop_2012
gop.percent.increase.gt.5 <- gop.percent.increase >
    5
# Borrowed from instructor codebase to
# create our predictors and reposenses
# for the carolinas data set.
fips <- carolinas[, 1]
Y <- round(100 * (carolinas$gop_2016 -
    carolinas$gop_2012)/carolinas$gop_2012,
    1)
Z <- Y > 5
these <- c(3, 7, 10, 15, 20, 21, 25, 27,
    31, 32, 47, 51, 22, 44:45)
X <- as.matrix(carolinas[, these + 15])
X <- scale(X)
names <- dict[these, ]
colnames(X) <- names[, 1]
county_plot <- function(fips, Y, main = "",
    units = "") {
    temp <- as.data.frame(list(region = fips,
        value = Y))
    # county_choropleth(temp,title=main,legend=units)
    county_choropleth(temp, title = main,
        county_zoom = fips)
}
county_plot(fips, Y, "Percent change in GOP support from 2012 to 2016",
    unit = "Percent increase")
```

Percent change in GOP support from 2012 to 2016



Fit with $\tau = 1$

```
n.chains <- 1
DEBUG <- FALSE
if(DEBUG)
{
nSamples <- 20000
} else
{
nSamples <- 20000
}
n <- nrow(X)
tau <- 1
p <- ncol(X)
logistic_model <- "model{
# Likelihood
for(i in 1:n){
Z[i] ~ dbern(q[i])
logit(q[i]) <- intercept +inprod(X[i,],beta[])
}
#Priors
```

```
intercept ~ dnorm(0,tau)
for(j in 1:p){
beta[j] ~ dnorm(0,tau)
}
}"
model.carolinas <- jags.model(textConnection(logistic_model), data = list(Z=Z,X=X,n=n,p=p,tau=
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 146
##    Unobserved stochastic nodes: 16
##    Total graph size: 2941
##
## Initializing model
```

```
update(model.carolinas, nSamples, progress.bar="none"); # Burnin
samp.coeff <- coda.samples(model.carolinas, variable.names=c("intercept","beta"),n.iter=2*nSamp
```

Fit with $\tau = 100$

```
tau <- 100
model.carolinas.uninformative <- jags.model(textConnection(logistic_model), data = list(Z=Z,X=
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 146
##    Unobserved stochastic nodes: 16
##    Total graph size: 2941
##
## Initializing model
```

```
update(model.carolinas.uninformative, nSamples, progress.bar="none"); # Burnin
samp.coeff.uninformative <- coda.samples(model.carolinas.uninformative, variable.names=c("inte
```

(2) Assess convergence of the sampler for both priors.

In this section we sample from our model after burn in. Although all of the plots are not presented we assesed convergence by; - viewing the time sereies for the intercept and each of the predictors. For this we utilized the coda package. - ran multiple chains and viewed evaluated the autocorrelation plots. - calculated the posterior means for the intercept and the

j - utilized the mlv funtions in the modeest to calculate the MAP estimated of the posterior modes - we fit a frequentist model an evaluated the estimated coefficients against the posterior means and modes - compared the 95% prediction intervals for the intercepts against the p-values from the logistic regression maximum likelihood model

Code for this is below, we run some of it conditionlally though the DEBUG variable.

We did run the model without standardizing the feature data and noted evidence that the chain might be experienceing convergence issues. There was significant autocorrelation of the chains when the data was not standardized.

$\tau = 1$ **Posterior quantiles**

```r
summary(samp.coeff)
```

```
##
## Iterations = 21001:61000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 40000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean     SD Naive SE Time-series SE
## beta[1]    1.0408 0.4795 0.002398       0.006819
## beta[2]   -0.6735 0.3614 0.001807       0.004439
## beta[3]   -1.8571 0.3924 0.001962       0.004742
## beta[4]   -0.7579 0.2937 0.001469       0.003720
## beta[5]   -0.3584 0.4846 0.002423       0.007755
## beta[6]   -1.3122 0.5644 0.002822       0.009472
## beta[7]    0.8543 0.3977 0.001989       0.004886
## beta[8]    0.1582 0.5663 0.002832       0.010244
## beta[9]   -0.4906 0.6228 0.003114       0.011812
## beta[10]  -0.3019 0.4808 0.002404       0.007729
## beta[11]  -0.2552 0.3347 0.001674       0.003444
## beta[12]  -0.2598 0.5901 0.002950       0.006962
## beta[13]   0.1869 0.5528 0.002764       0.007206
## beta[14]   0.1025 0.4971 0.002485       0.005765
## beta[15]  -0.6462 0.5914 0.002957       0.006879
## intercept  0.9763 0.2688 0.001344       0.002283
##
## 2. Quantiles for each variable:
##
##               2.5%     25%     50%      75%    97.5%
## beta[1]    0.13120  0.7108  1.0293  1.35505  2.01150
## beta[2]   -1.39078 -0.9164 -0.6708 -0.42669  0.02598
## beta[3]   -2.65851 -2.1129 -1.8459 -1.58974 -1.11376
## beta[4]   -1.33816 -0.9545 -0.7554 -0.56113 -0.18577
## beta[5]   -1.32575 -0.6767 -0.3564 -0.03475  0.58604
## beta[6]   -2.43483 -1.6885 -1.3085 -0.92882 -0.21926
## beta[7]    0.08282  0.5841  0.8517  1.11840  1.64232
## beta[8]   -0.96550 -0.2230  0.1577  0.53499  1.27152
## beta[9]   -1.71970 -0.9036 -0.4876 -0.07393  0.72904
## beta[10]  -1.23583 -0.6257 -0.3053  0.01622  0.65662
## beta[11]  -0.91377 -0.4802 -0.2512 -0.03075  0.39535
## beta[12]  -1.46637 -0.6482 -0.2399  0.14407  0.85474
## beta[13]  -0.91966 -0.1840  0.1977  0.56543  1.24668
## beta[14]  -0.90124 -0.2302  0.1129  0.44393  1.04928
## beta[15]  -1.86923 -1.0330 -0.6241 -0.23940  0.44953
## intercept  0.45849  0.7943  0.9723  1.15480  1.51474
```

$\tau = 1$ **Sample again and estimate the mean and MAP mode of the posterior dostributions.**

```
samp.coeff.jags <- jags.samples(model.carolinas,
    variable.names = c("intercept", "beta"),
    n.iter = nSamples, progress.bar = "none")
posterior_means <- lapply(samp.coeff.jags,
    apply, 1, "mean")
pander(posterior_means, caption = "posterior means second sample")
```

- **beta**: *1.049*, *-0.6789*, *-1.863*, *-0.7704*, *-0.3932*, *-1.29*, *0.8376*, *0.1557*, *-0.5126*, *-0.3495*, *-0.2556*, *-0.2784*, *0.1817*, *0.1081* and *-0.6509*
- **intercept**: *0.9806*

```
posterior_modes <- lapply(samp.coeff.jags,
    apply, 1, "mlv")
posterior_modes
```

```
## $beta
## $beta[[1]]
## Mode (most likely value): 1.015403
## Bickel's modal skewness: 0.0431
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[2]]
## Mode (most likely value): -0.6699021
## Bickel's modal skewness: -0.0074
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[3]]
## Mode (most likely value): -1.806363
## Bickel's modal skewness: -0.0905
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[4]]
## Mode (most likely value): -0.7649989
## Bickel's modal skewness: -0.0086
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[5]]
## Mode (most likely value): -0.4030568
## Bickel's modal skewness: 0.0241
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[6]]
## Mode (most likely value): -1.277611
## Bickel's modal skewness: -0.0067
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[7]]
## Mode (most likely value): 0.8242716
## Bickel's modal skewness: 0.0182
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[8]]
## Mode (most likely value): 0.1648691
## Bickel's modal skewness: -0.0063
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[9]]
## Mode (most likely value): -0.4921973
## Bickel's modal skewness: -0.0292
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[10]]
## Mode (most likely value): -0.3130962
## Bickel's modal skewness: -0.0625
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[11]]
## Mode (most likely value): -0.2426614
## Bickel's modal skewness: -0.0151
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[12]]
## Mode (most likely value): -0.2866481
## Bickel's modal skewness: 0.0285
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[13]]
## Mode (most likely value): 0.1967491
## Bickel's modal skewness: -0.0179
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

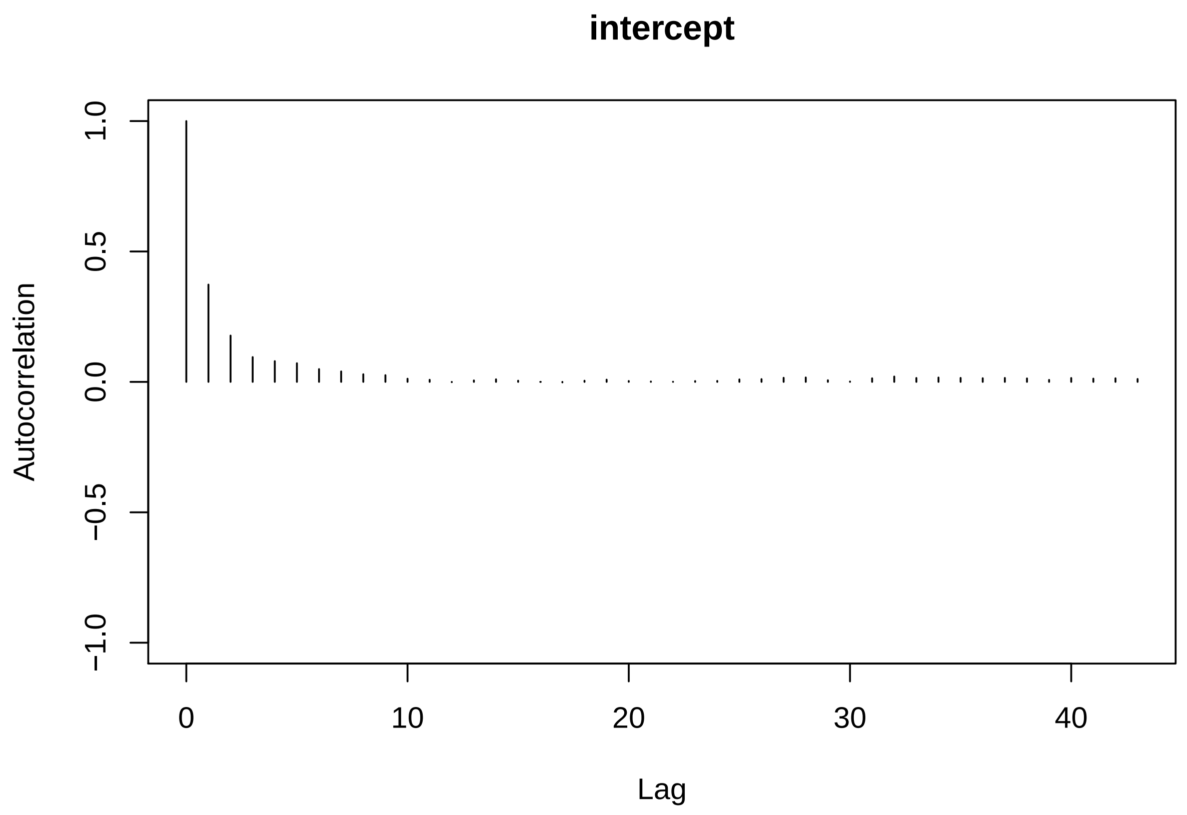
```
##
## $beta[[14]]
## Mode (most likely value): 0.1165615
## Bickel's modal skewness: 0.0082
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[15]]
## Mode (most likely value): -0.598256
## Bickel's modal skewness: -0.0511
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
##
## $intercept
## $intercept[[1]]
## Mode (most likely value): 0.9459394
## Bickel's modal skewness: 0.0869
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

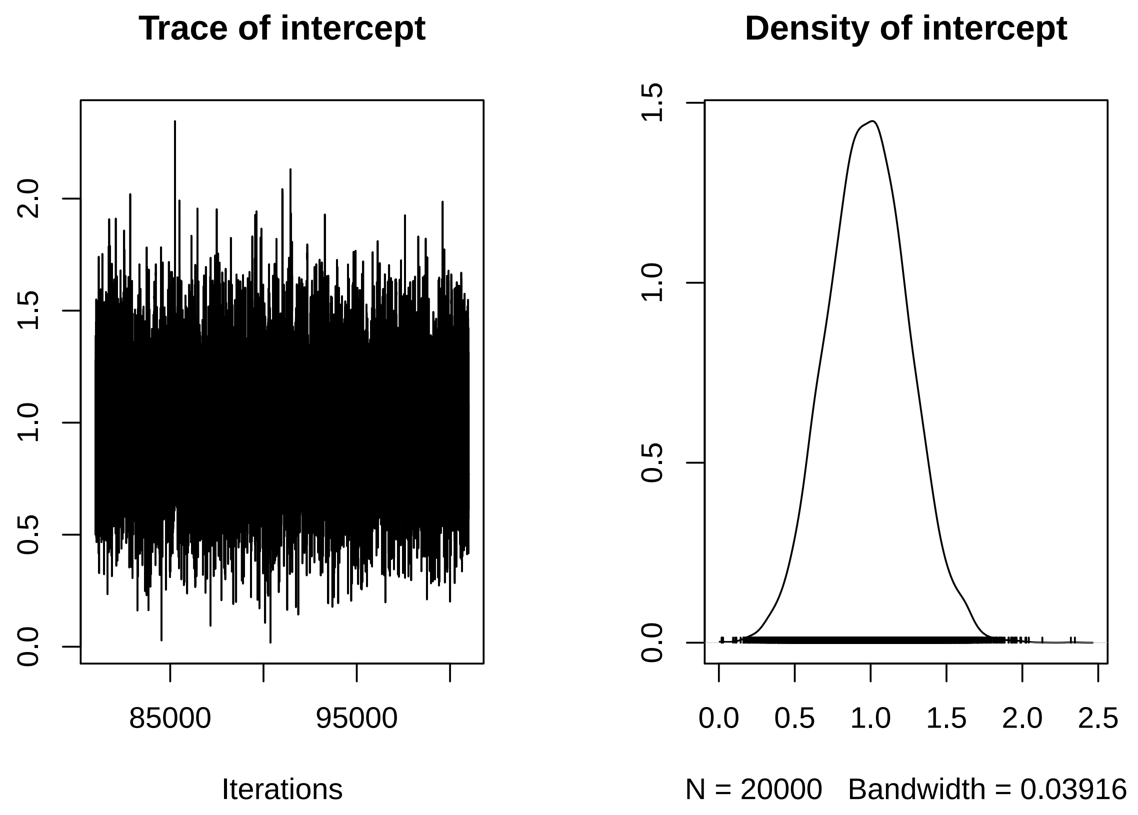
$\tau = 1$ Plot the time series, empirical posterior distribution, and the autocoerrelation

fucntion for the coefficients

We only plot the intercept for the final report. Set the DEBUG flag to TRUE in order to include all of the coefficients.

```r
if (DEBUG) {
    for (i in 1:p) {
        samp.coeff <- coda.samples(model.carolinas,
            variable.names = c(paste("beta[",
                i, "]", sep = "")), n.iter = nSamples,
            progress.bar = "none")
        autocorr.plot(samp.coeff)
        plot(samp.coeff)
    }
    samp.coeff <- coda.samples(model.carolinas,
        variable.names = "intercept", n.iter = nSamples,
        progress.bar = "none")
    autocorr.plot(samp.coeff)
    plot(samp.coeff)
} else {
    samp.coeff <- coda.samples(model.carolinas,
        variable.names = "intercept", n.iter = nSamples,
        progress.bar = "none")
    autocorr.plot(samp.coeff)
    plot(samp.coeff)
}
```


intercept
Autocorrelation
Lag
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40



$\tau = 100$ Posterior quantiles

```
summary(samp.coeff.uninformative)
```

```
##
## Iterations = 21001:61000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 40000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean      SD  Naive SE Time-series SE
## beta[1]    0.063605 0.09099 0.0004549      0.0006229
## beta[2]    0.008030 0.08982 0.0004491      0.0005844
## beta[3]   -0.227159 0.08925 0.0004462      0.0005869
## beta[4]   -0.027618 0.08812 0.0004406      0.0005712
## beta[5]   -0.020652 0.09287 0.0004644      0.0006559
## beta[6]   -0.102612 0.09264 0.0004632      0.0006474
## beta[7]    0.175155 0.09008 0.0004504      0.0006023
## beta[8]   -0.005129 0.09255 0.0004628      0.0006413
## beta[9]    0.002410 0.09273 0.0004637      0.0006409
## beta[10]  -0.084911 0.09173 0.0004587      0.0006570
## beta[11]  -0.063141 0.08888 0.0004444      0.0005802
## beta[12]  -0.075420 0.09435 0.0004718      0.0006501
## beta[13]  -0.052609 0.09296 0.0004648      0.0006348
## beta[14]  -0.057473 0.09215 0.0004607      0.0006045
## beta[15]  -0.082264 0.09287 0.0004644      0.0006230
## intercept  0.153660 0.08642 0.0004321      0.0005424
##
## 2. Quantiles for each variable:
##
##                 2.5%       25%       50%       75%    97.5%
## beta[1]   -0.1138879  0.001723  0.063437  0.125255  0.24082
## beta[2]   -0.1672744 -0.052260  0.007721  0.068294  0.18525
## beta[3]   -0.4027000 -0.287571 -0.227272 -0.166979 -0.05182
## beta[4]   -0.2007425 -0.087089 -0.027883  0.032415  0.14477
## beta[5]   -0.2025705 -0.084081 -0.020521  0.041899  0.16273
## beta[6]   -0.2843156 -0.164683 -0.102124 -0.040463  0.07945
## beta[7]   -0.0003341  0.114121  0.174991  0.235795  0.35183
## beta[8]   -0.1862736 -0.067812 -0.005152  0.057240  0.17638
## beta[9]   -0.1781511 -0.059833  0.001899  0.064998  0.18269
## beta[10]  -0.2644380 -0.146995 -0.085034 -0.023101  0.09546
## beta[11]  -0.2371333 -0.123824 -0.063141 -0.002858  0.10946
## beta[12]  -0.2610721 -0.138967 -0.074865 -0.011401  0.10888
## beta[13]  -0.2357536 -0.115240 -0.052236  0.010253  0.12999
## beta[14]  -0.2381314 -0.119554 -0.057207  0.004369  0.12491
## beta[15]  -0.2640671 -0.144673 -0.081751 -0.020377  0.10067
## intercept -0.0145893  0.095173  0.153496  0.211451  0.32426
```

$\tau = 100$ **Sample again and estimate the mean and MAP mode of the posterior dostributions.**

```r
samp.coeff.jags.uninformative <- jags.samples(model.carolinas.uninformative,
    variable.names = c("intercept", "beta"),
    n.iter = 2 * nSamples, progress.bar = "none")
posterior_means.uninformative <- lapply(samp.coeff.jags.uninformative,
    apply, 1, "mean")
pander(posterior_means.uninformative, caption = "posterior_means.uninformative")
```

- **beta**: *0.06403*, *0.01002*, *-0.2267*, *-0.02848*, *-0.01999*, *-0.1034*, *0.1738*, *-0.006085*, *0.003423*, *-0.08442*, *-0.06289*, *-0.0762*, *-0.05313*, *-0.05755* and *-0.08132*
- **intercept**: *0.1537*

```r
posterior_means.uninformative <- lapply(samp.coeff.jags.uninformative,
    apply, 1, "mlv")
posterior_means.uninformative
```

```
## $beta
## $beta[[1]]
## Mode (most likely value): 0.06397314
## Bickel's modal skewness: -0.0028
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[2]]
## Mode (most likely value): 0.00950086
## Bickel's modal skewness: 0.00405
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[3]]
## Mode (most likely value): -0.2257998
## Bickel's modal skewness: -0.0034
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[4]]
## Mode (most likely value): -0.0301949
## Bickel's modal skewness: 0.01295
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[5]]
## Mode (most likely value): -0.02017456
## Bickel's modal skewness: 0.00105
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[6]]
## Mode (most likely value): -0.106469
## Bickel's modal skewness: 0.0275
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[7]]
## Mode (most likely value): 0.1749479
## Bickel's modal skewness: -0.0141
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[8]]
## Mode (most likely value): -0.0002385341
## Bickel's modal skewness: -0.05055
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[9]]
## Mode (most likely value): 0.005033418
## Bickel's modal skewness: -0.01575
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[10]]
## Mode (most likely value): -0.08611902
## Bickel's modal skewness: 0.0171
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[11]]
## Mode (most likely value): -0.06720499
## Bickel's modal skewness: 0.03665
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[12]]
## Mode (most likely value): -0.0748319
## Bickel's modal skewness: -0.01135
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[13]]
## Mode (most likely value): -0.05042434
## Bickel's modal skewness: -0.02255
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[14]]
## Mode (most likely value): -0.05612208
## Bickel's modal skewness: -0.014
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[15]]
## Mode (most likely value): -0.07795454
## Bickel's modal skewness: -0.0291
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
##
## $intercept
## $intercept[[1]]
## Mode (most likely value): 0.1552686
## Bickel's modal skewness: -0.01975
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

Fit frequentist logistic model for reference.

```
df <- data.frame(cbind(Z, X))
lm.logistic <- glm(Z ~ ., family = binomial,
    df)
summary(lm.logistic)
```

```
##
## Call:
## glm(formula = Z ~ ., family = binomial, data = df)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.1986  -0.4521   0.1467   0.4048   3.0622
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.22344    0.33351   3.668 0.000244 ***
## PST120214    1.44242    0.68408   2.109 0.034984 *
## AGE775214   -0.81589    0.46258  -1.764 0.077771 .
## RHI225214   -2.05358    0.48616  -4.224  2.4e-05 ***
## RHI725214   -1.00849    0.37201  -2.711 0.006709 **
## EDU635213   -0.55048    0.64123  -0.858 0.390634
## EDU685213   -1.99593    0.88355  -2.259 0.023883 *
## HSG445213    0.96082    0.50725   1.894 0.058199 .
## HSG495213    0.87243    0.81258   1.074 0.282975
## INC110213   -0.97476    0.91641  -1.064 0.287475
## PVY020213   -0.36784    0.64185  -0.573 0.566582
## RTN131207   -0.44836    0.41945  -1.069 0.285097
## POP060210   -0.03261    0.76264  -0.043 0.965888
## VET605213    0.46278    0.81324   0.569 0.569321
## MAN450207    0.46521    0.69222   0.672 0.501550
## WTN220207   -1.01084    0.93321  -1.083 0.278726
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 190.144  on 145  degrees of freedom
## Residual deviance:  93.976  on 130  degrees of freedom
## AIC: 125.98
##
## Number of Fisher Scoring iterations: 6
```