Bayesian Inference/ Complaints Prediction

August 19, 2017

Objective:

To demonstrate the use of Bayesian Analysis with MCMC (Monte Carlo Markov Chain), using an example where the likelihood of complaints is to be predicted. To compare with the classical method, the same is also performed at the end. In both cases, the Logistic Regression approach is used.
The problem and the solution approach are explained with a technical audience in mind, using code comments and explanatory calls, like summary(). Look for these key components of Bayesian Analysis in the code below: Model, Likelihood, Logit function, Priors, Data and Parameter Initialization.

```
#Libraries
library(pROC)
library(caret)
library(dplyr)

#Data loading and exploration
dat_happiness <-
read.csv("D:/Users/balaks/Documents/Learning/BayseianTrainingHCL/hw2data.csv"
)
head(dat_happiness)

##   Type Pasthapp Responsible Futurehapp FTP complain
## 1    2        2          10          5  12        1
## 2    3       10           5          5  18        1
## 3    1        2          18          3   9        0
## 4    3        5          15          0  15        1
## 5    1        7          17          7  17        1
## 6    1        0          20          5  15        0

summary(dat_happiness)

##       Type          Pasthapp       Responsible      Futurehapp
##  Min.   :1.000   Min.   : 0.000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.:1.000   1st Qu.: 2.000   1st Qu.: 5.00   1st Qu.: 2.000
##  Median :2.000   Median : 4.000   Median :14.00   Median : 5.000
##  Mean   :1.937   Mean   : 4.778   Mean   :11.02   Mean   : 4.254
##  3rd Qu.:3.000   3rd Qu.: 7.000   3rd Qu.:16.00   3rd Qu.: 5.000
##  Max.   :3.000   Max.   :15.000   Max.   :20.00   Max.   :15.000
##       FTP           complain
##  Min.   : 3.00   Min.   :0.0000
##  1st Qu.:10.00   1st Qu.:0.0000
##  Median :13.00   Median :0.0000
```

```
##  Mean   :12.43   Mean   :0.4762
##  3rd Qu.:15.00   3rd Qu.:1.0000
##  Max.   :19.00   Max.   :1.0000
```

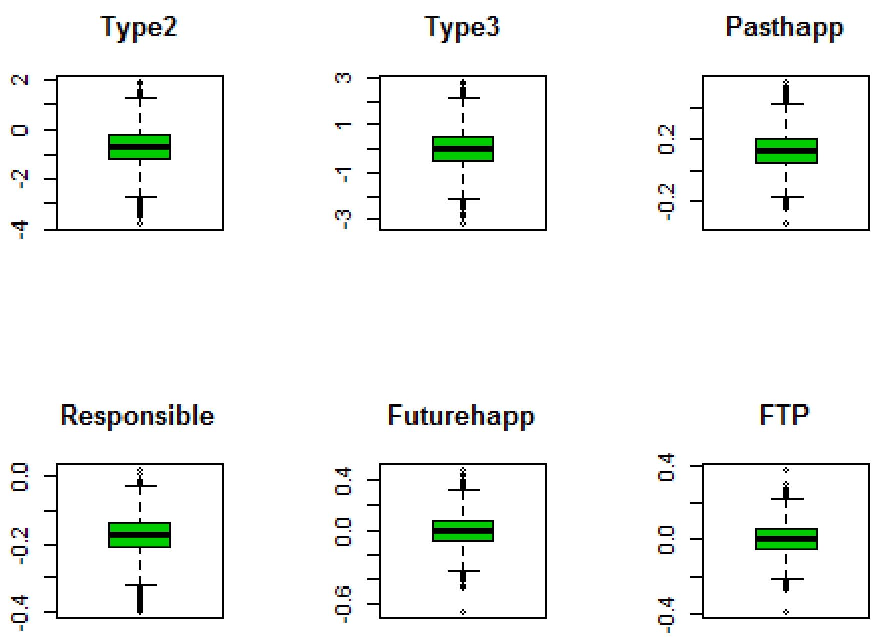
```
str(dat_happiness)
```

```
## 'data.frame':    63 obs. of  6 variables:
##  $ Type       : int  2 3 1 3 1 1 2 3 1 2 ...
##  $ Pasthapp   : int  2 10 2 5 7 0 2 10 8 4 ...
##  $ Responsible: int  10 5 18 15 17 20 20 14 14 6 ...
##  $ Futurehapp : int  5 5 3 0 7 5 5 11 8 5 ...
##  $ FTP        : int  12 18 9 15 17 15 19 7 14 15 ...
##  $ complain   : int  1 1 0 1 1 0 0 0 1 1 ...
```

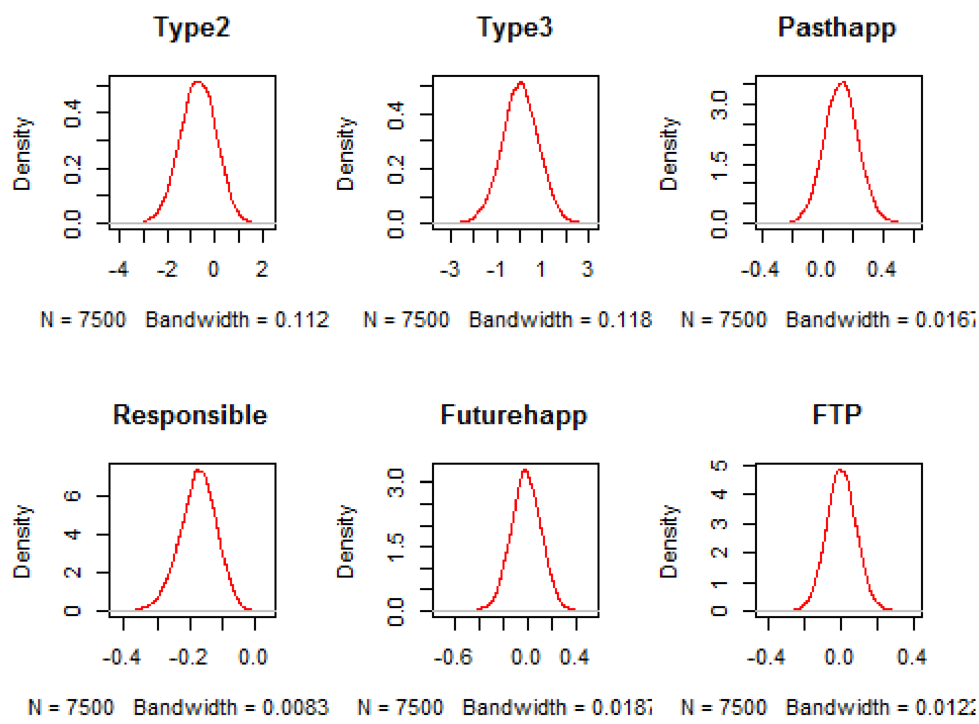
```
#Note that "Type" is converted into a "factor" explanatory variable. The
"complain" dependent variable is also converted to "factor", as expected for
Logistic Regression.



require(R2WinBUGS)
HappPred<-function()
{
  #This portion covers the model
  for (i in 1:n) {
    #Expressing our assumption that the observed complaints, which is the
dependent variable, were generated from a Bernoulli distribution. The
parameter for the distribution of the dependent variable is "p". Note that a
separate value of "p" is assumed for each observation. Hence, p[i] is in a
for loop.
    complain[i] ~ dbern(p[i])
    #Linking  the parameter for the dependent variable to the parameters of
the explanatory variables. Since this is a logistic regression model, the
logit of the parameter for the dependent variable is in the LHS.
    #The parameters for the dependent and independent variables collectively
express the "Likelihood" of the observations.
    logit(p[i]) <- alpha + b.Type2*S2[i] + b.Type3*S3[i] +
b.Pasthapp*Pasthapp[i] + b.Responsible*Responsible[i] +
b.Futurehapp*Futurehapp[i] + b.FTP*FTP[i]

    S2[i]           <- equals(Type[i], 2) #<- ref category:1
    S3[i]           <- equals(Type[i], 3) #<- ref category:1

  }
  #This portion covers the priors. Note that priors are defined only for
parameters of the explanatory variables. The prior parameter of the dependent
variable is arrived at in the logit equation above.
  alpha ~ dnorm(1,1.0E-4) # Prior for intercept
  b.Type2 ~ dnorm(0.0,1.0E-4) # Prior for slope of Type2
  b.Type3 ~ dnorm(0.0,1.0E-4) # Prior for slope of Type3
  b.Pasthapp ~ dnorm(0.0,1.0E-4) # Prior for slope of Pasthapp
```

```
  b.Responsible ~ dnorm(0.0,1.0E-4) # Prior for slope of Responsible
  b.Futurehapp ~ dnorm(0.0,1.0E-4) # Prior for slope of Futurehapp
  b.FTP ~ dnorm(0.0,1.0E-4) # Prior for slope of FTP

}
#Writing BUGS File
HappPred_FILE<- file.path(tempdir(), "HappPred.bug")
write.model(HappPred, HappPred_FILE)
#This portion covers the data or the observations
dat_happiness_list<-list(
  Type=dat_happiness$Type,
  Pasthapp=dat_happiness$Pasthapp,
  Responsible=dat_happiness$Responsible,
  Futurehapp=dat_happiness$Futurehapp,
  FTP=dat_happiness$FTP,
  complain=dat_happiness$complain,n=dim(dat_happiness)[1])
Type = factor(dat_happiness_list$Type, levels=c(0,1))
complain = factor(dat_happiness_list$complain, levels=c(0,1))


k=length(dat_happiness$Futurehapp)
ns=50000  #Number of simulations in BUGS
inits <- function()
{
  #This portion covers the parameters initialization

  list(alpha=0.5,b.Type2=0.5,b.Type3=0.5,b.Pasthapp=0.5, b.Responsible=0.5,
b.Futurehapp=0.5, b.FTP=0.5)
  list(alpha=0, b.Type2=1, b.Type3=1, b.Pasthapp=1, b.Responsible=1,
b.Futurehapp=1, b.FTP=1)
  list(alpha=-0.3, b.Type2=0.5,b.Type3=0.5,b.Pasthapp=0.5, b.Responsible=1,
b.Futurehapp=1, b.FTP=1)

}
#Finding the values of these parameters is the objective of exercise.
parameters <- c("alpha", "b.Type2", "b.Type3", "b.Pasthapp", "b.Responsible",
"b.Futurehapp", "b.FTP")
#The MCMC simulation below, simulates values of the above parameters, given
initial values and the prior distribution. Depending on how well such
simulated parameter values match the observed data, the simulated parameter
value is either accepted or rejected. Accepted values are then thinned to
prevent auto-correlation. Multiple chains, initialized with different values,
and the large number of simulations makes it more likely that the desired
"stationary" distribution is reached.
post_HappPred.sim <- bugs(dat_happiness_list, inits, parameters,
model.file=HappPred_FILE,n.chains=3, n.thin=10,
n.iter=ns,bugs.directory="C:/Program Files
(x86)/winbugs14_unrestricted/WinBUGS14/",debug =TRUE,digits=3)
```

```
mmm=post_HappPred.sim$sims.matrix

#######################################
#windows()
par(mfrow=c(2,3))
boxplot(mmm[,2],col=3,main="Type2")
boxplot(mmm[,3],col=3,main="Type3")
boxplot(mmm[,4],col=3,main="Pasthapp")
boxplot(mmm[,5],col=3,main="Responsible")
boxplot(mmm[,6],col=3,main="Futurehapp")
boxplot(mmm[,7],col=3,main="FTP")
```



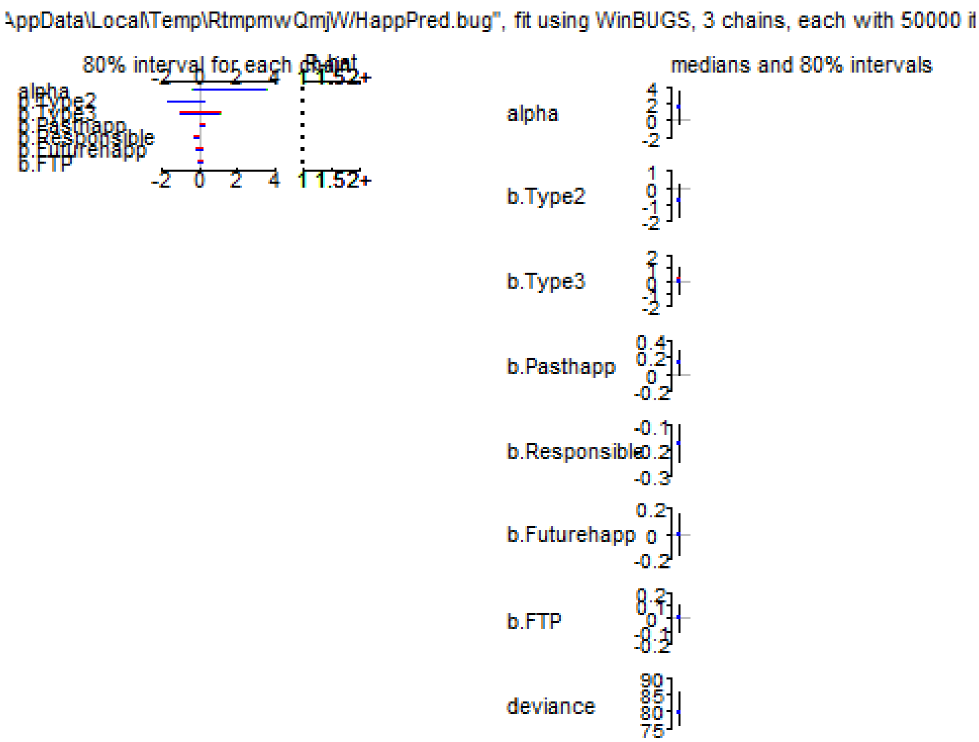

```
#Kernel Density plots
#windows()
par(mfrow=c(2,3))
plot(density(mmm[,2]),col=2,main="Type2")
plot(density(mmm[,3]),col=2,main="Type3")
plot(density(mmm[,4]),col=2,main="Pasthapp")
plot(density(mmm[,5]),col=2,main="Responsible")
plot(density(mmm[,6]),col=2,main="Futurehapp")
plot(density(mmm[,7]),col=2,main="FTP")
```




```
#################################################################################
#Results
#Summary of the simulation
print(post_HappPred.sim,digits=5)

## Inference for Bugs model at
"D:\Users\balaks\AppData\Local\Temp\RtmpmwQmjW/HappPred.bug", fit using
WinBUGS,
##  3 chains, each with 50000 iterations (first 25000 discarded), n.thin = 10
##  n.sims = 7500 iterations saved
##                   mean      sd     2.5%      25%      50%      75%
## alpha          1.51952 1.50714 -1.34353  0.49930  1.48950  2.49500
## b.Type2       -0.71630 0.74418 -2.19600 -1.21200 -0.70590 -0.20310
## b.Type3        0.01352 0.79754 -1.57700 -0.50505  0.01501  0.54848
## b.Pasthapp     0.12555 0.11157 -0.08846  0.04930  0.12460  0.19810
## b.Responsible -0.17551 0.05603 -0.29060 -0.21100 -0.17340 -0.13710
## b.Futurehapp  -0.00635 0.12533 -0.25047 -0.08846 -0.00780  0.07759
## b.FTP          0.00157 0.08145 -0.15690 -0.05328  0.00087  0.05539
## deviance      80.49028 3.89468 74.94475 77.63000 79.77000 82.65250
##                  97.5%    Rhat n.eff
## alpha          4.62505 1.00132  3800
## b.Type2        0.69712 1.00090  7500
## b.Type3        1.56552 1.00096  7500
## b.Pasthapp     0.34706 1.00095  7500
## b.Responsible -0.07272 1.00086  7500
## b.Futurehapp   0.23665 1.00124  4500
## b.FTP          0.16355 1.00118  5200
```

```
## deviance      89.52525 1.00123  4600
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 7.1 and DIC = 87.6
## DIC is an estimate of expected predictive error (lower deviance is
better).
```

```r
#Summary plot
plot(post_HappPred.sim)
```




```r
#plots may be obtained from storing from chain
#Know your runs
mmm=post_HappPred.sim$sims.matrix
head(mmm)
```

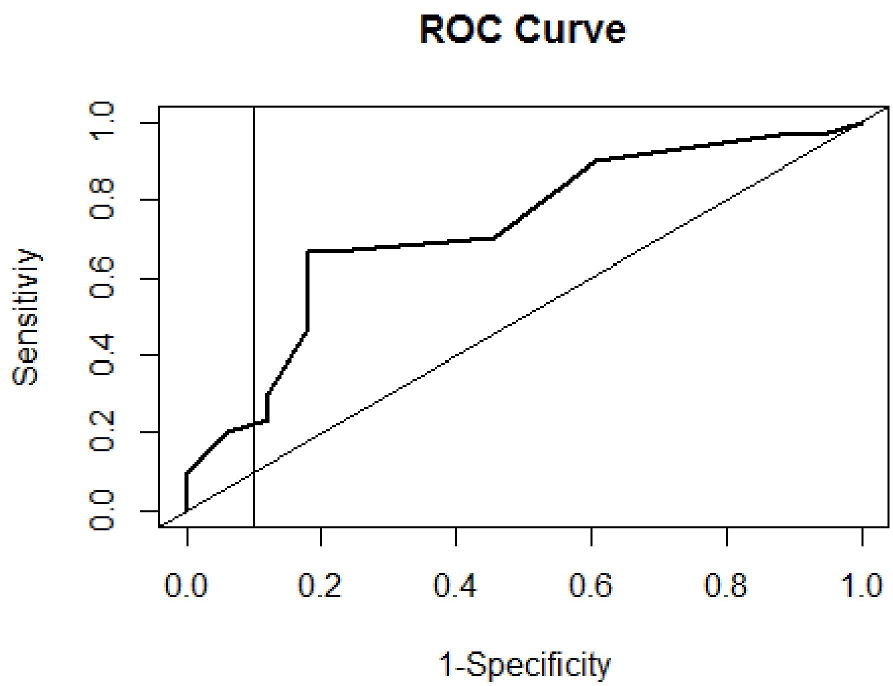
```
##        alpha b.Type2  b.Type3 b.Pasthapp b.Responsible b.Futurehapp
## [1,] 0.06506 -0.1582  1.25700    0.23050       -0.2175      0.03655
## [2,] 2.28800  0.6154 -0.37290    0.01319       -0.1706     -0.12010
## [3,] 3.10700 -0.5639 -0.05809    0.10370       -0.2141     -0.03036
## [4,] 0.85530 -1.0270 -0.76800    0.05408       -0.1093      0.07812
## [5,] 0.99460 -0.9438 -0.48830    0.32170       -0.2228      0.04587
## [6,] 2.34800 -0.3076  0.74560    0.13940       -0.2067     -0.05211
##         b.FTP deviance
## [1,]  0.04728    79.76
```

```
## [2,]  0.02974    87.62
## [3,] -0.07465    75.77
## [4,]  0.02661    77.44
## [5,]  0.05291    82.62
## [6,] -0.01286    78.79
```

```
me_alpha=mean(mmm[,1])
me_Type2=mean(mmm[,2])
me_Type3=mean(mmm[,3])
me_Pasthapp=mean(mmm[,4])
me_Responsible=mean(mmm[,5])
me_Futurehapp=mean(mmm[,6])
me_FTP=mean(mmm[,7])

LL_alpha=quantile(mmm[,1],0.025)
UL_alpha=quantile(mmm[,1],0.925)
LL_Type2=quantile(mmm[,2],0.025)
UL_Type2=quantile(mmm[,2],0.925)
LL_Type3=quantile(mmm[,3],0.025)
UL_Type3=quantile(mmm[,3],0.925)
LL_Pasthapp=quantile(mmm[,4],0.025)
UL_Pasthapp=quantile(mmm[,4],0.925)
LL_Responsible=quantile(mmm[,5],0.025)
UL_Responsible=quantile(mmm[,5],0.925)
LL_Futurehapp=quantile(mmm[,6],0.025)
UL_Futurehapp=quantile(mmm[,6],0.925)
LL_FTP=quantile(mmm[,7],0.025)
UL_FTP=quantile(mmm[,7],0.925)
###############RESULTS
res=rbind(cbind(me_alpha,LL_alpha,UL_alpha),cbind(me_Type2,LL_Type2,UL_Type2)
,cbind(me_Type3,LL_Type3,UL_Type3),cbind(me_Pasthapp,LL_Pasthapp,UL_Pasthapp)
,

cbind(me_Responsible,LL_Responsible,UL_Responsible),cbind(me_Futurehapp,LL_Fu
turehapp,UL_Futurehapp),cbind(me_FTP,LL_FTP,UL_FTP))
row.names(res)=c("alpha",
"Type2","Type3","Pasthapp","Responsible","Futurehapp","FTP")
colnames(res)<-c("mean","95% CrI_LL","95% CrI_UL")
#Revisiting the simulation summary with mean, lower limit and upper limit
values. Note that only the "Responsible" explanatory variable is
statistically significant, since both the lower and upper limits have the
same sign.
res
```

```
##                     mean 95% CrI_LL  95% CrI_UL
## alpha        1.519518812 -1.3435250  3.70457500
## Type2       -0.716295632 -2.1960000  0.33777250
## Type3        0.013524103 -1.5770000  1.16300000
## Pasthapp     0.125553024 -0.0884640  0.28800000
## Responsible -0.175506637 -0.2906050 -0.09684275
```

```
## Futurehapp  -0.006345347 -0.2504675  0.17330000
## FTP          0.001565434 -0.1569000  0.11945750

#######################
#Prediction using Bayesian parameters

ComplaintPredOP<-function(dat_happiness, me_alpha, me_Type2, me_Type3,
me_Pasthapp, me_Responsible, me_Futurehapp, me_FTP){
  ComplaintPredOP=0
  ComplaintPredLogOdds=0
  n= dim(dat_happiness)[1]
  for (i in 1:n){

    #Note that only the valid predictor, "Responsble", is used for prediction
    ComplaintPredLogOdds[i] <- me_Responsible*dat_happiness$Responsible[i]

    ComplaintPredOP[i] <- inv.logit((ComplaintPredLogOdds)[i])
  }
  return(ComplaintPredOP)
}
options(scipen = 999)
RawPredictions <- ComplaintPredOP(dat_happiness, me_alpha, me_Type2,
me_Type3, me_Pasthapp, me_Responsible, me_Futurehapp, me_FTP)


#######################
#Optimizing the probability threshold for class assignment

ComplaintPredResults=0
Prediction <- rep(0, length(RawPredictions))
Actual <- as.numeric(dat_happiness$complain)
ComplaintPredResults <- data.frame(RawPredictions, Prediction, Actual)

#apply roc function
analysis <- roc(response=ComplaintPredResults$Actual,
predictor=ComplaintPredResults$RawPredictions)

#Find t that minimizes error
e <- cbind(analysis$thresholds,analysis$sensitivities+analysis$specificities)
opt_t <- subset(e,e[,2]==max(e[,2]))[,1]


#Plot ROC Curve
par(mfrow=c(1,1))
plot(1-analysis$specificities,analysis$sensitivities,type="l",
     ylab="Sensitiviy",xlab="1-Specificity",col="black",lwd=2,
     main = "ROC Curve")
abline(a=0,b=1)
abline(v = opt_t) #add optimal t to ROC curve
```




```
#The plot shows the ideal threshold setting and the value for the same is
opt_t #print

## [1] 0.1005834

#Class assignment and accuracy calculation
ThreshProbComplain = opt_t

BayesianModPredictions <- rep(0, length(RawPredictions))
BayesianModPredictions[RawPredictions>=ThreshProbComplain] <- 1

ComplaintPredResultsDF <- cbind(dat_happiness$complain,
BayesianModPredictions)
colnames(ComplaintPredResultsDF) <- c("Actual", "Bayesian Prediction")

BayesianPredClassTable <- table(ComplaintPredResultsDF[,2],
ComplaintPredResultsDF[,1])
BayesianPredAccuracy <-
(BayesianPredClassTable[1,1]+BayesianPredClassTable[2,2])/(sum(BayesianPredCl
assTable))

###############Classical approach, to identify statistically significant
explanatory variables
dat_happiness <-
read.csv("D:/Users/balaks/Documents/Learning/BayseianTrainingHCL/hw2data.csv"
)
```

```r
#Coding the response/ dependant variable
dat_happiness$complain = factor(dat_happiness$complain, levels=c(0,1))

#Classical binomial logistic regression model details
summary(glm(dat_happiness$complain ~ ., data = dat_happiness, family =
binomial))

##
## Call:
## glm(formula = dat_happiness$complain ~ ., family = binomial,
##     data = dat_happiness)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7858  -0.9072  -0.5907   1.0400   1.5973
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.944769   1.466403   0.644  0.51940
## Type        -0.037584   0.380673  -0.099  0.92135
## Pasthapp     0.122392   0.101609   1.205  0.22838
## Responsible -0.139901   0.048790  -2.867  0.00414 **
## Futurehapp  -0.010960   0.114169  -0.096  0.92352
## FTP          0.004264   0.074520   0.057  0.95437
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 87.194  on 62  degrees of freedom
## Residual deviance: 74.126  on 57  degrees of freedom
## AIC: 86.126
##
## Number of Fisher Scoring iterations: 4

#Note that in this specific case, the parameter importance of the Bayesian
and classical approaches match. Only the "Responsible" variable is found to
be statistically relevant in both cases. This is seen in the lower and upper
limits being of the same sign/ not crossing the zero mark, in case of
Bayesian Analysis. The same "Responsible" variable is the only one for which
the null hypothesis can be rejected in case of Classical Analysis.

#Repeating classical logistic regression with 10 fold cross validation to
estimate model accuracy
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions =
TRUE)
mod_fit <- train(complain ~ Responsible, data = dat_happiness, method =
"glm", family = "binomial", trControl = ctrl, tuneLength = 5)

#######################
```

```r
#Accuracy comparison
print(paste("Bayesian accuracy (on training data) = ",
format(BayesianPredAccuracy, digits=2)))

## [1] "Bayesian accuracy (on training data) =  0.75"

print(paste("Classical accuracy (more realistic, using CV) = ",
format(mod_fit$results[2], digits=2)))

## [1] "Classical accuracy (more realistic, using CV) =  0.73"
```

Parameter simulations converging to stationary distributions. (This content has been added manually to the output of an .Rmd)