Bounded Regression Models

A Preliminary analysis

Jorge Luis Bazán- https://jorgeluisbazan.weebly.com

Data: Votes vs MHD in Sergipe State in Brazil

MHDI data set is the Municipal Human Development Index (MHDI) of the municipalities of Sergipe state of Brazil. The MHDI is a summary measure of long-term progress in three basic dimensions of human development that takes into account education, income and longevity indexes in municipalities. The MHDI data is the geometric mean of normalized indexes for each of the three dimensions of human development.

Votes is a data frame containing 75 observations on 4 variables with information about the proportion of votes for a political party (Partido dos Trabalhadores) in presidential elections in Brazil by the different municipalities of Sergipe state. In this case we consider the column 4 (Votes2006) corresponding to the votes in 2006.

This data are available in the llbayesireg R package:

https://mran.microsoft.com/snapshot/2019-04-05/web/packages/llbayesireg/index.html

```
#library(betareg)
#data("GasolineYield", package = "betareg")
#head(GasolineYield)
#attach(GasolineYield)
#data=GasolineYield
#y=yield
#x=pressure
library(llbayesireg)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: Rcpp
```

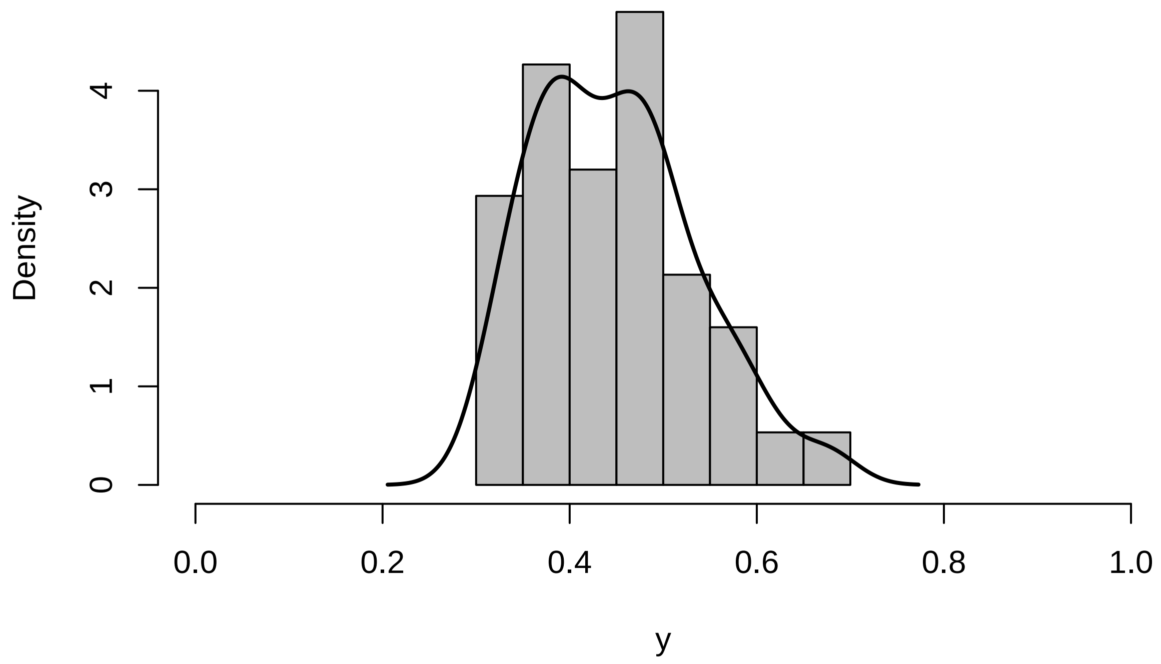
```
data("Votes","MHDI")
y=Votes[,4]
x=MHDI
data=data.frame(y,x)
head(data)
```

```
##        y     x
## 1 0.4925 0.611
## 2 0.3651 0.578
## 3 0.3749 0.770
## 4 0.4264 0.595
## 5 0.3676 0.579
## 6 0.3210 0.649
```

```
dim(data)
```

```
## [1] 75  2
```

```
hist(y, prob=T,col="grey",xlab="y",main="",xlim=c(0,1))
lines(density(y),lwd=2)
```



Fitting Beta Regression Models

```
#Beta Regression
#################
#ML estimation
##############
library(betareg)
betaml <- betareg(y ~ x,data=data)
summary(betaml)
```

```
##
## Call:
## betareg(formula = y ~ x, data = data)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -2.0032 -0.7886 -0.0695  0.7191  2.7346
##
## Coefficients (mean model with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.7097     0.6113   2.797  0.00516 **
## x            -3.2103     1.0231  -3.138  0.00170 **
##
## Phi coefficients (precision model with identity link):
##       Estimate Std. Error z value Pr(>|z|)
## (phi)   37.914      6.112   6.203 5.54e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 83.47 on 3 Df
## Pseudo R-squared: 0.1172
## Number of iterations: 20 (BFGS) + 2 (Fisher scoring)
```

```
#vgam
library(VGAM)
```

```
## Loading required package: stats4

## Loading required package: splines
```

```
betaml2<-vglm(y ~ x,betaff)
summary(betaml2)
```

```
##
## Call:
## vglm(formula = y ~ x, family = betaff)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept):1   1.4289     0.5110   2.796  0.00517 **
## (Intercept):2   0.1157     2.6055   0.044  0.96459
## x:1            -2.7440     0.8417  -3.260  0.00111 **
## x:2             5.9244     4.3578   1.359  0.17399
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: logitlink(mu), loglink(phi)
##
## Log-likelihood: 84.1232 on 146 degrees of freedom
##
## Number of Fisher scoring iterations: 12
##
## No Hauck-Donner effect found in any of the estimates
```

```
#Bayesian estimation
####################
#INLA
library(INLA)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loading required package: parallel

## Loading required package: sp

## This is INLA_21.10.03 built 2021-10-03 11:08:20 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - Save 68.6Mb of storage running 'inla.prune()'
```

```
betab <- inla(y ~ x,family="beta",data=data)
round(betab$summary.fixed,4)
```

```
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept)  1.7028 0.6538     0.4305   1.6980     3.0017  1.6888   0
## x           -3.1969 1.0945    -5.3761  -3.1884    -1.0673 -3.1721   0
```

```
round(betab$summary.hyperpar,4)
```

```
##                                                 mean     sd 0.025quant
## precision parameter for the beta observations 34.5554 5.5259    24.5766
##                                              0.5quant 0.975quant    mode
## precision parameter for the beta observations  34.2579     46.234 33.6532
```

The VGAM package fit a double beta regression with covariates on the mean and on the precision parameter, thus, it is no possible to estimate the precision parameter. A similar model can be obtained in INLA but it is no possible in betareg.

Fitting Simplex Regression Models

```
#Simplex Regression
##################
#ML estimation
###############
#simplexreg
library(simplexreg)
```

```
## Loading required package: Formula

## Loading required package: plotrix

##
## Attaching package: 'simplexreg'

## The following objects are masked from 'package:VGAM':
##
##     dsimplex, rsimplex
```

```
simplexml <- simplexreg(y ~ x,data=data)
summary(simplexml)
```

```
##
## Call:
## simplexreg(formula = y ~ x, data = data)
##
## standard Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.9086 -0.8025 -0.0931  0.7018  2.6369
##
## Coefficients (mean model with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.6937     0.5961   2.841  0.00449 **
## x            -3.1794     0.9957  -3.193  0.00141 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -65.19,  p-value: 0.4779857
## Deviance: 73
## Number of Fisher Scoring iterations:  5
```

```
simplexml$Dispersion
```

```
## [1] 0.4653043
```

```
#vgam
library(VGAM)
simplexml2<-vglm(y ~ x,simplex)
summary(simplexml2)
```

```
##
## Call:
## vglm(formula = y ~ x, family = simplex)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  1.69369    0.58863   2.877  0.00401 **
## (Intercept):2 -0.39605    0.08165  -4.851 1.23e-06 ***
## x             -3.17937    0.98325  -3.234  0.00122 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: logitlink(mu), loglink(sigma)
##
## Log-likelihood: 83.9135 on 147 degrees of freedom
##
## Number of Fisher scoring iterations: 5
##
## No Hauck-Donner effect found in any of the estimates
```

```
#dispersion parameter
exp(coef(simplexml2)[2])^2
```

```
## (Intercept):2
##     0.4528961
```

```
#Bayesian estimation
####################
#INLA
library(INLA)
simplexb <- inla(y ~ x| 1,data=data,family="simplex")
round(simplexb$summary.fixed,4)
```

```
##                  mean      sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept)   -0.2021 31.6228   -62.2882  -0.2029    61.8323 -0.2021   0
## x | 1 + 1TRUE  0.0000 31.6227   -62.0861  -0.0009    62.0343  0.0000   0
```

```
library(brinla)
bri.hyperpar.summary(simplexb)
```

```
##                                        mean         sd    q0.025      q0.5
## SD for the Simplex observations 0.7196783 0.05882448 0.6154011 0.7156524
##                                      q0.975      mode
## SD for the Simplex observations 0.8462717 0.7082956
```

Estimate of dispersion parameter under INLA are less than ML estimatimates.

Fitting Kumaraswamy Regression Models

```
#Kumaraswamy Regression
#######################
#ML estimation
kumarml<-vglm(y ~ x,kumar)
summary(kumarml)
```

```
##
## Call:
## vglm(formula = y ~ x, family = kumar)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1   0.6774     0.9216   0.735    0.462
## (Intercept):2  -7.9070     4.9318  -1.603    0.109
## x:1             1.8323     1.5317   1.196    0.232
## x:2            20.4929     8.4207      NA       NA
##
## Names of linear predictors: loglink(shape1), loglink(shape2)
##
## Log-likelihood: 80.7323 on 146 degrees of freedom
##
## Number of Fisher scoring iterations: 8
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## 'x:2'
```

```
coef(kumarml, matrix = TRUE)
```

```
##             loglink(shape1) loglink(shape2)
## (Intercept)       0.6773797       -7.907019
## x                 1.8322579       20.492928
```

```
Coef(kumarml)
```

```
## (Intercept):1 (Intercept):2           x:1           x:2
##     0.6773797    -7.9070190     1.8322579    20.4929279
```

The VGAM package fit a double Kumar regression with covariates on the mean and on the precision parameter.

Fitting Llogistic Regression Models

```
#Llogistic Regression
########################
#Bayesian estimation
library(llbayesireg)
data("Votes","MHDI")
fitll = llbayesireg(y,x)
```

```
##
## SAMPLING FOR MODEL '93e56cc741fce493bb30905382ac9cae' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.824 seconds (Warm-up)
## Chain 1:                2.63 seconds (Sampling)
## Chain 1:                5.454 seconds (Total)
## Chain 1:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```
summary(fitll$object, pars = c("beta","delta"), probs = c(0.025,0.975))
```

```
## $summary
##               mean     se_mean         sd       2.5%     97.5%     n_eff
## beta[1]   1.708886 0.083860915 0.64124189  0.6140861  3.052848  58.46885
## beta[2]  -3.244536 0.140875231 1.07385853 -5.4952081 -1.427102  58.10652
## delta[1]  1.642416 0.005790777 0.08576259  1.4734094  1.788389 219.34215
##               Rhat
## beta[1]  1.0201802
## beta[2]  1.0199963
## delta[1] 0.9987116
##
## $c_summary
## , , chains = chain:1
##
##           stats
## parameter       mean         sd       2.5%     97.5%
##   beta[1]   1.708886 0.64124189  0.6140861  3.052848
##   beta[2]  -3.244536 1.07385853 -5.4952081 -1.427102
##   delta[1]  1.642416 0.08576259  1.4734094  1.788389
```