Steven Maharaj 695281 Assignment 2, Question 3
MAST90125: Bayesian Statistical Learning

Due: Friday 20 September 2019

There are places in this assignment where R code will be required. Therefore set the random seed so assignment is reproducible.

```
set.seed(695281)  #Please change random seed to your student id number.
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyr)
library(TruncatedNormal)
```

Question Three (18 marks)

A group of 453 Bangladeshi women in 5 districts were asked about contraceptive use. The response variable *use* is an indicator for contraceptive use (coded N for no and Y for yes). Other covariates of interest are categorical variables for geographical location *district* (5 levels), and *urban* (2 levels), and number of living children *livch* (4 levels), and the continuous covariate for standardised age *age*.A random intercept for the district was suggested. This suggested the following model should be fitted,

$$\boldsymbol{\theta} = \mathbf{Z}\mathbf{u} + \mathbf{X}\boldsymbol{\beta},$$

where $\boldsymbol{\theta}$ is a link function, $\mathbf{Z}$ is an indicator variable for district, $\mathbf{u}$ is a random intercept with prior $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \sigma_u^2\mathbf{I})$, and $\mathbf{X}$ is a design matrix for fixed effects $\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ includes the coefficients for the intercept, urban status, living children, and age.

Data can be downloaded from LMS as `Contraceptionsubset.csv`.

a) Fit a generalised linear mixed model assuming a logistic link using `Stan`. The R and stan code below covers the following steps.

- Importing the data.
- Constructing design matrices.
- Provides code to go into the stan file.
- Running stan in R. This assumes your stan file is called *logitmm.stan*, and that you will run the sampler for 2000 iterations and 4 chains.

Note that provided code assumes everything required is located in your working directory in R.

```r
#Step one: Importing data,  constructing design matrices and calculating matrix dimensions.
dataX= read.csv("Contraceptionsubset.csv",header=TRUE)

n<-dim(dataX)[1]
Z    = table(1:n,dataX$district)     #incidence matrix for district
Q    = dim(Z)[2]
D1   = table(1:n,dataX$livch) #Dummy indicator for living children
D2   = table(1:n,dataX$urban) #Dummy indicator for urban status

#fixed effect design matrix
X    = cbind(rep(1,n),dataX$age,D1[,-1],D2[,-1])
P    = dim(X)[2]
y    = rep(0,n)
y[dataX$use %in% 'Y'] = 1
```

An example stan file.

```
//
// This Stan program defines a logistic mixed model
//
// Learn more about model development with Stan at:
//
//    http://mc-stan.org/users/interfaces/rstan.html
//    https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
//

data {
  int<lower=0> n;   //number of observations
  int<lower=0> Q;   //number of random effect levels
  int<lower=0> P;   //number of fixed effect levels
  int y[n];      //response vector
  matrix[n,Q] Z;     //indicator matrix for random effect levels
  matrix[n,P] X;     //design matrix for fixed effects
}

// The parameters accepted by the model.
// accepts three sets of parameters 'beta', 'u' and 'sigma'.
parameters {
  vector[P] beta; //vector of fixed effects of length P.
  vector[Q] u; //vector of random effects of length Q.
  real<lower=0> sigma; //random effect standard deviation
}


// The model to be estimated. We model the output
// 'y' to be bernoulli with logit link function,
// and assume a i.i.d. normal prior for u.
model {
  u ~ normal(0,sigma);                //prior for random effects.
  y ~ bernoulli_logit(X*beta+ Z*u);  //likelihood

}
```

```r
library(rstan)
```

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

```
## rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
```

```
## rstan_options(auto_write = TRUE)
##
## Attaching package: 'rstan'
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
logistic.mm <-stan(file="logitmm.stan",data=c('Z','X','y','n','P','Q'),iter=2000,chains=4)
```

```
##
## SAMPLING FOR MODEL 'logitmm' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.9 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.56373 seconds (Warm-up)
## Chain 1:                1.23385 seconds (Sampling)
## Chain 1:                2.79758 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'logitmm' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.43 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 2:
## Chain 2:  Elapsed Time: 1.59453 seconds (Warm-up)
## Chain 2:                1.16174 seconds (Sampling)
## Chain 2:                2.75627 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'logitmm' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 4.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.43 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.44937 seconds (Warm-up)
## Chain 3:                1.25615 seconds (Sampling)
## Chain 3:                2.70552 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'logitmm' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 4.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.42 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.7457 seconds (Warm-up)
## Chain 4:                1.38607 seconds (Sampling)
## Chain 4:                3.13177 seconds (Total)
```

```
## Chain 4:
```

```r
print(logistic.mm)
```

```
## Inference for Stan model: logitmm.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## beta[1]    -2.01    0.03 0.71   -3.33   -2.39   -2.02   -1.66   -0.69   642
## beta[2]    -0.04    0.00 0.02   -0.08   -0.05   -0.04   -0.03   -0.01  2468
## beta[3]     1.24    0.01 0.35    0.57    1.01    1.24    1.48    1.93  2260
## beta[4]     1.46    0.01 0.37    0.74    1.21    1.46    1.71    2.21  2187
## beta[5]     1.81    0.01 0.38    1.06    1.54    1.81    2.06    2.57  1849
## beta[6]     1.21    0.00 0.26    0.71    1.03    1.21    1.39    1.75  3370
## u[1]       -1.07    0.03 0.67   -2.42   -1.38   -1.03   -0.71    0.06   623
## u[2]       -0.27    0.03 0.68   -1.56   -0.57   -0.24    0.09    0.97   631
## u[3]        0.42    0.03 0.67   -0.86    0.10    0.43    0.76    1.64   615
## u[4]        0.18    0.03 0.67   -1.04   -0.14    0.19    0.53    1.41   642
## u[5]        0.68    0.03 0.67   -0.52    0.37    0.67    1.02    1.93   637
## sigma       1.14    0.03 0.79    0.42    0.68    0.92    1.32    3.17   623
## lp__     -274.52    0.08 2.58 -280.46 -276.09 -274.16 -272.63 -270.45  1161
##          Rhat
## beta[1]     1
## beta[2]     1
## beta[3]     1
## beta[4]     1
## beta[5]     1
## beta[6]     1
## u[1]        1
## u[2]        1
## u[3]        1
## u[4]        1
## u[5]        1
## sigma       1
## lp__        1
##
## Samples were drawn using NUTS(diag_e) at Mon Sep 23 10:36:41 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Note that in Stan, defaults for burn-in (warm-up) is one half of all iterations in stan, and no thinning. Note the code is written using the stan file and csv is in your working directory. Use the `print` function to report posterior means, standard deviations, 95 % central credible intervals and state from the output whether you believe the chains have converged. Also report the reference categories for *urban* and *livch*.

Reporting for PART A:

posterior means, standard deviations can be found in the above table. The lower limit of the 95 % central credible intervals given by the column labeled "2.5%" while the upperer limit of the 95 % central credible intervals given by the column labeled "97.5%".

```r
print(summary(logistic.mm)$summary)
```

```
##                 mean      se_mean         sd         2.5%          25%
## beta[1]   -2.01099308 0.0279965047 0.70946569   -3.33449620   -2.38946953
```

```
## beta[2]    -0.04295001 0.0003521243 0.01749236   -0.07815811   -0.05459747
## beta[3]     1.24325109 0.0072812395 0.34613194    0.57340710    1.00861655
## beta[4]     1.46352506 0.0080167605 0.37494188    0.73676952    1.20830568
## beta[5]     1.80572045 0.0089176262 0.38345736    1.06096227    1.54466692
## beta[6]     1.21453241 0.0045583884 0.26462671    0.71162499    1.03214224
## u[1]       -1.07381667 0.0269916432 0.67377403   -2.41698388   -1.37911413
## u[2]       -0.26721280 0.0272399411 0.68449734   -1.56242242   -0.57469241
## u[3]        0.41832407 0.0270562570 0.67078550   -0.85772209    0.10025676
## u[4]        0.18127185 0.0263757755 0.66834362   -1.03777263   -0.13773489
## u[5]        0.68136569 0.0264603269 0.66783668   -0.52009352    0.36717850
## sigma       1.13655381 0.0316092861 0.78872899    0.41805922    0.68156501
## lp__     -274.52251710 0.0758140006 2.58339599 -280.45888798 -276.08636043
##                   50%          75%        97.5%     n_eff     Rhat
## beta[1]    -2.0224113   -1.65718968 -6.941163e-01  642.1776 1.002302
## beta[2]    -0.0430541   -0.03097084 -9.660813e-03 2467.7715 1.001572
## beta[3]     1.2395327    1.48446282  1.933659e+00 2259.8142 1.000065
## beta[4]     1.4568984    1.71231504  2.212323e+00 2187.4094 1.000015
## beta[5]     1.8079857    2.05923746  2.566838e+00 1848.9945 1.000926
## beta[6]     1.2116489    1.38997495  1.748676e+00 3370.1147 1.001087
## u[1]       -1.0303874   -0.71288160  6.242366e-02  623.1174 1.002417
## u[2]       -0.2443410    0.08823265  9.712708e-01  631.4387 1.001814
## u[3]        0.4317889    0.75832452  1.639977e+00  614.6557 1.001684
## u[4]        0.1910809    0.53395180  1.413318e+00  642.0800 1.001682
## u[5]        0.6734846    1.02474037  1.925693e+00  637.0157 1.001925
## sigma       0.9166456    1.32004449  3.172199e+00  622.6245 1.003215
## lp__     -274.1591601 -272.63476831 -2.704475e+02 1161.1361 1.003517
```

Using the Gelman-Rubin diagnostic (Rhat) only 6 out of the 12 parameters These were beta[2],beta[3],beta[4],beta[5],beta[6],sigm had an Rhat value close to 1. All u parameters and beta[1] do not appear to converge thus the chain did not converge.

For *urban* and *livch* the reference categories are "N" and "0" respectivley. This is infered from the way the DataFrame X was constructed. `X = cbind(rep(1,n),dataX$age,D1[,-1],D2[,-1])`

b) An alternative to the logit link when analysing binary data is the probit. The probit link is defined as,

$$
\begin{aligned}
y_i &= \begin{cases} 1 & \text{if } z_i \geq 0 \\ 0 & \text{if } z_i < 0 \end{cases} \\
z_i &= \mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i, \quad \epsilon \sim \mathcal{N}(0,1).
\end{aligned}
$$

In lecture 14, we showed how by letting z_i be normal, probit regression can be fitted using a Gibbs sampler, but to do so, it requires the ability to sample from a truncated normal defined on either $(-\infty, 0)$ (if $y_i = 0$) or $(0, \infty)$ (if $y_i = 1$). Check by comparing the empirical and the true density that a modified version of the inverse cdf method can be used to produce draws from a truncated normal. Do this for the case where $x \in (0, \infty)$ and $x \in (-\infty, 0)$ with parameters $\mu = 0.5$ and $\sigma = 1$.

Hints: If y is drawn from a truncated normal with lower bound a, upper bound b and parameters μ, σ^2 then then $p(y|\mu, \sigma^2, a, b)$ is

$$
\frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-\mu)^2/2}}{\int_{-\infty}^{b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-\mu)^2/2} dy - \int_{-\infty}^{a} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-\mu)^2/2} dy},
$$

which in `R` means the truncated normal density can be written as

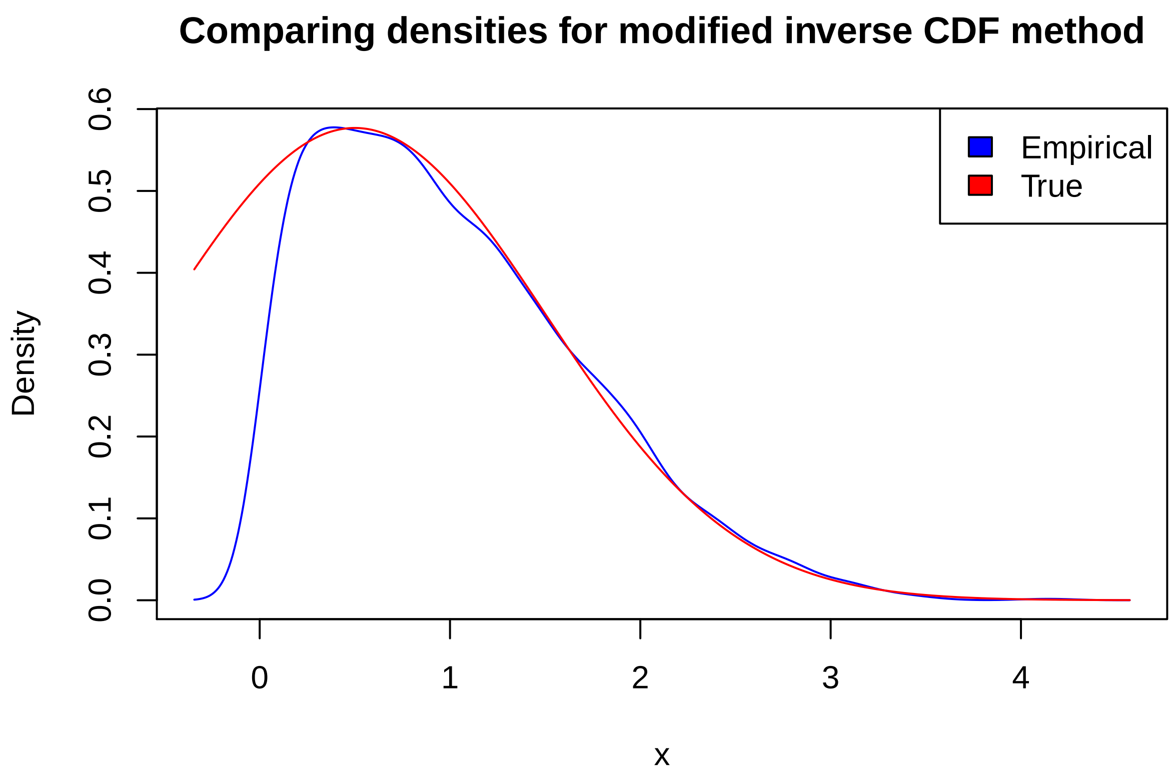
```
dnorm(x,mean=mu,sd=sigma)/(pnorm(b,mean=mu,sd=sigma)-pnorm(a,mean=mu,sd=sigma))
```

The inverse cdf method involves drawing v from $U(0,1)$ so that $x \sim p(x)$ can be found solving $x = F^{-1}(x)$, where F is the cdf. If the only change compared to drawing from a normal distribution is truncation, think about what happens to the bounds of the uniform distribution.

Answer: Part B

Case $x \in (0, \infty)$

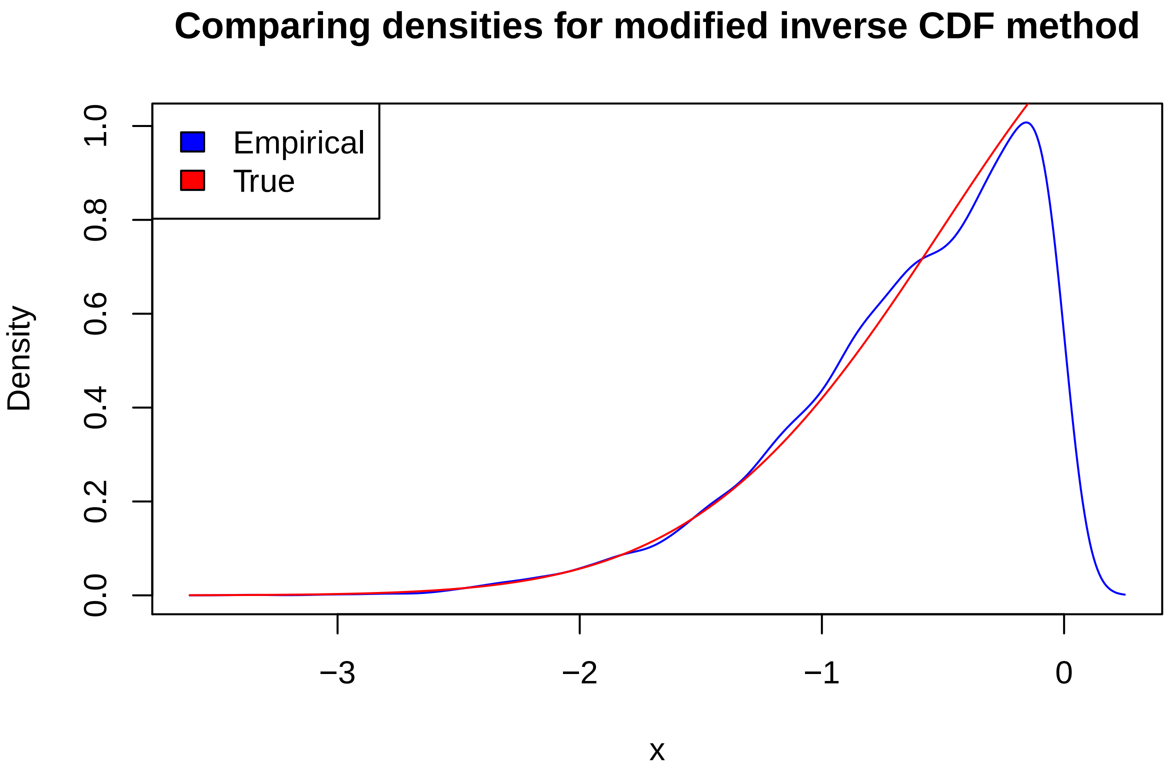
```r
mu = 0.5
sigma =  1 # sigma

a <- 0
b <- 10 # A number going to infinity
n <- 5000

invtn <- function(n,mu,sigma,a,b){
  u <- runif(n)
  return(qnorm(pnorm(a)+u*(pnorm(b) - pnorm(a)))*sigma + mu)
}

# Using my inverse cdf method
x_empircal <- invtn(n,mu,sigma,a,b)

# using the TruncatedNormal package
x_empircal <- rtnorm(n = n, mu = mu, lb = a, ub = b, method = "fast")
empircal_den <- density(x_empircal)

x <- seq(a,b,length.out = n)
y <- dnorm(empircal_den$x,mean=mu,sd=sigma)/(pnorm(b,mean=mu,sd=sigma)-pnorm(a,mean=mu,sd=sigma))

plot(empircal_den$x,empircal_den$y,
main="Comparing densities for modified inverse CDF method",
ylab="Density",
xlab = "x",
type="l",
col="blue")
lines(empircal_den$x, y, col="red")
legend("topright",
c("Empirical","True"),
fill=c("blue","red")
)
```



Case $x \in (-\infty, 0)$

```
mu = 0.5
sigma =  1 # sigma

a <- -10
b <- 0 # A number going to infinity
n <- 5000


# Using my inverse cdf method
x_empircal <- invtn(n,mu,sigma,a,b)

# using the TruncatedNormal package
x_empircal <- rtnorm(n = n, mu = mu, lb = a, ub = b, method = "fast")

empircal_den <- density(x_empircal)

x <- seq(a,b,length.out = n)

y <- dnorm(empircal_den$x,mean=mu,sd=sigma)/(pnorm(b,mean=mu,sd=sigma)-pnorm(a,mean=mu,sd=sigma))

plot(empircal_den$x,empircal_den$y,
main="Comparing densities for modified inverse CDF method",
ylab="Density",
xlab = "x",
type="l",
col="blue")
lines(empircal_den$x, y, col="red")
legend("topleft",
```

```r
c("Empirical","True"),
fill=c("blue","red")
)
```



c) Implement a Gibbs sampler to fit the same mixed model as fitted in Stan in a), but now with a probit link. As before, fit 4 chains, each running for 2000 iterations, with the first 1000 iterations discarded as burn-in. Perform graphical convergence checks and Gelman-Rubin diagnostics. Report posterior means, standard deviations and 95 % central credible intervals for $\sigma, \boldsymbol{\beta}, \mathbf{u}$ by combining chains.

d) For the co-efficients $\boldsymbol{\beta}$, $\mathbf{u}$, calculate the mean of the ratio of the posterior means $\boldsymbol{\beta}_{i,\text{logit}}/\boldsymbol{\beta}_{i,\text{probit}}, \mathbf{u}_{i,\text{logit}}/\mathbf{u}_{i,\text{probit}}$ obtained when fitting the logistic mixed model and the probit mixed model. To do this, you will need to apply the `extract` function to the stan model object. Once calculated, multiply the iterations obtained assuming a probit link by this constant and compare to the iterations obtained assuming a logit link.

e) The logistic link can be written in the same way as the probit link, but instead of $e_i \sim \mathcal{N}(0,1)$, the error term is $e_i \sim \text{Logistic}(0,1)$. By evaluating the standard normal and logistic inverse cdfs and superimposing the line $y = mx$ where m is the posterior ratio, do you think the results in d) were surprising.