Anexo tarea 4

Diego Hernández Jiménez

22/4/2022

Preprocesamiento de variables

```r
# eliminación por lista de 144 observaciones

fulldata <- read.csv(paste0(path,'\\datamemory.csv'))

tokeep <- c('distracted','importance','memType')
df <- fulldata[fulldata$memType != 'retold',tokeep] |> na.omit()

df$memType <- as.factor(df$memType)
df$z_distracted <- scale(df$distracted)
df$z_importance <- scale(df$importance)

str(df)
```

```
## 'data.frame':    5391 obs. of  5 variables:
##  $ distracted  : num  1 1 1 1 2 1 3 1 1 1 ...
##  $ importance  : num  3 4 4 5 3 5 4 5 4 3 ...
##  $ memType     : Factor w/ 2 levels "imagined","recalled": 1 2 1 2 1 2 1 2 1 2 ...
##  $ z_distracted: num [1:5391, 1] -0.482 -0.482 -0.482 -0.482 0.605 ...
##   ..- attr(*, "scaled:center")= num 1.44
##   ..- attr(*, "scaled:scale")= num 0.92
##  $ z_importance: num [1:5391, 1] -0.661 0.1 0.1 0.861 -0.661 ...
##   ..- attr(*, "scaled:center")= num 3.87
##   ..- attr(*, "scaled:scale")= num 1.31
##  - attr(*, "na.action")= 'omit' Named int [1:144] 40 84 126 128 161 163 174 267 283 292 ...
##   ..- attr(*, "names")= chr [1:144] "49" "106" "157" "160" ...
```

Definición del modelo y parametrización del proceso de muestreo de Gibbs

```r
data_jags <- list(N=nrow(df),
                  y=as.numeric(df$memType)-1, # niveles originales: 1:imagined y 2:recalled
                  z_distracted=as.numeric(df$z_distracted),
                  z_importance=as.numeric(df$z_importance),
                  burn=500,
                  samples=10000)

params <- c('beta0','beta')
```

```
library(R2jags)
```

```
## Warning: package 'R2jags' was built under R version 4.1.3

## Loading required package: rjags

## Warning: package 'rjags' was built under R version 4.1.3

## Loading required package: coda

## Warning: package 'coda' was built under R version 4.1.3

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs

##
## Attaching package: 'R2jags'

## The following object is masked from 'package:coda':
##
##     traceplot
```

```
# modelo predictores estandarizados

bayes_logit <- function(){

   # priors normales
   # beta0 ~ dnorm(0,0.01)
   # beta[1] ~ dnorm(0,0.16)
   # beta[2] ~ dnorm(0,0.16)

   # recomendaciones Gelman et. al (2008)
   # Cauchy(loc=0,scale=2.5) (t student con 1 gl == Cauchy)
   beta0 ~ dt(0,0.01,1) # prior menos informativo para intersección
   beta[1] ~ dt(0,0.16,1) # (1/2.5^2) = 0.16
   beta[2] ~ dt(0,0.16,1)

   # likelihood
   for (i in 1:N){
      prob[i] <- ilogit(beta0 + beta[1]*z_distracted[i] + beta[2]*z_importance[i])
      y[i] ~ dbern(prob[i])
   }
}


jagsfit <- jags.parallel(data=data_jags,
                         parameters.to.save=params,
                         n.chains=3,n.iter=samples+burn,
                         n.burnin=burn,
                         n.thin=1,
                         model.file=bayes_logit,
                         jags.seed=13)

save(jagsfit,file=paste0(path,'\\logitbayes.Rdata')) # se guarda para evitar ajustar el modelo cada vez
```

Resumen MCMC

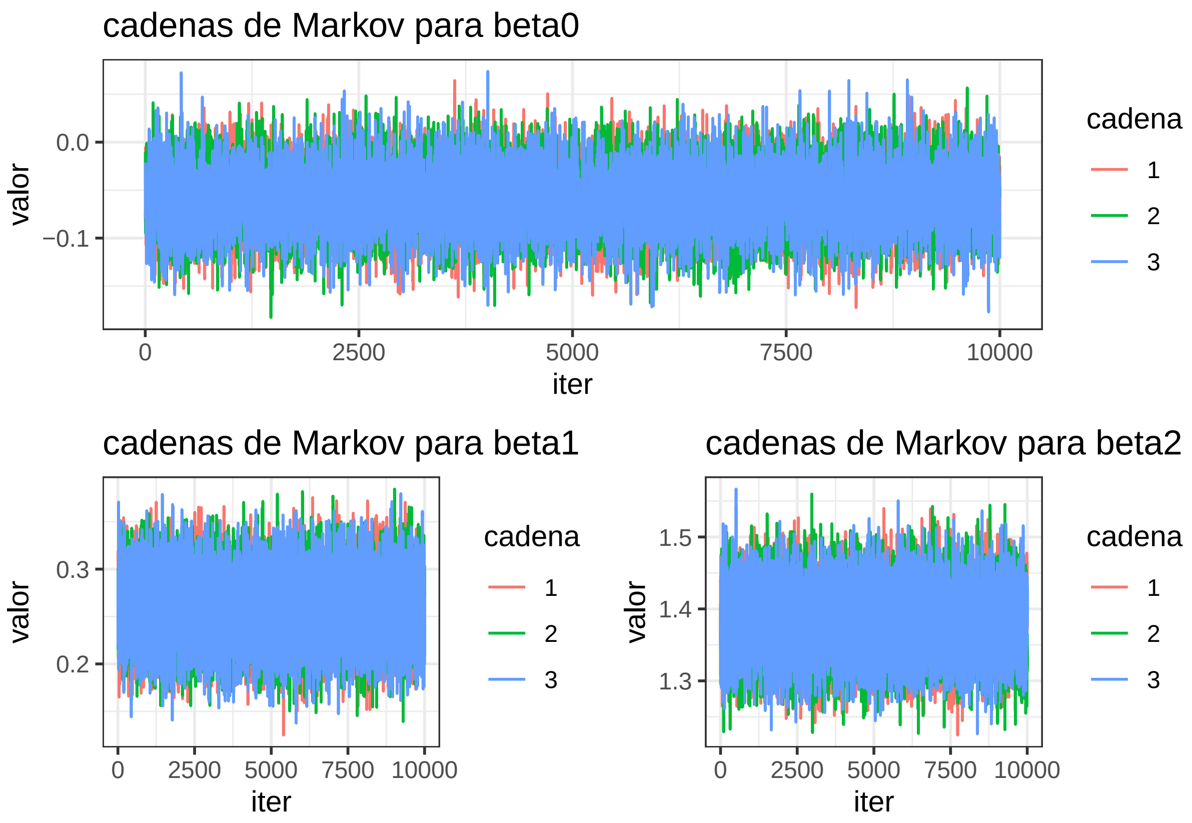
```
load(paste0(path,'\\logitbayes.Rdata'))
print(jagsfit)
```

```
## Inference for Bugs model at "bayes_logit", fit using jags,
##  3 chains, each with 10500 iterations (first 500 discarded)
##  n.sims = 30000 iterations saved
##           mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
## beta[1]     0.259   0.034    0.195    0.237    0.259    0.282    0.326 1.001
## beta[2]     1.382   0.044    1.298    1.352    1.382    1.412    1.469 1.001
## beta0      -0.057   0.033   -0.123   -0.079   -0.057   -0.035    0.007 1.001
## deviance 5830.271   2.424 5827.490 5828.492 5829.657 5831.398 5836.547 1.001
##          n.eff
## beta[1]  12000
## beta[2]   8600
## beta0     4600
## deviance 30000
##
## 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 = var(deviance)/2)
## pD = 2.9 and DIC = 5833.2
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Traceplots

```
library(ggplot2)
chains <- jagsfit$BUGSoutput$sims.array
nchain <- rep(1:3,each=data_jags$samples)
iter <- rep(1:data_jags$samples,times=3)
beta1post <- as.numeric(chains[,,1]) |>  # flatten to (10000*3)x1 vector
   cbind(nchain) |>
   data.frame()
beta2post <- as.numeric(chains[,,2]) |>  # flatten to (10000*3)x1 vector
   cbind(nchain) |>
   data.frame()
beta0post <- as.numeric(chains[,,3]) |>  # flatten to (10000*3)x1 vector
   cbind(nchain) |>
   data.frame()
```

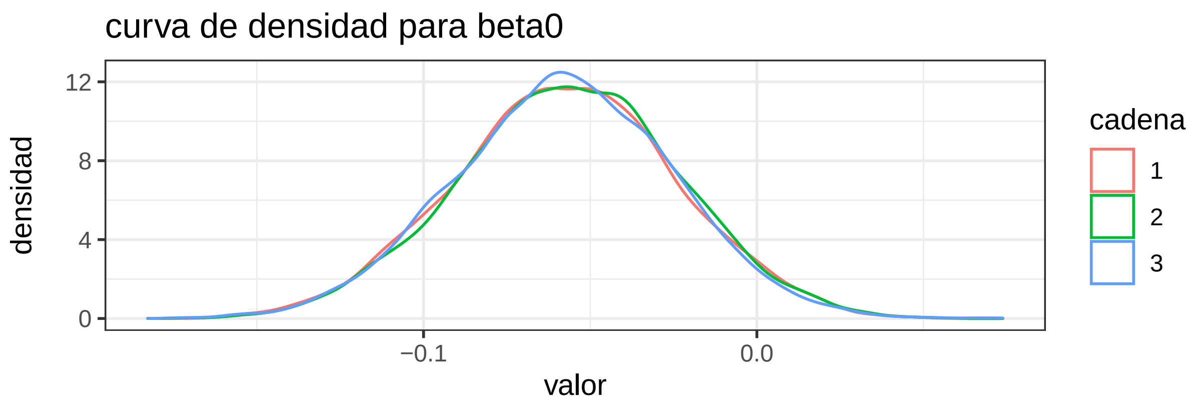
```r
beta0chains <- ggplot(beta0post,aes(x=iter,y=V1,col=factor(nchain))) +
   geom_line() +
   labs(title='cadenas de Markov para beta0',
        y='valor',
        col='cadena') +
   theme_bw()
beta1chains <- ggplot(beta1post,aes(x=iter,y=V1,col=factor(nchain))) +
   geom_line() +
   labs(title='cadenas de Markov para beta1',
        y='valor',
        col='cadena') +
   theme_bw()
beta2chains <- ggplot(beta2post,aes(x=iter,y=V1,col=factor(nchain))) +
   geom_line() +
   labs(title='cadenas de Markov para beta2',
        y='valor',
        col='cadena') +
   theme_bw()

library(patchwork)
beta0chains / (beta1chains + beta2chains)
```

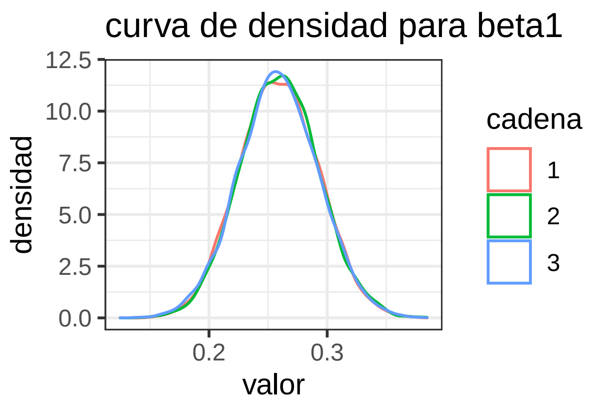


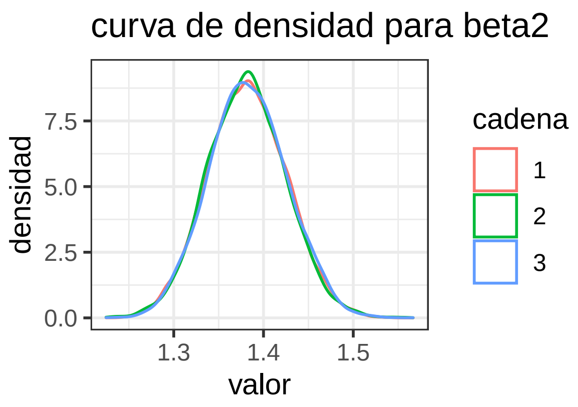
```r
# ggsave(filename=paste0(path,'\\traceplots.jpg'),dpi=300)
```

Curvas de densidad

```
beta0dens <- ggplot(beta0post,aes(x=V1,col=factor(nchain))) +
   geom_density() +
   labs(title='curva de densidad para beta0',
        x='valor',
        y='densidad',
        col='cadena') +
   theme_bw()
beta1dens <- ggplot(beta1post,aes(x=V1,col=factor(nchain))) +
   geom_density() +
   labs(title='curva de densidad para beta1',
        x='valor',
        y='densidad',
        col='cadena') +
   theme_bw()
beta2dens <- ggplot(beta2post,aes(x=V1,col=factor(nchain))) +
   geom_density() +
   labs(title='curva de densidad para beta2',
        x='valor',
        y='densidad',
        col='cadena') +
   theme_bw()


beta0dens / (beta1dens + beta2dens)
```



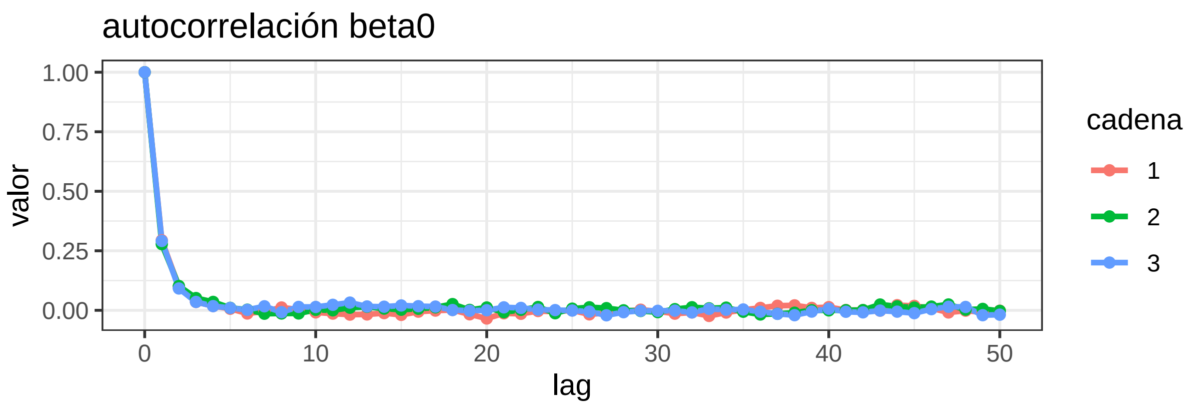


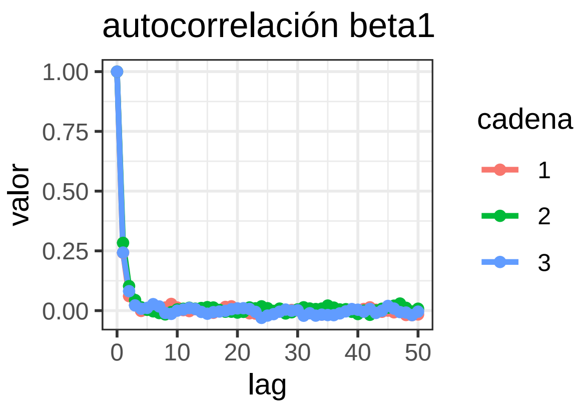


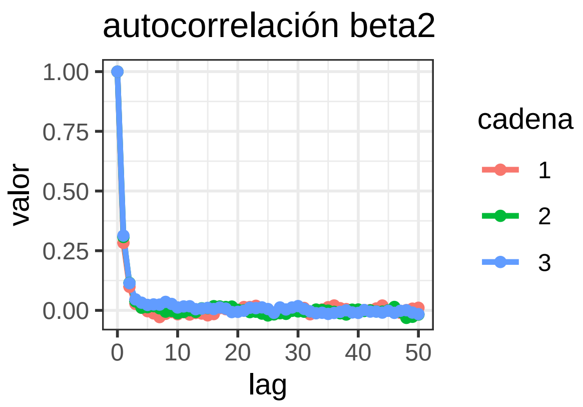
```
# ggsave(filename=paste0(path,'\\curvasdens.jpg'),dpi=300)
```

autocorrelación

```
step <- 1
lag <- seq(0,50,by=step)
nchain <- rep(1:3,each=1+(50/step))

autocorr_b1 <- coda::as.mcmc(chains[,,1]) |>
   coda::autocorr.diag(lags=lag) |>
   as.numeric() |>
   cbind(lag,nchain) |>
   as.data.frame()

autocorr_b2 <- coda::as.mcmc(chains[,,2]) |>
   coda::autocorr.diag(lags=lag) |>
   as.numeric() |>
   cbind(lag,nchain) |>
   as.data.frame()

autocorr_b0 <- coda::as.mcmc(chains[,,3]) |>
   coda::autocorr.diag(lags=lag) |>
   as.numeric() |>
   cbind(lag,nchain) |>
   as.data.frame()


beta0auto <- ggplot(autocorr_b0,aes(x=lag,y=V1,col=factor(nchain))) +
   geom_line(lwd=1) +
   geom_point() +
   labs(title='autocorrelación beta0',
        y='valor',
        col='cadena') +
   theme_bw()

beta1auto <- ggplot(autocorr_b1,aes(x=lag,y=V1,col=factor(nchain))) +
   geom_line(lwd=1) +
   geom_point() +
   labs(title='autocorrelación beta1',
        y='valor',
        col='cadena') +
   theme_bw()

beta2auto <- ggplot(autocorr_b2,aes(x=lag,y=V1,col=factor(nchain))) +
   geom_line(lwd=1) +
   geom_point() +
   labs(title='autocorrelación beta2',
        y='valor',
        col='cadena') +
   theme_bw()

beta0auto / (beta1auto + beta2auto)
```



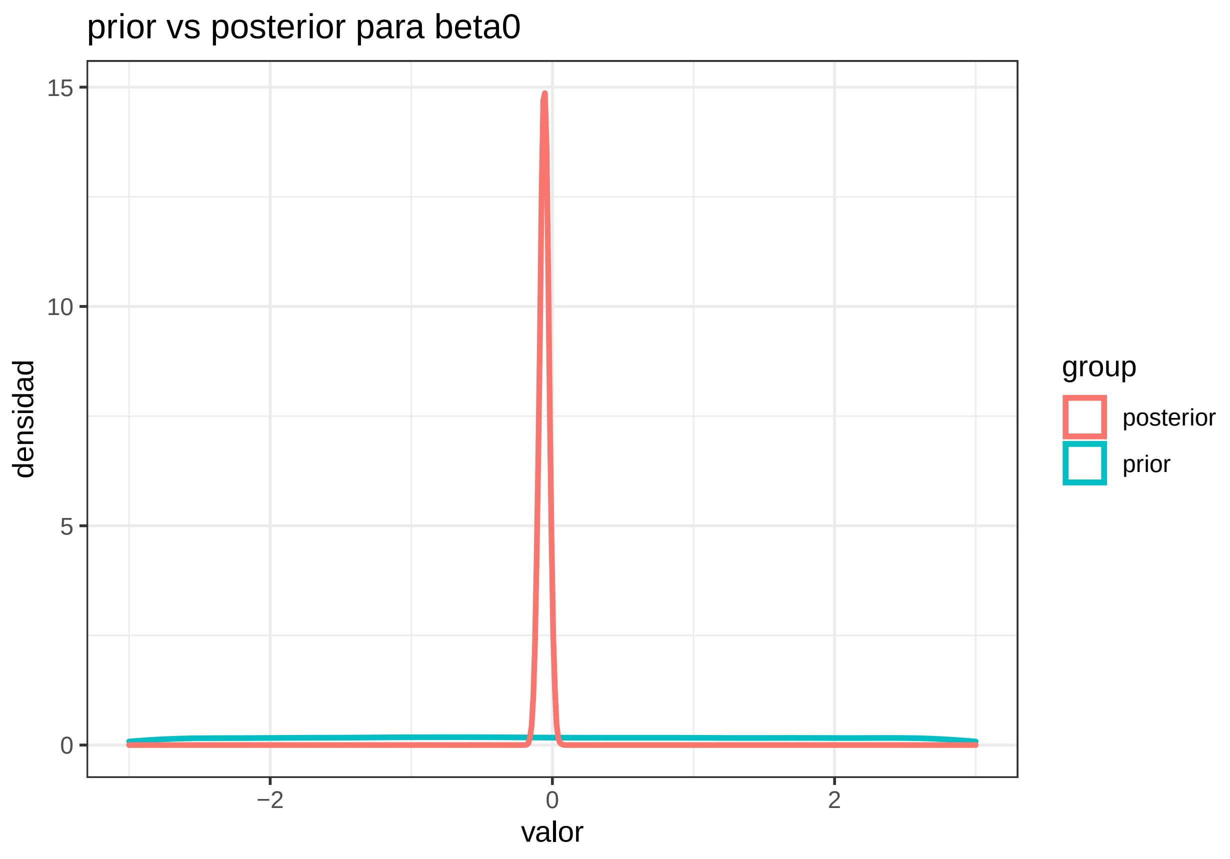




```
# ggsave(filename=paste0(path,'\\autocorr.jpg'),dpi=300)
```

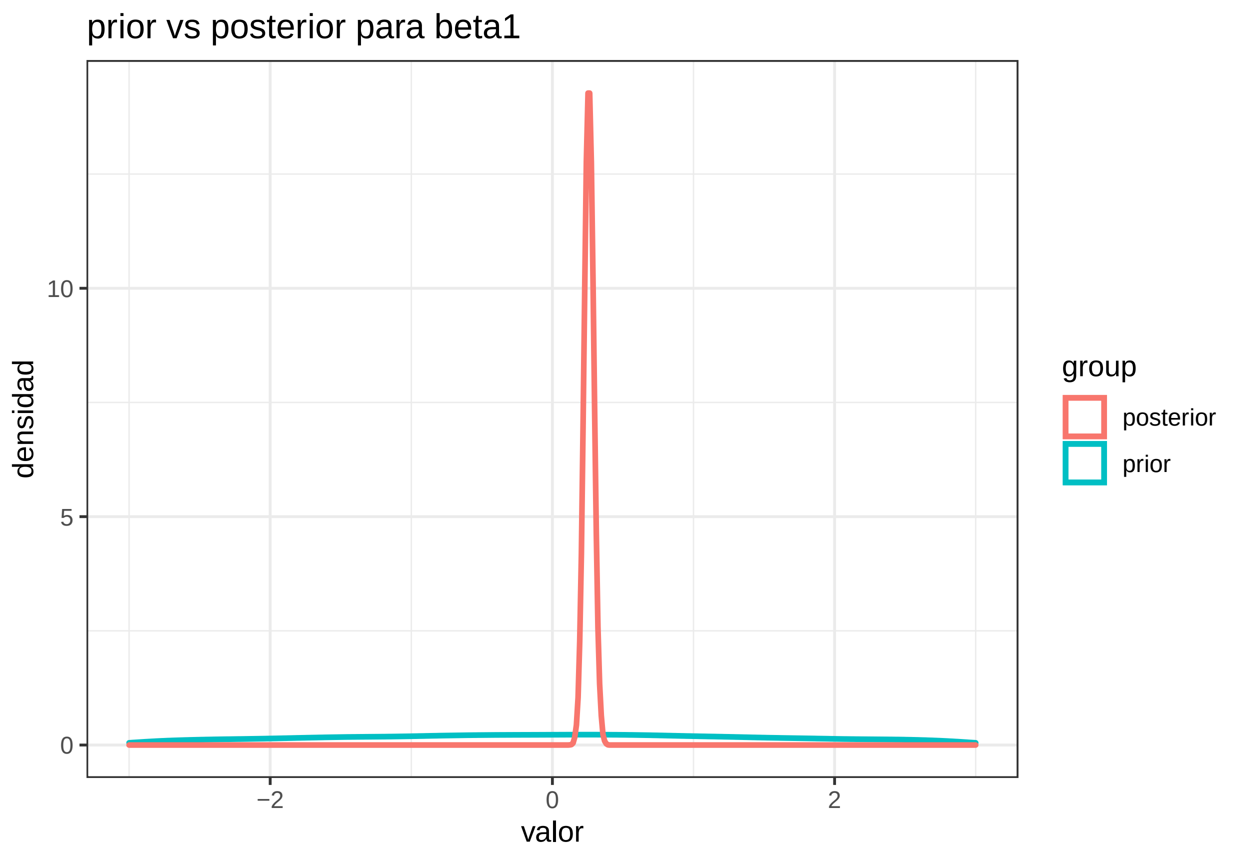
```
posterior <- data.frame(jagsfit$BUGSoutput$sims.matrix)
posterior$group <- 'posterior'

# prior <- data.frame(beta0=rnorm(nrow(posterior),mean=0,sd=10),
#                      beta1=rnorm(nrow(posterior),mean=0,sd=2.5),
#                      beta2=rnorm(nrow(posterior),mean=0,sd=2.5),
#                      group='prior')
prior <- data.frame(beta0=rcauchy(nrow(posterior),location=0,scale=10),
                    beta1=rcauchy(nrow(posterior),location=0,scale=2.5),
                    beta2=rcauchy(nrow(posterior),location=0,scale=2.5),
                    group='prior')



pbeta0 <- ggplot() +
   geom_density(data=prior,aes(beta0,color=group),lwd=1) +
   geom_density(data=posterior,aes(beta0,color=group),lwd=1) +
   xlim(-3,3) +
   labs(title='prior vs posterior para beta0',
        x='valor',
        y='densidad') +
   theme_bw()
```

```r
pbeta1 <- ggplot() +
   geom_density(data=prior,aes(beta1,color=group),lwd=1) +
   geom_density(data=posterior,aes(beta.1.,color=group),lwd=1) +
   xlim(-3,3) +
   labs(title='prior vs posterior para beta1',
        x='valor',
        y='densidad') +
   theme_bw()

pbeta2 <- ggplot() +
   geom_density(data=prior,aes(beta2,color=group),lwd=1) +
   geom_density(data=posterior,aes(beta.2.,color=group),lwd=1) +
   xlim(-3,3) +
   labs(title='prior vs posterior para beta2',
        x='valor',
        y='densidad') +
   theme_bw()



pbeta0
```

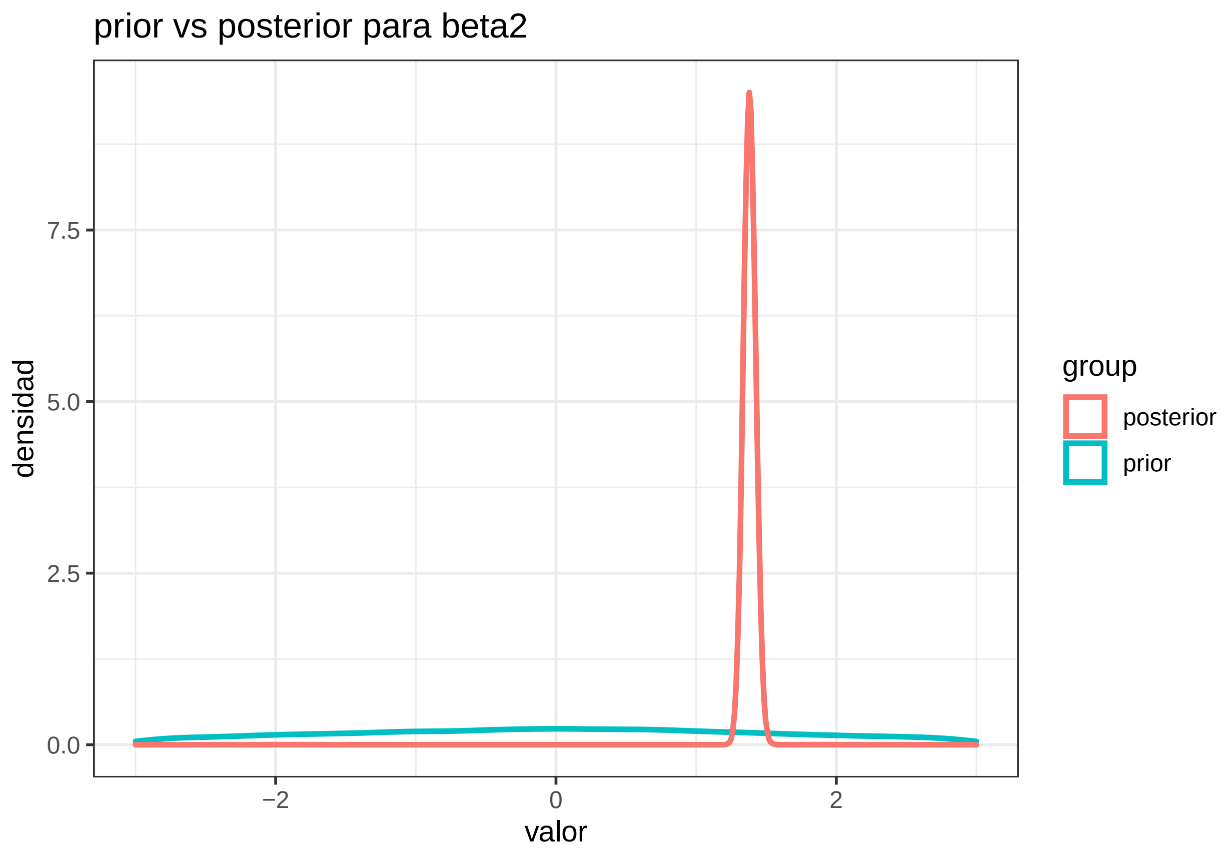


```r
# ggsave(filename=paste0(path,'\\pripostbeta0.jpg'),dpi=300)
```

```
pbeta1
```

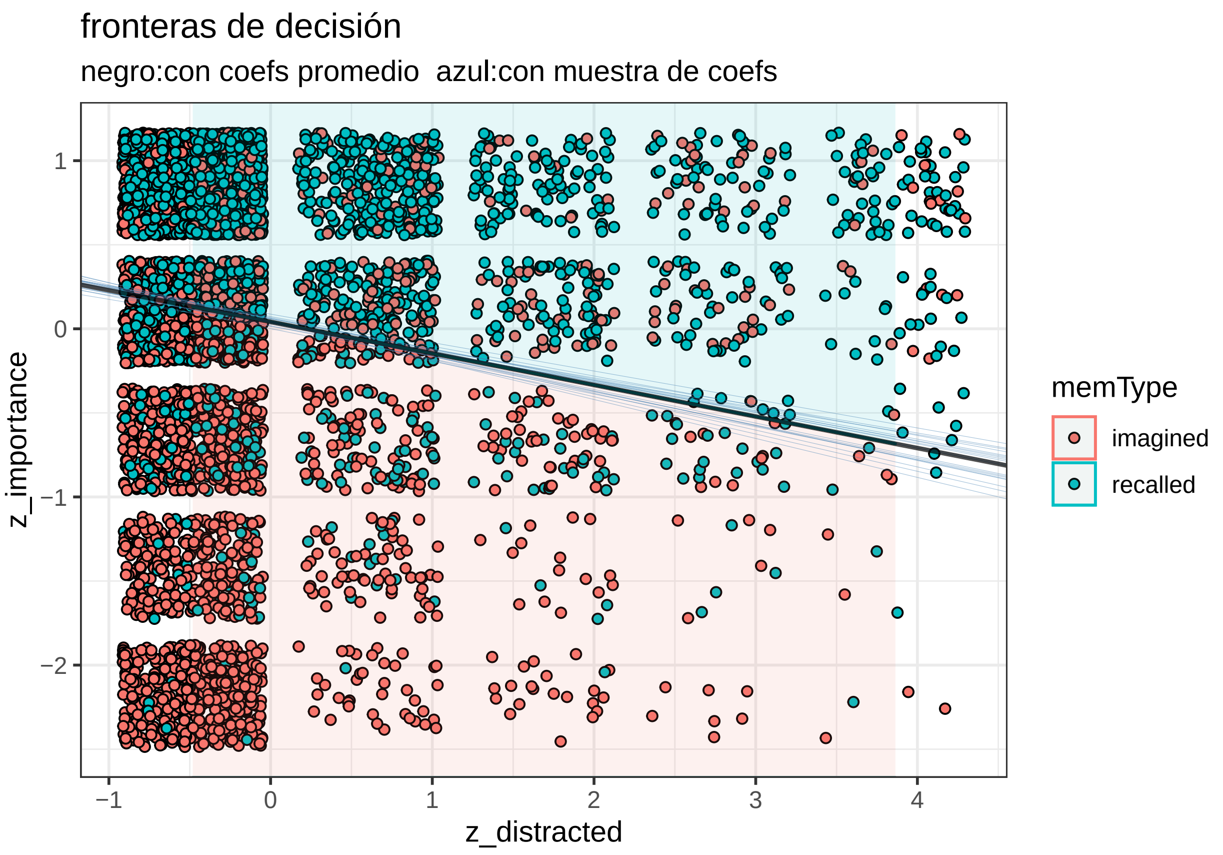


```
# ggsave(filename=paste0(path,'\\pripostbeta1.jpg'),dpi=300)

pbeta2
```



```r
# ggsave(filename=paste0(path,'\\pripostbeta2.jpg'),dpi=300)
```

```r
# frontera de decisión y = ax + b (importance = a*z_distracted + b)
mean_beta0 <- jagsfit$BUGSoutput$mean$beta0
mean_betas <- jagsfit$BUGSoutput$mean$beta

b <- c(-mean_beta0/mean_betas[2])
a <- c(-mean_betas[1]/mean_betas[2])


ids <- sample.int(nrow(posterior),size=20,replace=F)
sample_coefs <- posterior[ids,]
sample_coefs$bs <- -sample_coefs$beta0/sample_coefs$beta.2.
sample_coefs$as <- -sample_coefs$beta.1./sample_coefs$beta.2.

bound <- data.frame('dist'=c(-4,df$z_distracted,4),
                    'shadelim'=a*c(-4,df$z_distracted,4)+b)

ggplot(df,aes(x=z_distracted,y=z_importance,col=memType)) +
   geom_jitter(aes(fill=memType),col='black',shape=21) +
   geom_ribbon(ymin=-Inf,aes(ymax=a*z_distracted+b,xmax=Inf),fill='#F8766D',alpha=0.05) +
   geom_ribbon(aes(ymin=a*z_distracted+b),ymax=Inf,fill='#00BFC4',alpha=0.05) +
   geom_abline(data=sample_coefs,aes(slope=as,intercept=bs),lwd=0.1,col='steelblue',alpha=0.5) +
   geom_abline(aes(slope=a,intercept=b),lwd=0.8,col='black',alpha=0.7) +
   labs(title='fronteras de decisión',
      subtitle='negro:con coefs promedio  azul:con muestra de coefs') +
  theme_bw()
```



```
# ggsave(filename=paste0(path,'\\decbound_bayes.jpg'),dpi=300)
```

"Predicciones" y clasificación (con mismos datos)

```
meanbetas <- jagsfit$BUGSoutput$mean
linearpred <- as.matrix(cbind(1,df[c("z_distracted","z_importance")])) %*% c(meanbetas$beta0,
                                                                              meanbetas$beta)
preds <- 1/(1+exp(-linearpred))
classes <- ifelse(preds > 0.5,'recalled','imagined')
cat('\n accuracy:',mean(classes == df$memType))
```

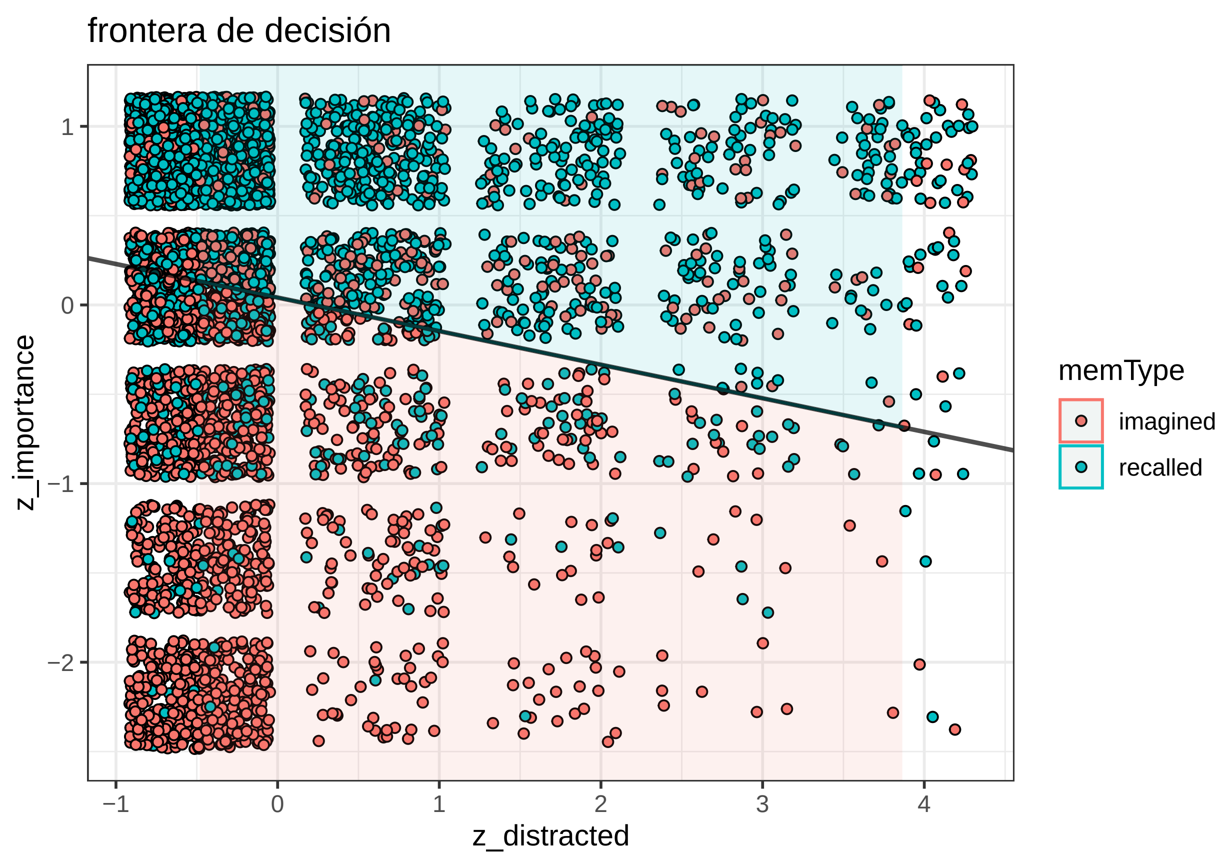
```
##
##  accuracy: 0.7288073
```

Enfoque frecuentista

```r
logit_model <- glm(memType ~ z_distracted +z_importance,family='binomial',data=df)
summary(logit_model)
```

```
##
## Call:
## glm(formula = memType ~ z_distracted + z_importance, family = "binomial",
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1194  -0.8572   0.5399   0.7892   2.5434
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.05698    0.03324  -1.714   0.0865 .
## z_distracted  0.25935    0.03369   7.698 1.38e-14 ***
## z_importance  1.38016    0.04345  31.768  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7468.3  on 5390  degrees of freedom
## Residual deviance: 5827.3  on 5388  degrees of freedom
## AIC: 5833.3
##
## Number of Fisher Scoring iterations: 4
```

```r
# frontera de decisión y = ax + b (importance = a*z_distracted + b)
b <- -coef(logit_model)[1]/coef(logit_model)[3]
a <- -coef(logit_model)[2]/coef(logit_model)[3]


ggplot(df,aes(x=z_distracted,y=z_importance,col=memType)) +
   geom_jitter(aes(fill=memType),col='black',shape=21) +
   geom_ribbon(ymin=-Inf,aes(ymax=a*z_distracted+b),fill='#F8766D',alpha=0.05) +
   geom_ribbon(aes(ymin=a*z_distracted+b),ymax=Inf,fill='#00BFC4',alpha=0.05) +
   geom_abline(aes(slope=a,intercept=b),lwd=0.8,col='black',alpha=0.7) +
   labs(title='frontera de decisión') +
   theme_bw()
```



```
# ggsave(filename=paste0(path,'\\decbound_freq.jpg'),dpi=300)
```