```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### A (CONJUGATE) linear regression example
### Example 2.15 - S. Jackman (2009)
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
## In November 1993, the state of Pennsylvania conducted elections for its state legislature.
## The result in the Senate election in the 2nd district (based in Philadelphia) was challenged
## in court, and ultimately overturned. The Democratic candidate won 19,127 of the votes cast
## by voting machine, while the Republican won 19,691 votes cast by voting machine, giving
## the Republican a lead of 564 votes. However, the Democrat won 1,396 absentee ballots, while
## the Republican won just 371 absentee ballots, more than offsetting the Republican lead based
## on the votes recorded by machines on election day. The Republican candidate sued, claiming
## that many of the absentee ballots were fraudulent. The judge in the case solicited expert
## analysis from Orley Ashenfelter, an economist at Princeton University. Ashenfelter examined
## the relationship between absentee vote margins and machine vote margins in 21 previous
## Pennsylvania Senate elections in seven districts in the Philadelphia area over the preceding decade.
## OUTLIER detection problem in the contect of simple linear
## regression in the Pennsylvania state senate election
## Response Y= abseentee vote margins (difference of
##             Democrat percentage and Republican percentage)
##             in the absentee ballot
## Absentee ballot = VOTO A DISTANZA
## See Jackman, from p. 87
## COVARIATE x = machine vote margins (difference of Democrat
##             percentage and Republican percentage)
##             (at the machine)
## R package pscl where you can find the dataset

library(pscl)      <- for the data
```

```
data(absentee)
help(absentee)      # A data frame with 22 observations on 8 variables.
```

```
summary(absentee)
```

```
##       year          district         absdem          absrep
##  Min.   :82.00   Min.   :1.000   Min.   : 308.0   Min.   :  70.0
##  1st Qu.:84.00   1st Qu.:2.000   1st Qu.: 561.8   1st Qu.: 216.2
##  Median :88.00   Median :4.000   Median : 680.5   Median : 321.5
##  Mean   :87.41   Mean   :4.182   Mean   : 797.1   Mean   : 506.1
##  3rd Qu.:90.00   3rd Qu.:6.500   3rd Qu.: 907.0   3rd Qu.: 673.5
##  Max.   :93.00   Max.   :8.000   Max.   :1923.0   Max.   :1610.0
##     machdem         machrep          dabs            dmach
##  Min.   :19127   Min.   : 3939   Min.   :-829.0   Min.   :-13194
##  1st Qu.:41832   1st Qu.:13030   1st Qu.: 175.8   1st Qu.: 12564
##  Median :53815   Median :20516   Median : 353.0   Median : 26237
##  Mean   :52120   Mean   :23445   Mean   : 291.0   Mean   : 28674
##  3rd Qu.:63757   3rd Qu.:28316   3rd Qu.: 399.2   3rd Qu.: 45240
##  Max.   :78270   Max.   :56721   Max.   :1329.0   Max.   : 71797
```

```
attach(absentee)

# Create variables for regression analysis
y <- (absdem - absrep)/(absdem + absrep)*100
x <- (machdem - machrep)/(machdem + machrep)*100
x[22]
```

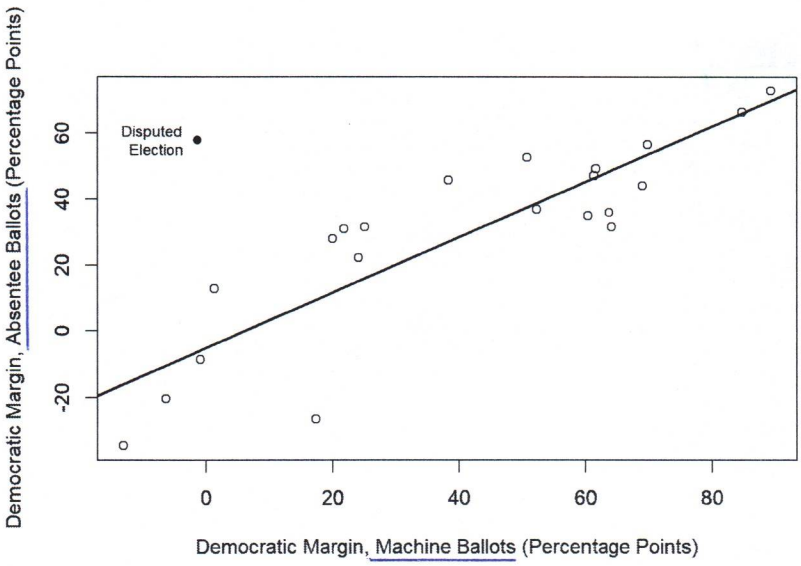
```
## [1] -1.452934
```

this couple (x[22], y[22]) is the debated point (the discussed point).
This point is weird w.r.t. the others: we will use 21 data to fit the model and then we'll predict y[22] given x[22].
Spoiler: we'll see that (x[22], y[22]) is in the tail of the predictive distribution (i.e. the predictive distribution cannot explain this point)

```
y[22]
```

```
## [1] 58.00792
```

```
# The 22nd datapoint is suspect!

## DATA PLOT
x11()
plot(y~x,type="n",xlim=range(x),ylim=range(y),
     xlab="Democratic Margin, Machine Ballots (Percentage Points)",
     ylab="Democratic Margin, Absentee Ballots (Percentage Points)")

ols <- lm(y ~ x, subset=c(rep(TRUE,21),FALSE)) ## OLS analysis of the
                                               ## dataset, but DROP data point 22

abline(ols,lwd=2)            ## overlay ols (retta di regressione stimata, senza il dato 22)
points(x[-22],y[-22],pch=1)  ## data
points(x[22],y[22],pch=16)   ## disputed data point
text(x[22],y[22],"Disputed\nElection",cex=.75,adj=1.25)
axis(1)
axis(2)
```




```
graphics.off()
summary(ols)
```

```
##
## Call:
## lm(formula = y ~ x, subset = c(rep(TRUE, 21), FALSE))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.973 -10.002   0.873  15.242  18.747
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.0849     5.4524  -0.933    0.363
## x             0.8395     0.1080   7.774 2.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.8 on 19 degrees of freedom
## Multiple R-squared:  0.7608, Adjusted R-squared:  0.7482
## F-statistic: 60.43 on 1 and 19 DF,  p-value: 2.559e-07
```

```
## DATA ANALYSIS WITHOUT the 22nd observation
y=y[1:21]
x=x[1:21]
X=as.matrix(cbind(rep(1,21),x))
n=length(y)
k=dim(X)[2] #lo abbiamo chiamato p alla lavagna!

## PARAMETERS
## beta0, beta1, sigma^2

## ORDINARY LEAST SQUARE ESTIMATION OF beta
inv=function(X){# RETURN THE INVERSE OF THE SYMMETRIC MATRIX X
  EV=eigen(X)
  EV$vector%*%diag(1/EV$values)%*%t(EV$vector)
}

inv(t(X)%*%X)
```

```
##              [,1]         [,2]
## [1,]  0.135703130 -2.165475e-03
## [2,] -0.002165475  5.323641e-05
```

```
solve(t(X)%*%X)  #use either these commands to compute the inverse of a square matrix
```

```
##                              x
##    0.135703130 -2.165475e-03
## x -0.002165475  5.323641e-05
```

```
betahat=inv(t(X)%*%X)%*%t(X)%*%y
betahat
```

```
##            [,1]
## [1,] -5.0848616
## [2,]  0.8395403
```

Std frequentist estimate

frequentist

```
## FREQUENTIST UNBIASED ESTIMATION OF sigma2
S2=t(y-X%*%betahat)%*%(y-X%*%betahat) ## SOMMA dei RESIDUI al quadrato
sigma2hat=S2/(n-k)
sigma2hat
```

```
##          [,1]
## [1,] 219.0732
```

```
sqrt(sigma2hat)
```

```
##          [,1]
## [1,] 14.80112
```

```
## ESTIMATION OF THE VARIANCE OF betahat
c(sigma2hat)*diag(inv(t(X)%*%X))
```

```
## [1] 29.72891311  0.01166267
```

```
## ESTIMATION OF THE sd OF betahat
sqrt(c(sigma2hat)*diag(inv(t(X)%*%X)))
```

```
## [1] 5.4524227 0.1079938
```

An alternative (we don't procede this way) is the **CONJUGATE PRIOR**.
We set $\underline{b}_0 = \underline{0}$, $c = \log(n)$, we try with some ν_0, σ_0^2 and we end up with a predictive distribution for $\underline{Y}^{new}$

$$Y_{zz}^{(new)} = (X_{zz})^T \underline{\beta} + \varepsilon^{new}$$

(we created a model using 21 data, we obtained a predictive distribution $\Sigma(Y^{new} | y, X, X^{new})^*$ and we predict the value y for x[22]. Then we can compare the predicted y with y[22])

* we know that in the conjugate prior case this is a t distribution

→ WE USE **JAGS**

```
### ------------------------------------------------------------------------
###                 JAGS use
###         Just Another Gibbs Sampler
### ------------------------------------------------------------------------
###    http://mcmc-jags.sourceforge.net/
###    WEB SITE: info and instructions on JAGS downloading
### ------------------------------------------------------------------------
## JAGS is Just Another Gibbs Sampler. It is a program for the analysis of Bayesian models
## using Markov Chain Monte Carlo (MCMC) which is not wholly unlike
## OpenBUGS (http://www.openbugs.info).
## JAGS was written with three aims in mind: to have an engine for the BUGS language that runs on Unix;
## to be extensible, allowing users to write their own functions, distributions, and samplers;
## and to be a platform for experimentation with ideas in Bayesian modelling.
## JAGS is designed to work closely with the R language and environment for statistical
## computation and graphics (http://www.r-project.org).
## You will find it useful to install the coda package for R to analyze the output.
## You can also use the rjags package to work directly with JAGS from within R.
```

```
rm(list=ls())
detach(absentee)
library(rjags)   # to interface R with JAGS
```

BAYESIAN APPROACH

```
library(pscl)
data(absentee)
# help(absentee)
summary(absentee)
```

```
##       year          district        absdem          absrep
##  Min.   :82.00   Min.   :1.000   Min.   : 308.0   Min.   :  70.0
##  1st Qu.:84.00   1st Qu.:2.000   1st Qu.: 561.8   1st Qu.: 216.2
##  Median :88.00   Median :4.000   Median : 680.5   Median : 321.5
##  Mean   :87.41   Mean   :4.182   Mean   : 797.1   Mean   : 506.1
##  3rd Qu.:90.00   3rd Qu.:6.500   3rd Qu.: 907.0   3rd Qu.: 673.5
##  Max.   :93.00   Max.   :8.000   Max.   :1923.0   Max.   :1610.0
##     machdem         machrep          dabs            dmach
##  Min.   :19127   Min.   : 3939   Min.   :-829.0   Min.   :-13194
##  1st Qu.:41832   1st Qu.:13030   1st Qu.: 175.8   1st Qu.: 12564
##  Median :53815   Median :20516   Median : 353.0   Median : 26237
##  Mean   :52120   Mean   :23445   Mean   : 291.0   Mean   : 28674
##  3rd Qu.:63757   3rd Qu.:28316   3rd Qu.: 399.2   3rd Qu.: 45240
##  Max.   :78270   Max.   :56721   Max.   :1329.0   Max.   : 71797
```

```
attach(absentee)

## create variables for regression analysis
y <- (absdem - absrep)/(absdem + absrep)*100
x <- (machdem - machrep)/(machdem + machrep)*100
x.sosp=x[22]
y.sosp=y[22]
y=y[1:21]
x=x[1:21]
## 21 (bivariate) data points
```

```
## JAGS takes a user's description of a Bayesian model for data,
## and returns an MCMC sample of the posterior distribution
```

we specify the Bayesian model and we also pass the data ⟹ **Jags** returns an MCMC sample which is an approximation of our posterior distribution

```
## Define the data (in this case, datapoints (y_i), the covariate vector (x_i), the sample size,
## the new covariate value xstar) in a list
data = list(y=y[1:21],
            x=x[1:21],
            n=21,
            xstar=x.sosp)
```

```
## A list of initial value for the MCMC algorithm that JAGS will implement
inits = function() {list(beta=c(0,0),
                         sigma=5,
                         ystar=0) }
## JAGS can automatically inizialize the chain, but the efficiency of the MCMC can be improved
## if we intelligently provide reasonable starting values, i.e. values in the midst of the
## posterior, e.g. MLE, or values close to the MLE
```

```
## The Bayesian model is in the text file "regression.bug". This file, in addition to the list
## "data", is taken as an input by JAGS for generating the MCMC in 3 steps.
## * The first step (jags.model) gets all the info into JAGS and let JAGS figure out
##   appropriate sampler for the model.
## * The second step (update) runs the chain for a burn-in period.
## * The third step (coda.samples) runs and records the MCMC sample we will
##   subsequently examine (using R).

1. modelRegress=jags.model("regression.bug",data=data,inits=inits,n.adapt=1000,n.chains=1)
```

we can run chains in parallel

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 4
##    Total graph size: 98
##
## Initializing model
```

```
### Command jags.model compiles and initializes the model described in the text file
### "regression.bug" (it is a text file, but we use *.bug to understand what file it is,
### but we could also name it *.txt); as an input, we have "data" and JAGS then run
### the sampler for n.adapt iterations. By default n.adapt=1000. When a JAGS model
### is compiled, it may require an initial sampling phase during which the samplers adapt
### their behaviour to maximize their efficiency (e.g. a Metropolis-Hastings random walk
### algorithm may change its step size). The sequence of samples generated during
### this adaptive phase is not a Markov chain, and therefore may not be used for
### posterior inference on the model.
### File "regression.bug" specify both the likelihood and the prior;
### beta has 2 components which are iid Gaussian, with mean = 0 and variance = 10thousand
### sigma is Uniform on the interval (0,100)
### In this case, the PRIOR IS NOT CONJUGATE.
### From rjags manual:
# jags.model is used to create an object representing a Bayesian graphical model,
# specified with a BUGS-language description of the prior distribution, and a set of data.
### USAGE:
# jags.model(file, data=sys.frame(sys.parent()), inits, n.chains = 1, n.adapt=1000, quiet=FALSE)
# Se inits è omessa, i valori iniziali saranno specificati automaticamente. ATTENZIONE:
# talvolta la catena si blocca subito se inits NON è BEN specificato (anche nel caso in cui
# venga calcolato automaticamente)
# Durante le prime n.adapt iterazioni, la catena è adattativa, per es. potrebbe NON essere
# markoviana, o la proposal distribution potrebbe cambiare
```

```
# save(modelRegress,file='modelregress.Rdata')

2. update(modelRegress,n.iter=19000) #this is the burn-in period
## this function DOES NOT record the sampled values of parameters
```

we suppose that after this we're not very far from the stationary distribution. We tell Jags to start collecting the generated samples.

```
## THIRD STEP: we tell JAGS to generate MCMC samples that we will use
## to represent the posterior distribution
variable.names=c("beta", "sigma", "ystar") # parameters - see the file .bug - that will have
                                           # their values recorded

n.iter=50000
thin=10
## the final sample size, i.e. the number of iterations to build the ergodic average, will be 5K

3. outputRegress = coda.samples(model          = modelRegress,
                                variable.names = variable.names,
                                n.iter         = n.iter,
                                thin           = thin)
# Lancio la catena per altre n.iter iterazioni, specificando l'eventuale thinning;
# salvo l'intera catena dei parametri monitorati;
# the OUTPUT is mcmc.list object - coda-formatted object;
# it needs to be converted into a matrix, in order to be "readable".
```

because thin = 10 means that we save 1 sample (value) out of 10 ⇒ 50.000 iters / 10

"regression.bug"

in Jags the 2nd parameter is the reciprocal of the variance !

```
model{                                  ## regression model for y
    for(i in 1:n){                        # loop
          mu[i] <- beta[1] + beta[2]*x[i] # <- is a deterministic assignment
          y[i]  ~ dnorm(mu[i],tau)        # ~ denotes "distributed as"
                                          # BE CAREFUL, tau is the PRECISION of the Gaussian distribution
    }                                     # Manual JAGS, Ch 6: list of the distributions
    ## priors
    beta[1] ~ dnorm(0,.0001)   ## intercept
    beta[2] ~ dnorm(0,.0001)   ## slope
    sigma ~ dunif(0,100)       ## residual std dev  # Gelman (2006) suggests this noninformative prior
                                                    # in the context of hierarchical models (grouped data)
    tau <- pow(sigma,-2)       ## convert to precision

    ## out of sample predicton for suspect case
    mustar <- beta[1] + beta[2]*xstar
    ## posterior predictive density for y[22] given
    ## historical data obs 1 - 21 and x[22]
    ystar ~ dnorm(mustar,tau)
}
```

Model that we're assuming:

$$Y_i = \beta[1] + \beta[2] \cdot X_i + \varepsilon_i \qquad \varepsilon_i \overset{iid}{\sim} N(0,\sigma^2) = N\left(0, \frac{1}{\tau}\right)$$

the **prior** is:

$$\underline{\beta} = (\beta[1], \beta[2]) \sim N_2(\underline{0}, 10000 I_2)$$

$$\sigma \sim \mathcal{U}([0,\sigma_0]) \qquad (:= \text{Gellman's proposal (2006)})$$

```
# SUMMING UP, to produce the output (the MCMC chain) using rjags 4 steps are required:
# 1. Define the model using the BUGS language in a separate file.
# 2. Read in the model file using the jags.model function. This creates an object of
#    class "jags".
# 3. Update the model using the update method for "jags" objects. This constitutes a
#    "burn-in" period.
# 4. Extract samples from the model object using the coda.samples function. This creates
#    an object of class "mcmc.list" which can be used to summarize the posterior
#    distribution. The coda package also provides convergence diagnostics to check that
#    the output is valid for analysis (see Plummer et al 2006).

# save(outputRegress,file='Jackman_regr_output.Rdata')
```

```
## ANALISI DELL'OUTPUT
library(coda)       # per leggere catena mcmc
library(plotrix)    # per fare plot CIs

## CARICA le traiettorie della catena
## load('Jackman_regr_output.Rdata')

data.out=as.matrix(outputRegress) # trasform the mcmc.list into a matrix,
data.out=data.frame(data.out)     # or, better, into a dataframe (easiest to handle in R)
attach(data.out)
n.chain=dim(data.out)[1]
n.chain # this is the final sample size
```

```
## [1] 5000
```

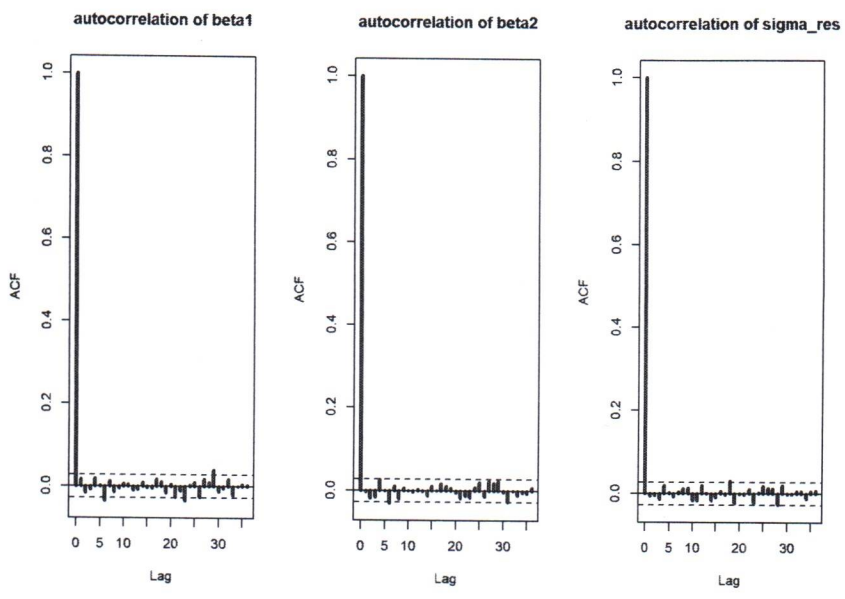
```
## Summary statistics for the posterior
summary(data.out)
```

```
##     beta.1.           beta.2.           sigma            ystar
##  Min.   :-32.129   Min.   :0.3378   Min.   : 9.522   Min.   :-71.578
##  1st Qu.: -8.703   1st Qu.:0.7636   1st Qu.:13.837   1st Qu.:-16.920
##  Median : -5.060   Median :0.8366   Median :15.401   Median : -5.574
##  Mean   : -5.064   Mean   :0.8384   Mean   :15.798   Mean   : -5.797
##  3rd Qu.: -1.323   3rd Qu.:0.9136   3rd Qu.:17.342   3rd Qu.:  4.964
##  Max.   : 18.021   Max.   :1.3710   Max.   :31.283   Max.   : 63.128
```

```
head(data.out)
```

```
##       beta.1.   beta.2.    sigma      ystar
## 1 -17.968734 1.0109528 25.51431   8.369565
## 2   4.346455 0.7418041 14.46945  -7.200389
## 3  -1.542601 0.8379011 18.81427 -41.308706
## 4  -4.690946 0.8197588 17.23902  12.088540
## 5  -7.292059 0.8563665 15.44180  48.286028
## 6  -6.203336 0.8001956 13.87144 -25.952665
```

```
## Use either packege CODA, or standard tools in R

x11()
par(mfrow=c(1,3))
acf(data.out[,'beta.1.'],lwd=3,col="red3",main="autocorrelation of beta1")
acf(data.out[,'beta.2.'],lwd=3,col="red3",main="autocorrelation of beta2")
acf(data.out[,'sigma'],lwd=3,col="red3",main="autocorrelation of sigma_res")
```



(These are too good as autocorrelation plots: apart from some cases, these autocorrelations are below the 5% threshold (that for coda means auto=0))

```
## Some nice graph to presents the posterior samples
```

```
# sub-chain containing the beta sample
beta.post <- data.out[,1:2]
# posterior mean of the beta parameters
beta.bayes  <- apply(beta.post,2,"mean")
beta.bayes
```

```
##    beta.1.    beta.2.
## -5.0637685  0.8384359
```

→ now we want to compare them with the std frequentist estimations

```
# sub-chain whit the sigma_res samle
sig.post= data.out[,'sigma']
```

```
## MLE estimate
pippo= lm(y ~ x)
summary(pippo)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.973 -10.002   0.873  15.242  18.747
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.0849     5.4524  -0.933    0.363
## x             0.8395     0.1080   7.774 2.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.8 on 19 degrees of freedom
## Multiple R-squared:  0.7608, Adjusted R-squared:  0.7482
## F-statistic: 60.43 on 1 and 19 DF,  p-value: 2.559e-07
```

```
pippo$coefficients
```

(MLE)

```
## (Intercept)           x
##  -5.0848616   0.8395403
```

very similar but not equivalent

```
beta.bayes
```

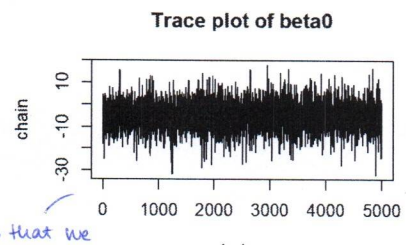
(Bayesian)

```
##    beta.1.    beta.2.
## -5.0637685  0.8384359
```

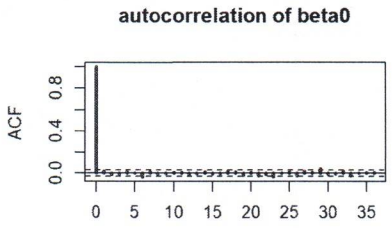
```
## Representation of the posterior chain of  beta0
chain <- beta.post[,1]
# Divide the plot device in three sub-graph regions
# two square on the upper and a rectangle on the bottom
x11()
layout(matrix(c(1,2,3,3),2,2,byrow=T))
# trace-plot of the posterior chain
plot(chain,type="l",main="Trace plot of beta0")
# autocorrelation plot
acf(chain,lwd=3,col="red3",main="autocorrelation of beta0")
# Histogram
hist(chain,nclass="fd",freq=F,main="Posterior of beta0",col="gray")
## Overlap the kernel-density
lines(density(chain),col="blue",lwd=2)
## Posterior credible interval of beta0
quantile(chain,prob=c(0.025,0.5,0.975))
```

```
##       2.5%        50%      97.5%
## -16.783115  -5.059833   6.424498
```

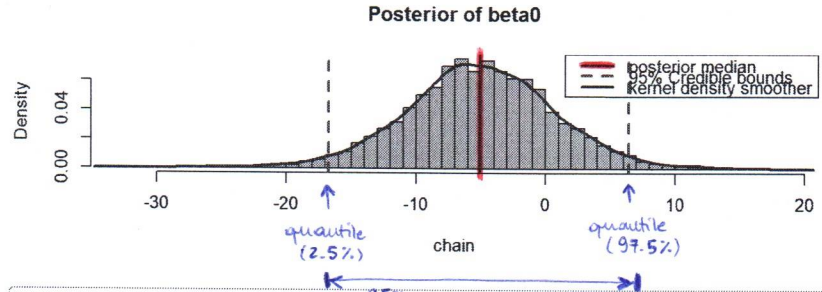
```
## Display the posterior credible interval on the graph
abline(v=quantile(chain,prob=c(0.025)),col="red",lty=2,lwd=2)
abline(v=quantile(chain,prob=c(0.5)),col="red",lty=1,lwd=2)
abline(v=quantile(chain,prob=c(0.975)),col="red",lty=2,lwd=2)
## Add the legend to the plot
legend("topright",legend=c("posterior median", "95% Credible bounds","kernel density smoother"),lwd=c(2,2,2), col=c("red","r
ed","blue"),lty=c(1,2,1))
```



it seems that we reached stationarity since the support is constant through iterations



very good autocorrelation plot

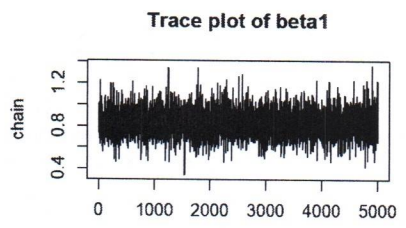


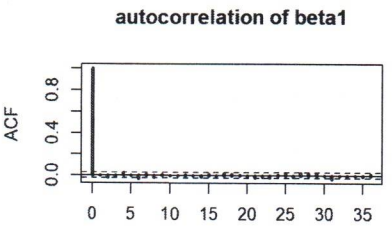
95% posterior credible interval

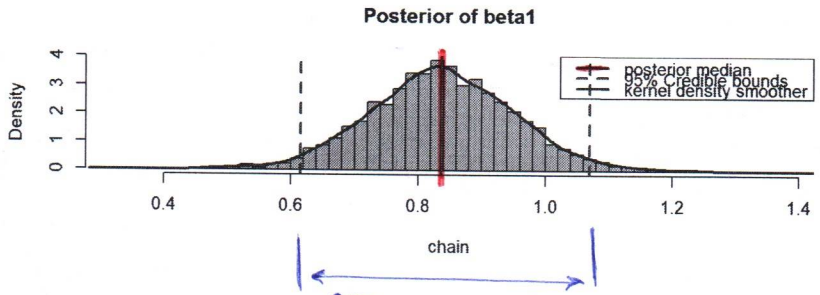
```
x11()
## Representation of the posterior chain of beta1
chain <- beta.post[,2]
# Divide the plot device
layout(matrix(c(1,2,3,3),2,2,byrow=T))
# trace-plot of the posterior chain
plot(chain,type="l",main="Trace plot of beta1")
# autocorrelation plot
acf(chain,lwd=3,col="red3",main="autocorrelation of beta1")
# Histogram
hist(chain,nclass="fd",freq=F,main="Posterior of beta1",col="gray")
## Overlap the kernel-density
lines(density(chain),col="blue",lwd=2)
## Posterior credible interval of  beta1
quantile(chain,prob=c(0.025,0.5,0.975))
```

```
##      2.5%       50%     97.5%
## 0.6146442 0.8365779 1.0703441
```

```
## Display the posterior credible interval on the graph
abline(v=quantile(chain,prob=c(0.025)),col="red",lty=2,lwd=2)
abline(v=quantile(chain,prob=c(0.5)),col="red",lty=1,lwd=2)
abline(v=quantile(chain,prob=c(0.975)),col="red",lty=2,lwd=2)
## Add the Legend to the plot
legend("topright",legend=c("posterior median", "95% Credible bounds","kernel density smoother"),lwd=c(2,2,2), col=c("red","red","blue"),lty=c(1,2,1))
```







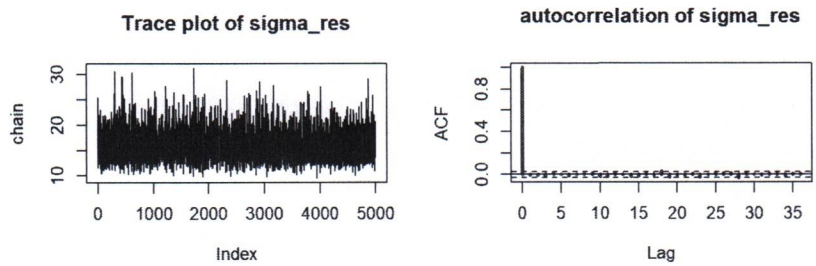
95% posterior credible interval

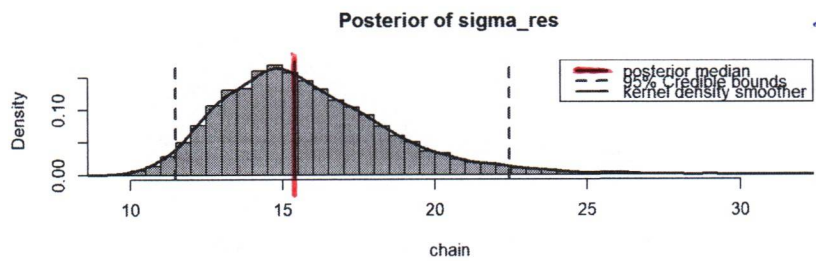
(we see that the 0 is not in it, we will use it as a criterium to say that this parameter is significant (in a test : $\beta_1 = 0$ vs. $\beta_1 \neq 0$ in this case we would say that $\beta_1 \neq 0$ (from a bayesian point of view)))

```
x11()
## Representation of the posterior chain of sigma_res
chain <-sig.post
# Divide the plot device
layout(matrix(c(1,2,3,3),2,2,byrow=T))
# trace-plot of the posterior chain
plot(chain,type="l",main="Trace plot of sigma_res")
# autocorrelation plot
acf(chain,lwd=3,col="red3",main="autocorrelation of sigma_res")
# Histogram
hist(chain,nclass="fd",freq=F,main="Posterior of sigma_res",col="gray")
## Overlap the kernel-density
lines(density(chain),col="blue",lwd=2)
## Posterior Credible bound of sigma_res
quantile(chain,prob=c(0.05,0.5,0.95))
```

```
##       5%      50%      95%
## 11.99671 15.40088 20.99817
```

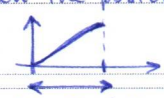
```
## Display the posterior credible interval on the graph
abline(v=quantile(chain,prob=c(0.025)),col="red",lty=2,lwd=2)
abline(v=quantile(chain,prob=c(0.5)),col="red",lty=1,lwd=2)
abline(v=quantile(chain,prob=c(0.975)),col="red",lty=2,lwd=2)
## Add the legend to the plot
legend("topright",legend=c("posterior median", "95% Credible bounds","kernel density smoother"),lwd=c(2,2,2), col=c("red","red","blue"),lty=c(1,2,1))
```





Remember that apriori the support for σ was $[0, 100]$.
A posteriori the support is the same, however we haven't see in our MCMC values larger than 35.
What we can do is re-run the chain assuming as a prior for σ not $\mathcal{U}([0,100])$ but, for instance, $\mathcal{U}([0,50])$

Note: usually this is done the other way: we impose some support for the prior and then the posterior comes out like:



support of the prior

$\Rightarrow$ in this case we go back to the prior and we increase the support (and we re-run everything)

we should always try to find a good support: not too **large**, not too **small**.

```
#close all the graphic windows
graphics.off()
```

```
### Predictive distibution of Y_new at x=xstar=suspected value
### "ystar" column in the output contains one simulated value
### from N(beta0+beta1*xstar,1/tau) for each
### value (beta0,beta1,tau) simulated a posteriori, i.e.
### "ystar" is a MCMC sample from the predictive distribution
### of Ynew corresponding at xstar
ystar.pred=data.out[,'ystar']

## Representation of the chain y.star
x11()
chain <- ystar.pred
# Divide the plot device in three sub-graph regions
# two square on the upper and a rectangle on the bottom
layout(matrix(c(1,2,3,3),2,2,byrow=T))
# trace-plot of the posterior chain
plot(chain,type="l",main="Trace plot of y.star")
# autocorrelation plot
acf(chain,lwd=3,col="red3",main="autocorrelation of ystar")
# Histogram
hist(chain,nclass="fd",freq=F,main="Posterior of ystar",col="gray")
## Overlap the kernel-density
lines(density(chain),col="blue",lwd=2,xlim=c(0,60))
## Posterior credible interval of ystar
quantile(chain,prob=c(0.025,0.5,0.975))
```

The predictive distribution is:

$$\mathcal{L}(Y^{new} | x^{new}, y, X) = \int \mathcal{L}(Y^{new} | x^{new}, \beta, \sigma^2)\, \pi(\beta, \sigma^2 | y, X)\, d\beta\, d\sigma^2$$

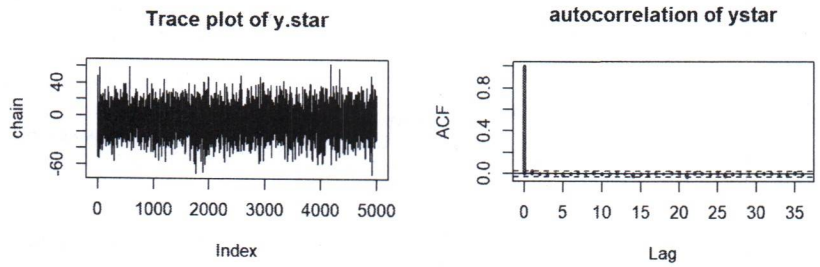
likelihood depending on parameters, and these parameters are the simulated values from the posterior distribution. We sample from this as:

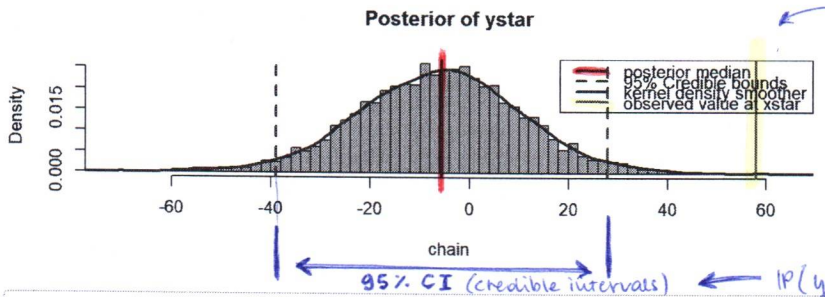
$$\mathcal{L}(Y^{new} | x^{new}, \beta^{(j)}, \sigma^{2(j)})$$

MCMC simulated values

```
##       2.5%        50%      97.5%
## -39.046881  -5.574413  27.807360
```

```
## Display the posterior credible interval on the graph
abline(v=quantile(chain,prob=c(0.025)),col="red",lty=2,lwd=2)
abline(v=quantile(chain,prob=c(0.5)),col="red",lty=1,lwd=2)
abline(v=quantile(chain,prob=c(0.975)),col="red",lty=2,lwd=2)
## Add the legend to the plot
# Aggiungiamo una retta relativa all'osservazione di
# y[22]=y.sops
abline(v=y.sosp,col="magenta",lwd=2)

legend("topright",legend=c("posterior median", "95% Credible bounds","kernel density smoother", "observed value at xstar"),
 lwd=c(2,2,2,2), col=c("red","red","blue","magenta"), lty=c(1,2,1,1))
```





the value is not contained in the CI 95%, it's in the tail of the distribution
⟹ the model is not able to explain this datum
⟹ this datum is WEIRD (OUTLIER)

95% CI (credible intervals) ← $\mathbb{P}(y.star \in CI_{95\%}) = 0.95$
(since are credible intervals, not confidence intervals)

```
### The observed value at xstar is OUTSIDE the 95% CI of the
### predictive distribution, i.e. it is
### rather extreme w.r.t. the predictive,
### i.e. it is in the tails of the predictive distribution...
### IT IS pretty suspect!!
```

```
###  Credible band for the regression line
### For any x in a grid of point, we compute 90% CI of Y_new(x)| x
### Valori sui quali andremo a valutare la retta di regressione bayesiana,
### cioè la retta y = E[ beta[1]|dati]+E[ beta[2]|dati]*x

x.gr=seq(-50,100,length=200)

# Pred is a matrix: each column contains MCMC values from the predictive distribution of Y_new(x)| x
# each row is a draw from the predictive of the regression line (as a function of x)
Pred <- matrix(ncol=200,nrow=n.chain)
for(i in 1:n.chain){
    #print(i)
    Pred[i,]=beta.post[i,1]+beta.post[i,2]*x.gr+rnorm(1,mean=0,sd=sig.post[i])
# For any value of x.gr, we generate from the cond distribution of data, given
# the current value of the parameter vector
# This amounts to sample from the posterior predictive distribution of Y_x^new, given x.gr=x.
}
dim(Pred)
```

Considering *: we're sampling from * $(\underline{Y}(Y^{new}|x^{new}, \underline{\beta}^{(j)}, \sigma^{2(j)})$, where x^{new} is the whole grid).
This means that we're doing:
$\underline{\beta}^{(j)}(\underline{x}^{new})^T + \varepsilon^{new}$
simulated from $N(0, \sigma^{2(j)})$

```
## [1] 5000  200
```

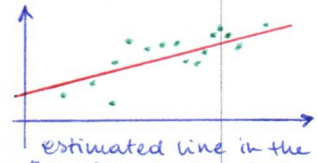
```
head(Pred)
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,]   1.780242   2.542267   3.304292   4.066316   4.828341   5.590366
## [2,] -46.624667 -46.065518 -45.506369 -44.947220 -44.388072 -43.828923
## [3,] -68.065030 -67.433446 -66.801862 -66.170279 -65.538695 -64.907111
## [4,] -62.787926 -62.170018 -61.552109 -60.934200 -60.316292 -59.698383
## [5,] -79.650514 -79.005012 -78.359509 -77.714007 -77.068505 -76.423002
## [6,] -41.740988 -41.137825 -40.534663 -39.931500 -39.328338 -38.725175
##           [,7]       ..
## [1,]   ..
```

```
x11()
layout(matrix(c(1,2,3,3),1,1,byrow=T))
### Plot the mean (over columns) of  all these values
predit=apply(Pred,2,"mean") # this is the mean of the predictive distribution for any fixed x.gr
plot(x.gr,predit,type="l",col="red",lwd=2)
# this is a line:
# y = E(beta0|data)+E(beta_1|data)*x
# This is the Bayesian estimate of the regression line

### Questa che ho disegnato con i comandi qui sopra è una retta, perchè è la retta che associa
### ad ogni valore di x.gr il valore E(Y^*(x.gr)|dati)= E[E(Y^*(x.gr)|par,dati)|dati]
###                                                  = E[ beta[1]+beta[2]*x.gr |dati]
###                                                  = E[ beta[1]|dati]+E[ beta[2]|dati]*x.gr
points(x,y,col="green",pch="*",cex=2) #datapoints in green!
gra=gray(1:100/100)
gra=rep(gra,10)

#### Disegno le prime 600 rette di regressione simulate
for(i in 1:600){lines(x.gr,Pred[i,],col=gra[i])}
### These are lines, since we used the same simulated value
### rnorm(1,mean=0,sd=sig.post[i]) for all x.gr
### Remember that we have defined
### Pred[i,]=beta.post[i,1]+beta.post[i,2]*x.gr+rnorm(1,mean=0,sd=sig.post[i])

predit.qua=apply(Pred,2,"quantile",prob=c(0.05,0.95))
### QUANTILES of each predictive distribution of  Y_new(x)| x
### the function "quantile" has been applied column by column
### For any value of x.gr in the grid, we compute 3 quantiles
### of the (simulated) distribution of y=beta0+beta1*x.gr+err
### and then we plot it as a function of x.gr
lines(x.gr,predit.qua[1,],col="red",lwd=3,lty=2)
### THIS IS NOT a line! the function "quantile" is NOT linear
lines(x.gr,predit.qua[2,],col="red",lwd=3,lty=2)
### THIS IS NOT a line!
lines(x.gr,predit,type="l",col="red",lwd=3)
### this IS a line, i.e. it is the Bayesian regression line computed before
### Invece le curve dei quantili sono curve (NON SONO NECESSARIAMENTE RETTE) che rappresentano
### le bande di credibilità della retta di regressione  y = E[ beta[1]|dati]+E[ beta[2]|dati]*x
### Se "affetto" questo plot con una retta x= x0, ottengo la distribuzione predittiva
### corrispondente a x0

points(x.sosp,y.sosp,pch=16,col="magenta") ## disputed data point
text(x.sosp,y.sosp,"Disputed\nElection",cex=.75,adj=1.25,col="magenta")
```

Instead of plotting the frequentist:
$y = \hat{\beta}_0 + \hat{\beta}_1 x$
we plot the Bayesian estimate:
$y = \mathbb{E}[\beta_0 | y, X] + \mathbb{E}[\beta_1 | y, X]\, x$

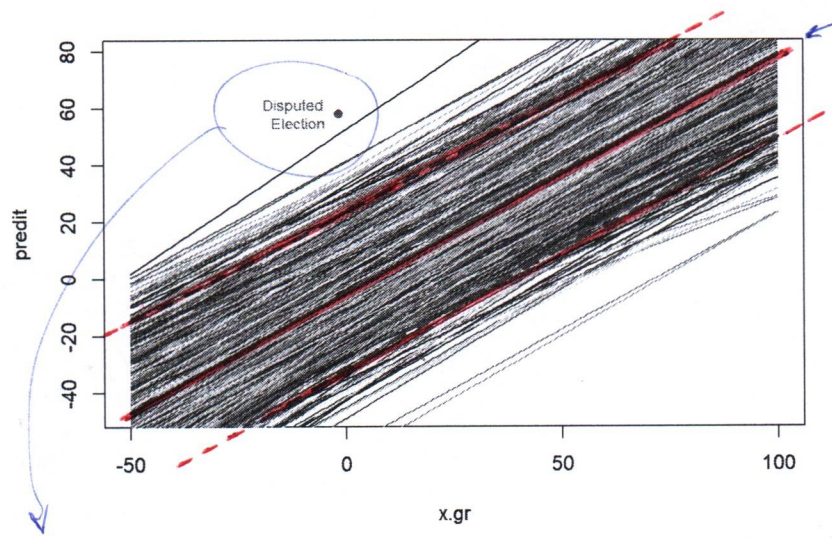
[missing plot]



estimated line in the Bayesian context, instead of the frequentist context

as a function of x.gr we have a function for any set of coefficients

(we can, in principle, plot any of the 5000 lines, however since 5000 is too much we plot only 600 (Note: these are the lines that right before we were averaging for a single bayesian line estimation))



estimated regression line in the bayesian context

The dashed (- - -) lines are not actually lines, they're the union of the quantiles (the quantiles are not disposed perfectly linearly in general)

Those are the quantiles meaning that: if we fix x then we'll have MCMC samples from the predictive distribution of the response (corresponding to these values of the covariates); then we evaluate the quantiles. We procede for every x and we get this band.

⇒ **CREDIBILITY BANDS**

They give us an idea on where should be the data if the model shows a good fit for the data.

very far from the whole credibility band

Once we've fitted the model, how can we measure the goodness of the fit?

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### LINEAR REGRESSION with iid N(0,sigma^2) errors
### Section 9.2.6 Albert's book
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
library(LearnBayes) # DO NOT USE LearnBayes package for your project (only educational purpose)
```

(Predictive goodness of fit)

```
data(birdextinct)
attach(birdextinct)
birdextinct
```

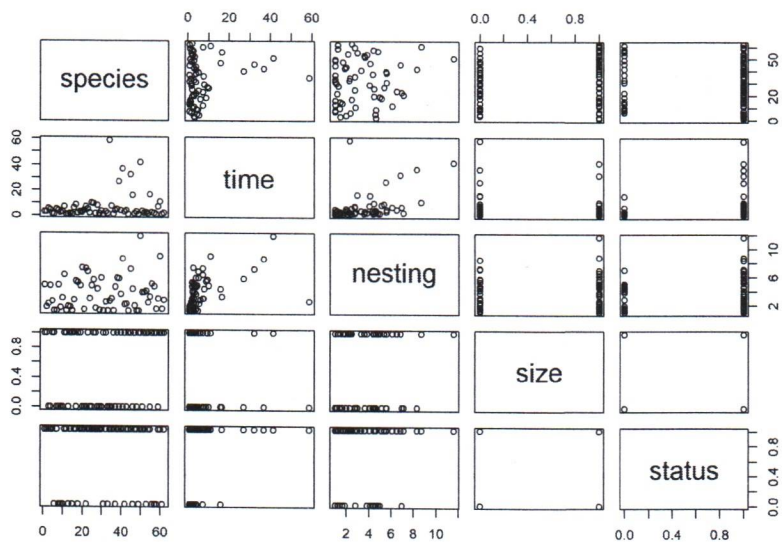
average time of extinction on the island

1/0 if the species is large or not

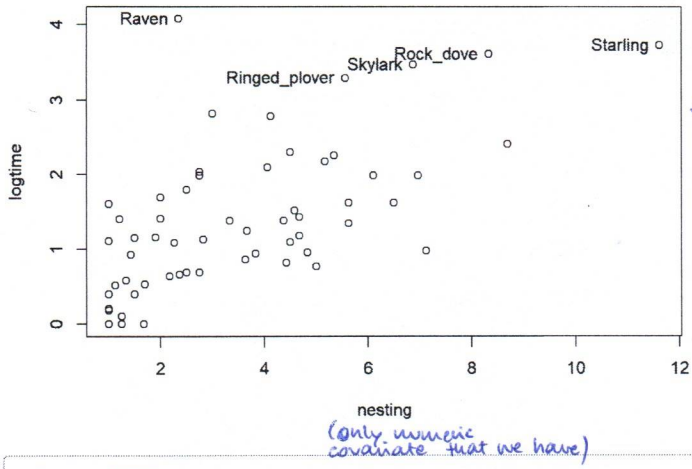
1 if it's resident, 0 if it's migrant

```
##                 species  time nesting size status
## 1           Sparrowhawk 3.030   1.000    0      1
## 2               Buzzard 5.464   2.000    0      1
## 3               Kestrel 4.098   1.210    0      1
## 4             Peregrine 1.681   1.125    0      1
## 5        Grey_partridge 8.850   5.167    0      1
## 6                 Quail 1.493   1.000    0      0
## 7  Red-legged_partridge 7.692   2.750    0      1
## ..
## 60             Starling 41.667 11.620    1      1
## 61      Pied_flycatcher 1.000   1.000    1      0
## 62               Siskin 1.000   1.000    1      1
```

```
# The DATASET concerns EXTINCTION of BIRDS;
# Measurements on breedings ("in cova") pairs of landbird species were collected from 16 islands
# about Britain over several decades. For each pairs (62 in total), the dataset records:
#   SPECIES = name of bird species
#   TIME    = average time of extinction on the islands
#   NESTING = average number of nesting pairs
#   SIZE    = size of the species, 1 or 0 if large or small
#   STATUS  = status of the species, 1 or 0 if resident or migrant.
## LINEAR MODEL for TIME, using NESTING, SIZE, STATUS as covariates.

plot(birdextinct)
```



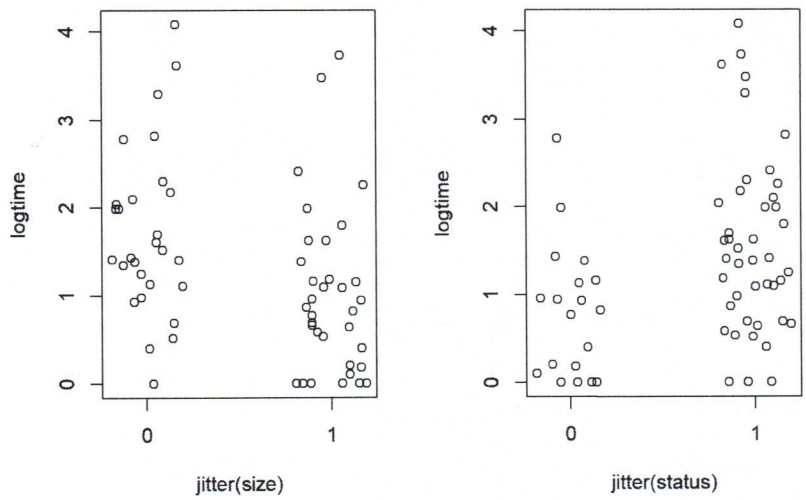

```
logtime = log(time)   # Better consider TIME in the log scale
plot(nesting,logtime) # POSITIVE CORRELATION between these variables

# OUTLIER: probably datapoints with log(time)>3
out = (logtime > 3)
# aggiungo le specie relativa agli OUTLIER sull'ultimo grafico che ho generato
text(nesting[out], logtime[out], label=species[out], pos = 2)
```



It seems like a linear model could be a good choice, however there are some data that are "strange". We'll have to remember about them.

(only numeric covariate that we have)

```
# Command JITTER adds a small amount of noise to a numeric vector
par(mfrow=c(1,2))
plot(jitter(size),logtime,xaxp=c(0,1,1))
plot(jitter(status),logtime,xaxp=c(0,1,1))
```




```
### ---------------------------------------------------------------------------
### ML estimates
### ---------------------------------------------------------------------------
fit = lm(logtime~nesting+size+status,data=birdextinct,x=TRUE,y=TRUE)
summary(fit)
```

```
##
## Call:
## lm(formula = logtime ~ nesting + size + status, data = birdextinct,
##     x = TRUE, y = TRUE)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8410 -0.2932 -0.0709  0.2165  2.5167
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.43087    0.20706   2.081 0.041870 *
## nesting      0.26501    0.03679   7.203 1.33e-09 ***
## size        -0.65220    0.16667  -3.913 0.000242 ***
## status       0.50417    0.18263   2.761 0.007712 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6524 on 58 degrees of freedom
## Multiple R-squared:  0.5982, Adjusted R-squared:  0.5775
## F-statistic: 28.79 on 3 and 58 DF,  p-value: 1.577e-11
```

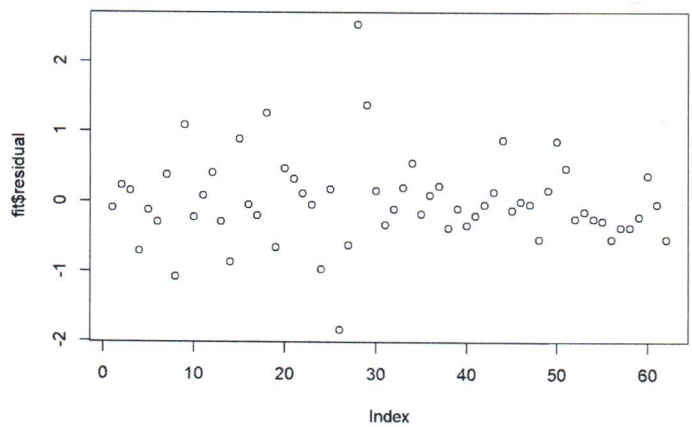
```
# NESTING is a strong effect
# estimate of the effect of SIZE is negative: animali piccoli (SIZE=1) hanno tempi di estinzione più brevi
fit$x
```

```
##    (Intercept) nesting size status
## 1            1   1.000    0      1
## 2            1   2.000    0      1
## 3            1   1.210    0      1
## 4            1   1.125    0      1
## 5            1   5.167    0      1
## 6            1   1.000    0      0
## 7            1   2.750    0      1
## ..
## 60           1  11.620    1      1
## 61           1   1.000    1      0
## 62           1   1.000    1      1
## attr(,"assign")
## [1] 0 1 2 3
```

```
fit$y
```

```
##         1         2         3         4         5         6         7         8
## 1.1085626 1.6981811 1.4104990 0.5193889 2.1804175 0.4007875 2.0401808 1.3470336
##         9        10        11        12        13        14        15        16
## 2.8134307 1.4395981 2.0955609 1.6094379 1.9877374 0.0000000 3.2968364 1.1333357
##        17        18        19        20        21        22        23        24
## 1.3862944 2.7806189 1.2481811 3.6119174 1.9877374 0.9262411 1.4187616 0.6931472
##        25        26        27        28        29        30        31        32
## 2.3025851 0.9809542 1.5232262 4.0745499 3.4737661 0.9442949 0.7701082 0.0000000
##        33        34        35        36        37        38        39        40
## 1.0875513 2.2538149 2.4079356 1.9877374 1.3862944 0.8675206 0.9597333 1.1808065
##        41        42        43        44        45        46        47        48
## 0.5312163 0.5850050 0.1806535 1.1584523 0.8211005 0.1052605 0.0000000 0.0000000
##        49        50        51        52        53        54        55        56
## 0.2070142 1.8018748 1.1553076 0.6931472 1.6245235 0.6595904 0.4007875 0.0000000
##        57        58        59        60        61        62
## 1.6296326 1.0996118 0.6408007 3.7297094 0.0000000 0.0000000
```

```
plot(fit$residual)
```



ADVISE: never use it in the project since it's improper

Zellner's g-prior

Parameters: $\underline{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4)$, σ^2 so that the model is:

$$\underbrace{Y_i}_{\text{logarithm of the extinction time}} \overset{iid}{\sim} N(\underline{\beta}^T \underline{x}_i, \sigma^2), \quad \sigma^2 = \frac{1}{\tau}$$

$\Rightarrow$ parameters:

$(\underline{\beta}, \sigma^2) \sim$ **Zellner's - g** prior

(which assumes for σ^2 an improper marginal distribution ($\propto 1/\sigma^2$) and the $\underline{\beta}$ distr. is gaussian with mean $= \underline{0}$ and variance $\sigma^2 B$, $B = c\,(X^T X)^{-1}$, where c is the weight that we want to give to the data apostheon, usually $c = \log(n)$)

Now we're going to fit a bayesian model for our regression parameters:

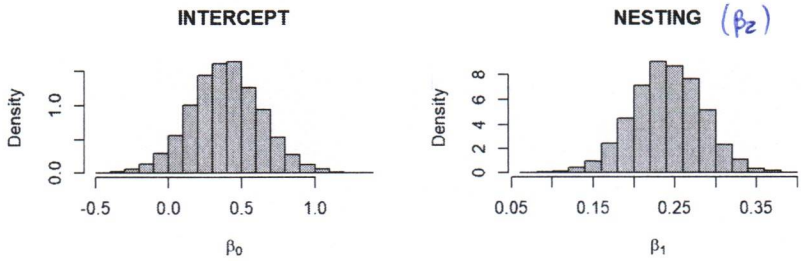
```
### ------------------------------------------------------------------------------
### Zellner's g-prior
### ------------------------------------------------------------------------------
help(blinreg)
```

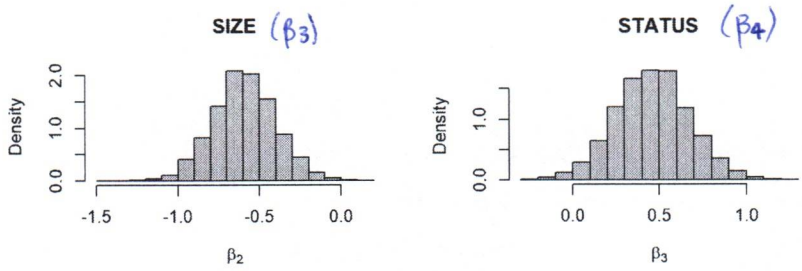
```
# The output is a Monte Carlo sample (iid) from the posterior of theta=(beta,sigma)
# Gives a simulated sample from the joint posterior distribution of the regression vector and the
# error standard deviation for a linear regression model with a (noninformative or) Zellner g prior.
# INPUT: response vector y, design matrix X, num. of iterations, prior list with components c0 and beta0 of Zellner g prior

# Zellner g prior N_p(b0, sigma^2 (c0 (Xt X)^{-1})) * (1/sigma^2)
prior          = list(b0=c(0,0,0,0), c0=10)  # c0\in [10,100]
theta2.sample = blinreg(fit$y,fit$x,5000,prior=prior)

# La distribuzione finale si fattorizza come al solito;
# La distribuzione di beta, dato tau e y, è
# N_p(c0/(c0+1)(b0/c0 + hatbeta), sigma^2 * c0/(c0+1)* (Xt X)^{-1})) ),
# mentre la distribuzione di tau, dato y, è gamma(n/2, s^2/2+ ...)

par(mfrow=c(2,2))
hist(theta2.sample$beta[,1],main="INTERCEPT",xlab=expression(beta[0]),prob=T)
hist(theta2.sample$beta[,2],main="NESTING",  xlab=expression(beta[1]),prob=T)
hist(theta2.sample$beta[,3],main="SIZE",     xlab=expression(beta[2]),prob=T)
hist(theta2.sample$beta[,4],main="STATUS",   xlab=expression(beta[3]),prob=T)
```

here we have the simulated values from the posterior distribution

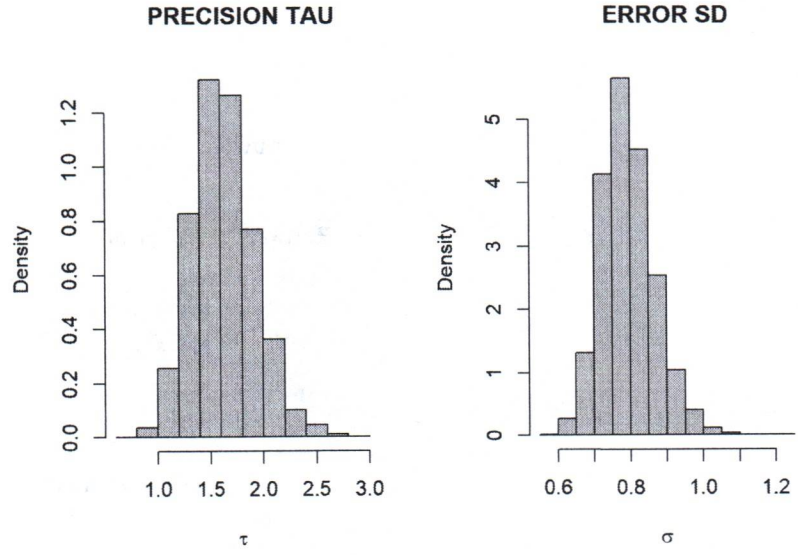
Marginals: $\underline{\beta}$, σ^2

A good thing is that the posterior can be simply sampled iid ($\underline{\beta} \sim N$ and $\sigma^2 \sim$ inv-gamma) (MC can procede easily (not even MCMC))






```
par(mfrow=c(1,2))
hist(1/((theta2.sample$sigma)^2),main="PRECISION TAU",xlab=expression(tau),prob=T)
hist(theta2.sample$sigma,main="ERROR SD",xlab=expression(sigma),prob=T)
```




```
# posteriori quantiles and mean of beta parameters
apply(theta2.sample$beta,2,quantile,c(.05,.5,.95))
```

```
##     X(Intercept)  Xnesting      Xsize   Xstatus
## 5%  -0.001370107 0.1695021 -0.9156057 0.1074238
## 50%  0.389251410 0.2409184 -0.5934178 0.4565169
## 95%  0.789390987 0.3107257 -0.2756947 0.8082835
```

```
apply(theta2.sample$beta,2,mean)
```

```
## X(Intercept)     Xnesting        Xsize      Xstatus
##    0.3927824    0.2406088   -0.5926610    0.4575863
```

```
fit$coefficients # stime frequentiste
```

```
## (Intercept)     nesting        size      status
##   0.4308716   0.2650140  -0.6521982   0.5041655
```

```
quantile(theta2.sample$sigma,c(.05,.5,.95))
```

```
##        5%       50%       95%
## 0.6881560 0.7886346 0.9200982
```

```
# stima bayesiana di sigma^2 - media a posteriori - è maggiore della stima ai minimi quadrati
mean(theta2.sample$sigma)
```

```
## [1] 0.7936692
```

```
# stima frequentista ai mimini quadrati di sigma
sqrt(sum(fit$residuals^2)/(62-4))
```

```
## [1] 0.6524084
```

```
### ------------------------------------------------------------------------------
### Estimating mean extinction times
### ------------------------------------------------------------------------------
# We'd like to estimate the function of beta:  E(y_new|beta, x1)=x_1*beta,
# x_1 is a  "new" covariate matrix;
# this function is a LINEAR COMBINATION of the beta parameters
cov1 = c(1,4,0,0) #A
cov2 = c(1,4,1,0) #B
cov3 = c(1,4,0,1) #C
cov4 = c(1,4,1,1) #D
X1   = rbind(cov1,cov2,cov3,cov4)
# La funzione blinregexpected prende in INPUT la nuova matrice disegno
# e il campione generato dalla posterior e ne fa la combinazione lineare
mean.draws = blinregexpected(X1,theta2.sample)

c.labels = c("A","B","C","D")
par(mfrow=c(2,2))
for (j in 1:4)
  hist(mean.draws[,j], main=paste("Covariate set",c.labels[j]),xlab="log TIME",prob=T)
```

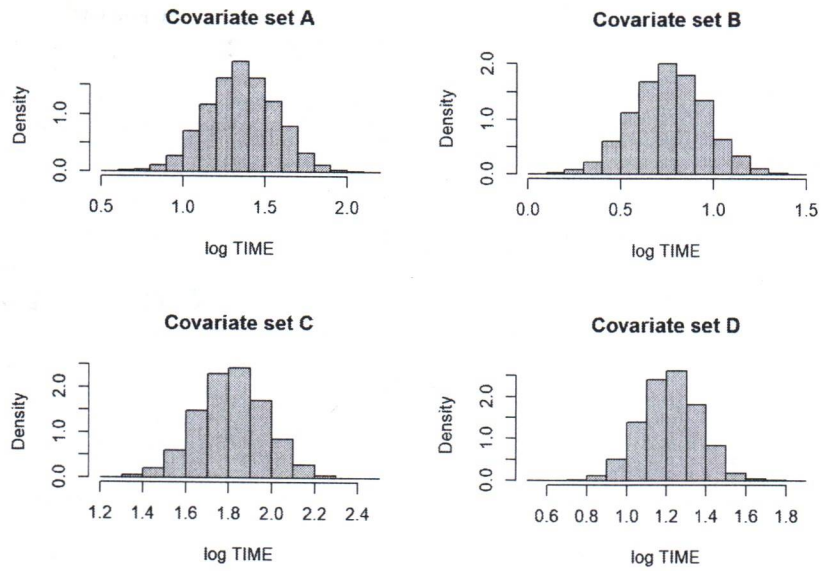
we're not interested in estimate the parameters marginally, we're interested in the estimate of the effect of the betas on the mean response.
What we are interested in is:

$$\mathbb{E}[Y^{new} | \underline{x}^{new}, y, X] = \mathbb{E}[(\underline{\beta})^T \underline{x}^{new} | y, X]$$
$$= \mathbb{E}[\beta_1 + \beta_2 x^{new} + \beta_3 x^{new} + \beta_4 x^{new} | data]$$

we fix 4 different covariates (4 different vectors $\underline{x}^{new}$)

we are interested in the posterior mean of a specific linear combination of $(\underline{\beta})^T \underline{x}^{new}$

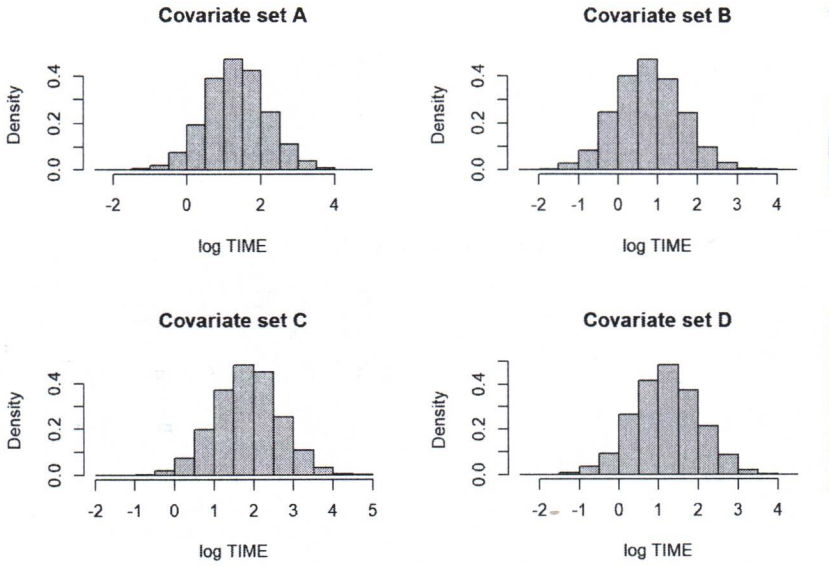
Instead of monitoring singularly the betas we're interested in a specific linear combination of these betas.
Why this particular linear combination?
Because this combination express the response (average/mean response) corresponding to the new values of the covariates.



full posterior distributions of $\underline{\beta}^T \underline{x}^{new}$ (from which we can compute the mean and everything else)

We can compute not only the function which express the expectation of the predictive distribution, but we can compute the predictive distribution itself.

```
### ------------------------------------------------------------------------------
### Calcolo delle PREDITIVE  con la PRIOR g di ZELLNER - Predicting extinction time
### ------------------------------------------------------------------------------
# Prima abbiamo simulato la distribuzione finale di E(y_new|X1) - che è una funzione dei
# 4 prametri di regressione. Ora invece vogliamo la distribuzione predittiva di y_new,
# cioè la legge condizionale di y_new, dato il vettore delle osservazioni y.
cov1 = c(1,4,0,0)
cov2 = c(1,4,1,0)
cov3 = c(1,4,0,1)
cov4 = c(1,4,1,1)
X1   = rbind(cov1,cov2,cov3,cov4)
# La funzione blinregpred genera un valore dalle verosimiglianze valutate
# in ognuno dei valori di theta generati dalla posterior, cioè generata da una
# gaussiana di dim m=4 con media X_1 beta^(j) e matrice di var =(X^tX *tau^(j))^(-1)
pred.draws = blinregpred(X1,theta2.sample)

c.labels = c("A","B","C","D")
par(mfrow=c(2,2))
for (j in 1:4)
  hist(pred.draws[,j],main=paste("Covariate set",c.labels[j]),xlab="log TIME",prob=T)
```



Plots of the predictive distribution of the response (≠ from what we did before. Before we were plotting the posterior of a function of our parameters, which express the expectation of this (←) response)

How do we check if the model is explaining sufficiently well the data?

```
### -----------------------------------------------------------------------------
#### MODEL CHECKING via POSTERIOR PREDICTIVE DISTRIBUTIONS
### -----------------------------------------------------------------------------
# From the book Gelman et al. (2014) - $Y_i^{new}$ is "the replicated data that could have been
# observed, or, to think predictively, as the data that we would see tomorrow if the experiment
# that produced y_i today were replicated with the same model and the same value of $\theta$ that
# produced the observed data". If the model fits, then replicated data generated under the model
# should look similar to observed data. To put another way, the observed data should look PLAUSIBLE
# under the posterior predictive distribution. This is a self-consistency test! Any systematic
# differences between the simulations and the data indicate potential failings of the model.
#### As an ALTERNATIVE, apply the CROSS-VALIDATION approach - see BELOW!

help(blinregpred)
# Simulo la distribuzione predittiva di Y_1,...,Y_62, ottenendo un campione iid da questa,
# lungo quanto era il campione dalla finale, e ne calcolo i quantili di ordine 0.05 e 0.95
pred.draws = blinregpred(fit$x,theta2.sample)
pred.sum   = apply(pred.draws,2,quantile,c(.05,.95))
# Plot the 90% CI and the datapoints
par(mfrow=c(1,1))
ind=1:length(logtime)
matplot(rbind(ind,ind),pred.sum,type="l",lty=1,col=1,xlab="INDEX",ylab="log TIME")
points(ind,logtime,pch=19)

# Consider species datapoints outside these intervals as OUTLIERS
out = (logtime>pred.sum[2,]| logtime<pred.sum[1,])
text(ind[out], logtime[out], label=species[out], pos = 4)
```

we mark them with their names

REPLICATED DATA APPROACH :

! for every covariate vector in the dataset we're going to compute the predictive distribution of a new item which will join the study with the same covariate value that we have already in the dataset.

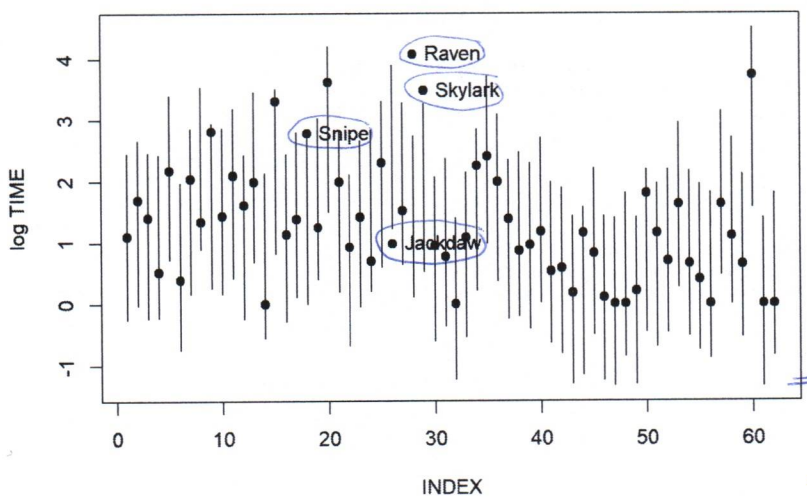
We're going to compute:

$$\mathcal{L}(Y_i^{new} \mid \underline{x}^{new} = \underline{x}_i, \underline{y}, X) \simeq y_i$$

predictive distribution of a new potential item joining the study with covariates equal to all the covariates in the dataset.

Then we compare the values y_i and this distribution. (we want to measure their distance)

We can for instance compute 90% / 95% credibility intervals of the predictive distributions ("predictive intervals" more than "credibility intervals") and see if our observed responses are within these intervals. (We're gonna do it with all the items $(\underline{x}, y)$ in the sample)



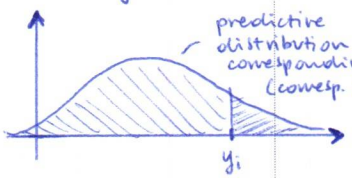
⟹ There are 4 species for which the observed logarithm of time extinction lies outside these intervals.

⟹ OUTLIERS

what else we can do?

$= \min \{ \mathbb{P}(Y_i^{new} > y_i \mid \underline{x}^{new} = \underline{x}_i, y, X),\ \mathbb{P}(Y_i^{new} < y_i \mid \cdot\cdot) \}$

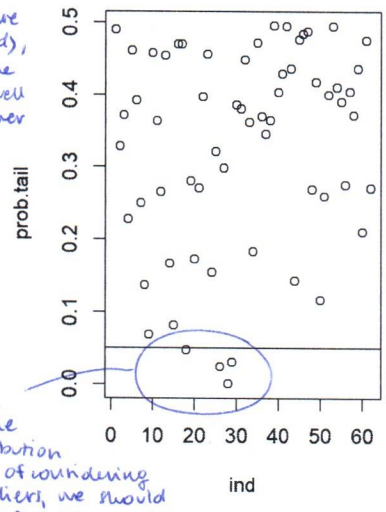
```
### ---------------------------------------------------------------------------
### BAYESIAN PREDICTIVE p-values
### ---------------------------------------------------------------------------
# min( P(Y_i^{new}>y_i|x_i,data), P(Y_i^{new}<y_i|x_i, data))
# Compute how many simulated values from the predictive are > y_i
# (over the total num of simulations): this is an estimate of P(Y_i^{new}>y_i|data, x_i)
ind_pvalue    = t((t(pred.draws)>logtime))
pred.suptail = apply(ind_pvalue,2,sum)/5000
prob.tail     = rep(0, 62)
for (i in 1 : 62)
  prob.tail[i] = min(pred.suptail[i],1-pred.suptail[i]) # this is the BAYESIAN PREDICTIVE p-value of item i

# if this number is close to 1/2, the corresponding predictive "explain" well the datapoint
# if this number is close to 0, the datapoint is an "outlier" for the model
par(mfrow=c(1,2))
plot(ind, prob.tail)
abline(h=0.05)

plot(ind,logtime,pch=19)
# OUTLIERS: those species such that the BAYESIAN PREDICTIVE p-value is smaller than 0.05 (or than 0.1)
out2 = (prob.tail<0.05)
text(ind[out2], logtime[out2], label=species[out2], pos = 4)
```

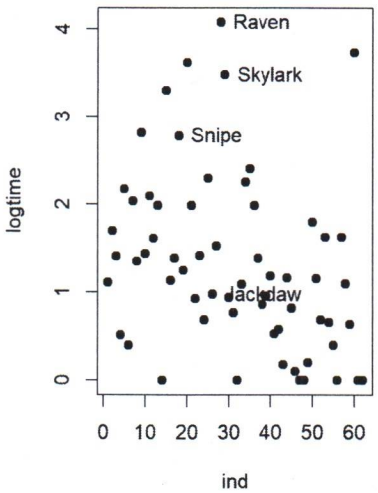


If the model explains well y_i, then we should find y_i in the middle, so that the minimum of those two values should be close to 1/2.
If the minimum is extremely low then the model is not explaining well y_i.

⇒ We're going to compute all these tail probabilities of the predictive distribution (let call them "bayesian p-values") and plot them on a graph. Then we'll look for outliers (→ values for which the p-values are extremely low)

for all the i we plot the tail probability. A lot of values are close to 1/2 (good), it means that the model explain well those data. However there are some outliers.

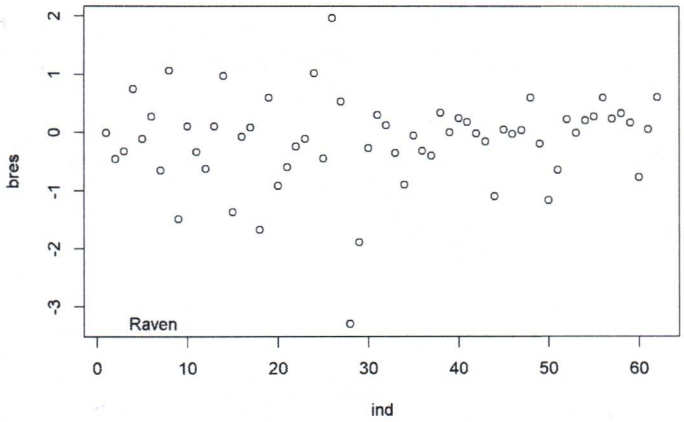




these data lie in the tail of the predictive distribution. Maybe, instead of considering those data outliers, we should consider a more flexible model.

```
### ---------------------------------------------------------------------------
### Predictive Bayesian residuals
### ---------------------------------------------------------------------------
## Compute mean and sd of all n predictive distributions first. Then compute the difference
## between the predictive mean and observed datapoint, over the sd of th epredictive distribution.
## This is the "predictive Bayesian residual".
## Consider as an OUTLIER all data such that the corresponding residual is LARGE
## e.g. its absolute value is larger than 2 (or 3)
# Simulo la distribuzione predittiva di Y_1,...,Y_62; la prior è quella di Zellner, quindi
# è una caso particolare della coniugata, e quindi riesco a simulare un campione iid dalla
# posterior usando la funzione blinreg(fit$y,fit$x,5000,prior=prior) - FATTO DI SOPRA!

pred.mean = apply(pred.draws,2,mean)
pred.sd   = apply(pred.draws,2,sd)
bres      = (pred.mean- logtime)/pred.sd   # residui bayesiani
out2      = (abs(bres) > 2) #as a reference value we take 2 or 1.8

par(mfrow=c(1,1))
plot(ind,bres,cex=1)
text(nesting[out2], bres[out2], label=species[out2], pos = 4)
```

Compute the predictive distributions of all y_i^{new}, then compute the difference between the means of these predictive distributions and observed data and divide by the standard deviation of the predictive distribution. We consider as "outlier" a datum for which the corresponding residual is large.



Comparison between models:

```
# Predictive goodness-of-fit: SUM of the predictive Bayesian residuals to compare different models.
# The "best" model is the one with the smallest value for this index
sum(bres^2)
```

```
## [1] 38.98543
```

```
### ------------------------------------------------------------------------
### LPML and  CPO_i: a CROSS-VALIDATION approach
### ------------------------------------------------------------------------
# Conditional density of each Y_i, given parameters, is Gaussian with mean X*beta and variance sigma^2
mean.draws.data = blinregexpected(fit$x,theta2.sample)
dim(mean.draws.data)
```

```
## [1] 5000   62
```

```
cpo = seq(1,62)
for (i in 1:62){
  cpo[i] = 1/mean(1/dnorm(fit$y[i],mean.draws.data[,i], theta2.sample$sigma))
}

1/mean(1/dnorm(fit$y[2],mean.draws.data[,2], theta2.sample$sigma)) # this is  CPO[2] of item i=2
```

```
## [1] 0.4380306
```

```
LPML=sum(log(cpo))
LPML
```

```
## [1] -67.52007
```

```
### ------------------------------------------------------------------------
### Repeat the analysis WITHOUT covariate "status"
### ------------------------------------------------------------------------
### Stima ai MINIMI QUADRATI (stime frequentiste)
### ------------------------------------------------------------------------
fit2 = lm(logtime~nesting+size,data=birdextinct,x=TRUE,y=TRUE)
summary(fit2)
```

(since the LPML is meaningless for one model, we've using it for the comparison of models)

```
##
## Call:
## lm(formula = logtime ~ nesting + size, data = birdextinct, x = TRUE,
##     y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76083 -0.38290 -0.09753  0.38512  2.68075
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.72970    0.18615   3.920 0.000233 ***
## nesting      0.28260    0.03821   7.395 5.79e-10 ***
## size        -0.66904    0.17565  -3.809 0.000335 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.688 on 59 degrees of freedom
## Multiple R-squared:  0.5454, Adjusted R-squared:   0.53
## F-statistic:  35.4 on 2 and 59 DF,  p-value: 7.923e-11
```

```
fit2$x
```

```
##    (Intercept) nesting size
## 1            1   1.000    0
## 2            1   2.000    0
## 3            1   1.210    0
## 4            1   1.125    0
## 5            1   5.167    0
## 6            1   1.000    0
## 7            1   2.750    0
## ..
## 60           1  11.620    1
## 61           1   1.000    1
## 62           1   1.000    1
## attr(,"assign")
## [1] 0 1 2
```

```
fit2$y
```

```
##         1         2         3         4         5         6         7         8
## 1.1085626 1.6981811 1.4104990 0.5193889 2.1804175 0.4007875 2.0401808 1.3470336
##         9        10        11        12        13        14        15        16
## 2.8134307 1.4395981 2.0955609 1.6094379 1.9877374 0.0000000 3.2968364 1.1333357
##        17        18        19        20        21        22        23        24
## 1.3862944 2.7806189 1.2481811 3.6119174 1.9877374 0.9262411 1.4187616 0.6931472
##        25        26        27        28        29        30        31        32
## 2.3025851 0.9809542 1.5232262 4.0745499 3.4737661 0.9442949 0.7701082 0.0000000
##        33        34        35        36        37        38        39        40
## 1.0875513 2.2538149 2.4079356 1.9877374 1.3862944 0.8675206 0.9597333 1.1808065
##        41        42        43        44        45        46        47        48
## 0.5312163 0.5850050 0.1806535 1.1584523 0.8211005 0.1052605 0.0000000 0.0000000
##        49        50        51        52        53        54        55        56
## 0.2070142 1.8018748 1.1553076 0.6931472 1.6245235 0.6595904 0.4007875 0.0000000
##        57        58        59        60        61        62
## 1.6296326 1.0996118 0.6408007 3.7297094 0.0000000 0.0000000
```

```
### ---------------------------------------------------------------------------
### SAMPLING from the psoterior with Zellner g prior
### ---------------------------------------------------------------------------
# Mi restituisce un campione Monte Carlo - dunque iid - dalla distribuzione finale di
# theta=(beta,sigma) con prior(sigma^2)=1/sigma^2
# INPUT: vettore risposta y, matrice disegno X, numerosità del campione in uscita
prior3        = list(b0=c(0,0,0), c0=10)  # c0\in [10,100]
theta3.sample = blinreg(fit2$y,fit2$x,5000, prior=prior3)

### ---------------------------------------------------------------------------
### LPML
### ---------------------------------------------------------------------------
mean.draws3.data=blinregexpected(fit2$x,theta3.sample)
dim(mean.draws3.data)
```

```
## [1] 5000   62
```

```
cpo2 = seq(1,62)
for (i in 1:62){
  cpo2[i] = 1/mean(1/dnorm(fit$y[i],mean.draws3.data[,i], theta3.sample$sigma))
}

LPML2 = sum(log(cpo2))
LPML2
```

```
## [1] -70.01025
```

```
### The best model is the one with LARGEST LPML: which model is better, then?
LPML
```

```
## [1] -67.52007
```

```
LPML2
```

```
## [1] -70.01025
```

CROSS-VALIDATION

We split the dataset in **2** parts:

- training set: used to compute the estimates
- test set: we compute the prediction using our model and we compare the prediction with the data

LEAVE-ONE-OUT CROSS VALIDATION

We put apart **1** observation and consider the other $n-1$ to compute the posterior distribution.

$$\underline{y} = (y_1, \dots, y_n) \longrightarrow \begin{cases} \{y_i\} & \text{test set} \quad (\#=1) \\ \underline{y}_{-i} & \text{training set} \quad (\#=n-1) \end{cases}$$

We compute the predictive distribution $\Sigma(Y_i^{new} | \underline{y}_{-i})$. Since it's a predictive distribution, this is just the integral of a likelihood integrated w.r.t. the posterior (a posterior where we used only $n-1$ observations). We'll compute $\Sigma(Y_i^{new} | \underline{y}_{-i})$ $\forall i = 1, \dots, n$ and each time we'll compare it with y_i (the observation that we have). To comparison, we use an index which is called CONDITIONAL PREDICTIVE ORDINATE$_i$ (**CPO$_i$**):

$$\underline{\mathbf{CPO_i}} = \text{predictive density (from } \Sigma(Y_i^{new} | \underline{y}_{-i})) \text{ evaluated in } y_i = f(y_i | \underline{y}_{-i})$$

If it's <u>large</u> is good, the model explains well the i-th observation.

We also compute the LOG-PSEUDO-MARGINAL-LIKELIHOOD (**LPML**)

$$\underline{\mathbf{LPML}} = \sum_{i=1}^{n} \log(CPO_i)$$

(useful only for comparison between models (e.g. when we have to compare models with different covariates. The <u>higher</u> the better.)

How can we not compute n $f_i(y_i | \underline{y}_{-i})$? (More precisely: $f_i(\cdot | \underline{y}_{-i})$?)

$$f(y_i | \underline{y}_{-i}) = \int f_i(y_i | \theta)\, \pi(\theta | \underline{y}_{-i})\, d\theta$$

If we assume: $Y_i | \theta \overset{iid}{\sim} f_i(\cdot | \theta)$ then:

$$f(y_i | \underline{y}_{-i}) = \int \underbrace{f_i(y_i | \theta)}_{\text{likelihood}} \frac{\pi(\theta) \prod_{j \neq i} f_j(y_j | \theta)}{\int_\Theta \prod_{j \neq i} f_j(y_j | \theta)\, \pi(\theta)\, d\theta}\, d\theta$$

— integrated w.r.t. the posterior distribution using all the data but the i-th

$$\Rightarrow \frac{1}{f(y_i | \underline{y}_{-i})} = \frac{\int_\Theta \prod_{j \neq i} f_j(y_j | \theta)\, \pi(\theta)\, d\theta}{\int_\Theta \prod_{j=1}^{n} f_j(y_j | \theta)\, \pi(\theta)\, d\theta} = \frac{\int_\Theta \frac{1}{f_i(y_i | \theta)} \prod_{j=1}^{n} f_j(y_j | \theta)\, \pi(\theta)\, d\theta}{\int_\Theta \prod_{j=1}^{n} f_j(y_j | \theta)\, \pi(\theta)\, d\theta}$$

posterior distribution according to bayes theorem (posterior using <u>all</u> the data)

$$= \int_\Theta \frac{1}{f_i(y_i | \theta)}\, \pi(\theta | y_1, \dots, y_n)\, d\theta$$

Using MCMC we can estimate it as →

$$\approx \underbrace{\frac{1}{M} \sum_{m=1}^{M} \frac{1}{f_i(y_i | \theta^{(m)})}}_{\text{MCMC estimation}} \quad (\theta^{(m)} = \text{MCMC sample})$$

Then, if we want the CPO$_i$ we do $\dfrac{1}{1/f(y_i | \underline{y}_{-i})}$

Another goodness of fit criterion: **WAIC** $= \widehat{elpd}_{WAIC} = LPML - \hat{p}_{WAIC}$.
(Used to compares models, it's just the LPML corrected via penalization that considers how big is the model (bigger the model, bigger the penalization))

R-package = "**LOO**" (Documentation:

- Vehtari, Gelman, Gabry (2017): Statistics & Computing
- Vehtari, Simpson, Gelman, Yao, Gabry (2019))

Some tools for model comparison and predictive goodness-of-fit

Alessandra Guglielmi

Dipartimento di Matematica
Politecnico di Milano
Milano, Italia
e-mail: alessandra.guglielmi@polimi.it

November 12, 2020

1.
only 2 models comparison

Bayes factor for 2-model comparison

(case we already examined)

M_0: $\boldsymbol{X}|\theta \sim f(\boldsymbol{x}|\theta_0)$, $\theta_0 \in \Theta_0$, $\theta_0 \sim g_0$ prior

M_1: $\boldsymbol{X}|\theta \sim f(\boldsymbol{x}|\theta_1)$, $\theta_1 \in \Theta_1$, $\theta_1 \sim g_1$ prior

where $\Theta = \Theta_0 \cup \Theta_1$, Θ_0, Θ_1 disjoint.

Model choice by Bayes factor:

$$BF_{01} = \frac{m_0(\boldsymbol{x})}{m_1(\boldsymbol{x})}$$

with

$$m_i(\boldsymbol{x}) = \int_{\Theta_i} f(\boldsymbol{x}|\theta_i) g_i(\theta_i) d\theta_i, \ i = 1, 2.$$

Bayes factor computation

If we use the previous formula, 2 MC samples would be required, one from g_0 and one from g_1. What if we had only one? For instance, if the model were complicated, it would be expensive even to sample from the posterior.

In this case, if $\Theta_0 = \Theta_1 = \Theta$, and $f_0(\boldsymbol{x}|\theta) := f(\boldsymbol{x}|\theta_0)$ for $\theta \in \Theta_0$, $f_1(\boldsymbol{x}|\theta) := f(\boldsymbol{x}|\theta_1)$ for $\theta \in \Theta_1$:

$$BF_{01} = \frac{m_0(\boldsymbol{x})}{m_1(\boldsymbol{x})} = \frac{\int_{\Theta_0} f(\boldsymbol{x}|\theta_0) g_0(\theta_0) d\theta_0}{\int_{\Theta_1} f(\boldsymbol{x}|\theta_1) g_1(\theta_1) d\theta_1} = \frac{\int_{\Theta} f_0(\boldsymbol{x}|\theta) g_0(\theta) d\theta}{\int_{\Theta} f_1(\boldsymbol{x}|\theta) g_1(\theta) d\theta}$$

$$= \frac{\int_{\Theta} \frac{f_0(\boldsymbol{x}|\theta) g_0(\theta)}{f_1(\boldsymbol{x}|\theta) g_1(\theta)} f_1(\boldsymbol{x}|\theta) g_1(\theta) d\theta}{\int_{\Theta} f_1(\boldsymbol{x}|\theta) g_1(\theta) d\theta} = \int_{\Theta} r(\theta) g_1(\theta|\boldsymbol{x}) d\theta$$

Bayes factor computation

Therefore, if a MCMC (or MC) sample from $g_1(\theta|\boldsymbol{x})$ is available, we can compute BF_{01} as the ergodic mean of the function

$$r(\theta) = \frac{f_0(\boldsymbol{x}|\theta) g_0(\theta)}{f_1(\boldsymbol{x}|\theta) g_1(\theta)}.$$

- Generally, BF computation is *numerically heavy*!
- If there are more that two models, how shall I use the BF to choose among those different models?

2. Maximum marginal likelihood criterion

Model M: likelihood $f(\boldsymbol{x}|\theta_M, M)$ and prior $\pi_M(\theta_M)$

index of the model

$$m(\boldsymbol{x}|M) = \int_{\Theta_M} f(\boldsymbol{x}|\theta_M, M)\pi_M(\theta_M)d\theta_M$$ marginal d. under model M

Choose model $\hat{M}$ which maximizes the chance (density) to see actual data, i.e.

] choose the model for which data maximizes the marginal density

$$\hat{M} : m(\boldsymbol{x}|\hat{M}) = \max_M m(\boldsymbol{x}|M)$$

Maximum marginal *likelihood* criterion

3. BIC - Bayesian Information Criterion

Models $M_1, \ldots, M_K$

M_i: $\boldsymbol{X}|\theta_i \sim f(\boldsymbol{x}|\theta_i)$, $\theta_i \sim g_i$ prior

Assume that $dim(\theta_i) = r_i$ DOES NOT depend on $n = dim(\boldsymbol{X})$

$\hat{\theta}_i$ MLE of θ_i in model M_i

Under suitable regularity conditions, for large n, we have that

$$m(\boldsymbol{x}|M_i) \simeq f(\boldsymbol{x}|\hat{\theta}_i)n^{-r_i/2}$$

Then, for instance, the Bayes factor between M_1 and M_2 is:

$$BF_{12} = \frac{m(\boldsymbol{x}|M_1)}{m(\boldsymbol{x}|M_2)} \simeq \frac{f(\boldsymbol{x}|\hat{\theta}_1)n^{-r_1/2}}{f(\boldsymbol{x}|\hat{\theta}_2)n^{-r_2/2}}$$

BIC - Bayesian Information Criterion

$$2\log BF_{12} \simeq (2\log f(\boldsymbol{x}|\hat{\theta}_1) - r_1\log n) - (2\log f(\boldsymbol{x}|\hat{\theta}_2) - r_2\log n)$$
$$= \quad BIC(1) \quad - \quad BIC(2),$$

where

$$BIC(i) = 2\log f(\boldsymbol{x}|\hat{\theta}_i) - r_i\log n$$ ⟶ we want the model with MAXIMUM BIC

- Check the definition in R packages computing *BIC*!
- *BIC* does NOT depend on the prior! Hence, it is NOT a subjectivist tool, and therefore I do not love it!

CRITERION: choose the model with the maximum *BIC*

- There is the term $-r_i\log n$: this is a penalization for large dimension models
- Approximation: $BIC(i) = 2\log f(\boldsymbol{x}|\tilde{\theta}_i) - r_i\log n$, where $\tilde{\theta}_i$ is the posterior mean or mode, computed via MCMC

4. AIC - Aikake Information Criterion

$$AIC(i) = 2\log f(\boldsymbol{x}|\hat{\theta}_i) - 2r_i$$

It is NOT a Bayesian tool: it was introduced as an approximation of the Kullback-Leibler divergence between the *true* and estimated models

Note the *BIC* and *AIC* have different penalizations

5.

Deviance information criterion (Spiegelhalter et al., 2002)

deviance=$D(\theta_i, M_i) = -2 \log f(\boldsymbol{x}|\theta_i, M_i)$ of i

$$DIC(i) = \mathbb{E}[D(\theta_i, M_i)|\boldsymbol{x}] + p_D = 2\mathbb{E}[D(\theta_i, M_i)|\boldsymbol{x}] - D(\tilde{\theta}_i, M_i),$$

con $p_D = \mathbb{E}[D(\theta_i, M_i)|\boldsymbol{x}] - D(\tilde{\theta}_i, M_i) =$ num of *effective parameters* for model M_i

CRITERION: choose the model with smallest *DIC*!

- Check the definition in R packages computing *DIC*!
- caution using it, since the assumption is that $\tilde{\theta}_i$ is a good summary of the posterior
- It is not defined for hierarchical models, and in this case it is not yet clear any generalization

DIC

DIC is a generalization of *AIC* (up to the sign); when prior information is weak: $DIC \simeq -AIC$
In this case, minimizing *DIC* is equivalent to maximizing *AIC*

DIC values are not important *per se*; differences between *DIC* values must be monitored:

escludere

- differences of more than 10 might definitely rule out the model with the higher *DIC*
- differences between 5 and 10 are substantial, in favour of the model with the lower *DIC*
- if the difference between the two *DIC*s is less than 5, better choose the model according other criteria, since, in this case, it could be misleading just to report the model with the lowest DIC

6.

WAIC (Watanabe-Akaike information criterion)

It is a predictive goodness-of-fit tool which approximate the expected log pointwise predictive density

$$elppd = \sum_{i=1}^{n} E_f(\log f_i(y_i^{new}|\boldsymbol{y}))$$

evaluated at $y_i^{new} = y_i$ for each i

$$WAIC = \text{computed lppd} - p_{WAIC2}$$

where $\text{computed MCMC lppd} = \widehat{lppd} = \sum_{1}^{n} \log\left(\frac{1}{M}\sum_{j=1}^{M} f_i(y_i|\theta^{(j)})\right)$

$$p_{WAIC2} = \sum_{1}^{n} \text{MCMC variance of } \log(f_i(y_i|\theta))$$

WAIC and the package `loo`

WAIC is fully Bayesian!

leave one out

See Vehtari, Gelman, Gabry (2017), and the R package `loo` at `http://https://mc-stan.org/loo/articles/`

Please check the formula implemented in `loo` for WAIC:
either

$WAIC = \widehat{lppd} - p_{WAIC2}$: choose the model with the largest *WAIC*

or

$WAIC = -2\left(\widehat{lppd} - p_{WAIC2}\right)$: choose the model with the smallest *WAIC*

In this last case, WAIC and DIC are on the same scale, hence

- differences of more than 10 might rule out the model with the higher *WAIC*

(another (more formal) approach)

Formal approach to model choice

Fix prior probability masses for any model, then use Bayes Theorem to compute the posterior distribution.

The best models (more than one) are those with highest posterior probability.

Let m be the index describing K models $M_1, \ldots, M_K$

- $\mathbb{P}(m = j)$ = prior probability to choose model M_j (typically $=1/K$)
- model M_j: likelihood $f(\boldsymbol{y}|\theta_j, M_j)$, prior $\pi(\theta_j|M_j) = \pi(\theta_j|m = j)$, θ_j vector of parameters

Formal approach to model choice

1. For any model M_j compute the posterior of parameter θ_j:

$$\pi(\theta_j|\boldsymbol{y}, M_j) = \frac{f(\boldsymbol{y}|\theta_j, M_j)\pi(\theta_j|M_j)}{m(\boldsymbol{y}|M_j)}, \qquad \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal}}$$

   where $m(\boldsymbol{y}|M_j) = \int f(\boldsymbol{y}|\theta_j, M_j)\pi(\theta_j|M_j)d\theta_j$.

2. Compute the posterior probability masses of $M_1, \ldots, M_K$:

$$\mathbb{P}(m = j|\boldsymbol{y}) = \frac{m(\boldsymbol{y}|M_j)\mathbb{P}(m = j)}{m(\boldsymbol{y})}, \; j = 1, \ldots, K,$$ can be very high: $K = 2^{\#\text{covariates}}$

   with $m(\boldsymbol{y}) = \sum_{j=1}^{K} m(\boldsymbol{y}|M_j)\mathbb{P}(m = j)$.

3. Choose the model with the highest $\mathbb{P}(m = j|\boldsymbol{y})$ (or 2-3 models with the highest posterior probabilities), and then compare these few models via predictive goodness-of-fit criteria

Bayesian Model Averaging

(however this is not a "popular" approach)

Alternative: BMA

Calculate inference from ALL models, weighing them with

$$w_j = \mathbb{P}(m = j|\boldsymbol{y}) = \frac{m(\boldsymbol{y}|M_j)\mathbb{P}(m = j)}{m(\boldsymbol{y})}, \; j = 1, \ldots, K.$$

GENERALIZED LINEARS MODELS

Data: $(y_i, \underline{x}_i)$ $\quad i = 1, \dots, n$

y_i = response variable (univariate) $\rightarrow$ continuous, discrete, 0-1, categorical

$\underline{x}_i$ = vector of covariates (multidimensional) $\rightarrow$ continuous, discrete, 0-1, categorical

— Dealing with "**dummy variables**":

x univariate covariate: categorical with K categories: $j = 1, \dots, K$.

How can we represent this covariate?

Via DUMMY VECTOR:

K-1 components: $x^{(1)}, x^{(2)}, \dots, x^{(q)}$, with $q = k-1$, $x^{(j)} \in \{0,1\}$ $\forall j$

$$x^{(j)} = \begin{cases} 1 & \text{if category 1 is observed} \\ 0 & \text{otherwise} \end{cases}$$

We need to:

→ Fix a reference category (for instance the last one) as the basic level

(so that, $x \in k^{th}$ category if and only if $(x^{(1)}, x^{(2)}, \dots, x^{(q)}) = (0,0,\dots,0)$)

$x \in 1^{st}$ $\quad (x^{(1)}, x^{(2)}, \dots, x^{(q)}) = (1,0,\dots,0)$

$x \in 2^{nd}$ $\quad (x^{(1)}, x^{(2)}, \dots, x^{(q)}) = (0,1,\dots,0)$

[...] $\quad$ [...]

In case of linear models:

$$\mathbb{E}[Y_i \mid (x_i^{(1)}, \dots, x_i^{(k-1)})] = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \dots + \beta_{K-1} x_i^{(k-1)}$$

- $x_i \in k^{th}$: $(x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k-1)}) = (0,0,\dots,0) \rightarrow \mathbb{E}[Y_i \mid (\cdot\cdot)] = \beta_0$
- $x_i \in 1^{st}$: $(x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k-1)}) = (1,0,\dots,0) \rightarrow \mathbb{E}[Y_i \mid (\cdot\cdot)] = \beta_0 + \beta_1$

  ⋮

- $x_i \in k-1$: $(x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k-1)}) = (0,0,\dots,1) \rightarrow \mathbb{E}[Y_i \mid (\cdot\cdot)] = \beta_0 + \beta_{k-1}$

effect on the average response corresponding to items for which the only categorical covariate assume values in the last class

Are you from group **j**? On average your response will be: $\mathbb{E}[Y_i \mid (\cdot\cdot)] = \beta_0 + \beta_j$ $\quad (\beta_j = 0 \text{ if } j = k)$

—

Let's talk about GLM.

Data: $(y_i, \underline{x}_i)$ $\quad i = 1, \dots, n$.

A generalized linear model has **3** components:

1. RANDOM COMPONENT: $\mathcal{L}(Y_i \mid \underline{x}_i)$ $\forall i$ assuming $Y_1, \dots, Y_n \mid \underline{x}_1, \dots, \underline{x}_n$ are $\perp\!\!\!\perp$

   more in general: $\mathcal{L}(\underline{Y} \mid X)$, X = design matrix

   How are we going to choose it?

   $Y_i \mid \underline{x}_i \sim$ exponential family, i.e. it's density is of the form:

   $$f(y_i \mid \theta_i, \phi) = \exp\left\{ \frac{y_i \theta_i - b(\theta_i)}{\phi} + c(y_i, \phi) \right\}$$

   where:

   θ_i = natural parameter

   ϕ = scale / dispersion parameter

   $$\mu_i := \mathbb{E}[Y_i \mid \underline{x}_i]$$

2. LINEAR PREDICTOR : we need to specify the covariates $\underline{x}_i$ which appear in the linear predictor term :

$$\eta_i := \underline{x}_i^T \underline{\beta} \qquad (\dim(\underline{x}_i) = \dim(\underline{\beta}) = p)$$

(linear pred. term)

3. LINK FUNCTION: it establishes the link between the random component and the linear predictor. We assume that $\exists$ a deterministic relationship between μ_i and η_i :

$$g(\mu_i) = \eta_i = \underline{x}_i^T \underline{\beta}$$

(link function)

(or :)

$$\mu_i = h(\underline{x}_i^T \underline{\beta})$$

(response function)

Examples:

1. $Y_i \sim N(\mu_i, \sigma^2)$: $\theta_i = \mu_i$, $\phi = \sigma^2$, $b(\theta) = b(\mu) = \frac{\mu^2}{2}$

Natural link function: $\mu_i = \underline{x}_i^T \underline{\beta} \longrightarrow$ g and h are the identity funct.

2. $Y_i | \underline{x}_i \sim Be(\pi(\underline{x}_i))$

$Y_i \sim Be(\pi)$:

$$\mathbb{P}(Y_i = y) = \pi^y (1-\pi)^{1-y} \mathbb{1}_{\{0,1\}}(y) = e^{\log(\frac{\pi}{1-\pi}) \cdot y + \log(1-\pi)} \mathbb{1}_{\{0,1\}}(y)$$

$\theta = \log\left(\frac{\pi}{1-\pi}\right)$, $\phi = 1$, $b(\theta) = \dots$

Natural link function: it would be something like: $\pi_i = \underline{x}_i^T \underline{\beta}$, but $\pi_i \in (0,1)$?
To prevent / solve the problem we transform:

$$\pi_i = F(\underline{x}_i^T \underline{\beta}) \qquad (F \text{ distribution function})$$

according to F we have different models:

- PROBIT LINK : $F = \Phi_{N(0,1)}$: $\pi_i = \Phi(\underline{x}_i^T \underline{\beta})$ (Φ distr. of std. normal
- LOGIT LINK : (natural link) $g(\pi_i) = \log\left(\frac{\pi_i}{1-\pi_i}\right) = \underline{x}_i^T \underline{\beta}$: $\pi_i = \frac{e^{\underline{x}_i^T \underline{\beta}}}{1 + e^{\underline{x}_i^T \underline{\beta}}}$

(which corresponds to: $F(x) = \frac{e^x}{1+e^x}$, which is the logistic distribution function)

- COMPLEMENTARY LOG-LOG : $g(\pi_i) = \log(-\log(1-\pi_i)) = \eta_i = \underline{x}_i^T \underline{\beta}$

which corresponds to: $F(x) = 1 - e^{-e^x}$, $x \in \mathbb{R}$

3. $Y_i \sim \text{Poiss}(\lambda_i)$

$$f(y_i|\lambda_i) = \frac{1}{y_i!}\mathbb{1}_{\{0,1,2,\dots\}}(y_i)\, e^{y_i \log(\lambda_i) - \lambda_i}$$

$\theta_i = \log(\lambda_i)$, $\lambda_i = \mathbb{E}[Y_i]$

Natural link function: $\log(\lambda_i) \overset{\text{link}}{=} \underline{x}_i^T \cdot \underline{\beta} \longrightarrow \lambda_i \overset{\text{response}}{=} e^{\underline{x}_i^T \cdot \underline{\beta}}$

Bayesian model:

we specify: the likelihood $L(\underline{\beta}, \phi | \underline{y}, X) = \prod_{i=1}^{n} f(y_i | \theta_i(\underline{x}_i^T\underline{\beta}), \phi)$

(θ_i depending on: $\underline{x}_i^T\underline{\beta}$; ϕ: dispersion parameter)

the prior $\pi(\underline{\beta}, \phi) = \pi(\underline{\beta}|\phi)\,\pi(\phi)$

(usually we'll assume: $\underline{\beta}|\phi \sim N_p(\underline{\beta}_0, \Sigma_p)$)

We want an: **MCMC algorithm for PROBIT MODEL** (Jackman, Chp. 8)

Data: $(y_i, \underline{x}_i)$ $i = 1, \dots, n$

$y_i \in \{0,1\}$ binary response

$\dim(\underline{x}_i) = p = \dim(\underline{\beta})$

(Note: in the case (for instance) of the LOGIT model, the Gibbs sampler is not efficient, for the logit model is better to design a METROPOLIS-HASTING algorithm)

We assume:

$$Y_i | \underline{x}_i, \underline{\beta} \overset{iid}{\sim} Be(\pi_i)$$

$$\pi_i = F(\underline{x}_i, \underline{\beta}) = \Phi(\underline{x}_i^T \cdot \underline{\beta})$$

(PROBIT)

this means: $\mathbb{P}(Y_i = 1) = \pi_i$, $\pi_i = \Phi(\underline{x}_i^T \cdot \underline{\beta})$

Prior for the parameters:

$\underline{\beta} \sim \pi(\underline{\beta})$ [it'll be convenient: $\underline{\beta} \sim N_p(\underline{b}_0, B_0)$]

Let's introduce a Data Augmentation Technique

We introduce: $\underline{z} = (z_1, \dots, z_n)$ and then:

$$Y_i = \begin{cases} 1 & \text{if } z_i > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } z_i = \underline{x}_i^T\underline{\beta} + \underline{\varepsilon}_i \quad \varepsilon_i \overset{iid}{\sim} N(0,1)$$

$$(z_i \overset{iid}{\sim} N(\underline{x}_i^T\underline{\beta}, 1))$$

This new model is equivalent to the original:

$$\mathbb{P}(Y_i = 1) = \mathbb{P}(z_i > 0) = 1 - \mathbb{P}(z_i \le 0) = 1 - \mathbb{P}\left(\frac{z_i - \underline{x}_i^T\underline{\beta}}{1} \le \frac{0 - \underline{x}_i^T\underline{\beta}}{1}\right)$$

$$= 1 - \Phi(-\underline{x}_i^T\underline{\beta}) = \Phi(\underline{x}_i^T\underline{\beta})$$

$\Rightarrow$ starting from the model $z_i = \dots$ we found that: $\mathbb{P}(Y_i = 1) = \Phi(\underline{x}_i^T\underline{\beta}) = \pi_i$

$\Rightarrow$ the original model and the augmented are equivalent

This is a useful data augmentation.

Now the parameters are: $(\underline{\beta}, \underline{z})$ $\Longrightarrow$ we need to characterize the joint posterior of the whole bunch of parameters:

$$\pi(\underline{\beta}, \underline{z} | \underline{y}) \quad , \quad \underline{y} = (y_1, \dots, y_n)$$

We'll see that it's straightforward to design a Gibbs sampler to sample from the posterior: $\pi(\underline{\beta}, \underline{z} \mid \underline{y})$, $\underline{y} = (y_1, \dots, y_n)$

Then we'll focus on the marginal of $\underline{\beta}$ (since it's the goal of the analysis).

$$\pi(\underline{\beta}, \underline{z} \mid \underline{y}) \propto \mathcal{L}(\underline{\beta}, \underline{z}, \underline{Y}) = \underbrace{\mathcal{L}(\underline{Y} \mid \cancel{\underline{\beta}}, \underline{z})}_{\text{since we defined } Y_i = \begin{cases} 1 & z_i > 0 \\ 0 & z_i \le 0 \end{cases} \text{ then } Y_i \perp\!\!\!\perp \underline{\beta}} \cdot \mathcal{L}(\underline{z} \mid \underline{\beta}) \cdot \mathcal{L}(\underline{\beta})$$

$$\propto \prod_{i=1}^{n} \Big[\underbrace{\left(\mathbb{1}_{\{Y_i=1\}} \cdot \mathbb{1}_{\{z_i>0\}} + \mathbb{1}_{\{Y_i=0\}} \cdot \mathbb{1}_{\{z_i \le 0\}} \right)}_{\mathcal{L}(\underline{Y} \mid \underline{z}) \text{ (actually } \mathcal{L}(Y_i \mid \underline{z}) \text{ and then } \prod_{i=1}^n)} \cdot \underbrace{N(z_i ; \underline{x}_i^T \underline{\beta}, 1)}_{\mathcal{L}(\underline{z} \mid \underline{\beta}) \text{ is the normal evaluated in } z_i \text{ with mean } \underline{x}_i^T\underline{\beta} \text{ and var. } 1 \ (z_i, \cancel{\underline{z}} \text{ because } \prod_{i=1}^n)} \Big] \cdot \underbrace{\pi(\underline{\beta})}_{\mathcal{L}(\underline{\beta})}$$

Gibbs sampler:

- $[\underline{\beta} \mid \underline{z}, \underline{y}] \propto \mathcal{L}(\underline{Y}, \underline{z}, \underline{\beta}) = \cancel{\mathcal{L}(\underline{Y} \mid \underline{\beta}, \underline{z})} \cdot \mathcal{L}(\underline{z} \mid \underline{\beta}) \cdot \mathcal{L}(\underline{\beta})$

  ($\mathcal{L}(\underline{Y} \mid \underline{\beta}, \underline{z})$ constant w.r.t. $\underline{\beta}$ (since $\underline{Y} \mid \underline{z}$ is $\perp\!\!\!\perp \underline{\beta}$))

  $$\propto \mathcal{L}(\underline{z} \mid \underline{\beta}) \cdot \pi(\underline{\beta}) \quad \leftarrow \text{ in the model for the } \underline{z}, \ \mathcal{L}(\underline{z} \mid \underline{\beta}) \text{ is a gaussian regression model}$$

  If $\pi(\underline{\beta}) \propto \text{const}$ (however, we do **not** use it) $\Rightarrow$ $[\underline{\beta} \mid \underline{z}, \underline{y}] = N_p\big(\underbrace{(X^TX)^{-1}X\underline{z}}_{\hat{\underline{\beta}}}, (X^TX)^{-1}\big)$ (X design matrix)

  If $\pi(\underline{\beta}) = N_p(\underline{b}_0, B_0)$ $\Rightarrow$ $[\underline{\beta} \mid \underline{z}, \underline{y}] = N_p(\tilde{\underline{\beta}}, \tilde{B})$: (easy to sample from this full conditional! (it's a gaussian))

  $$\tilde{B} = (X^TX + B_0^{-1})^{-1}$$
  $$\tilde{\underline{\beta}} = \tilde{B}(X^TX\hat{\underline{\beta}} + B_0^{-1}\underline{b}_0)$$
  $$\hat{\underline{\beta}} = (X^TX)^{-1}X\underline{z}$$

- $[\underline{z} \mid \underline{\beta}, \underline{y}] \propto \mathcal{L}(\underline{Y}, \underline{z}, \underline{\beta}) = \mathcal{L}(\underline{Y} \mid \underline{z}, \cancel{\underline{\beta}}) \mathcal{L}(\underline{z} \mid \underline{\beta}) \cancel{\pi(\underline{\beta})}$ ($\underline{\beta} \perp\!\!\!\perp \underline{z}$)

  $$\propto \prod_{i=1}^{n} \left[\left(\mathbb{1}_{\{Y_i=1\}} \cdot \mathbb{1}_{\{z_i>0\}} + \mathbb{1}_{\{Y_i=0\}} \mathbb{1}_{\{z_i \le 0\}} \right) \cdot N(z_i ; \underline{x}_i^T \underline{\beta}, 1) \right]$$

  because of the product:

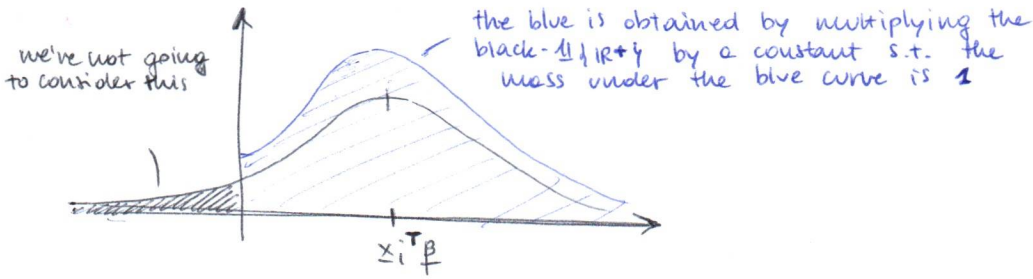
  $$z_i \mid \underline{\beta}, y_i = \begin{cases} \underline{\mathbb{1}_{\{z_i>0\}}} \cdot N(z_i ; \underline{x}_i^T\underline{\beta}, 1) & \text{if } y_i = 1 \\ \underline{\mathbb{1}_{\{z_i \le 0\}}} \cdot N(z_i ; \underline{x}_i^T\underline{\beta}, 1) & \text{if } y_i = 0 \end{cases}$$

  (it changes the support under $y_i = 1$ or $y_i = 0$)

  $\Rightarrow$ $z_i \mid \underline{\beta}, y_i = 1$ is a $N(\underline{x}_i^T\underline{\beta}, 1)$ **TRONCATED** (restricted) to $z_i \in (0, \infty)$

  $z_i \mid \underline{\beta}, y_i = 0$ is a $N(\underline{x}_i^T\underline{\beta}, 1)$ TRONCATED to $z_i \in (-\infty, 0]$

What do we mean by "restricted"? (case $z_i \in (0, +\infty)$)



How to write the distribution function of a truncated random variable X? (which is the distribution of X conditioning to $X \in (a,b)$, $(a,b) \subseteq \mathbb{R}$)

Suppose $X \sim F$ distr. function.

$\Longrightarrow$ $\mathcal{L}(X \mid X \in (a,b))$ has distr. as follow:

$$F_{X|X\in(a,b)}(t) = \begin{cases} 0 & t < a \\ \dfrac{F(t)-F(a)}{F(b)-F(a)} & a \le t < b \\ 1 & t \ge b \end{cases}$$

that's because:

$$F_{X|X\in(a,b)}(t) = \mathbb{P}(X \le t \mid X \in (a,b)) = \frac{\mathbb{P}(X \le t,\ X \in (a,b))}{\mathbb{P}(X \in (a,b))}$$

Now that we have it, we want to simulate from $\mathcal{L}(X \mid X \in (a,b))$.

Remark: $X \sim F_X$, $U \sim \mathcal{U}([0,1]) \longrightarrow F_X^{-1}(U) \sim F_X$

Hence, to sample from $\mathcal{L}(X \mid X \in (a,b))$:

- sample from: $U \sim \mathcal{U}([0,1])$
- $T := F^{-1}\big((F(b)-F(a))\cdot U + F(a)\big)$ $\left(\sim F_{X|X\in(a,b)}\right)$

proof (of $T \sim F_{X|X\in(a,b)}$)

$$\begin{aligned} \mathbb{P}(T \le t) &= \mathbb{P}\left(F^{-1}\big((F(b)-F(a))\cdot U + F(a)\big) \le t\right) \\ &= \mathbb{P}\left((F(b)-F(a))\cdot U + F(a) \le F(t)\right) \\ &= \mathbb{P}\left(U \le \frac{F(t)-F(a)}{F(b)-F(a)}\right) \\ &= \begin{cases} 0 & t < a \\ \dfrac{F(t)-F(a)}{F(b)-F(a)} & a \le t < b \\ 1 & t \ge b \end{cases} \end{aligned}$$

■

We want to apply this to sample (for example) on: $N(1,1)\cdot \mathbb{1}_{(0,\infty)}$ $\left[(a,b) = (0,+\infty)\right]$

$\Longrightarrow$ $F(b) = F(+\infty) = 1$

$F(a) = F(0) = \phi\left(\frac{0-1}{1}\right) = \left(= \mathbb{P}(X \le 0),\ X \sim N(1,1)\right) = \phi(-1)$

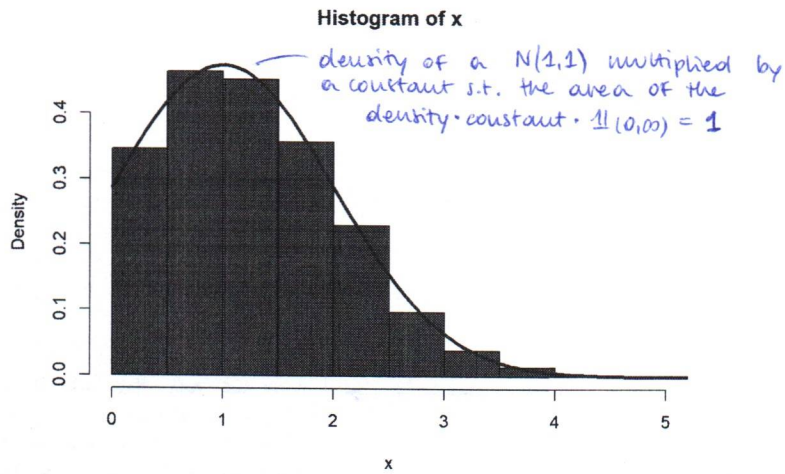
$\Longrightarrow$

- $U \sim \mathcal{U}([0,1])$
- $T := F^{-1}\left[\big(1 - F(-1)\big)\cdot U + F(-1)\right] \sim N(1,1)\cdot \mathbb{1}_{(0,+\infty)}$

where F is the density function of a $N(1,1)$

$\sim N(1,1) \cdot \mathbb{1}_{(0,+\infty)}$

```
# Sampling from a TRUNCATED distribution, i.e conditioning to the r.v. to belong to an interval
# (strictly contained in the original support). Gaussian r.v. X with, parameters mu=1, sigma=1
# TRUNCATED in A=(a1,a2)=(0,+\infinity), that is X, but conditioning X to assume values in A

rm(list=ls())
set.seed(3)
mu    = 1
sigma = 1

Fa1 <- pnorm(0,mu,sigma)
Fa2 <- 1
u1  <- runif(2000)
u1  <- u1*(Fa2-Fa1)+Fa1
#u1 = runif(2000,min=Fa1,max=Fa2)
#par(mfrow=c(1,2))
x = qnorm(p=u1,mu,sigma) # qnorm is the inverse function of the Gaussian d.f.

x11()
hist(x,freq=F,col="red")
curve( dnorm(x,mu,sigma)/(1-pnorm(0,mu,sigma)),from=0,to=6,col="blue",add=T,lwd=3)
```



density of a $N(1,1)$ multiplied by a constant s.t. the area of the density · constant · $\mathbb{1}_{(0,\infty)} = 1$

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Example 10.3 from J. Albert's book "Bayesian computations with R".
### GLMs for binary data: a regression model with PROBIT link
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
# Data from DONNER party, a group of 'pioneers' who crossed
# Sierra Nevada, in a wagon train (in carovana) in 1846-47.
# Most pioneers starved to death.
# Binary response representing death/survival.
# We analyze this dataset via a GLM with PROBIT link.
# n=45
#
# Analisi bayesiana col package LearnBayes
library(LearnBayes)
```

```
data(donner)
attach(donner)
names(donner)
```

covariates

```
## [1] "age"      "male"     "survival"
```

binary response

```
age
```

```
##  [1] 23 40 40 30 28 40 45 62 65 45 25 28 28 23 22 23 28 15 47 57 20 18 25 60 25
## [26] 20 32 32 24 30 15 50 21 25 46 32 30 25 25 25 30 35 23 24 25
```

```
male
```

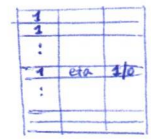
```
##  [1] 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1
## [39] 1 1 1 1 1 1 0
```

```
survival
```

```
##  [1] 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 0
## [39] 0 0 0 0 1 0 1
```

```r
# Response = survival: 0-> dead, 1-> alive
# p_i      = survival probability
# COVARIATE: age (min age: 15 years)
#            male: 0-> female, 1-> male

# matrice disegno con la prima colonna di '1' per includere l'intercetta
X = cbind(1,age,male)

# La funzione glm calcola la MLE per i parametri di regressione;
# è necessario specificare la distribuzione nella famiglia esponenziale e la link function
# Si deve togliere 'intercetta', perchè è già stata aggiunta alla matrice disegno
fit = glm(survival~X-1,family=binomial(link=probit))
# Equivalently: glm(survival ~ male + age,family=binomial(link = probit))
summary(fit) # frequentist MLE of the parameters in the GLM
```

(β_0) →




```
##
## Call:
## glm(formula = survival ~ X - 1, family = binomial(link = probit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7420  -1.0555  -0.2756   0.8861   2.0339
##
## Coefficients:
##       Estimate Std. Error z value Pr(>|z|)
## X      1.91730    0.76438   2.508   0.0121 *
## Xage  -0.04571    0.02076  -2.202   0.0277 *
## Xmale -0.95828    0.43983  -2.179   0.0293 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.383  on 45  degrees of freedom
## Residual deviance: 51.283  on 42  degrees of freedom
## AIC: 57.283
##
## Number of Fisher Scoring iterations: 5
```

all significant

Now we run the Gibbs sampler for the Bayesian model. We assume (a priori) that β_0, β_1 and β_2 are gaussian distributed : $N(0,$ variance from the g of Zellner)

```r
### ------------------------------------------------------------------------------
### Section 10.3.2  Proper priors and model selection
### ------------------------------------------------------------------------------
# library(LearnBayes)
# data(donner)
# y=donner$survival
# X=cbind(1,donner$age,donner$male)

# PRIOR of beta: N_p with mean=beta00 and PRECISION matrix P0
# Fisso la media pari al vettore nullo e la matrice di varianza c*(X^T X)^(-1)
#  (è simile alla prior g di Zellner per il caso lineare classico)
beta00 = c(0,0,0)
c0     = 100        # prior weight is 1% wrt data weight
## HOMEWORK: c0=1 e c0=10
P0 = t(X)%*%X/c0
# prior = list with components beta, the prior mean, and P, the prior PRECISION matrix
#         P è quella che abbiamo indicato con B0^(-1)
inv=function(X){
  # RETURN THE INVERSE OF THE SYMMETRIC MATRIX X
  EV=eigen(X)
  EV$vector%*%diag(1/EV$values)%*%t(EV$vector)
}
B0= inv(P0)
B0    #B0= c0*(t(X)%*%X)^{-1}
```

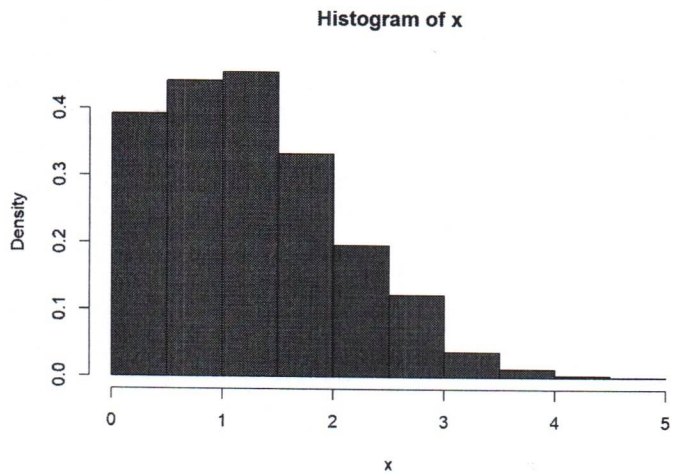
```
##             [,1]        [,2]        [,3]
## [1,] 20.7048302 -0.45187222 -6.16960723
## [2,] -0.4518722  0.01454524 -0.01599977
## [3,] -6.1696072 -0.01599977 10.01759974
```

```r
# bayes.probit function implements a Gibbs sampler to compute the posterior  of
# (beta, Z) parameter, beta= 3 regression parameters, Z= vector of  n Latent variables
# Gaussian PRIOR: full conditionals have closed-form analytic expressions

help(bayes.probit)
```

```r
print(bayes.probit)
```

```
## function (y, X, m, prior = list(beta = 0, P = 0))
## {
##     rtruncated = function(n, lo, hi, pf, qf, ...) qf(pf(lo, ...) +
##         runif(n) * (pf(hi, ...) - pf(lo, ...)), ...)
##     if (sum(prior$P) == 0)
##         log.marg = NULL
##     beta0 = prior$beta
##     BI = prior$P
##     N = length(y)
##     fit = glm(y ~ X - 1, family = binomial(link = probit))
##     beta.s = fit$coef
##     p = length(beta.s)
##     beta = array(beta.s, c(p, 1))
##     beta0 = array(beta0, c(p, 1))
##     BI = array(BI, c(p, p))
##     Mb = array(0, dim = c(m, p))
##     lo = c(-Inf, 0)
##     hi = c(0, Inf)
##     LO = lo[y + 1]
##     HI = hi[y + 1]
##     postvar = solve(BI + t(X) %*% X)
##     aa = chol(postvar)
##     BIbeta0 = BI %*% beta0
##     post.ord = 0
##     for (i in 1:m) {
##         z = rtruncated(N, LO, HI, pnorm, qnorm, X %*% beta, 1)
##         mn = solve(BI + t(X) %*% X, BIbeta0 + t(X) %*% z)
##         beta = t(aa) %*% array(rnorm(p), c(p, 1)) + mn
##         post.ord = post.ord + dmnorm(beta.s, mn, postvar)
##         Mb[i, ] = beta
##     }
##     if (sum(BI) > 0) {
##         log.f = sum(y * log(fit$fitted) + (1 - y) * log(1 - fit$fitted))
##         log.g = dmnorm(beta.s, beta0, solve(BI), log = TRUE)
##         log.marg = log.f + log.g - log(post.ord/m)
##     }
##     return(list(beta = Mb, log.marg = log.marg))
## }
## <bytecode: 0x000000000e63a718>
## <environment: namespace:LearnBayes>
```

```
## Keypoint is simulation from the full-conditionals of Z_i| beta, y_i
## Nella funzione, questo è il passo z = rtruncated(N, LO, HI, pnorm, qnorm, X %*% beta, 1)
## IN GENERAL, to simulate a rv Z from N(mu,1) truncated on (a,b):
## Z=\Phi^{-1}(\phi(a-mu)+ U*(\Phi(b-mu)-\Phi(a-mu)) ) + mu,
## where U~ U(0,1)
## HERE a=0, b=+\infty
u1  <- runif(2000)
mu=1
Fa1 <- pnorm(0,mu) #\Phi(a-mu)
Fa2 <- 1  #\Phi(b-mu)
u1  <- runif(2000)
u1  <- u1*(Fa2-Fa1)+Fa1
x=qnorm(p=u1)+mu
x11()
hist(x,freq=F,col="red")
```



```
## Full conditional of the beta: posterior of a Gaussian linear model
## when data are the latent variables Z_i

# La funzione vuole in input il vettore delle risposte binarie y, la matrice disegno X, e
# il numero di iterazioni del Gibbs sampler m e i parametri della PRIOR gaussiana per beta
# Gives a simulated sample from the joint posterior distribution of the regression vector for
# a binary response regression model with a probit link and a informative normal(beta, P) prior.
# bayes.probit(y,X,m,prior=list(beta=0,P=0))
## Se NON specifico la lista "prior", assumo la prior IMPROPRIA per i parametri di regressione beta,
## cioè proporzionale ad una costante...

# In uscita dà beta, la matrice dei valori simulati dei parametri di regressione
# e log.marg, una stima Monte Carlo della "log marginal likelihood of the model"

m=10000
fitBayes=bayes.probit(survival,X,m,list(beta=beta00,P=P0)) # PROPER PRIOR
head(fitBayes)
```

we run Gibbs sampler for 10000 iterations

```
## $beta
##              [,1]          [,2]          [,3]
##      [1,] 2.610115464 -6.507676e-02 -0.9034605385
##      [2,] 2.357362938 -5.935107e-02 -1.2097489897
##      [3,] 2.742370181 -4.949714e-02 -1.7156731899
##      [4,] 3.814792311 -7.768734e-02 -1.7572837432
##      [5,] 2.771811795 -7.668783e-02 -0.9131793992
##      [6,] 3.258375830 -8.881271e-02 -0.5352991919
##      [7,] ..
```

$beta: dataframe that contains the output of the Gibbs sampler

(β_0) (β_1) (β_2)

moreover fitBayes has: $log.marg, which is the logarithm of the marginal density in this model evaluated in the dataset (here $log.marg = -31.52555)

```
# Posterior mean and standard deviations of the regression
# coefficients via the MCMC
apply(fitBayes$beta,2,mean)
```

```
## [1]  1.97434765 -0.04730765 -0.97437206
```

```
apply(fitBayes$beta,2,sd)
```

```
## [1] 0.76541155 0.02009488 0.45154202
```

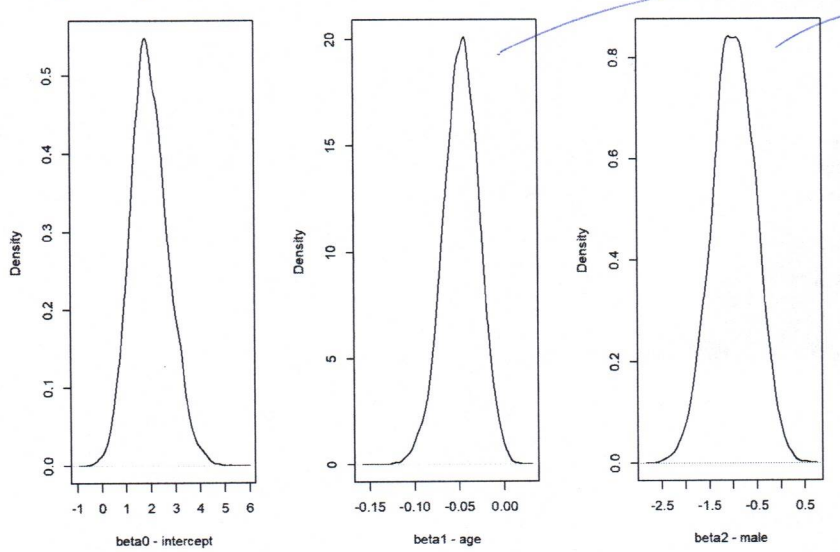
```
# Le medie a posteriori sono abbastanza simili  alle stime frequentiste, perchè costruite a partire
# dal Gibbs Sampler in cui campiono i beta da una Gaussiana, con media data dalla media ponderata
# della stima di max ver (che pesa 100 volte di più della media prior) e della media a priori
fit$coefficients
```

```
##           X        Xage       Xmale
##  1.91730001 -0.04570867 -0.95827865
```

```
x11()
par(mfrow=c(1,3))
plot(density(fitBayes$beta[,1]), xlab='beta0 - intercept',main=' ')
plot(density(fitBayes$beta[,2]), xlab='beta1 - age',main=' ')
plot(density(fitBayes$beta[,3]), xlab='beta2 - male',main=' ')
```

Marginal posteriors of $\beta_0, \beta_1, \beta_2$:

both of the supports are mostly on the negative axes. We can compute the masses assigned by these posteriors to $(-\infty, 0)$:




```
mean(fitBayes$beta[,2]<0)
```

```
## [1] 0.995
```

β_1

β_2

```
mean(fitBayes$beta[,3]<0)
```

```
## [1] 0.9881
```

```
# Both beta1 and beta2 are significative, and in both cases the
# marginal posterior is concentrated on negative values.
# Survival probability decreases as age increases and it is
# smaller for men than for women.
# Survival probability, given AGE and MALE values, is
# \Phi(beta0+beta1*age+beta2*male), i.e. it is a function of
#                                    the betas
# Let us compute its posterior distribution
# for a man (male=1), as age varies.
# Coincide con la banda di previsione della sopravvivenza predittiva
#     di un "nuovo" paziente maschio.
a=seq(15,65) #all integers from 15 to 65
X1=cbind(1,a,1)
#C'è una funzione del package che fa questo:
p.male=bprobit.probs(X1,fitBayes$beta) #posterior draws of
                                       #\Phi(beta0+beta1*age+beta2)
# This is a whole distribution: we plot, for each value of age,
# the posterior median and quantiles (0.05 and 0.95), i.e.
# 90% CI of the posterior of this function of betas
x11()
par(mfrow=c(1,1))
plot(a,apply(p.male,2,quantile,.5),type="l",ylim=c(0,1), xlab="age",ylab="Probability of Survival")
lines(a,apply(p.male,2,quantile,.05),lty=2)
lines(a,apply(p.male,2,quantile,.95),lty=2)
# For each fixed age, this is the 90% CI of the survival probability
# of a male
# \Phi(beta0+beta1*age+beta2*male)
lines(a,apply(p.male,2,mean),lty=3,col='red')
# this is the posterior predictive probability that Y_i^new
# at all ages would have survived


### ------------------------------------------------------------------------------
### Model choice - we compute the log of the marginal density of the data for 4 different models
### and make 2-by-2 comparisons.
### ------------------------------------------------------------------------------
# Choose the model corresponding to the highest marginal (NON è un criterio OTTIMO):
y=donner$survival
X=cbind(1,donner$age,donner$male)

# Modello 1: tutte e tre le covariate (compresa l'intercetta)
bayes.probit(y,X,1000,list(beta=beta00,P=P0))$log.marg
```

plot on the other page (dietro)

```
## [1] -31.5995
```

```
# Modello 2: intercetta, male
bayes.probit(y,X[,-2],1000,
   list(beta=beta00[-2],P=P0[-2,-2]))$log.marg
```

```
## [1] -32.7634
```

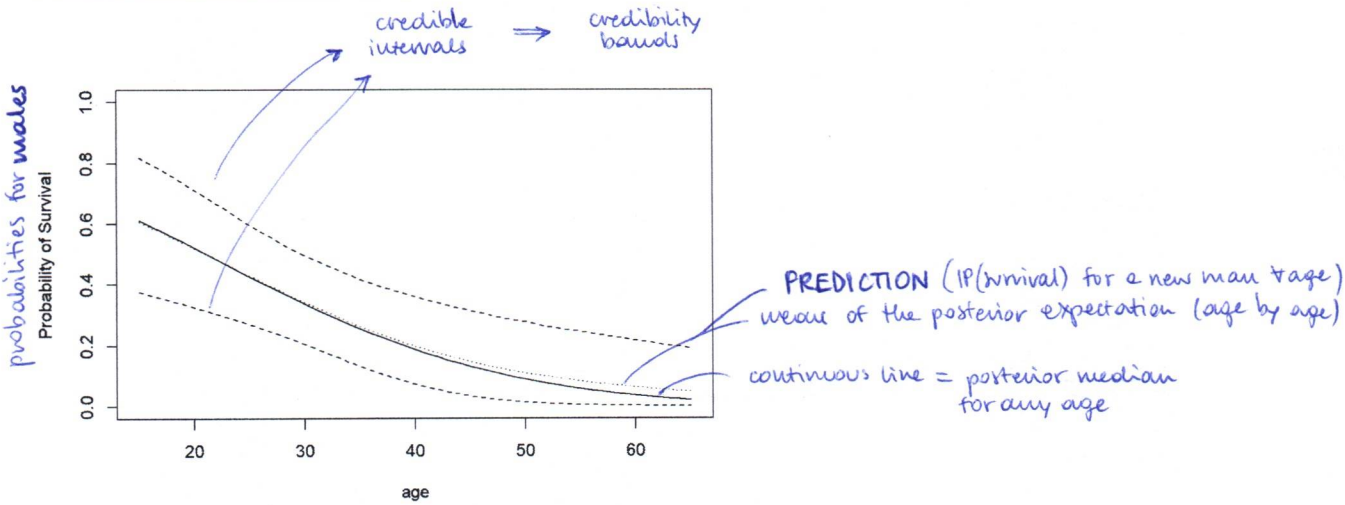
```
# Modello 3: intercetta, age
bayes.probit(y,X[,-3],1000,
   list(beta=beta00[-3],P=P0[-3,-3]))$log.marg
```

```
## [1] -32.02235
```

```
# Modello 4: solo intercetta
bayes.probit(y,X[,-c(2,3)],1000,
   list(beta=beta00[-c(2,3)],P=P0[-c(2,3),-c(2,3)]))$log.marg
```

```
## [1] -32.9966
```

```
### ------------------------------------------------------------------------------
### Model 1 is the best according to this criterion


### ------------------------------------------------------------------------------
#### Package MCMCpack  http://mcmcpack.berkeley.edu/
#### Paper on JStatSoft http://www.jstatsoft.org/article/view/v042i09
#### Better download the package from https://cran.r-project.org/web/packages/MCMCpack/index.html
#### In alternativa: arm, MCMCglm
#
# MODELS implemented in the package: linear regression (with Gaussian errors),
# a hierarchical longitudinal model with Gaussian errors,
# a probit model, a logistic regression model, a one-dimensional item response theory model,
# a K-dimensional item response theory model, a normal theory factor analysis model,
# a mixed response factor analysis model, an ordinal factor analysis model,
# a Poisson regression, a tobit regression, a multinomial logit model,
# a dynamic ecological inference model, a hierarchial ecological inference model,
# and an ordered probit model.
# The package also contains densities and random number generators for
# commonly used distributions that are not part of the
# standard R distribution, a general purpose Metropolis sampling algorithm,
# and some utilities for visualization and data manipulation.

### ------------------------------------------------------------------------------
###### https://cran.r-project.org/web/views/Bayesian.html  is a site for R packages
######         for Bayesian inference
### ------------------------------------------------------------------------------
#install.packages("MCMCpack")
library(MCMCpack)
```




```
?MCMCprobit
?MCMClogit

library(LearnBayes)
data(donner)
y=donner$survival
X=cbind(1,donner$age,donner$male)

## mlfit <- glm(y ~ X, family = binomial(link = probit), x = TRUE, y = TRUE)

# Input: response vector, covariate (design) matrix and number of simulations to run
# Output: MCMC sample forom the posterior distribution of beta
# The fitted prior is Gaussian, beta ~ N_p(b0,B0^(-1))
#################### B0 HERE is the PRECISION matrix in the prior
## Here we use Zellner g prior: beta ha distribuzione Gaussiana con media beta00 e matrice di varianza c*(X^T X)^(-1)

beta00=c(0,0,0); c0=100 #La prior ha un peso dell'1% rispetto al campione
## Provate con c0=1 e c0=10
P0=t(X)%*%X/c0
P0
```

```
##       [,1]   [,2] [,3]
## [1,]  0.45  14.31 0.30
## [2,] 14.31 523.93 9.65
## [3,]  0.30   9.65 0.30
```

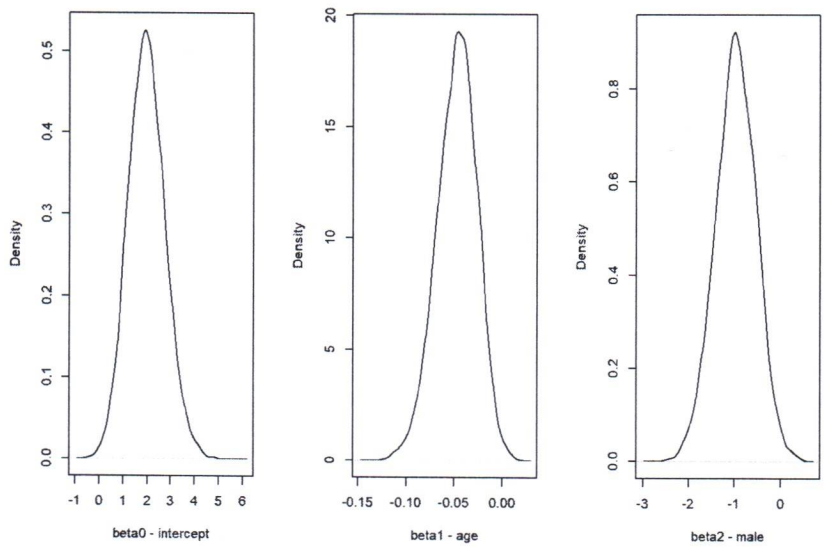
```
bfit <- MCMCprobit(y ~ donner$age +as.factor(donner$male), b0=beta00, B0=P0,marginal.likelihood="Chib95")
 # CAREFUL: here B0 denotes the PRECISION matrix for beta

bfitout<-as.matrix(bfit)

apply(bfitout,2,mean)
```

```
##              (Intercept)              donner$age as.factor(donner$male)1
##               1.98531685             -0.04768716             -0.97363200
```

```
apply(bfitout,2,sd)
```

```
##              (Intercept)              donner$age as.factor(donner$male)1
##               0.76831631              0.02090254              0.44429764
```

```
x11()
par(mfrow=c(1,3))
plot(density(bfitout[,1]), xlab='beta0 - intercept',main=' ')
plot(density(bfitout[,2]), xlab='beta1 - age',main=' ')
plot(density(bfitout[,3]), xlab='beta2 - male',main=' ')
```




```
## HOMEWORK: fit the logit model to this dataset using
##       MCMCpack
```