Lezioni

In *corsivo* sono i commenti ai risultati.

20 11 17 Lezione 1: Esempio Bayes' Billard

Modello Beta-Binomiale

Simulo i valori di p.

```
#totale palline blu
n <- 10

#simuliamo delle realizzazioni pseudo-casuali con runif
#creo quindi il vettore delle probabilità
nsim <- 10^6
p <- runif(nsim)
head(p)
```

```
## [1] 0.942715691 0.319666586 0.895402866 0.002267912 0.312874190 0.323390847
```

Genero da binomiale tramite rbinom, cioè vado a creare una probabilità condizionata X|p. Essa rappresenta la mia verosimiglianza, cioè conto ogni volta quante palline blu ho lanciato nelle fissate n posizioni. Distribuzione beta.

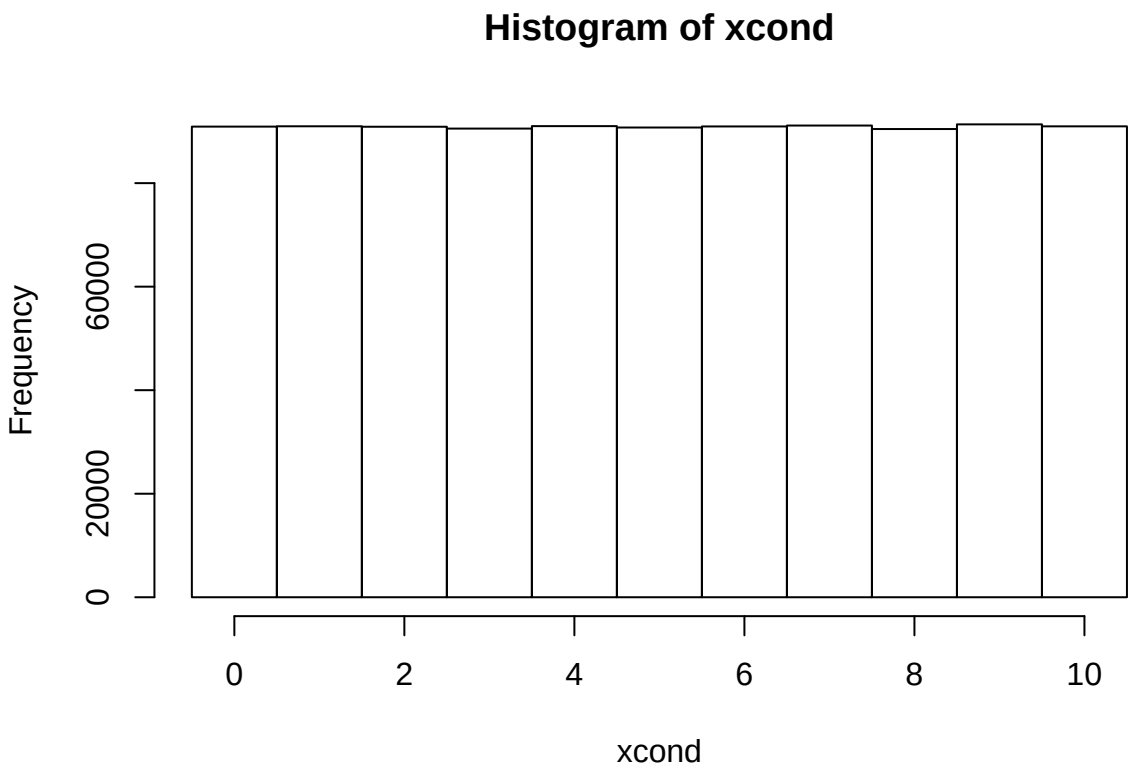
```
xcond <- rbinom(nsim,size = n,prob =  p)
head(xcond)
```

```
## [1] 9 2 8 0 3 1
```

Abbiamo generato real da una binomiale, cioè una condizionata da evento precedente che in questo caso erano realizzazioni pseudo-casuali

Faccio istogramma dei valori generati dalla binomiale, ognuno generato in modo uniforme. cioè il primo valore di xcond è associato al primo valore di p, il secondo col secondo ecc.

```
hist(xcond,
     breaks = seq(-0.5,n+0.5,1))
```




```r
table(xcond)
```

```
## xcond
##     0     1     2     3     4     5     6     7     8     9    10
## 90925 90988 90901 90545 91017 90751 90955 91131 90454 91359 90974
```

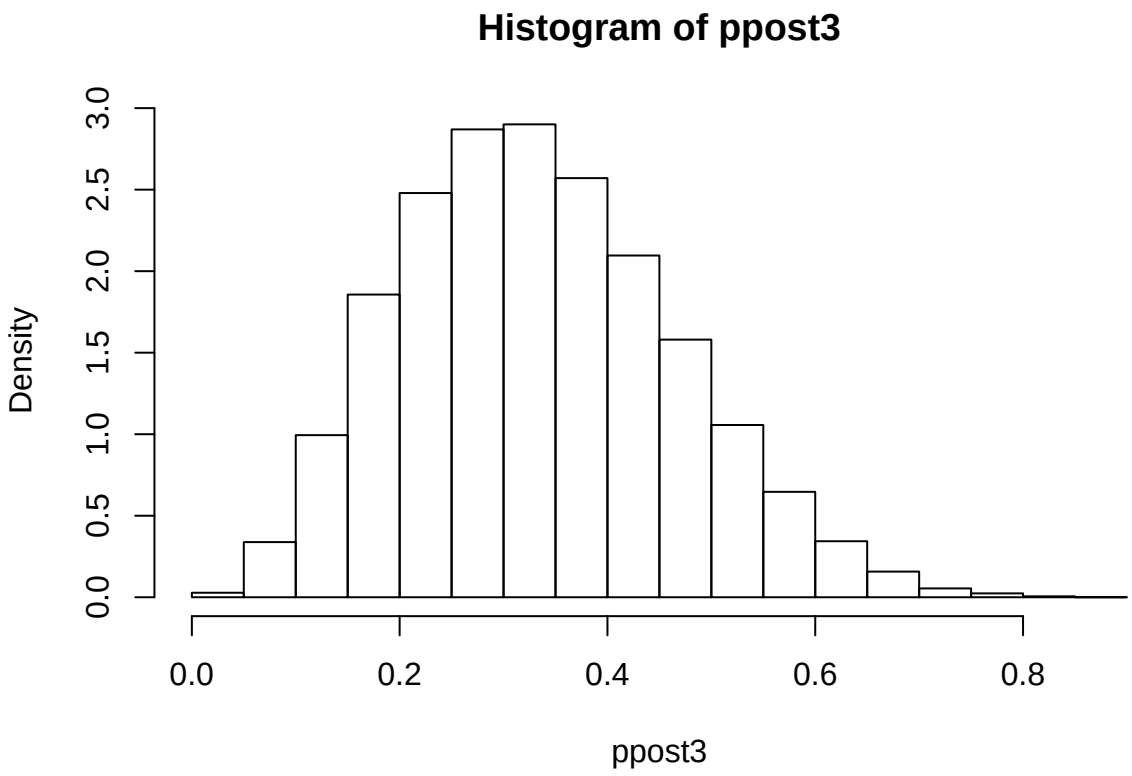
Dato n molto elevato, ho generato gli stessi elementi circa lo stesso numero di volte, Abbiamo generato una sequenza di numeri distribuiti uniformemente tra 0 e 10.

Possiamo calcolare la prob a posteriori di tutti gli eventi ottenuti che hanno valore 3. (cioè per i quali si hanno 3 palline blu prima di una gialla)

```r
#estraggo un vettore che prende solo gli elementi di xcond uguali a 3
A3 <- xcond==3
#creo un sottovettore di p
ppost3 <- p[xcond==3]
```

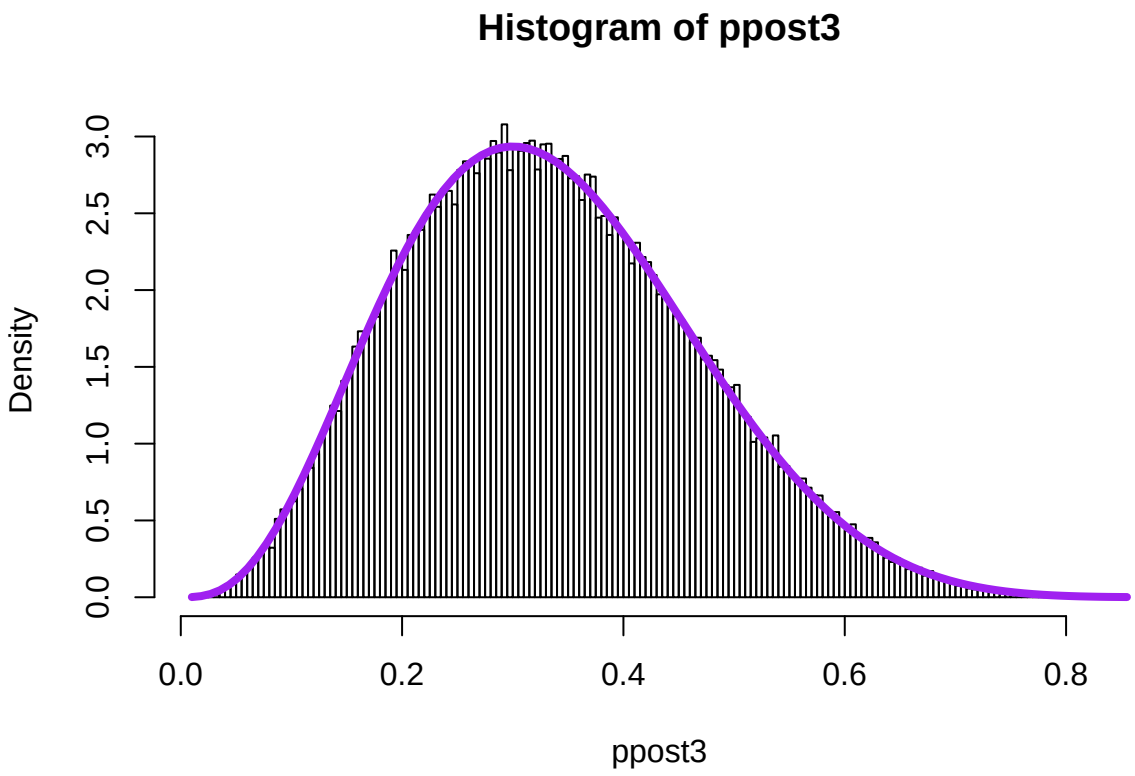
Istogramma della distribuzione a posteriori, condizionata a 3.

```r
hist(ppost3, freq = F)
```



Posso ottenere la stessa distribuzione anche tramite la funzione di R che genera da una vc Beta. Per confronto sovrappongo la densità della Beta all'istogramma ottenuto per la ppost condizionata a x==3.

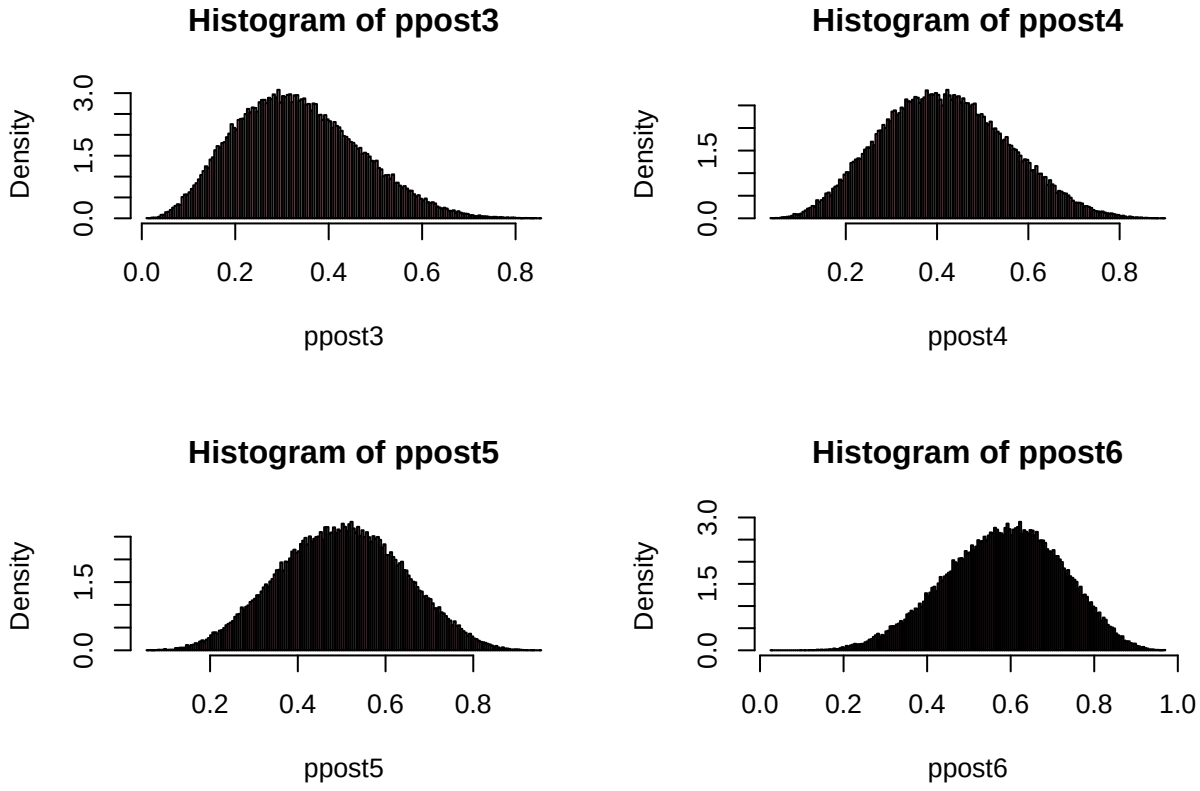
```
#genero realizzazioni dalla vc Beta con parametri di posizione 4 e shape 8
dteb <- rbeta(10^6,4,8)

hist(ppost3, freq = F, breaks = 200)
curve(dbeta(x,4,8),add=T, col="purple", lwd=4)
```



Ogni istogramma è centrato sull'intero.

Posizionare nella stessa finestra grafica le distribuzioni condizionate ppost rispetto ai valori x==4, 5 e 6. Visualizzo la forma delle distribuzioni condizionate.

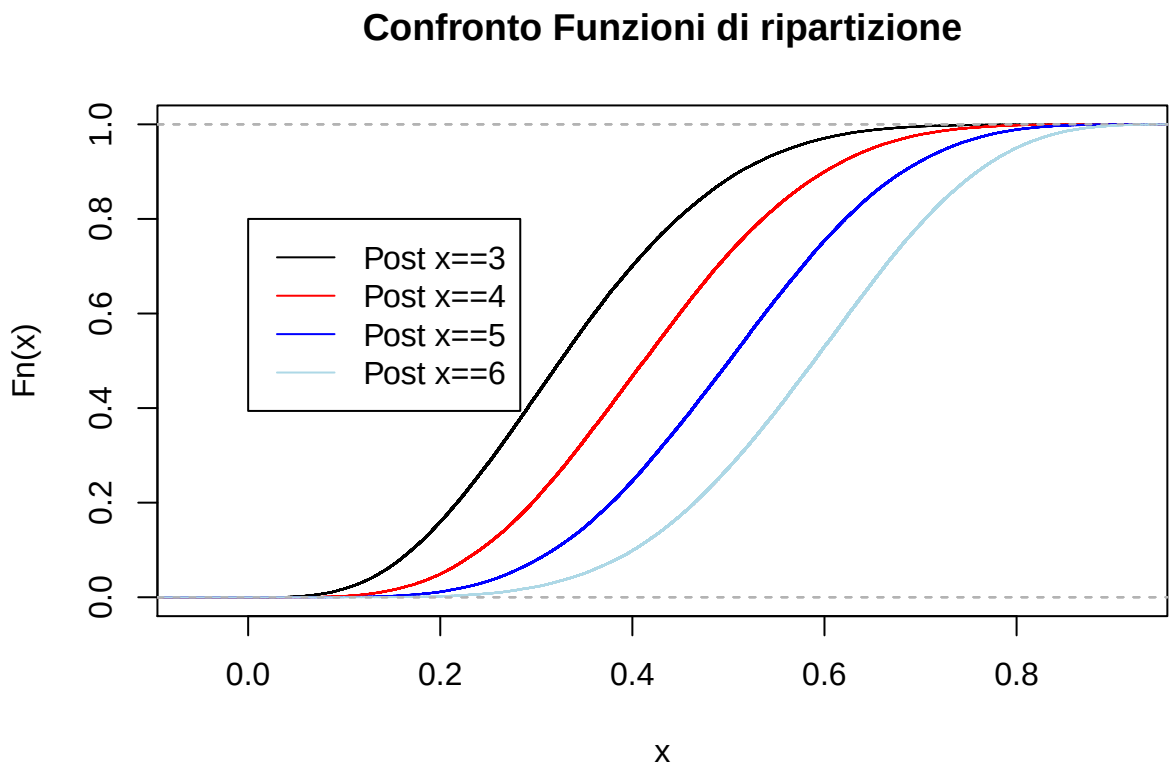
```
ppost4 <- p[xcond==4]
ppost5 <- p[xcond==5]
ppost6 <- p[xcond==6]

par(mfrow=c(2,2))
hist(ppost3, col="pink", breaks = 200, freq = F)
hist(ppost4, col="pink", breaks = 200, freq = F)
hist(ppost5, col="pink", breaks = 200, freq = F)
hist(ppost6, col="pink", breaks = 200, freq = F)
```



Gli istogrammi differiscono per la posizione del valore medio

Riporto le funzioni di ripartizione empirica a confronto.

```
plot(ecdf(ppost3), main="Confronto Funzioni di ripartizione")
plot(ecdf(ppost4),add=T, col="red")
plot(ecdf(ppost5),add=T, col="blue")
plot(ecdf(ppost6),add=T, col="lightblue")
legend(0,0.8,c("Post x==3","Post x==4", "Post x==5", "Post x==6"),
       col=c("black","red","blue","lightblue"), lty=1)
```



All'aumentare del valore a cui condizioniamo,in questo grafico possiamo vedere come cambia la probabilità cumulata.

22 11 17 Lezione 3

Uso della funzione beta.select.

Fa una selezione di beta prior dato che si conoscono 2 quantità d'interesse: restituisce i parametri shape della forma della Beta quando si conoscono 2 quantili d'interesse della distribuzione. Come input vanno specificati i valori di quantile 1 e quantile 2.

Esempio: ore di sonno degli studenti

Voglio conoscere il numero di studenti che dormono almeno un tot di ore di sonno durante una notte. I quantili di interesse di questa distribuzione possono essere ad esempio la mediana che pongo 0.3 e il 90esimo percentile che pongo 0.5

```
#install.packages("LearnBayes")
require(LearnBayes)
```

```
## Loading required package: LearnBayes
```

```
quantile1 <- list(p=0.5,x=0.3) #mediana è il quantile p=0.5 e la pongo pari a 0.3
quantile2 <- list(p=0.9,x=0.5)
```

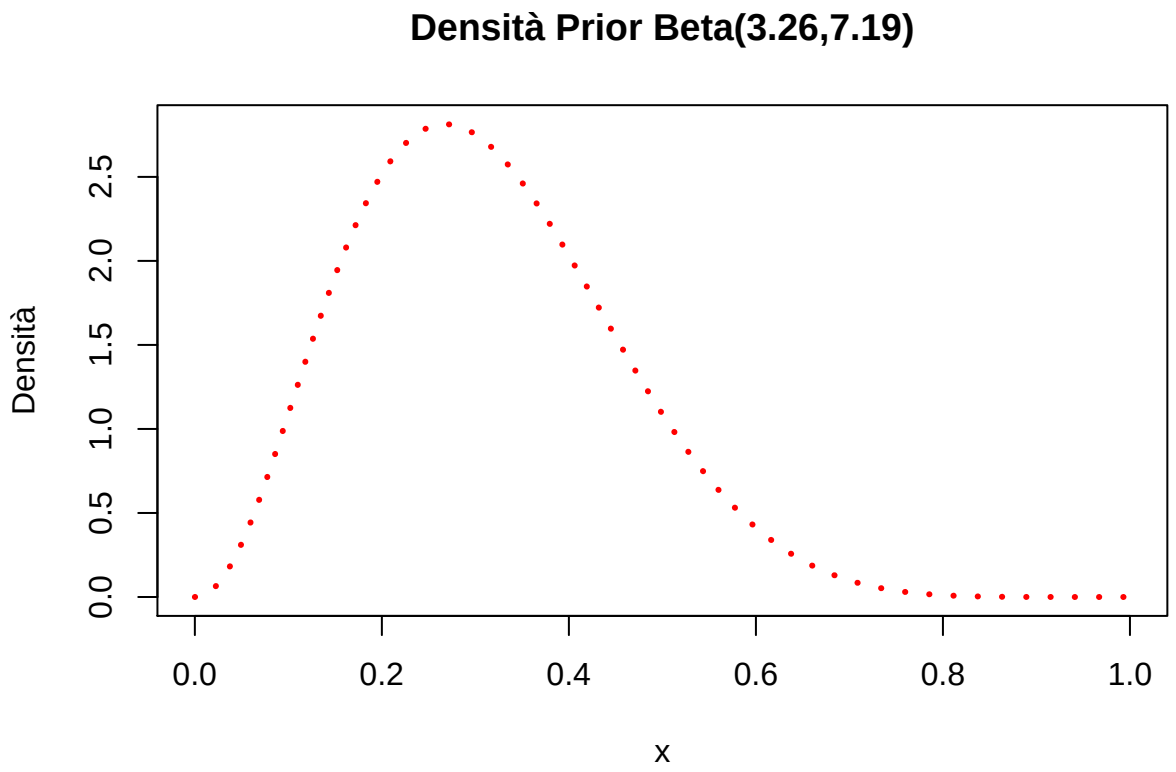
```
beta.select(quantile1,quantile2)
```

```
## [1] 3.26 7.19
```

Otteniamo shape1=3.26=alpha e shape2=7.19=beta

Disegno una beta con questi parametri. Essa rappresenta la nostra distribuzione a priori per il parametro.

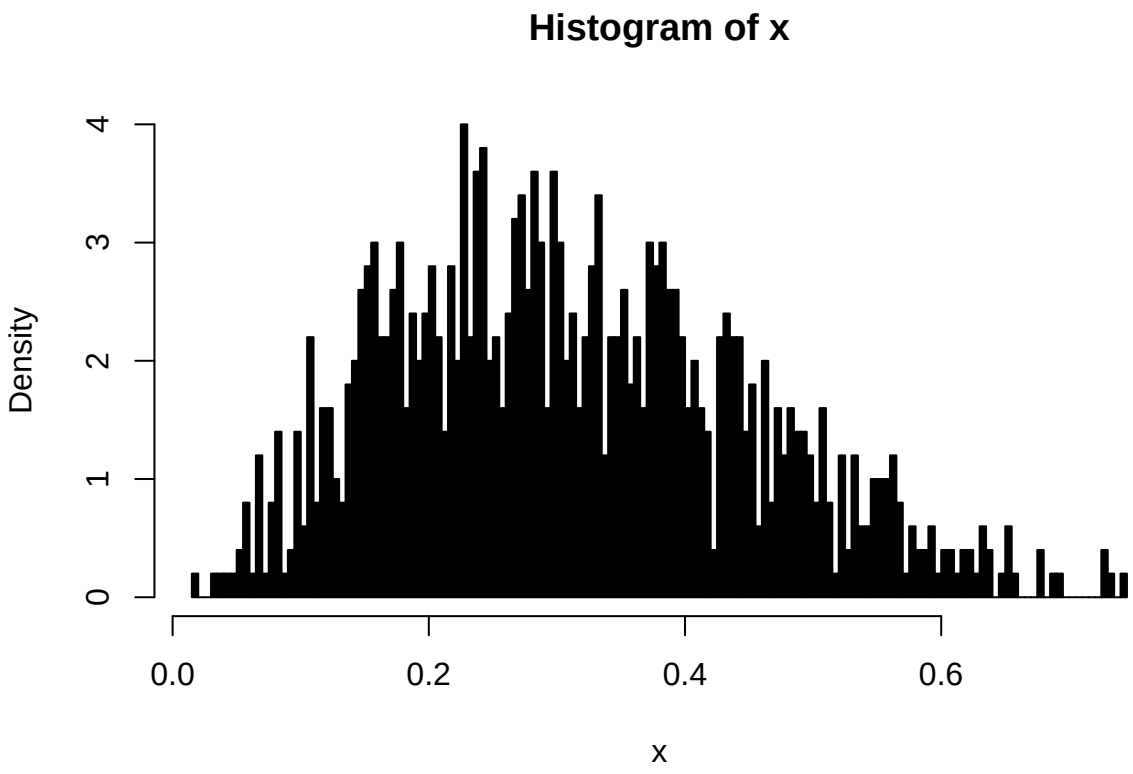
```
a <- 3.26
b <- 7.19
curve(dbeta(x,a,b),col="red", main="Densità Prior Beta(3.26,7.19)",
      from = 0, to = 1, lwd=3, lty=3, ylab="Densità")
```



Ha una coda più pesante a destra. Essendo la beta una distribuzione molto flessibile, la sua forma dipende molto dai parametri e può variare molto.

Generiamo 1000 realizzazioni dalla prior Beta(3.26,7.19) e visualizziamo al forma della dsitribuzione.

```
x <- rbeta(1000,a,b)
hist(x,col="black", freq=F, breaks = 200)
```



Disegniamo la funzione di ripartizione empirica evidenziano la mediana e il 90esimo percentile della distribuzione.

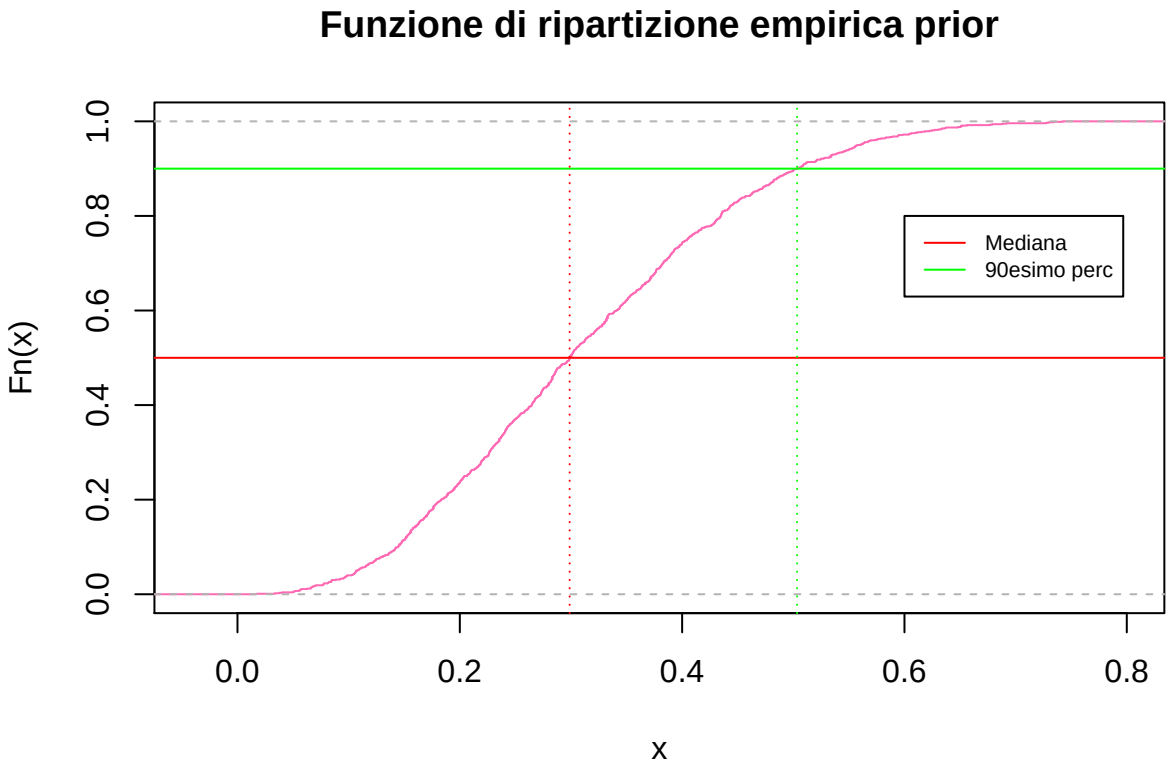
I due quantili d'interesse sui dati generati dovrebbero essere molto simili a quelli supposti in partenza.

```
qq <- quantile(x,c(0.5,0.9)); qq
```

```
##       50%       90%
## 0.2987889 0.5034278
```

Effettivamente sono 0.3 e 0.5 come posto a priori. La generazione è esatta

```
plot(ecdf(x), col="hotpink", do.point=F,
     main="Funzione di ripartizione empirica prior")
abline(h=c(0.5,0.9), col=c("red","green"))
abline(v=c(qq[1],qq[2]),col=c("red","green"), lty=3)

legend(0.6,0.8,c("Mediana","90esimo perc"),
       col=c("red","green"), lty=1, cex = 0.7)
```



Come interpreto i valori? Il 50% degli studenti dorme almeno fino a 7 ore e quindi meno di 8 ore (0,3 che è pari a 7/24), il 90% degli studenti dorme fino a 12 ore e quindi meno di 12 ore. QUesta interpretazione è realistica.

In media mi aspetto che siano 7 ore/notte. 7/24

```
7/24
```

```
## [1] 0.2916667
```

A priori in base al mio giudizio fisso il valore medio di ore dormite pari a 0.3

Da un campione di intervistati risulta che su 27 studenti 11 dichiarano di aver dormito almeno 8 ore nella notte precedente l'intervista. Fisso il numero di successi s, ovvero la numerosità complessiva che può assumere l'evento successo a priori, cioè sulla base della conoscenza fornita dalle osservazioni.

```
s <- 11
f <- 27-11 #16
```

Dimostro che la prior è una binomiale e quindi una Beta, secondo il modello Beta-Binomiale.
??????????????????????????????

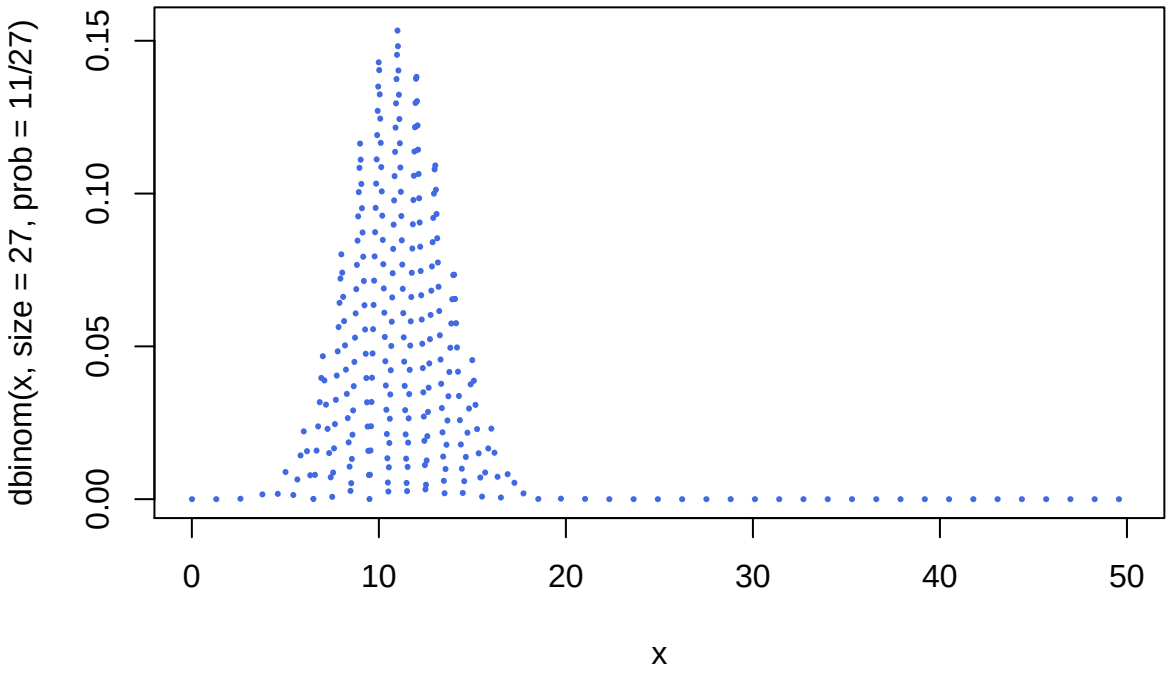
Disegno la distribuzione binomiale a priori considerando i dati osservati. Essa è uguale alla beta di parametri a e b.

```
curve(dbinom(x,size = 27,prob = 11/27), xlim = c(0,50),
      col="royalblue", lwd=3,lty=3)
```

```
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 0.500000
```

```
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 1.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 2.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 3.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 4.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 5.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 6.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 7.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 8.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 9.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 10.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 11.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 12.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 13.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 14.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 15.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 16.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 17.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 18.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 19.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 20.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 21.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 22.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 23.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 24.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 25.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 26.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 27.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 28.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 29.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 30.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 31.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 32.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 33.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 34.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 35.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 36.500000
```

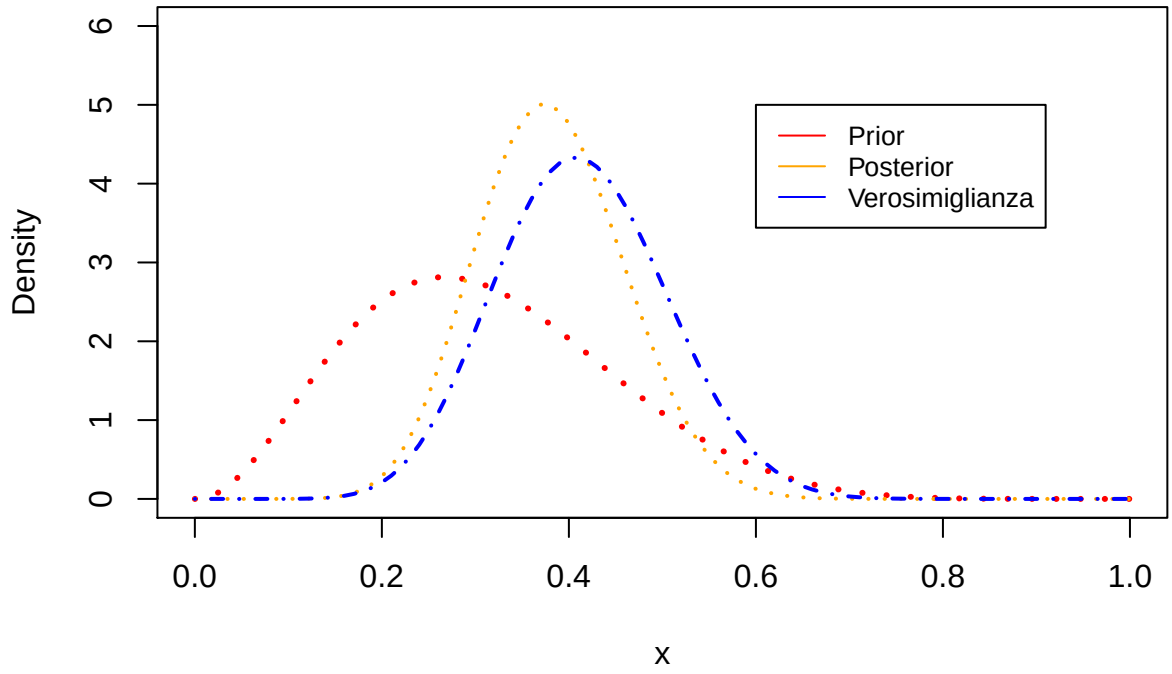
```
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 37.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 38.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 39.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 40.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 41.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 42.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 43.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 44.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 45.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 46.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 47.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 48.500000
## Warning in dbinom(x, size = 27, prob = 11/27): non-integer x = 49.500000
```



Posterior per la proporzione d'interesse

Confronto la prior, la curva di verosimiglianza e la distribuzione a posteriori ottenuta applicando la Bayes' rule.

```r
#a priori fissata coi 2 quantili della distribuzione
curve(dbeta(x,a,b),col="red",   ylim=c(0,6),ylab="Density",
      from = 0, to = 1, lwd=3, lty=3)

#a posteriori è una Beta
curve(dbeta(x,a+s,b+f),col="orange", lwd=2, lty=3, add=T)

#verosimiglianza è una Beta(k+1,n-k+1)
curve(dbeta(x,s+1,f+1),col="blue", lwd=2, lty=4, add=T)

legend(0.6,5,c("Prior","Posterior","Verosimiglianza"),
       col=c("red","orange","blue"), lty=1, cex = 0.8)
```



29 11 17 Lezione 4

Definire la matrice di transizione e passeggiata aleatoria.

La matrice richiede un vettore in cui mettere gli elementi, il numero di righe e il numero di colonne. Le matrici stocastiche sono matrici quadrate. Ad esempio 3x3 in cui mettiamo proboabilità comndizionate a somma 1 per riga.

```r
vett1 <- c(1/4,1/2,1/4)
vett2 <- c(0,1/2,1/2)
vett3 <- c(1/8,3/4,1/8)
```

```r
A <- matrix(c(vett1,vett2,vett3),nrow = 3,ncol = 3,byrow = T)
A
```

```
##       [,1] [,2]  [,3]
## [1,] 0.250 0.50 0.250
## [2,] 0.000 0.50 0.500
## [3,] 0.125 0.75 0.125
```

```r
#verifico somma a 1 delle righe: MATRICE STOCASTICA
apply(A,1,sum)
```

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

Verificando che le righe sommano a 1, abbiamo verificato che A è una matrice stocastica. Come la leggo: dato che parto in stato 1, la prob di arrivare in stato 1 è do 0.25; dato che parto in stato 1, la prob di arrivare in stato 2 è di 0.5; dato che parto in stato 1, la prov di arrivare in stato 3 è di 0.25; dato che parto in stato 2, la prob di arrivare in stato 1 è di 0; dato che parto in stato 2, la prob di arrivare in stato 2 è di 0.5; ecc...

Operazioni matriciali: %*% esegue il prodotto tra matrici AxA mi dà una matrice di dimensioni uguali ad A Anche AxAXA mi dà matrice quadrata di stesso ordine

```r
A2 <- A%*%A
A2
```

```
##          [,1]    [,2]     [,3]
## [1,] 0.093750 0.56250 0.343750
## [2,] 0.062500 0.62500 0.312500
## [3,] 0.046875 0.53125 0.421875
```

```r
A3 <- A2%*%A
A3
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.06640625 0.5859375 0.3476562
## [2,] 0.05468750 0.5781250 0.3671875
## [3,] 0.06445312 0.6054688 0.3300781
```

Matrice di equilibrio: tutte le probabilità condizionate per riga sono uguali

Per cercare la distribuzione di equilibrio, vado avanti con l'ordine, fino a ordine 10 (A10). Creo la passeggiata aleatoria.

```r
A4 <- A3%*%A
A5 <- A4%*%A
A6 <- A5%*%A
A7 <- A6%*%A
A8 <- A7%*%A
A9 <- A8%*%A
A10 <- A9%*%A
A10
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.05882344 0.5882347 0.3529419
## [2,] 0.05882456 0.5882390 0.3529364
## [3,] 0.05882183 0.5882292 0.3529490
```

Aumentano l'ordine n vedo che tendo ad ottnere la stessa distribuzione per riga.lo verifico tramite scomposizione in autovettore e autovalori.

04 12 17 Lezione 5

Catene di Markov: scomposizione della matrice in autovettori e autovalori

Miro ad ottenere la matrice di equilibrio. Riprendo la matrice A usata a lezione 4.

```
vett1 <- c(1/4,1/2,1/4)
vett2 <- c(0,1/2,1/2)
vett3 <- c(1/8,3/4,1/8)
A <- matrix(c(vett1,vett2,vett3),nrow = 3,ncol = 3,byrow = T)
A2 <- A%*%A
A3 <- A2%*%A
A4 <- A3%*%A
A5 <- A4%*%A
A6 <- A5%*%A
A7 <- A6%*%A
A8 <- A7%*%A
A9 <- A8%*%A
A10 <- A9%*%A
```

Autovettori e autovalori di A

Estraggo gli autovalori e gli autovettori di questa matrice, tramite la funzione che restituisce la scomposizione spettrale della matrice stocastica A –> "eigen". Dalla teoria, sappiamo che se la matrice è definita in un certo modo ho almeno un autovaloe= 1 ed esso se esiste è il maggiore.

Trasposizione di A: inverto righe e colonne. VAdo meglio con la funzione eigen.

```
t(A)
```

```
##      [,1] [,2]  [,3]
## [1,] 0.25  0.0 0.125
## [2,] 0.50  0.5 0.750
## [3,] 0.25  0.5 0.125
```

```
vA <- eigen(t(A))
vA
```

```
## eigen() decomposition
## $values
## [1]  1.0000000 -0.3201941  0.1951941
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] 0.08543577 -0.1702700  0.8144420
## [2,] 0.85435766 -0.6064256 -0.4573522
## [3,] 0.51261459  0.7766956 -0.3570898
```

Devo prendere l'autovettore corrispondente all'autovalore più alto. In questo caso identifico la prima colonna

Lo estraggo

```
vA$vectors[,1]
```

```
## [1] 0.08543577 0.85435766 0.51261459
```

Normalizzazione degli autovettori

La matrice di equilibrio la ottengo usando gli autovettori normalizzati.

```
SvA <- sum(vA$vectors[,1])
vA$vectors[,1]/SvA
```

```
## [1] 0.05882353 0.58823529 0.35294118
```

Sono le prob di ogni songolo stato prese a sè, al limite. In realtà questo limite non si raggiunge mai ma è importante per vedere dove tende il sistema.

Distribuzione di equilibrio

```
A10
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.05882344 0.5882347 0.3529419
## [2,] 0.05882456 0.5882390 0.3529364
## [3,] 0.05882183 0.5882292 0.3529490
```

E' uguale a quella ottenuta con il concatenamento

Random Walk: passeggiata casuale

Abbiamo 15 stati possibili. SUlla diagonale ho tutti 0 perchè non posso mai rimanere sullo stesso stato, posso solo andare negli stati vicini tramite +1 e -1. La partenza è posizionata sullo stato 5.

```
ini <- 0 #valore dello stato iniziale
fine <- 15  #valore dello stato finale
p <- 0.8  #prob di transizione
nsim <- 150 #numero di transizioni
x <- rep(ini,nsim)

#devo inizializzare il processo rispetto ad un valore del passo 1
x[1] <- 5
```

I passaggi simulati del random li ricreo con un ciclo for

```
for (i in 2:nsim){  #il ciclo parte dal passo 2
 #ora devo creare le due condizioni possibili
  if(x[i-1]==ini || x[i-1]==fine)
    {
    x[i] <- x[i-1]
    #stati assorbenti da cui non devo più uscire, perciò lo stato al passo i
    #deve essere posto uguale allo stato precedente quindi
    #sto creando uno stop: non esco più
  }
  else
  {
    x[i] <- x[i-1]+ sample(c(1,-1),1,prob = c(p,1-p))
    #riempio l'elemento alla posizione i-esima
    #con quello che ho al passo precedente più un valore di W che è Bernoulli
    #in 1 e -1
  }
```

```
}
x
```

```
##   [1]  5  4  5  6  7  8  9 10  9 10 11 12 11 12 13 14 15 15 15 15 15 15 15
##  [24] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
##  [47] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
##  [70] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
##  [93] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [116] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [139] 15 15 15 15 15 15 15 15 15 15 15 15
```

*Arrivo allo stato assorbente 15 (e non 0) più probabilmente perchè con una prob di 0.8 aggiungo +1 (quindi vado in avanti) piuttosto che

Quanti sono i passi che occorrono per arrivare allo stato assorbente?

```
sum(x!=15)
```

```
## [1] 16
```

```
sum(x!=15)/nsim
```

```
## [1] 0.1066667
```

Su 150 simulazioni dello stato, ho impiegato 10 passi per raggiungere lo stato assorbente 15. Per cui la freq relativa degli stati differenti da 15 è del 7%

Cosa succede cambiando p. Tengo p=0.5

```
p1 <- 0.5
for (i in 2:nsim){  #il ciclo parte dal passo 2
 #ora devo creare le due condizioni possibili
  if(x[i-1]==ini || x[i-1]==fine)
    {
    x[i] <- x[i-1]
    #stati assorbenti da cui non devo più uscire, perciò lo stato al passo i
    #deve essere posto uguale allo stato precedente quindi
    #sto creando uno stop: non esco più
  }
  else
  {
    x[i] <- x[i-1]+ sample(c(1,-1),1,prob = c(p1,1-p1))
    #riempio l'elemento alla posizione i-esima
    #con quello che ho al passo precedente più un valore di W che è Bernoulli
    #in +1 e -1
  }


}
x
```

```
##   [1] 5 4 3 4 3 2 3 4 5 6 5 4 5 4 3 2 3 2 3 4 3 4 3 2 1 2 1 2 1 2 1 2 3 2 1
##  [36] 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [106] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [141] 0 0 0 0 0 0 0 0 0 0
```

```
sum(x!=0)
```

```
## [1] 37
```

```
sum(x!=0)/nsim
```

```
## [1] 0.2466667
```

Impiego tot passi per arrivare ad uno dei due stati assorbenti, cioè il tot% degli stati simulati Arrivo più facilmente a 0 rispetto che a 15, nonostante le prob identiche per la direzione del passo, perchè lo stato iniziale è più vicino allo 0.

Cosa succede se lo stato di partenza è 15 e p=0.5.

```
p1 <- 0.5
x[1] <- 14
for (i in 2:nsim){  #il ciclo parte dal passo 2
 #ora devo creare le due condizioni possibili
  if(x[i-1]==ini || x[i-1]==fine)
    {
    x[i] <- x[i-1]
    #stati assorbenti da cui non devo più uscire, perciò lo stato al passo i
    #deve essere posto uguale allo stato precedente quindi
    #sto creando uno stop: non esco più
  }
  else
  {
    x[i] <- x[i-1]+ sample(c(1,-1),1,prob = c(p1,1-p1))
    #riempio l'elemento alla posizione i-esima
    #con quello che ho al passo precedente più un valore di W che è Bernoulli
    #in +1 e -1
  }
}
x
```

```
##   [1] 14 13 12 11 12 11 10 11 12 13 12 13 12 11 12 11 10  9  8  7  6  5  4
##  [24]  5  6  7  8  7  8  9 10  9  8  9 10  9  8  9 10  9 10 11 10  9 10 11
##  [47] 10  9  8  7  8  7  6  7  8  7  8  7  6  7  8  7  6  7  6  7  6  5  6
##  [70]  5  6  7  8  9 10 11 12 13 14 13 12 11 10  9  8  9  8  9  8  7  6  5
##  [93]  4  3  2  1  2  1  2  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [116]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [139]  0  0  0  0  0  0  0  0  0  0  0  0
```

Partendo molto vicino ad uno degli stati assorbenti, il processo va quasi sempre su quello stato assorbente.

Stima di un modello di transizione di Markov. Esempio datadrug.

Set of functions to fit latent Markov models. In particolare useremo la funzione est_mc_basic.

```
require(LMest)
```

```
## Loading required package: LMest
```

```
## Loading required package: MASS
```

```
## Loading required package: MultiLCIRT
```

```
## Loading required package: limSolve
```

Uso i dati data_drug. Longitudinal dataset about marijuana consumption measured by ordinal variables with 3 categories with increasing levels of consumption. 51 observations on the following 6 variables:reported drug

use at the 1st occasion,reported drug use at the 2nd occasion,reported drug use at the 3rd occasion,reported drug use at the 4th occasion,reported drug use at the 5th occasion,frequency of the response configuration. Le categorie sono: 1 –> mai nell'anno precedente 2 –> solo 1 volta al mese nell'anno precedente 3 –> più di una volta al mese

```
data("data_drug")
d <- data_drug
head(d)
```

```
##   V1 V2 V3 V4 V5  V6
## 1  1  1  1  1  1 111
## 2  1  1  1  1  2  18
## 3  1  1  1  1  3   7
## 4  1  1  1  2  1   6
## 5  1  1  1  2  2   6
## 6  1  1  1  2  3   1
```

```
sum(d$V6)
```

```
## [1] 237
```

Ho 51 patterns possibili con frequenza non nulla. Ho 237 soggetti totali rispondenti, di cui ho 111 soggetti che non hanno consumato mai. Ho 18 soggetti che hanno consumato solo una volta al mese nell'ultimo anno. Ho 7 soggetti che hanno consumato più di una volta al mese solo nell'ultimo anno. ecc... Non ho valori mancanti, quindi ho sempre le risposte per ogni anno per ogni soggetto

Costruzione del modello di transizione

Il modello richiede la matrice dei pattern e il vettore delle frequenze. S identifica la matrice dei pattern di risposta, mentre yv identifica il vettore delle freq. S deve avere come valore iniziale delle categorie 0; la matrice che abbiamo noi inizia da 1, per cui dobbiamo apportare modifiche scalando di 1 tutti i valori presenti in tutte le righe.

```
S <- d[,1:5]-1 #tutte le righe con valori categoriali scalate di 1
S <- as.matrix(S)
yv <- d[,6]
```

Stima dei parametri tramite est_mc_basic

```
mod1 <- est_mc_basic(S,yv,mod=1)
#mod=1 quando le matrici sono omogenee nel tempo
summary(mod1)
```

```
## Call:
## est_mc_basic(S = S, yv = yv, mod = 1)
##
## Coefficients:
##
## Initial probabilities:
##   est_piv
## 0  0.9198
## 1  0.0591
## 2  0.0211
##
## Transition probabilities:
```

```
##          category
## category      0      1      2
##        0 0.8342 0.1250 0.0408
##        1 0.2846 0.4472 0.2683
##        2 0.0787 0.1573 0.7640
```

Otteniamo le prob iniziali di trovarsi in uno stato e la matrice di transizione. Lo stato più probabile è 0: mai uso di cannabis. Usando le info campionarie (cioè una sequenza temporale di risposte) ottengo le prob di transizione. #Leggo la matrice!!!!! MANCA COMMENTO

05 12 17 Lezione 6

Matrice per diverse traiettorie di un Random Walk

Costruisco una matrice XX che racchiuda 150 percorsi per diverse traiettorie del RW. Righe XX = numero di percorsi simulati (nsim) Colonne XX = numero di traiettorie che vogliamo simulare

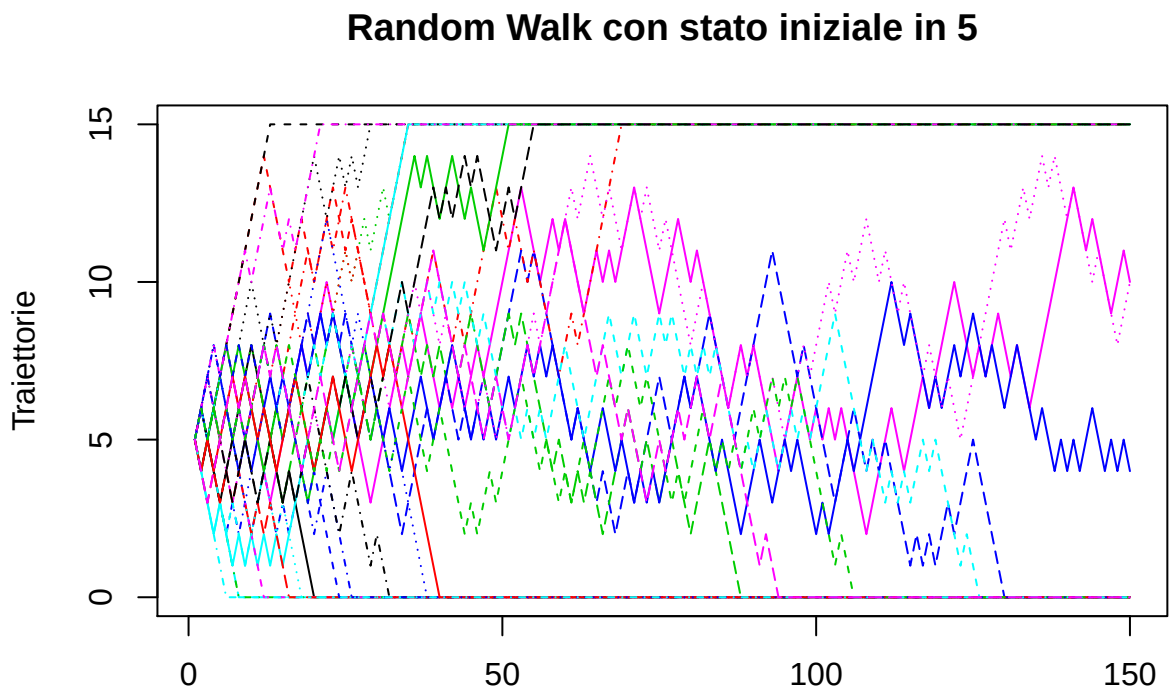
```
ini <- 0 #valore dello stato iniziale
fine <- 15  #valore dello stato finale
p <- 0.5  #prob di transizione
nsim <- 150 #numero di percorsi
x <- rep(ini,nsim)
x[1] <- 5

ttr <- 30 #traiettorie diverse che vogliamo testare

#predispongo la matrice vuota che si riempirà col for
XX <- matrix(0,nrow = nsim,ncol = ttr)
```

Creo il ciclo for, di cui uno esterno per ciascuna traiettoria che voglio simulare e che dovrà riempire la matrice XX.

```
for(j in 1:ttr){
for (i in 2:nsim){
  if(x[i-1]==ini || x[i-1]==fine)
    {
    x[i] <- x[i-1]
    }
  else
    {
    x[i] <- x[i-1]+ sample(c(1,-1),1,prob = c(p,1-p))
    }

}

 XX[,j] <- x

}
```

Visualizzo le 30 traiettorie simulate in un grafico a linee.

```
matplot(XX,type = "l", main="Random Walk con stato iniziale in 5",
        ylab = "Traiettorie")
```



Posso osservare le differenze nelle traiettorie generate e vedere le differenti velocità con cui raggiungono i due stati assorbenti.

Stato iniziale in 3 e in 10.

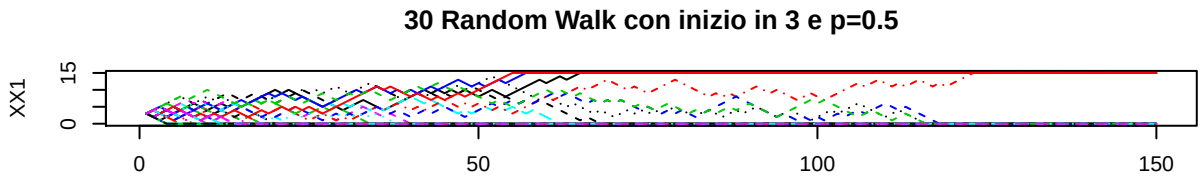
Rieseguo il RW cambiando lo stato iniziale.

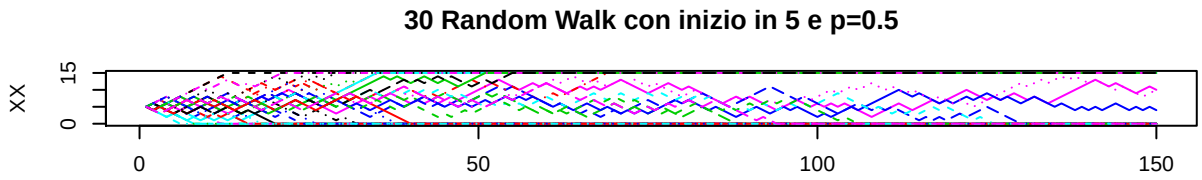
```
#Prima variazione: stato iniziale in 3
x1 <- rep(ini,nsim)
x1[1] <- 3
XX1 <- matrix(0,nrow = nsim,ncol = ttr)
for(j in 1:ttr){
for (i in 2:nsim){
  if(x1[i-1]==ini || x1[i-1]==fine)
    {
    x1[i] <- x1[i-1]
    }
  else
    {
    x1[i] <- x1[i-1]+ sample(c(1,-1),1,prob = c(p,1-p))
    }

}

 XX1[,j] <- x1
```

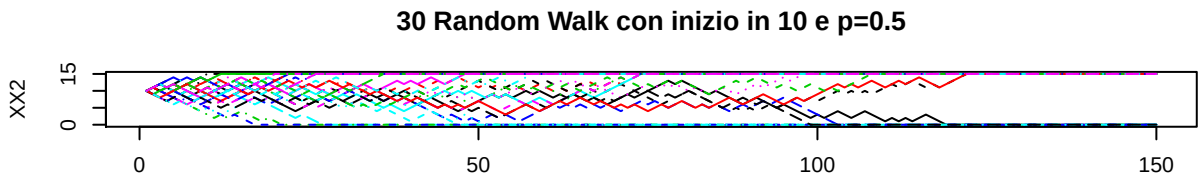
```
}

#seconda variazione: stato iniziale in 10
x2 <- rep(ini,nsim)
x2[1] <- 10
XX2 <- matrix(0,nrow = nsim,ncol = ttr)
for(j in 1:ttr){
for (i in 2:nsim){
  if(x2[i-1]==ini || x2[i-1]==fine)
    {
    x2[i] <- x2[i-1]
    }
  else
    {
    x2[i] <- x2[i-1]+ sample(c(1,-1),1,prob = c(p,1-p))
    }

}

 XX2[,j] <- x2

}
```

Grafici di confronto per gli stati iniziali differenti.

```
par(mfrow=c(3,1))
matplot(XX1,type = "l", main="30 Random Walk con inizio in 3 e p=0.5")
matplot(XX,type = "l", main= "30 Random Walk con inizio in 5 e p=0.5")
matplot(XX2,type = "l", main="30 Random Walk con inizio in 10 e p=0.5")
```







Come influisce lo stato iniziale? Se lo stato iniziale è vicino ad uno dei due stati assorbenti, la maggior parte delle traiettorie termineranno in quello stato assorbente. Incide quindi il punto di partenza, anche se la probabilità di incremento/decremento è la stessa e cioè 0.5

p=0.2

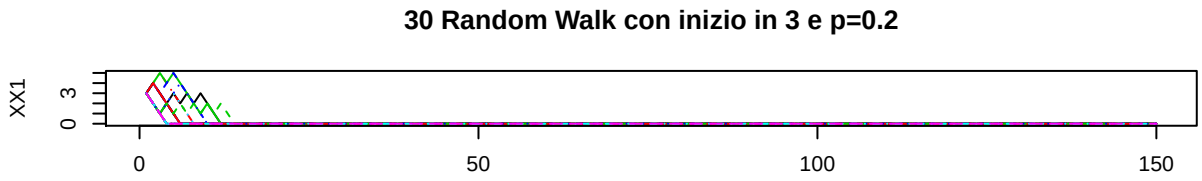
Rifaccio la stessa cosa cambiando la probabilità di incremento/decremento: p=0.2

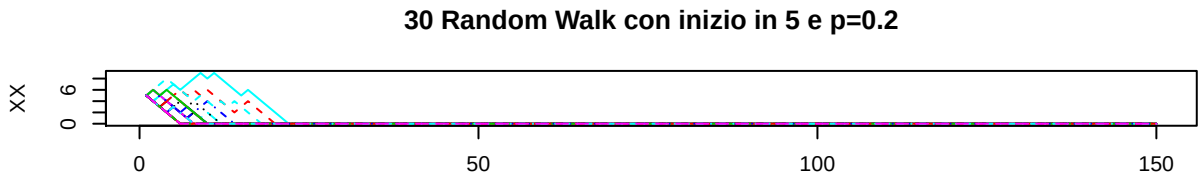
```
p1 <- 0.2

#stato iniziale 5
for(j in 1:ttr){
for (i in 2:nsim){
  if(x[i-1]==ini || x[i-1]==fine)
    {
    x[i] <- x[i-1]
    }
  else
    {
    x[i] <- x[i-1]+ sample(c(1,-1),1,prob = c(p1,1-p1))
    }
}
 XX[,j] <- x
}

#stato iniziale 3
```

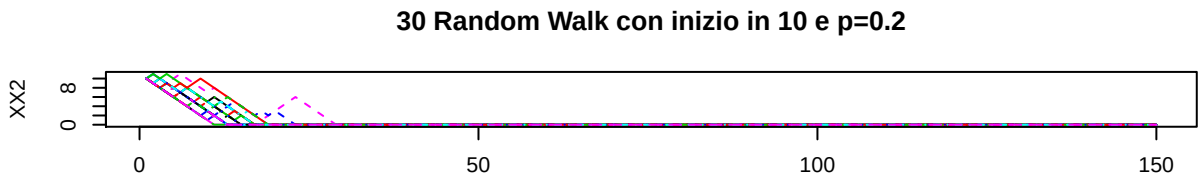
```
for(j in 1:ttr){
for (i in 2:nsim){
  if(x1[i-1]==ini || x1[i-1]==fine)
    {
    x1[i] <- x1[i-1]
    }
  else
    {
    x1[i] <- x1[i-1]+ sample(c(1,-1),1,prob = c(p1,1-p1))
    }

}
 XX1[,j] <- x1
}

#stato iniziale 10
for(j in 1:ttr){
for (i in 2:nsim){
  if(x2[i-1]==ini || x2[i-1]==fine)
    {
    x2[i] <- x2[i-1]
    }
  else
    {
    x2[i] <- x2[i-1]+ sample(c(1,-1),1,prob = c(p1,1-p1))
    }

}
   XX2[,j] <- x2
}
```

Grafici:

```
par(mfrow=c(3,1))
matplot(XX1,type = "l", main="30 Random Walk con inizio in 3 e p=0.2")
matplot(XX,type = "l", main= "30 Random Walk con inizio in 5 e p=0.2")
matplot(XX2,type = "l", main="30 Random Walk con inizio in 10 e p=0.2")
```







Nessuna traiettoria arriva allo stato assorbente 15, perchè la probabilità alla direzione +1 è molto bassa. Inoltre nessuna traiettoria non termina entro i 150 step. In media ci vogliono circa 30 step per terminare la traiettoria

p=0.8

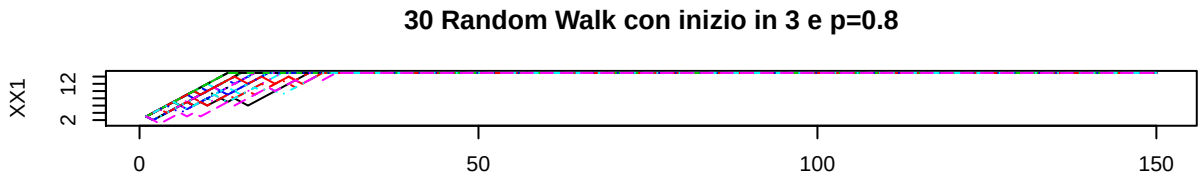
Rifaccio la stessa cosa cambiando la probabilità di incremento/decremento: p=0.8

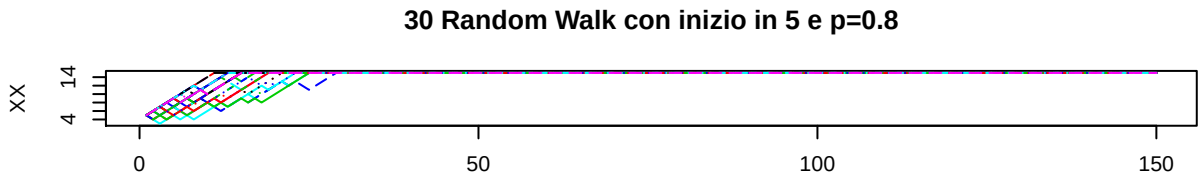
```
p2 <- 0.8

#stato iniziale 5
for(j in 1:ttr){
for (i in 2:nsim){
  if(x[i-1]==ini || x[i-1]==fine)
    {
    x[i] <- x[i-1]
    }
  else
    {
    x[i] <- x[i-1]+ sample(c(1,-1),1,prob = c(p2,1-p2))
    }
}
 XX[,j] <- x
}

#stato iniziale 3
```

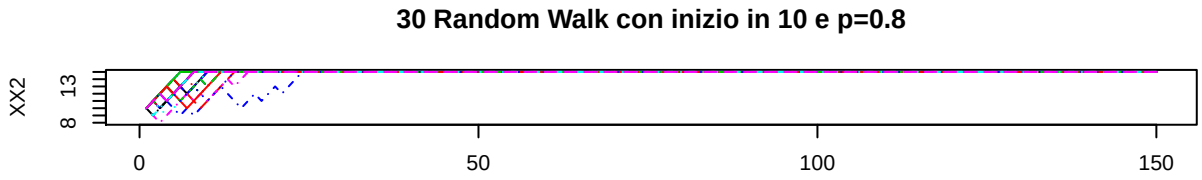
```
for(j in 1:ttr){
for (i in 2:nsim){
  if(x1[i-1]==ini || x1[i-1]==fine)
    {
    x1[i] <- x1[i-1]
    }
  else
    {
    x1[i] <- x1[i-1]+ sample(c(1,-1),1,prob = c(p2,1-p2))
    }

}
 XX1[,j] <- x1
}

#stato iniziale 10
for(j in 1:ttr){
for (i in 2:nsim){
  if(x2[i-1]==ini || x2[i-1]==fine)
    {
    x2[i] <- x2[i-1]
    }
  else
    {
    x2[i] <- x2[i-1]+ sample(c(1,-1),1,prob = c(p2,1-p2))
    }

}
   XX2[,j] <- x2
}
```

Grafici:

```
par(mfrow=c(3,1))
matplot(XX1,type = "l", main="30 Random Walk con inizio in 3 e p=0.8")
matplot(XX,type = "l", main= "30 Random Walk con inizio in 5 e p=0.8")
matplot(XX2,type = "l", main="30 Random Walk con inizio in 10 e p=0.8")
```







Simmetricamente al caso precedente, con una p=0.8 tutte le traiettorie finiscono nello stato assorbente 15. Nessuna traiettoria non termina entro i 150 step. In media anche in questo caso ci vogliono circa 30 step per terminare la passeggiata.

!!!!Costruisco una funzione per simulare facilmente la passeggiata.!!!!

Stima di un modello di transizione. Esempio datadr.

Carico i dati. La matrice delle configurazioni di risposta (datad) contiene i dati originali con i pattern per ogni individuo. Ho 1000 individui. Le colonne definiscono i tempi di osservazione: in questo caso 4 tempi. Il contesto è ancora l'uso di cannabis: 0 nessun uso nell'anno precedente 1 uso moderato nell'anno precedente 2 uso abituale nell'anno precedente.

```
load("datadr.Rdata")
```

Determinare i pattern di risposta più frequenti.

Sconf è la matrice che contiene tutti i pattern di risposta differenti trovati nei 1000. Sono 56 configurazioni differenti. Il vettore yvconf contiene le frequenze per le 56 differenti configurazioni.

```
XX <- cbind(Sconf,yvconf)
yvconf
```

```
##  [1] 50 86 39 46 29 47 93 13 31 27  2  5 82 24  6 15  5 18  9 31  1 26  1
```

```
## [24] 27 62  9 12  9 14 11 17 24 38 19  6  4  2  5  3  3  4  5  6  2  1  2
## [47]  5  9  1  3  1  4  2  2  1  1
```

Estraggo la posizione e in seguito la sequenza più frequente

```
pmax <- which(yvconf==93)  #7
Sconf[pmax,]
```

```
## [1] 0 2 2 2
```

La maggior parte dei rispondenti,93 soggetti, ha fatto uso abituale negli ultimi 3 periodi

Valutare le frequenze di risposta in ogni tempo sui dati campionari (datad).

```
p1 <- datad[,1]
p2 <- datad[,2]
p3 <- datad[,3]
p4 <- datad[,4]
table(p1)
```

```
## p1
##   0   1   2
## 511 300 189
```

```
table(p2)
```

```
## p2
##   0   1   2
## 247  69 684
```

```
table(p3)
```

```
## p3
##   0   1   2
## 275  65 660
```

```
table(p4)
```

```
## p4
##   0   1   2
## 233 295 472
```

Quindi ho così immagine delle frequenze osservate ai diversi tempi.

In alternativa posso usare la funzione apply sulle colonne.

```
apply(datad,2,FUN =table)
```

```
##   [,1] [,2] [,3] [,4]
## 0  511  247  275  233
## 1  300   69   65  295
## 2  189  684  660  472
```

Cercare il pattern 0 1 2 2

```r
#Subsetting both rows and columns of a dataframe
XX <- as.data.frame(XX)
subset(XX, V1==0 & V2==1 & V3==2 & V4==2, select = c(V1: yvconf))
```

```
##    V1 V2 V3 V4 yvconf
## 15  0  1  2  2      6
```

Cercare il pattern 0 0 0 0

```r
subset(XX, V1==0 & V2==0 & V3==0 & V4==0, select = c(V1: yvconf))
```

```
##    V1 V2 V3 V4 yvconf
## 51  0  0  0  0      1
```

C'è solo un soggetto, il 51, che rimane non consumatore.

Trovare tutti i pattern in cui c'è solo un soggetto

```r
subset(XX, yvconf==1, select = c(V1: V4))
```

```
##    V1 V2 V3 V4
## 21  2  2  1  1
## 23  2  0  0  0
## 45  1  0  0  0
## 49  0  1  0  2
## 51  0  0  0  0
## 55  0  1  1  2
## 56  0  0  1  2
```

Stima dei parametri di input del modello di transizione

La tipologia di transizione è omogenea rispetto al tempo(mod=1). L'omogeneità nel tempo permette di stimare il modello più parsimonioso. Il numero di parametri è pari a 8: 2 (c-1) prob iniziali (l'ultima è per complemento a 1) e le 6 (c*c-1) prob di transizione. La matrice ci mostra come ci si muove tra le categorie ai vari tempi. Nella formula dei parametri non mettiamo T-1 perchè abbiamo stimato con tempo omogeneo.

```r
require(LMest)
mod <- est_mc_basic(S = Sconf, mod = 1,yv = yvconf)
summary(mod)
```

```
## Call:
## est_mc_basic(S = Sconf, yv = yvconf, mod = 1)
##
## Coefficients:
##
## Initial probabilities:
##   est_piv
## 0   0.511
## 1   0.300
## 2   0.189
##
## Transition probabilities:
```

```
##          category
## category      0      1      2
##        0 0.1123 0.1152 0.7725
##        1 0.2834 0.1429 0.5737
##        2 0.3366 0.1618 0.5016
```

Lo stato più persistente è relativo a quello di chi fa uso abituale, infatti il 50% di coloro che partono in categoria 2 rimane in categoria 2 (0.5016). Per coloro che partono da un uso sporadico, è più probabile che finiscano ad essere consumatori usuali (0.5737). Soprattutto chi non fa uso, ha il 77.25% di probabilità di finire a fare uso quotidiano. Tra chi non fa uso e chi fa uso sporadico, la prob di rimanere in quella categoria durante tutto il periodo studiato è molto bassa e si aggira attorno al 11 e 14%. L eprobabilità nella diagonale sono le probabilityà di persistenza, cioè di rimanere nello stesso stato per tutto il periodo. Se ipotizziamo che il tempo non sia omogeneo, cioè che a un certo punto è ipotizzabile un cambiamento significativo, è meglio stimare un modello per tutti i tempi per il quale si avranno più parametri da stimare.

ottenere le prob stimate congiunte per Y

Estraggo la probabilità congiunta della variabile risposta per ogni categoria su ogni tempo, ottenuta dalle prob iniziali stimate * prob di transizione. Si calcola in base ai parametri stimati dal modello.

```
mod$Fy
```

```
##     category
## time         0         1         2
##    1 0.5110000 0.3000000 0.1890000
##    2 0.2060219 0.1322989 0.6616792
##    3 0.2833478 0.1496759 0.5669763
##    4 0.2650793 0.1457457 0.5891750
```

La prima riga identifica le prob iniziali di ogni stato. Questa tabella va letta in verticale, per colonna. Leggendo la prima colonna, dimostra che la progressione temporale sulla categoria 1 è decrescente e il salto principale viene fatto tra il primo e secondo tempo. Dal 2 al 4 tempo in tutte le categoria non si notano grosse differenze perchè c'è una tendenza leggera a decrescere. Il passaggio principale si ha per tutte le categoria dal primo al secondo tempo.

06 12 17 lez 7

Ancora esercizi sul dataset datadrug

Lo richiamo

```
data("data_drug")
yv <- data_drug[,6]
S <- data_drug[,1:5]-1 #tutte le righe con valori categoriali scalate di 1
S1 <- data.frame(S,yv)
```

Cercare il pattern 0 1 2 2 2 e la relativa frequenza

```
subset(S1, V1==0 & V2==1 & V3==2 & V4==2 & V5==2, select = c(V1: yv))
```

```
##    V1 V2 V3 V4 V5 yv
## 32  0  1  2  2  2  5
```

5 persone hanno questo specifico pattern

Cercare i pattern con solo 1 frequenza

```
subset(S1, yv==1, select = c(V1: V5))
```

```
##    V1 V2 V3 V4 V5
## 6   0  0  0  1  2
## 8   0  0  0  2  1
## 12  0  0  1  0  2
## 17  0  0  2  0  1
## 18  0  0  2  1  1
## 20  0  1  0  0  0
## 22  0  1  0  1  0
## 24  0  1  1  0  1
## 25  0  1  1  0  2
## 28  0  1  2  0  1
## 29  0  1  2  1  1
## 30  0  1  2  1  2
## 33  0  2  0  2  2
## 34  0  2  1  1  1
## 38  1  0  0  2  2
## 39  1  0  1  1  1
## 40  1  0  2  2  2
## 41  1  1  1  1  1
## 42  1  1  2  2  2
## 43  1  2  1  0  0
## 44  1  2  1  2  2
## 45  1  2  2  2  1
## 47  2  0  0  0  0
## 48  2  1  2  2  2
## 49  2  2  2  1  2
## 50  2  2  2  2  0
## 51  2  2  2  2  2
```

Ci sono 27 pattern con frequenza pari a 1

Cercare persone che non hanno mai usato cannabis

```
subset(S1, V1==0 & V2==0 & V3==0 & V4==0 & V5==0, select = c(V1: yv))
```

```
##   V1 V2 V3 V4 V5  yv
## 1  0  0  0  0  0 111
```

111 persone non fanno mai uso di cannabis durante il periodo

Ottenere le probabilità stimate congiunte per la risposta Y.

```
S2 <- as.matrix(S)
mod1 <- est_mc_basic(S2,yv,mod=1)
mod1$Fy
```

```
##     category
## time         0         1         2
##    1 0.9198312 0.05907173 0.02109705
##    2 0.7858275 0.14471173 0.06946073
##    3 0.7022094 0.17386335 0.12392723
##    4 0.6450310 0.18501412 0.16995490
##    5 0.6041236 0.19009324 0.20578314
```

La prima riga identifica le probabilità iniziali. Passando alla lettura per colonna, abbiamo che per la categoria 0 si ha un decremento pressochè costante, man mano che le persone non usano cannabis diminuiscono nei 5 anni della survey. Nella categoria 1 c'è un salto al primo periodo e poi continua con un leggero incremento costante. Per la categoria 2 si ha un andamento caratterizzato da salti di incremento più marcati su tutti i periodi.

Estraggo la matrice delle probabilità di transizione per ogni tempo. Attenzione che se ho messo il modello a tempi omogenei, avrò matrici uguali per tutti i tempi. Se avevo mod=0 (cioè a tempo non omogeneo) avrei avuto matrici diverse per ogni tempo, visto che ipotizzo che ci sia evoluzione temporale.

```
mod1$Pi
```

```
## , , time = 1
##
##          category
## category 0 1 2
##        0 0 0 0
##        1 0 0 0
##        2 0 0 0
##
## , , time = 2
##
##          category
## category          0         1          2
##        0 0.83423913 0.1250000 0.04076087
##        1 0.28455285 0.4471545 0.26829268
##        2 0.07865169 0.1573034 0.76404494
##
## , , time = 3
##
##          category
## category          0         1          2
##        0 0.83423913 0.1250000 0.04076087
##        1 0.28455285 0.4471545 0.26829268
##        2 0.07865169 0.1573034 0.76404494
##
## , , time = 4
##
##          category
## category          0         1          2
##        0 0.83423913 0.1250000 0.04076087
##        1 0.28455285 0.4471545 0.26829268
##        2 0.07865169 0.1573034 0.76404494
##
## , , time = 5
##
##          category
## category          0         1          2
```

```
##         0 0.83423913 0.1250000 0.04076087
##         1 0.28455285 0.4471545 0.26829268
##         2 0.07865169 0.1573034 0.76404494
```

Modello con tempo non omogeneo (mod=0)

```
mod2 <- est_mc_basic(S2,yv,mod=0)
mod1
```

```
## Call:
## est_mc_basic(S = S2, yv = yv, mod = 1)
##
## Convergence info:
##          LogLik np      AIC      BIC
## [1,] -669.3717  8 1354.743 1382.488
```

```
mod2
```

```
## Call:
## est_mc_basic(S = S2, yv = yv, mod = 0)
##
## Convergence info:
##          LogLik np      AIC      BIC
## [1,] -655.1622 26 1362.324 1452.494
```

confrontando i due modelli in base al BIC, scelgo quello con Bic minore. Risulta migliore quello omogeneo con solo 8 parametri, rispetto a quello con 26 parametri.

11 12 17 lez 8

Modello latente di Markov: esempio soddisfazione RLMS

Carico i dati. Longitudinal dataset deriving from the Russia Longitudinal Monitoring Survey (RLMS) about job satisfaction measured by an ordinal variable with 5 categories. Le 7 colonne riportano job satisfaction at seven occasions. Soddisfazione sul primo lavoro per 1718 soggetti. Livelli di soddisfazione: 1 completamente soddisfatto, 5 nessuna soddisfazione. Periodo dal 2008 al 2014. 7 tempi.

```
library(LMest)
data("RLMSdat")
```

Ottenere le frequenze per tutti i possibili pattern osservati

La funzione aggr_data genera la matrice con le frequenze delle configurazioni di risposta.

```
out <- aggr_data(RLMSdat)
yv <- out$freq  #vettore delle frequenze osservate
S <- out$data_dis #matrice delle conf di risposta
m <- cbind(out$data_dis, yv)
```

Cerco il massimo e la configurazione corrispondente

```
max(yv)
```

```
## [1] 168
```

Il pattern più frequente è composto da 168 idnividui

Trasformo la matrice m in dataframe ed eseguo il subset per ottenere i risultati.

```
mdf <- as.data.frame(m)
subset(mdf,yv=="168")
```

```
##    IKSJQ IKSJR IKSJS IKSJT IKSJU IKSJV IKSJW  yv
## 23     2     2     2     2     2     2     2 168
```

La configurazione maggiore è quella che vede 168 soggetti che risultano su tutto il periodo considerato abbastanza soddisfatti.

Cerco quanti soggetti hanno sempre voto 4 in tutti i tempi

```
subset(mdf,IKSJQ==4 & IKSJR==4 & IKSJR==4 & IKSJT==4 & IKSJU==4 & IKSJV==4 & IKSJW==4)
```

```
##     IKSJQ IKSJR IKSJS IKSJT IKSJU IKSJV IKSJW yv
## 680     4     4     4     4     4     4     4  1
```

Solo un soggetto ha questa configurazione, cioè è abbastanza insoddisfatto per tutto il periodo considerato.

Cerco soggetti con pattern misto decrescente da 5 (insoddisfatto) a 1 (soddisfatto)

Impostando solo i valori iniziali e finali desiderati, trovo tutto quello che può rientrare in questa casistica.

```
subset(mdf,IKSJQ==5 & IKSJW==1)
```

```
##     IKSJQ IKSJR IKSJS IKSJT IKSJU IKSJV IKSJW yv
## 381     5     2     5     5     4     2     1  1
## 732     5     4     4     1     2     3     1  1
## 980     5     3     3     2     2     2     1  1
```

Trovo solo 3 pattern, ciascuno di frequenza 1, che partono da 5 e arrivano a 1

Stima del modello latente

Fisso k, numero di stati latenti. Start è il tipo di inizializzazione dell'algoritmo EM: con start = 0 li fisso in modo deterministico. NB ricordiamoci di scalare S, in modo che parta da 0. S1 = 5-S ricategorizza le categorie: parte da 0 fino a 4

```
S1 <- 5-S
k=3 #stati del processo latente
mod <- est_lm_basic(S = S1, yv = yv, k=k,
                    mod=1,
                    start = 0)
```

```
## ------------|-------------|-------------|-------------|-------------|-------------|-------------|
##      mod    |      k      |    start    |     step    |     lk      |    lk-lko   | discrepancy |
## ------------|-------------|-------------|-------------|-------------|-------------|-------------|
```

```
##           1 |           3 |           0 |           0 |    -15606.8 |
##           1 |           3 |           0 |          10 |    -13577.1 |     4.87573 |  0.00765358 |
##           1 |           3 |           0 |          20 |    -13559.7 |    0.533973 |  0.00266113 |
##           1 |           3 |           0 |          30 |    -13557.6 |    0.073773 | 0.000832573 |
##           1 |           3 |           0 |          40 |    -13557.3 |   0.0144436 | 0.000366457 |
##           1 |           3 |           0 |          50 |    -13557.2 |  0.00374981 | 0.000198311 |
##           1 |           3 |           0 |          60 |    -13557.2 |  0.00116625 | 0.000106657 |
##           1 |           3 |           0 |          70 |    -13557.2 | 0.000413452 |  5.7188e-05 |
##           1 |           3 |           0 |          80 |    -13557.2 | 0.000164074 | 3.06113e-05 |
##           1 |           3 |           0 |          83 |    -13557.2 | 0.000126816 | 2.53723e-05 |
## ------------|-------------|-------------|-------------|-------------|-------------|-------------|
```

```
summary(mod)
```

```
## Call:
## est_lm_basic(S = S1, yv = yv, k = k, start = 0, mod = 1)
##
## Coefficients:
##
## Initial probabilities:
##      est_piv
## [1,]  0.3638
## [2,]  0.4623
## [3,]  0.1739
##
## Transition probabilities:
##      state
## state      1      2      3
##     1 0.8863 0.0991 0.0145
##     2 0.0472 0.9335 0.0193
##     3 0.0469 0.0774 0.8757
##
## Conditional response probabilities:
## , , item = 1
##
##         state
## category      1      2      3
##        0 0.0731 0.0000 0.0031
##        1 0.2442 0.0224 0.0233
##        2 0.4421 0.0965 0.0448
##        3 0.2197 0.8166 0.3480
##        4 0.0209 0.0645 0.5808
```

Output: step dà il numero di passi in salti di 10, lk è la logver, lk-lk0 è la differenza tra valori consecutivi di lk ogni 10 passi, disrepancy mi dà la quantità relativa di lk da confrontare con epsilon (10^-8). L'algoritmo converge con 83 Passi.

Lettura dei risultati

Il summary dà gli stessi risultati che potremmo ottenere con le seguenti istruzioni. Esse sono utili se lette in questo ordine:

```
mod$Psi #modello di misura --> caratterizzazione dei gruppi
```

```
## , , item = 1
```

```
##
##         state
## category          1            2           3
##        0 0.07311240 3.506373e-06 0.003075002
##        1 0.24424077 2.241661e-02 0.023329937
##        2 0.44205360 9.651902e-02 0.044762193
##        3 0.21968519 8.166013e-01 0.348021503
##        4 0.02090804 6.445956e-02 0.580811365
```

```
mod$piv #prob iniziali del modello latente
```

```
## [1] 0.3638439 0.4622935 0.1738625
```

```
mod$Pi #prob di transizione del modello latente
```

```
## , , time = 1
##
##       state
## state 1 2 3
##     1 0 0 0
##     2 0 0 0
##     3 0 0 0
##
## , , time = 2
##
##       state
## state          1          2          3
##     1 0.88633122 0.09914876 0.01452002
##     2 0.04723110 0.93351507 0.01925382
##     3 0.04687675 0.07737856 0.87574469
##
## , , time = 3
##
##       state
## state          1          2          3
##     1 0.88633122 0.09914876 0.01452002
##     2 0.04723110 0.93351507 0.01925382
##     3 0.04687675 0.07737856 0.87574469
##
## , , time = 4
##
##       state
## state          1          2          3
##     1 0.88633122 0.09914876 0.01452002
##     2 0.04723110 0.93351507 0.01925382
##     3 0.04687675 0.07737856 0.87574469
##
## , , time = 5
##
##       state
## state          1          2          3
##     1 0.88633122 0.09914876 0.01452002
##     2 0.04723110 0.93351507 0.01925382
##     3 0.04687675 0.07737856 0.87574469
##
```

```
## , , time = 6
##
##      state
## state          1          2          3
##     1 0.88633122 0.09914876 0.01452002
##     2 0.04723110 0.93351507 0.01925382
##     3 0.04687675 0.07737856 0.87574469
##
## , , time = 7
##
##      state
## state          1          2          3
##     1 0.88633122 0.09914876 0.01452002
##     2 0.04723110 0.93351507 0.01925382
##     3 0.04687675 0.07737856 0.87574469
```

Nella tabella del modello di misura caratterizzo i gruppi del processo latente sottostante –> leggo la matrice in verticale: ho 3 gruppi (3 stati latenti) –> ho creato dei gruppi omogenei in base alla variabile risposta. Il primo gruppo raccoglie in generale quelli con soddisfazione prevalentemente 2. Il secondo gruppo raccoglie quelli prevalentemente soddisfatti 3. Il terzo gruppo quelli soddisfatti 3 e 4. In secondo luogo guardo le prob iniziali di ogni gruppo. Infine guardo le prob di transizione, definite sugli stati latenti.

Cerco i valori dei criteri di informazione.

```
print(mod)
```

```
## Call:
## est_lm_basic(S = S1, yv = yv, k = k, start = 0, mod = 1)
##
## Convergence info:
##         LogLik np      AIC      BIC
## [1,] -13557.21 20 27154.41 27263.39
```

Stima del modello con k=2

```
k1=2 #stati del processo latente
mod1 <- est_lm_basic(S = S1, yv = yv, k=k1,
                     mod=1,
                     start = 0)
```

```
## ------------|-------------|-------------|-------------|-------------|-------------|-------------|
##      mod    |      k      |    start    |     step    |     lk      |    lk-lko   | discrepancy |
## ------------|-------------|-------------|-------------|-------------|-------------|-------------|
##           1 |           2 |           0 |           0 |    -16066.6 |             |             |
##           1 |           2 |           0 |          10 |    -13922.7 |    0.426408 |  0.00269075 |
##           1 |           2 |           0 |          20 |    -13921.2 |   0.0392978 | 0.000840968 |
##           1 |           2 |           0 |          30 |    -13921.1 |  0.00346979 | 0.000250364 |
##           1 |           2 |           0 |          40 |    -13921.1 | 0.000297542 | 7.34083e-05 |
##           1 |           2 |           0 |          44 |    -13921.1 | 0.000111064 | 4.48694e-05 |
## ------------|-------------|-------------|-------------|-------------|-------------|-------------|
```

```
summary(mod1)
```

```
## Call:
## est_lm_basic(S = S1, yv = yv, k = k1, start = 0, mod = 1)
##
```

```
## Coefficients:
##
## Initial probabilities:
##      est_piv
## [1,]  0.3974
## [2,]  0.6026
##
## Transition probabilities:
##      state
## state      1      2
##     1 0.9004 0.0996
##     2 0.0528 0.9472
##
## Conditional response probabilities:
## , , item = 1
##
##         state
## category      1      2
##        0 0.0649 0.0009
##        1 0.2252 0.0194
##        2 0.4273 0.0693
##        3 0.2671 0.7096
##        4 0.0155 0.2008
```

L'algoritmo è abbastanza veloce: 44 passi.

Confronto i due modelli in base al BIC, con 3 e 2 stati latenti.

```
print(mod)
```

```
## Call:
## est_lm_basic(S = S1, yv = yv, k = k, start = 0, mod = 1)
##
## Convergence info:
##         LogLik np      AIC      BIC
## [1,] -13557.21 20 27154.41 27263.39
```

```
print(mod1)
```

```
## Call:
## est_lm_basic(S = S1, yv = yv, k = k1, start = 0, mod = 1)
##
## Convergence info:
##         LogLik np      AIC      BIC
## [1,] -13921.09 11 27864.18 27924.12
```

Con 3 stati latenti stimo 20 parametri, mentre con 2 stati latenti ne stimo 11. Ma secondo il BIC è migliore quello con 3 categorie. Evidentemente, nonostante vi siano più parametri da stimare, l'aggiunta di uno stato latente è vantaggioso.

Estraggo le probabilità condizionate a posteriori della risposta | latente.

Cerco la probabilità a posteriori di appartenere ad ogni stato latente per la configurazione di risposta 1 e al primo tempo di osservazione. Facendo così riesco a fare ragionamenti sul processo latente sottostante.

```
V <- mod$V
V[1,1:3,1]
```

```
## [1] 0.03020969 0.82738341 0.14240689
```

L'oggetto V mi dà la distrib a posteriori per ogni osservazione delle probabilità di appartenere alle diverse classi latenti e nei diversi tempi osservati. Il primo soggetto osservato ha prob 0.8 di appartenere al 2 gruppo al tempo 1.

08 01 18 lez 14

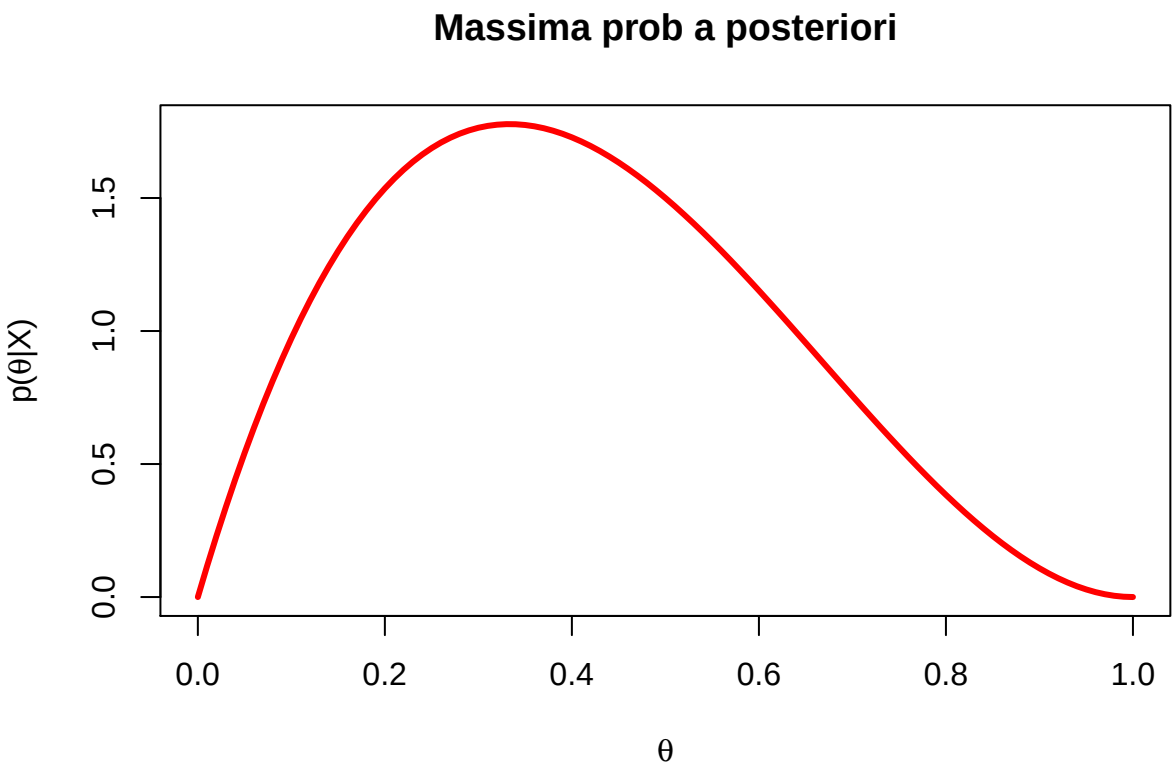
Regione di massima probabilità a posteriori: HPD

A partire da una Beta Binomiale, Uso di metodi numerici

```
densbeta <- function(x,a,b){   #a è alpha    b è beta
  x^(a-1)*(1-x)^(b-1)/beta(a,b)   #funzione beta già definita in R, integra a 1
}
```

Disegno la distribuzione a posteriori ottenuta: P(theta|x)

```
curve(densbeta(x, a=2,b=3),
      0,1,  #definisco l'intervallo
      col="red",
      lwd=3,
      ylab=expression(paste(p,"(", theta, "|X)")),
      xlab=expression(theta),
      main="Massima prob a posteriori"
      )
```



Intervallo di confidenza sulla posteriori

Intervallo calcolato tramite il Metodo del percentile con alpha=0.05

```
CI <- qbeta(c(0.025,0.975),2,3)
CI
```

```
## [1] 0.06758599 0.80587955
```

Questo ottenuto non è intervallo HPD. è un intervallo di confidenza dato rispetto ai quantili di un alpha scelto. Gli estremi dell'IC sono 0.07 e 0.8. La densità compresa nei due estremi è tutta quella area sotto la curva, che è di 0.95 per costruzione.

Calcolo HPD

Calcolo HPD partendo da k=0.5

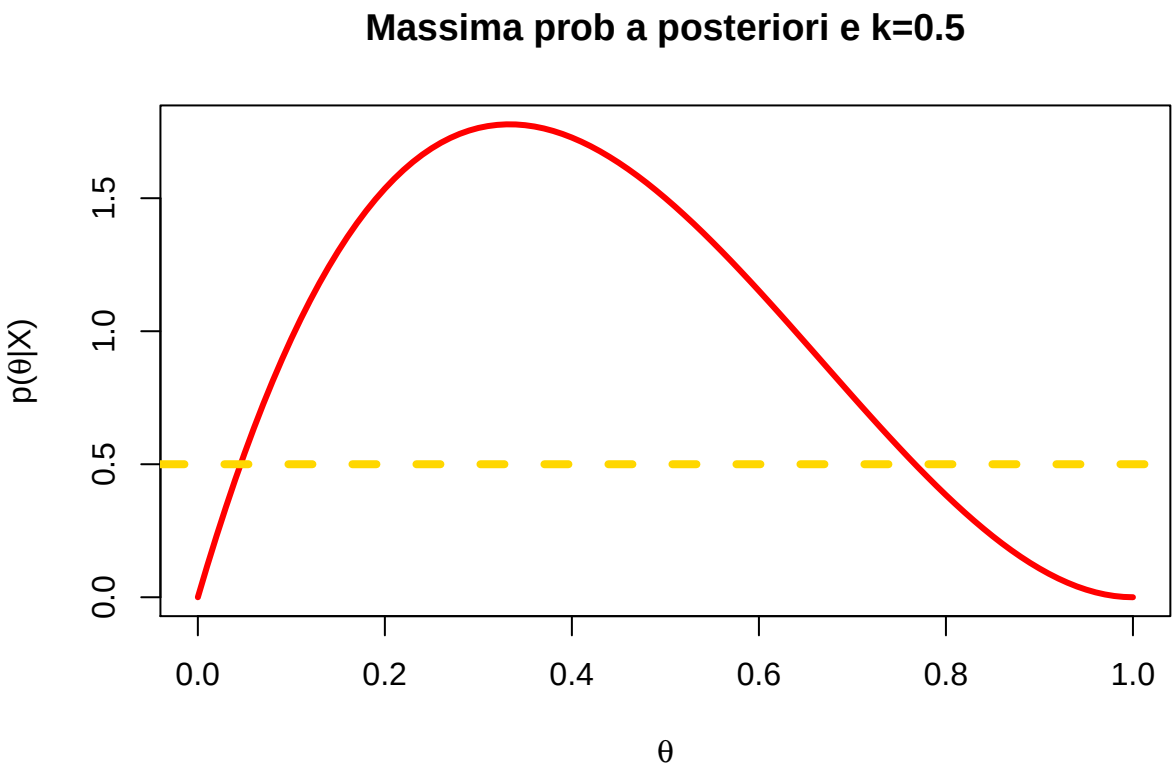
```
#cerco la densità per valori noti dei parametri della funzione
densbeta1 <- function(x){
  x^(2-1)*(1-x)^(3-1)/beta(2,3)
}

#cerco il punto di massimo della densità numericamente
optimise(densbeta1,
         c(0,1), #intervallo in cui cercare il punto di massimo
         maximum = T)
```

```
## $maximum
## [1] 0.3333205
##
## $objective
## [1] 1.777778
```

objective è il corrispondente valore del massimo sull'asse Y; maximum è il valore di theta che massimizza la funzione Beta(2,3), cioè è il valore del parametro con probabilità più alta.

Aggiungo nel grafico il valore k; disegno la funzione e ci aggiungo K come linea orizzontale

```
curve(densbeta(x, a=2,b=3),
      0,1,  #definisco l'intervallo
      col="red",
      lwd=3,
      ylab=expression(paste(p,"(", theta, "|X)")),
      xlab=expression(theta),
      main="Massima prob a posteriori e k=0.5"
      )
abline(h=0.5,lty=2, col="gold",lwd=4)   #aggiungo linea orizzontale
```



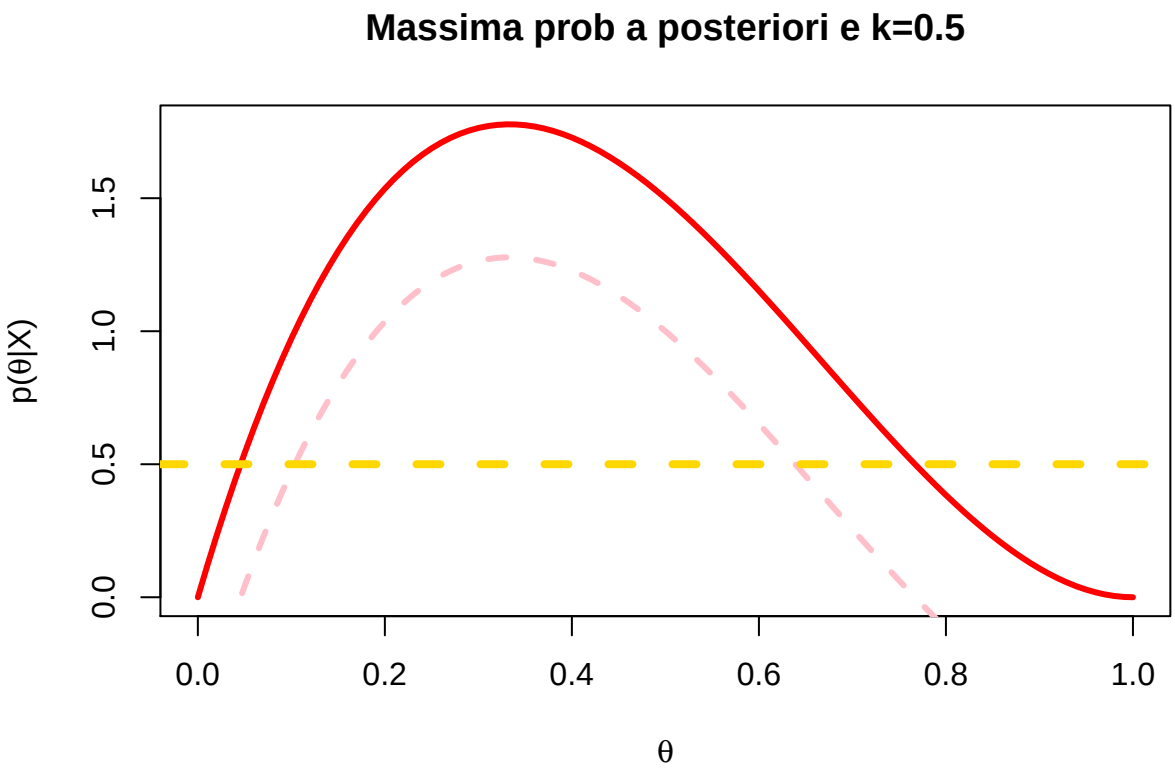
Traslazione rispetto a k

Prima devo ottenere la funzione traslata rispetto ai valori di k. Essa sarà la curva della regione massimizzata. K è un valore che rappresenta un'asticella (una soglia) per stringere l'area sotto la curva a nostro piacimento. Se vogliamo incrementare l'area, k scende. Se vogliamo diminuire l'area k va alzato.

```
betat <- function(x,a,b){
  densbeta(x,a,b) - 0.5
}
```

Disegno entrambe sullo stesso grafico.

```
curve(densbeta(x, a=2,b=3),
      0,1,  #definisco l'intervallo
      col="red",
      lwd=3,
      ylab=expression(paste(p,"(", theta, "|X)")),
      xlab=expression(theta),
      main="Massima prob a posteriori e k=0.5")
curve(betat(x, a=2,b=3),
      col="pink",
      lwd=3,
      lty=2,
      add=T)
abline(h=0.5,lty=2, col="gold",lwd=4)
```



Posso osservare la regione che massimizza la densità a posteriori (cioè la regione di credibilità, più probabile per i valori di theta in base alla densità a posteriori), che è l'area sottesa alla funzione traslata (la quale incorpora il valore di k). Gli estremi per l'intervallo di theta massimizzato sono i punti che si riflettono sull'asse X che riflettono l'intersezione tra la curva della traslata e la retta in k=0.5.

Calcolo gli estremi della regione di credibilità per il parametro theta

Li trovo facendo le radici della densità a posteriori traslata

```
hpd1 <- uniroot(betat,c(0.0001,0.1),a=2,b=3) #estremo inferiore
hpd2 <- uniroot(betat,c(0.07,0.9),a=2,b=3)   #estremo superiore
h1 <- hpd1$root;h1
```

```
## [1] 0.04576026
```

```
h2 <- hpd2$root;h2
```

```
## [1] 0.7669256
```

Calcolo ampiezza dell'area tra gli estremi

Vogliamo alpha=0.95

```
integrate(densbeta,lower=h1,upper = h2,a=2,b=3)
```

```
## 0.9463967 with absolute error < 1.1e-14
```

L'area non è esattamente 0.95.é leggermente inferiore alla probabilità che associamo come massima. Bisognerebbe trovare il preciso K che dà come risultato esattamente 0.95

Cerco K ottimale per ottenere area di credibilità pari a 0.95

Creo una sequenza di tutti i possibili k ed esploro per trovare l'ottimale, tramite ciclo for.

```
k <- seq(0.45,0.5,0.001)
risultati <- matrix(NA, ncol = 4, nrow = length(k))

for(i in 1:length(k)){
  traslata <- function(x,a,b)densbeta(x,a,b) - k[i]
  hpd1 <- uniroot(traslata,c(0.0001,0.1),a=2,b=3)
  h1 <-  hpd1$root
  hpd2 <- uniroot(traslata,c(0.7,0.9),a=2,b=3)
  h2 <- hpd2$root
  int <- integrate(traslata,
                   lower=h1,
                   upper=h2,
                   a=2,b=3)
  int1 <- int$value
  fine <- i
  if(int1 <= 0.95) break
  risultati [i,] <- c(h1,h2,int1,k[i])

}
fine
```

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

```
risultati
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]   NA   NA   NA   NA
##  [2,]   NA   NA   NA   NA
##  [3,]   NA   NA   NA   NA
##  [4,]   NA   NA   NA   NA
##  [5,]   NA   NA   NA   NA
##  [6,]   NA   NA   NA   NA
##  [7,]   NA   NA   NA   NA
##  [8,]   NA   NA   NA   NA
##  [9,]   NA   NA   NA   NA
## [10,]   NA   NA   NA   NA
## [11,]   NA   NA   NA   NA
## [12,]   NA   NA   NA   NA
## [13,]   NA   NA   NA   NA
## [14,]   NA   NA   NA   NA
## [15,]   NA   NA   NA   NA
## [16,]   NA   NA   NA   NA
## [17,]   NA   NA   NA   NA
## [18,]   NA   NA   NA   NA
## [19,]   NA   NA   NA   NA
## [20,]   NA   NA   NA   NA
## [21,]   NA   NA   NA   NA
## [22,]   NA   NA   NA   NA
```

```
## [23,]   NA   NA   NA   NA
## [24,]   NA   NA   NA   NA
## [25,]   NA   NA   NA   NA
## [26,]   NA   NA   NA   NA
## [27,]   NA   NA   NA   NA
## [28,]   NA   NA   NA   NA
## [29,]   NA   NA   NA   NA
## [30,]   NA   NA   NA   NA
## [31,]   NA   NA   NA   NA
## [32,]   NA   NA   NA   NA
## [33,]   NA   NA   NA   NA
## [34,]   NA   NA   NA   NA
## [35,]   NA   NA   NA   NA
## [36,]   NA   NA   NA   NA
## [37,]   NA   NA   NA   NA
## [38,]   NA   NA   NA   NA
## [39,]   NA   NA   NA   NA
## [40,]   NA   NA   NA   NA
## [41,]   NA   NA   NA   NA
## [42,]   NA   NA   NA   NA
## [43,]   NA   NA   NA   NA
## [44,]   NA   NA   NA   NA
## [45,]   NA   NA   NA   NA
## [46,]   NA   NA   NA   NA
## [47,]   NA   NA   NA   NA
## [48,]   NA   NA   NA   NA
## [49,]   NA   NA   NA   NA
## [50,]   NA   NA   NA   NA
## [51,]   NA   NA   NA   NA
```

Dovrei testare 31 valori di k per raggiungere 0.95

Trovo HPD sull'esempio del 22 novembre: sonno degli studenti

Parametri e k=0.5

```
a <- (3.26+11);a
```

```
## [1] 14.26
```

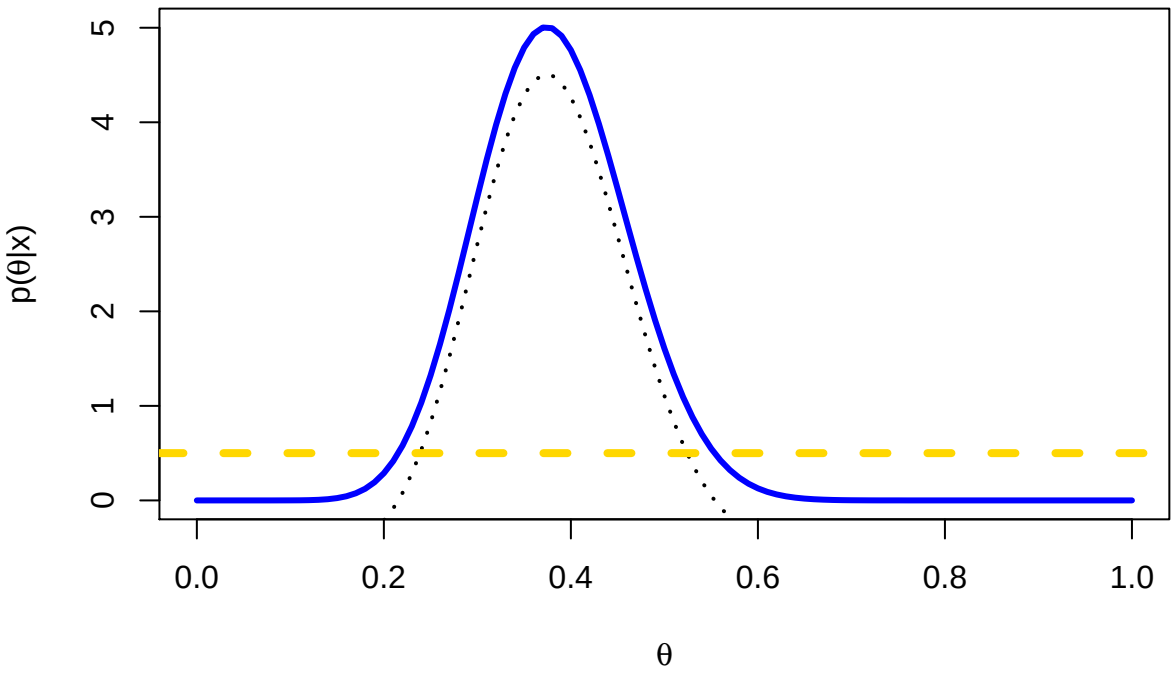
```
b <- (7.19+16);b
```

```
## [1] 23.19
```

I parametri della distrib a posteriori sono a=14.26 e b=23.19

Distribuzione a posteriori e traslata

```
densbeta <- function(x,a,b){x^(a-1)*(1-x)^(b-1)/beta(a,b)}
betat <- function(x,a,b) densbeta(x,a,b)-0.5
curve(densbeta(x,a= 14.26,b = 23.19),0,1,
      ylab=expression(paste(p,"(",theta,"|x)")),
      xlab=expression(theta),
      col ="blue",
      lwd = 3 )
curve(betat(x ,a=14.26,b=23.19),add=T,lty=3,lwd=2)
abline(h=0.5,lty=2, col="gold",lwd=4)
```



IC

```
ci <- qbeta(c(0.025,0.975),14.26 ,23.19); ci
```

```
## [1] 0.2343206 0.5392949
```

Estremi

```
hpd1 <- uniroot(betat,c(0.195,0.23),a=14.26,b=23.19);
hpd2 <-  uniroot(betat,c(0.45,0.6),a=14.26,b=23.19)

h1 <- hpd1$root; h1
```

```
## [1] 0.2154165
```

```
h2 <- hpd2$root; h2
```

```
## [1] 0.5536319
```

Area

```
integrate(densbeta,lower=h1,upper=h2,a=14.26,b=23.19)
```

```
## 0.9715052 with absolute error < 1.1e-12
```

K ottimale per area pari a 0.95. Devo alzare k per diminuire l'area tra la curva traslata e la retta. Da 0.97 voglio che mi scenda a 0.95.

```
k <- seq(0.5,0.8,by=0.001)
risultati <- matrix(NA,ncol=4,nrow=length(k))

for(i in 1:length(k)){
  traslata <- function(x,a,b)densbeta(x,a,b)-k[i]
  hpd1 <- uniroot(traslata,c(0.195,0.25),a=14.26,b=23.19)
  h1 <-  hpd1$root
  hpd2 <- uniroot(traslata,c(0.45,0.6),a=14.26,b=23.19)
  h2 <-  hpd2$root
  int <- integrate(densbeta,
                   lower=h1,
                   upper=h2,
                   a=14.26,b=23.19)
  int1<-int$value
  fine<-i
  if(int1 <= 0.95) break
  risultati[i,] <- c(h1,h2,int1, k[i])
}

fine
```

```
## [1] 293
```

```
risultati
```

```
##             [,1]      [,2]      [,3]  [,4]
##   [1,] 0.2154529 0.5536319 0.9714869 0.500
##   [2,] 0.2155129 0.5535545 0.9714181 0.501
##   [3,] 0.2155728 0.5534772 0.9713493 0.502
##   [4,] 0.2156326 0.5534001 0.9712804 0.503
##   [5,] 0.2156923 0.5533230 0.9712115 0.504
##   [6,] 0.2157520 0.5532461 0.9711425 0.505
##   [7,] 0.2158115 0.5531693 0.9710735 0.506
##   [8,] 0.2158710 0.5530926 0.9710045 0.507
##   [9,] 0.2159304 0.5530160 0.9709354 0.508
##  [10,] 0.2159897 0.5529395 0.9708663 0.509
##  [11,] 0.2160490 0.5528631 0.9707972 0.510
##  [12,] 0.2161081 0.5527869 0.9707281 0.511
##  [13,] 0.2161672 0.5527108 0.9706589 0.512
##  [14,] 0.2162262 0.5526347 0.9705896 0.513
##  [15,] 0.2162851 0.5525588 0.9705203 0.514
##  [16,] 0.2163439 0.5524830 0.9704510 0.515
##  [17,] 0.2164026 0.5524074 0.9703817 0.516
##  [18,] 0.2164613 0.5523318 0.9703123 0.517
##  [19,] 0.2165199 0.5522563 0.9702429 0.518
##  [20,] 0.2165783 0.5521810 0.9701735 0.519
##  [21,] 0.2166368 0.5521058 0.9701040 0.520
##  [22,] 0.2166951 0.5520306 0.9700345 0.521
##  [23,] 0.2167534 0.5519556 0.9699649 0.522
##  [24,] 0.2168115 0.5518807 0.9698953 0.523
##  [25,] 0.2168696 0.5518059 0.9698257 0.524
##  [26,] 0.2169277 0.5517312 0.9697561 0.525
##  [27,] 0.2169856 0.5516566 0.9696864 0.526
##  [28,] 0.2170435 0.5515821 0.9696167 0.527
##  [29,] 0.2171013 0.5515078 0.9695469 0.528
```

```
##  [30,] 0.2171590 0.5514335 0.9694771 0.529
##  [31,] 0.2172166 0.5513593 0.9694073 0.530
##  [32,] 0.2172742 0.5512853 0.9693374 0.531
##  [33,] 0.2173317 0.5512113 0.9692675 0.532
##  [34,] 0.2173891 0.5511375 0.9691976 0.533
##  [35,] 0.2174464 0.5510638 0.9691276 0.534
##  [36,] 0.2175037 0.5509901 0.9690576 0.535
##  [37,] 0.2175608 0.5509166 0.9689876 0.536
##  [38,] 0.2176179 0.5508432 0.9689175 0.537
##  [39,] 0.2176750 0.5507699 0.9688474 0.538
##  [40,] 0.2177319 0.5506967 0.9687773 0.539
##  [41,] 0.2177888 0.5506235 0.9687071 0.540
##  [42,] 0.2178456 0.5505505 0.9686369 0.541
##  [43,] 0.2179024 0.5504776 0.9685666 0.542
##  [44,] 0.2179590 0.5504048 0.9684964 0.543
##  [45,] 0.2180156 0.5503321 0.9684260 0.544
##  [46,] 0.2180721 0.5502595 0.9683557 0.545
##  [47,] 0.2181286 0.5501870 0.9682853 0.546
##  [48,] 0.2181849 0.5501146 0.9682149 0.547
##  [49,] 0.2182412 0.5500423 0.9681445 0.548
##  [50,] 0.2182975 0.5499701 0.9680740 0.549
##  [51,] 0.2183536 0.5498980 0.9680035 0.550
##  [52,] 0.2184097 0.5498259 0.9679329 0.551
##  [53,] 0.2184657 0.5497540 0.9678623 0.552
##  [54,] 0.2185216 0.5496822 0.9677917 0.553
##  [55,] 0.2185775 0.5496105 0.9677210 0.554
##  [56,] 0.2186333 0.5495389 0.9676503 0.555
##  [57,] 0.2186890 0.5494674 0.9675796 0.556
##  [58,] 0.2187447 0.5493959 0.9675089 0.557
##  [59,] 0.2188003 0.5493246 0.9674381 0.558
##  [60,] 0.2188558 0.5492534 0.9673672 0.559
##  [61,] 0.2189113 0.5491823 0.9672964 0.560
##  [62,] 0.2189666 0.5491112 0.9672255 0.561
##  [63,] 0.2190220 0.5490403 0.9671546 0.562
##  [64,] 0.2190772 0.5489694 0.9670836 0.563
##  [65,] 0.2191324 0.5488987 0.9670126 0.564
##  [66,] 0.2191875 0.5488280 0.9669416 0.565
##  [67,] 0.2192425 0.5487574 0.9668705 0.566
##  [68,] 0.2192975 0.5487063 0.9668104 0.567
##  [69,] 0.2193524 0.5486358 0.9667392 0.568
##  [70,] 0.2194072 0.5485654 0.9666680 0.569
##  [71,] 0.2194620 0.5484951 0.9665967 0.570
##  [72,] 0.2195167 0.5484249 0.9665254 0.571
##  [73,] 0.2195713 0.5483547 0.9664541 0.572
##  [74,] 0.2196259 0.5482847 0.9663828 0.573
##  [75,] 0.2196804 0.5482148 0.9663114 0.574
##  [76,] 0.2197349 0.5481449 0.9662400 0.575
##  [77,] 0.2197892 0.5480751 0.9661685 0.576
##  [78,] 0.2198435 0.5480055 0.9660970 0.577
##  [79,] 0.2198978 0.5479359 0.9660255 0.578
##  [80,] 0.2199520 0.5478664 0.9659539 0.579
##  [81,] 0.2200061 0.5477970 0.9658824 0.580
##  [82,] 0.2200601 0.5477277 0.9658107 0.581
##  [83,] 0.2201141 0.5476585 0.9657391 0.582
```

```
##  [84,] 0.2201680 0.5475894 0.9656674 0.583
##  [85,] 0.2202219 0.5475203 0.9655957 0.584
##  [86,] 0.2202757 0.5474514 0.9655239 0.585
##  [87,] 0.2203294 0.5473825 0.9654521 0.586
##  [88,] 0.2203831 0.5473137 0.9653803 0.587
##  [89,] 0.2204196 0.5472451 0.9653185 0.588
##  [90,] 0.2204732 0.5471765 0.9652466 0.589
##  [91,] 0.2205268 0.5471080 0.9651746 0.590
##  [92,] 0.2205803 0.5470395 0.9651026 0.591
##  [93,] 0.2206337 0.5469712 0.9650306 0.592
##  [94,] 0.2206871 0.5469030 0.9649585 0.593
##  [95,] 0.2207404 0.5468348 0.9648864 0.594
##  [96,] 0.2207937 0.5467667 0.9648143 0.595
##  [97,] 0.2208469 0.5466987 0.9647421 0.596
##  [98,] 0.2209000 0.5466308 0.9646699 0.597
##  [99,] 0.2209531 0.5465630 0.9645977 0.598
## [100,] 0.2210061 0.5464953 0.9645254 0.599
## [101,] 0.2210590 0.5464276 0.9644532 0.600
## [102,] 0.2211119 0.5463601 0.9643808 0.601
## [103,] 0.2211647 0.5462926 0.9643085 0.602
## [104,] 0.2212175 0.5462252 0.9642361 0.603
## [105,] 0.2212702 0.5461579 0.9641636 0.604
## [106,] 0.2213229 0.5460906 0.9640912 0.605
## [107,] 0.2213754 0.5460235 0.9640187 0.606
## [108,] 0.2214280 0.5459564 0.9639462 0.607
## [109,] 0.2214804 0.5458894 0.9638736 0.608
## [110,] 0.2215328 0.5458225 0.9638010 0.609
## [111,] 0.2215852 0.5457557 0.9637284 0.610
## [112,] 0.2216375 0.5456890 0.9636557 0.611
## [113,] 0.2216897 0.5456223 0.9635830 0.612
## [114,] 0.2217419 0.5455558 0.9635103 0.613
## [115,] 0.2217940 0.5454893 0.9634375 0.614
## [116,] 0.2218461 0.5454229 0.9633647 0.615
## [117,] 0.2218981 0.5453565 0.9632919 0.616
## [118,] 0.2219500 0.5452903 0.9632190 0.617
## [119,] 0.2220019 0.5452241 0.9631462 0.618
## [120,] 0.2220537 0.5451580 0.9630732 0.619
## [121,] 0.2221055 0.5450920 0.9630003 0.620
## [122,] 0.2221572 0.5450261 0.9629273 0.621
## [123,] 0.2222088 0.5449603 0.9628543 0.622
## [124,] 0.2222604 0.5448945 0.9627812 0.623
## [125,] 0.2223120 0.5448288 0.9627081 0.624
## [126,] 0.2223635 0.5447632 0.9626350 0.625
## [127,] 0.2224149 0.5446976 0.9625618 0.626
## [128,] 0.2224662 0.5446322 0.9624886 0.627
## [129,] 0.2225176 0.5445668 0.9624154 0.628
## [130,] 0.2225688 0.5445015 0.9623422 0.629
## [131,] 0.2226200 0.5444363 0.9622689 0.630
## [132,] 0.2226712 0.5443711 0.9621955 0.631
## [133,] 0.2227223 0.5443061 0.9621222 0.632
## [134,] 0.2227733 0.5442411 0.9620488 0.633
## [135,] 0.2228243 0.5441761 0.9619754 0.634
## [136,] 0.2228752 0.5441113 0.9619019 0.635
## [137,] 0.2229261 0.5440465 0.9618284 0.636
```

```
## [138,] 0.2229769 0.5439818 0.9617549 0.637
## [139,] 0.2230277 0.5439172 0.9616814 0.638
## [140,] 0.2230784 0.5438527 0.9616078 0.639
## [141,] 0.2231290 0.5437882 0.9615342 0.640
## [142,] 0.2231796 0.5437238 0.9614605 0.641
## [143,] 0.2232302 0.5436595 0.9613868 0.642
## [144,] 0.2232807 0.5435953 0.9613131 0.643
## [145,] 0.2233311 0.5435311 0.9612394 0.644
## [146,] 0.2233815 0.5434670 0.9611656 0.645
## [147,] 0.2234318 0.5434030 0.9610918 0.646
## [148,] 0.2234821 0.5433390 0.9610179 0.647
## [149,] 0.2235323 0.5432752 0.9609441 0.648
## [150,] 0.2235825 0.5432114 0.9608701 0.649
## [151,] 0.2236326 0.5431476 0.9607962 0.650
## [152,] 0.2236827 0.5430840 0.9607222 0.651
## [153,] 0.2237327 0.5430204 0.9606482 0.652
## [154,] 0.2237827 0.5429569 0.9605742 0.653
## [155,] 0.2238326 0.5428935 0.9605001 0.654
## [156,] 0.2238825 0.5428301 0.9604260 0.655
## [157,] 0.2239323 0.5427668 0.9603519 0.656
## [158,] 0.2239821 0.5427036 0.9602777 0.657
## [159,] 0.2240318 0.5426404 0.9602035 0.658
## [160,] 0.2240814 0.5425773 0.9601292 0.659
## [161,] 0.2241310 0.5425143 0.9600550 0.660
## [162,] 0.2241806 0.5424514 0.9599807 0.661
## [163,] 0.2242301 0.5423885 0.9599063 0.662
## [164,] 0.2242795 0.5423257 0.9598320 0.663
## [165,] 0.2243289 0.5422630 0.9597576 0.664
## [166,] 0.2243783 0.5422003 0.9596832 0.665
## [167,] 0.2244276 0.5421378 0.9596087 0.666
## [168,] 0.2244769 0.5420752 0.9595342 0.667
## [169,] 0.2245261 0.5420128 0.9594597 0.668
## [170,] 0.2245752 0.5419504 0.9593851 0.669
## [171,] 0.2246243 0.5418881 0.9593105 0.670
## [172,] 0.2246734 0.5418258 0.9592359 0.671
## [173,] 0.2247224 0.5417637 0.9591612 0.672
## [174,] 0.2247713 0.5417016 0.9590865 0.673
## [175,] 0.2248202 0.5416395 0.9590118 0.674
## [176,] 0.2248691 0.5415775 0.9589371 0.675
## [177,] 0.2249179 0.5415156 0.9588623 0.676
## [178,] 0.2249667 0.5414538 0.9587875 0.677
## [179,] 0.2250154 0.5413920 0.9587126 0.678
## [180,] 0.2250641 0.5413303 0.9586377 0.679
## [181,] 0.2251127 0.5412687 0.9585628 0.680
## [182,] 0.2251612 0.5412071 0.9584879 0.681
## [183,] 0.2252098 0.5411456 0.9584129 0.682
## [184,] 0.2252582 0.5410842 0.9583379 0.683
## [185,] 0.2253066 0.5410228 0.9582628 0.684
## [186,] 0.2253550 0.5409615 0.9581878 0.685
## [187,] 0.2254034 0.5409003 0.9581127 0.686
## [188,] 0.2254516 0.5408391 0.9580375 0.687
## [189,] 0.2254999 0.5407780 0.9579624 0.688
## [190,] 0.2255481 0.5407169 0.9578872 0.689
## [191,] 0.2255962 0.5406560 0.9578119 0.690
```

```
## [192,] 0.2256443 0.5405950 0.9577366 0.691
## [193,] 0.2256923 0.5405342 0.9576614 0.692
## [194,] 0.2257403 0.5404734 0.9575860 0.693
## [195,] 0.2257883 0.5404127 0.9575107 0.694
## [196,] 0.2258362 0.5403520 0.9574353 0.695
## [197,] 0.2258841 0.5402914 0.9573598 0.696
## [198,] 0.2259319 0.5402309 0.9572844 0.697
## [199,] 0.2259796 0.5401705 0.9572089 0.698
## [200,] 0.2260274 0.5401101 0.9571334 0.699
## [201,] 0.2260750 0.5400497 0.9570578 0.700
## [202,] 0.2261227 0.5399894 0.9569822 0.701
## [203,] 0.2261703 0.5399292 0.9569066 0.702
## [204,] 0.2262178 0.5398691 0.9568310 0.703
## [205,] 0.2262653 0.5398090 0.9567553 0.704
## [206,] 0.2263127 0.5397490 0.9566796 0.705
## [207,] 0.2263602 0.5396890 0.9566038 0.706
## [208,] 0.2264075 0.5396291 0.9565280 0.707
## [209,] 0.2264548 0.5395693 0.9564522 0.708
## [210,] 0.2265021 0.5395095 0.9563764 0.709
## [211,] 0.2265493 0.5394498 0.9563005 0.710
## [212,] 0.2265965 0.5393901 0.9562246 0.711
## [213,] 0.2266436 0.5393305 0.9561487 0.712
## [214,] 0.2266907 0.5392710 0.9560727 0.713
## [215,] 0.2267378 0.5392115 0.9559967 0.714
## [216,] 0.2267848 0.5391521 0.9559207 0.715
## [217,] 0.2268318 0.5390928 0.9558446 0.716
## [218,] 0.2268787 0.5390335 0.9557685 0.717
## [219,] 0.2269256 0.5389743 0.9556924 0.718
## [220,] 0.2269724 0.5389151 0.9556162 0.719
## [221,] 0.2270192 0.5388560 0.9555400 0.720
## [222,] 0.2270659 0.5387969 0.9554638 0.721
## [223,] 0.2271126 0.5387380 0.9553876 0.722
## [224,] 0.2271593 0.5386790 0.9553113 0.723
## [225,] 0.2272059 0.5386202 0.9552350 0.724
## [226,] 0.2272525 0.5385614 0.9551586 0.725
## [227,] 0.2272990 0.5385026 0.9550822 0.726
## [228,] 0.2273455 0.5384439 0.9550058 0.727
## [229,] 0.2273919 0.5383853 0.9549294 0.728
## [230,] 0.2274383 0.5383267 0.9548529 0.729
## [231,] 0.2274847 0.5382682 0.9547764 0.730
## [232,] 0.2275310 0.5382097 0.9546999 0.731
## [233,] 0.2275773 0.5381513 0.9546233 0.732
## [234,] 0.2276235 0.5380930 0.9545467 0.733
## [235,] 0.2276697 0.5380347 0.9544701 0.734
## [236,] 0.2277159 0.5379765 0.9543934 0.735
## [237,] 0.2277620 0.5379183 0.9543167 0.736
## [238,] 0.2278080 0.5378602 0.9542400 0.737
## [239,] 0.2278541 0.5378021 0.9541632 0.738
## [240,] 0.2279001 0.5377441 0.9540864 0.739
## [241,] 0.2279460 0.5376862 0.9540096 0.740
## [242,] 0.2279919 0.5376283 0.9539328 0.741
## [243,] 0.2280378 0.5375705 0.9538559 0.742
## [244,] 0.2280836 0.5375127 0.9537790 0.743
## [245,] 0.2281294 0.5374550 0.9537020 0.744
```

```
## [246,] 0.2281751 0.5373974 0.9536250 0.745
## [247,] 0.2282208 0.5373398 0.9535480 0.746
## [248,] 0.2282665 0.5372822 0.9534710 0.747
## [249,] 0.2283121 0.5372247 0.9533939 0.748
## [250,] 0.2283577 0.5371673 0.9533168 0.749
## [251,] 0.2284032 0.5371099 0.9532397 0.750
## [252,] 0.2284487 0.5370526 0.9531625 0.751
## [253,] 0.2284941 0.5369953 0.9530853 0.752
## [254,] 0.2285396 0.5369379 0.9530080 0.753
## [255,] 0.2285849 0.5368805 0.9529305 0.754
## [256,] 0.2286303 0.5368231 0.9528530 0.755
## [257,] 0.2286756 0.5367658 0.9527754 0.756
## [258,] 0.2287208 0.5367085 0.9526979 0.757
## [259,] 0.2287661 0.5366513 0.9526203 0.758
## [260,] 0.2288112 0.5365941 0.9525426 0.759
## [261,] 0.2288564 0.5365370 0.9524650 0.760
## [262,] 0.2289015 0.5364799 0.9523873 0.761
## [263,] 0.2289465 0.5364229 0.9523096 0.762
## [264,] 0.2289916 0.5363660 0.9522318 0.763
## [265,] 0.2290365 0.5363091 0.9521540 0.764
## [266,] 0.2290815 0.5362522 0.9520762 0.765
## [267,] 0.2291264 0.5361955 0.9519984 0.766
## [268,] 0.2291713 0.5361387 0.9519205 0.767
## [269,] 0.2292161 0.5360821 0.9518426 0.768
## [270,] 0.2292609 0.5360254 0.9517646 0.769
## [271,] 0.2293056 0.5359689 0.9516867 0.770
## [272,] 0.2293504 0.5359124 0.9516087 0.771
## [273,] 0.2293950 0.5358559 0.9515306 0.772
## [274,] 0.2294397 0.5357995 0.9514526 0.773
## [275,] 0.2294843 0.5357431 0.9513745 0.774
## [276,] 0.2295288 0.5356868 0.9512964 0.775
## [277,] 0.2295734 0.5356306 0.9512182 0.776
## [278,] 0.2296179 0.5355744 0.9511400 0.777
## [279,] 0.2296623 0.5355182 0.9510618 0.778
## [280,] 0.2297067 0.5354622 0.9509835 0.779
## [281,] 0.2297793 0.5354061 0.9508833 0.780
## [282,] 0.2298234 0.5353501 0.9508051 0.781
## [283,] 0.2298675 0.5352942 0.9507269 0.782
## [284,] 0.2299116 0.5352383 0.9506486 0.783
## [285,] 0.2299556 0.5351825 0.9505704 0.784
## [286,] 0.2299996 0.5351267 0.9504921 0.785
## [287,] 0.2300436 0.5350710 0.9504137 0.786
## [288,] 0.2300875 0.5350153 0.9503354 0.787
## [289,] 0.2301314 0.5349597 0.9502570 0.788
## [290,] 0.2301752 0.5349041 0.9501785 0.789
## [291,] 0.2302191 0.5348486 0.9501001 0.790
## [292,] 0.2302628 0.5347931 0.9500216 0.791
## [293,]        NA        NA        NA    NA
## [294,]        NA        NA        NA    NA
## [295,]        NA        NA        NA    NA
## [296,]        NA        NA        NA    NA
## [297,]        NA        NA        NA    NA
## [298,]        NA        NA        NA    NA
## [299,]        NA        NA        NA    NA
```

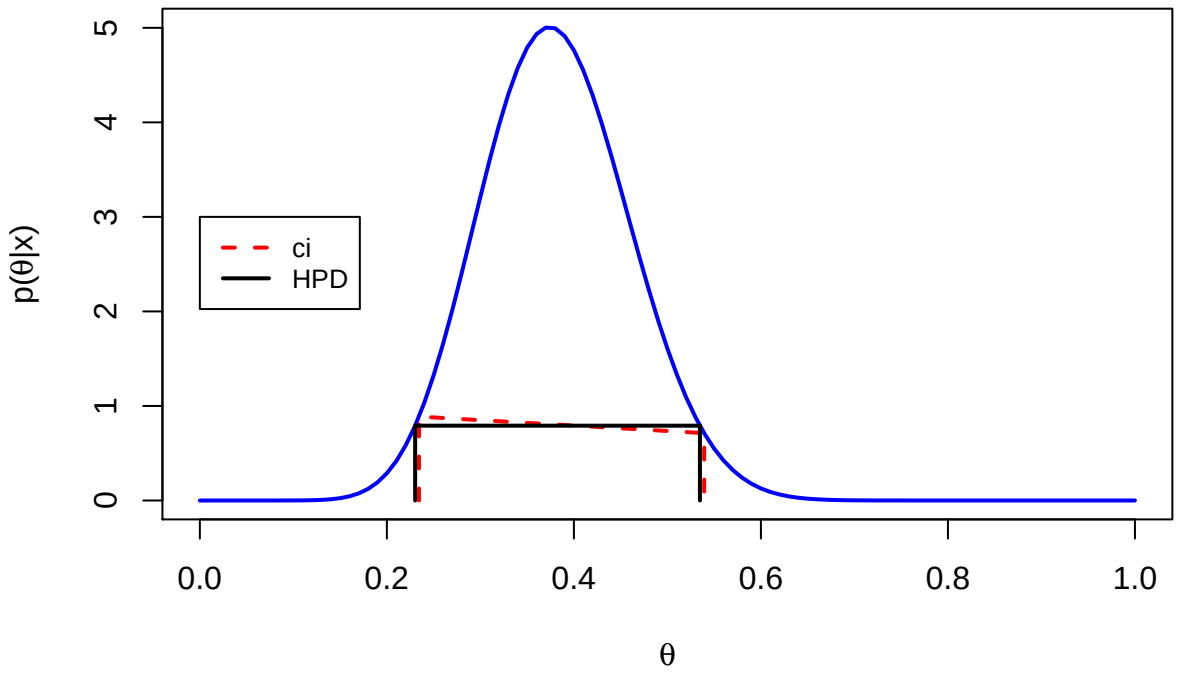
```
## [300,]         NA         NA         NA    NA
## [301,]         NA         NA         NA    NA
```

```
hpd <- risultati[fine-1,-3];hpd
```

```
## [1] 0.2302628 0.5347931 0.7910000
```

Il valore di k ottimale è pari a 0.791

```
curve(densbeta(x,a=14.26,b=23.19),0,1,
      ylab=expression(paste(p,"(",theta,"|x)")),
      xlab=expression(theta),lwd=2, col ="blue")
lines( x=c(ci[1],ci[1],ci[2],ci[2]),
       y=c(0,dbeta(c(ci[1],ci[2]),14.26,23.19),0),
       col="red",
       lwd=2,
       lty=2)
lines( x=c(hpd[1],hpd[1],hpd[2],hpd[2]),
       y=c(0,dbeta(c(hpd[1],hpd[2]),14.26,23.19),0),
       col=1,
       lwd=2)
legend(0,3,c("ci","HPD"),
       col=c("red",1),
       cex=0.8,
       lwd=2,
       lty=c(2,1))
```



Vediamo che i due intervalli si assomigliano ma non coincidono. il problema non si pone se la distribuzione a posteriori è simmetrica.

09 01 18 lez 15

Distribuzione predittiva: esempio trapianti di cuore

Esempio di modello coniugato Gamma-Gammma

PRIOR

Disegno la prior con i relativi parametri: -alpha è il numero di decessi osservati negli ultimi 30 gg in 10 ospedali osservati -beta è il numero dei rispettivi pazienti esposti a trapianto negli ultimi 30 gg in 10 ospedali osservati Inoltre fisso un valore medio di riferimento per il tasso di mortalità (lambda della Poisson)

```
alpha <- 16
beta <- 15174
lam <- alpha/beta;lam
```

```
## [1] 0.001054435
```

Il tasso è molto basso.

OSSERVAZIONI

Considero ora un solo ospedale per raccogliere osservazioni e sul quale improntare il modello. si osserva 1 solo decesso e 66 esposti.

```
ex <- 66
y <- 0:10   #osservazioni future
```

DISTRIB PREDITTIVA fy PER IL NUMERO DI DECESSI

fy = verosimiglianza*prior / posterior

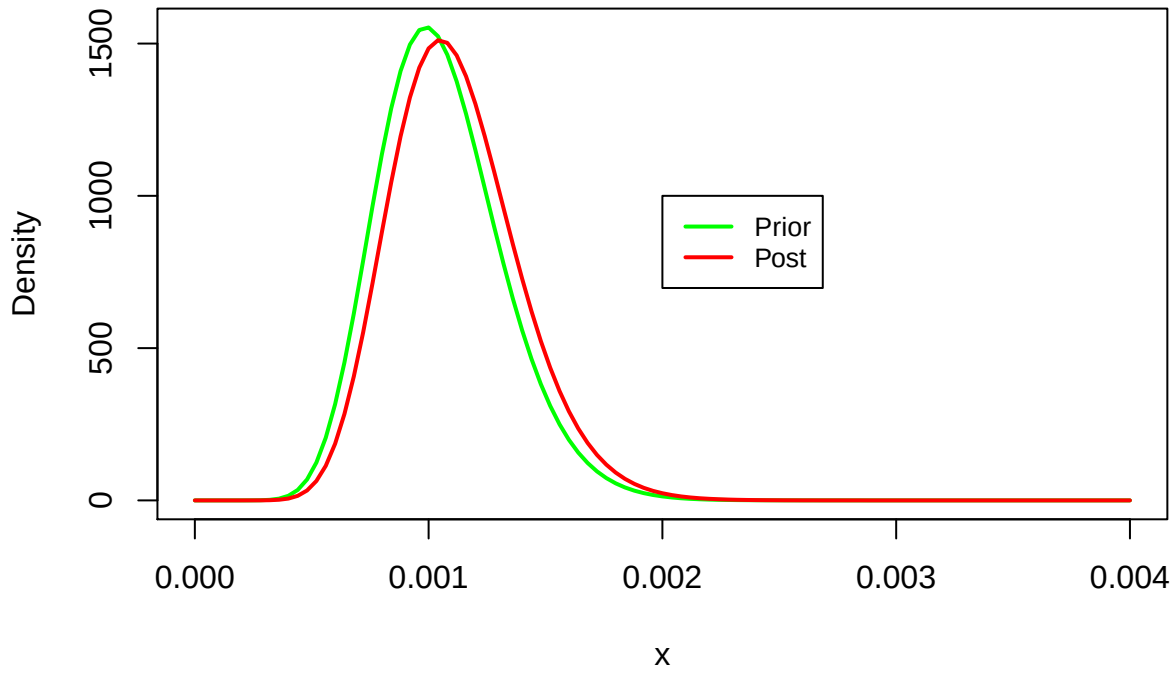
```
fy <- dpois(y,lam*ex)          *  #verosimiglianza per le osservazioni è una poisson
  dgamma(lam,                     #prior è una gamma di parametri fissati
         shape = alpha,
         rate = beta)         /   #divido per la posteriori in base alla regola della
  dgamma(lam,
         shape = alpha+y,
         rate = beta+ex)

cbind(y,round(fy,3))  #vediamo il risultato del calcolo del rapporto, arrotondato
```

```
##        y
##  [1,]  0 0.933
##  [2,]  1 0.065
##  [3,]  2 0.002
##  [4,]  3 0.000
##  [5,]  4 0.000
##  [6,]  5 0.000
##  [7,]  6 0.000
##  [8,]  7 0.000
##  [9,]  8 0.000
## [10,]  9 0.000
## [11,] 10 0.000
```

Per la distibuzione predittiva mi aspetto un prob molto alta per valori bassi dei decessi. Infatti è così.

Disegno la prior e la posterior per il parametro di interesse lambda, cioè il tasso di mortalità, e per y=1 osservato nell'ospedale preso a campione

```
curve(dgamma(x,          shape = alpha,         rate = beta),
      col="green" , lwd=2, xlim = c(0,0.004), ylab = "Density")
curve(dgamma(x,         shape = alpha+1,         rate = beta+ex),
      add=T, col="red", lwd=2)
legend(0.002,1000,c("Prior","Post"),
       col=c("green","red"),
       cex=0.8,
       lwd=2,
       lty=c(1,1))
```

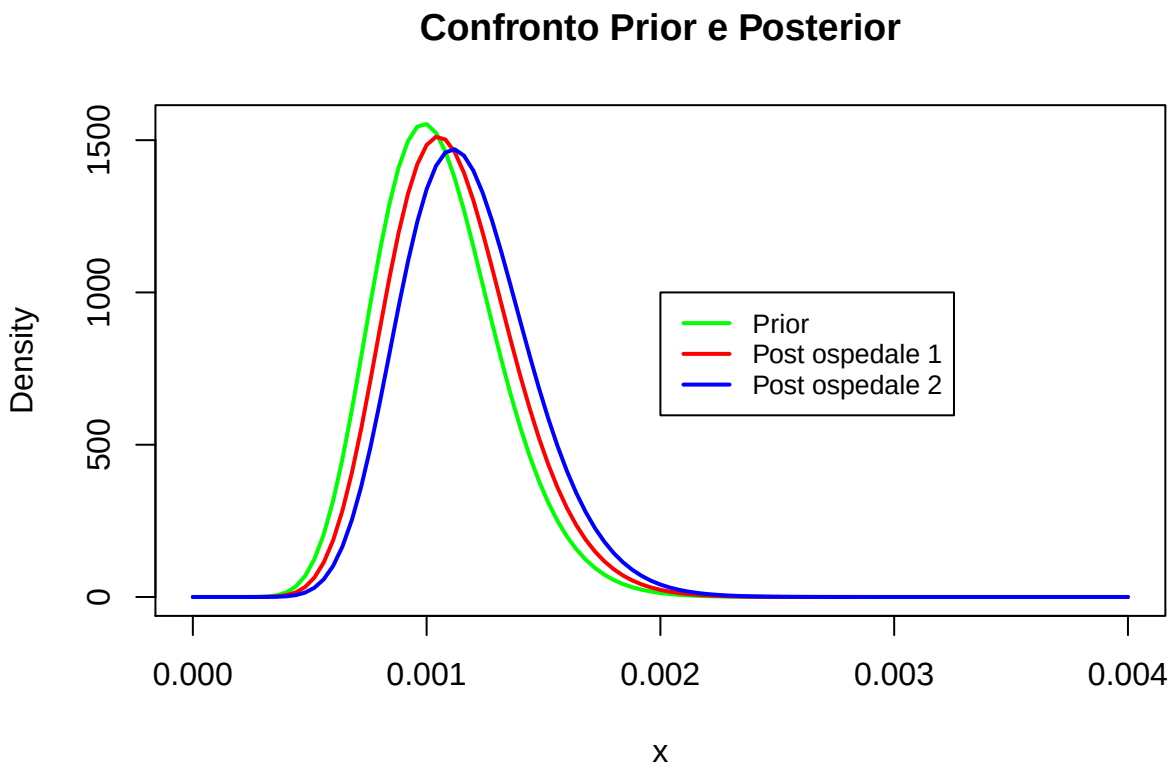


La prior è leggermente spostata a sinistra e più alta

Per un altro ospedale posso osservare una differente posterior. Osservo ad esempio 2 decessi per 100 interventi

```
curve(dgamma(x,          shape = alpha,         rate = beta),
      col="green" , lwd=2, xlim = c(0,0.004), ylab = "Density",
      main="Confronto Prior e Posterior")
curve(dgamma(x,         shape = alpha+1,         rate = beta+ex),
      add=T, col="red", lwd=2)
curve(dgamma(x,         shape = alpha+2,         rate = beta+100),
      add=T, col="blue", lwd=2)

legend(0.002,1000,c("Prior","Post ospedale 1", "Post ospedale 2"),
       col=c("green","red","blue"),
```

```
       cex=0.8,
       lwd=2,
       lty=1)
```



In generale la regola è: assegno una prior e fisso i relativi parametri in base alle conoscenze degli esperti (ad esempio dico che la prior è una Gamma di parametri fissati); se i dati osservati sul fenomeno in studio seguono una Poisson, allora posso seguire la regola per la famiglia coniugata, per cui so che la Posterior è anch'essa una Gamma con parametri facili da calcolare secondo formule note. Questo metodo è comodo perchè non serve svolgere di nuovo i calcoli per nuove osservazioni, ma basta aggiornare i parametri della Post. La posterior cambia se aggiungo/modifico osservazioni La distribuzione predittiva deriva dal rapporto tra prior X verosimiglianza / posterior, dove per la posterior posso mettere il valore ipotetico che mi interessa e che potrebbe essere osservato in futuro. In pratica ottengo le probabilità con cui esso può presentarsi in base al passato e alla conoscenza del fenomeno.

10 01 18 lez 16 Scelta della prior ed inferenza robusta; esempio test

Esempio famiglia coniugata normale: modello coniugato Gaussiano-Gaussiano

Vedremo la robustezza di questo approccio, andando a modificare la natura della prior: normale o T di Student, data una verosimiglianza dei dati osservati normale.

Elicitazione dice che mediana della prior per il punteggio medio è 100 e un valore plausibile per il 95esimo perc è 120.

Imposto la prior e rispettivi parametri

La funzione normal.select trova mu e tau2 della prior normal.select parte dai quantili (simile a beta.select)

```
require(LearnBayes)
quantile1 <- list(p=.5,x=100) #imposto la mediana che è il 50esimo perc
quantile2 <- list(p=.95,x=120)  #imposto il 95esimo perc
ris <- normal.select(quantile1, quantile2)
mu <- ris$mu; mu
```

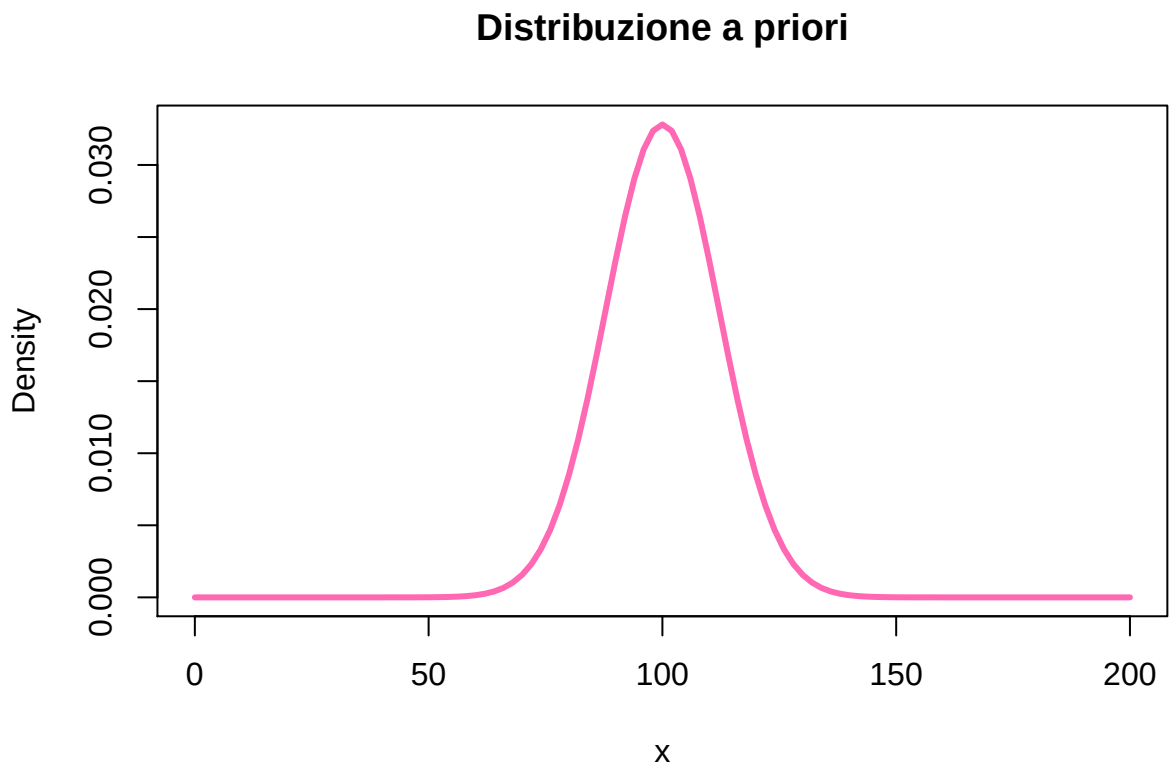
```
## [1] 100
```

```
tau <- ris$sigma; tau
```

```
## [1] 12.15914
```

Disegno la distribuzione a priori.

```
curve(dnorm(x,mu, tau), col="hotpink",
      main="Distribuzione a priori",
      lwd=3, xlim = c(0,200), ylab = "Density")
```



Notiamo una varianza abbastanza ridotta

Verosimiglianza Normale

Supponendo di voler lavorare nella famiglia coniugata, la verosimiglianza sarà una Normale(theta, sigma2) con sigma2 noto e pari a 15^2. Quindi la distribuzione della media campionaria theta è N(theta, sigma/radice di n). Ho 4 punteggi osservati: x1,x2,x3,x4

```
sigma <- 15  #dev std nota per la verosimiglianza
se <- sigma/sqrt(4) #dev std della media campionaria per n=4
xnn <- c(110, 125, 140) #insieme di valori possibili per il punteggio medio, serviranno per testare div
```

Caratterizzazione della posterior secondo il modello coniugato

```
tau1 <- 1/sqrt(1/se^2 + 1/tau^2)  #dev std della post
mu1 <- (xnn/se^2 + mu/tau^2) * tau1^2  #media della post scritta in funzione della varianza a posterior
```

Creo la tabella per vedere come cambia la media a posteriori al cambiare del punteggio medio osservato (basato sempre su 4 osservazioni); in pratica sto osservando diversi scenari per il campione osservato. Tau1 rimane sempre uguale perchè dipende solo da sigma2 e tau2 che sono fissati a priori.

```
summ1 <- cbind(xnn, mu1, tau1)
summ1
```

```
##      xnn      mu1     tau1
## [1,] 110 107.2439 6.383344
## [2,] 125 118.1098 6.383344
## [3,] 140 128.9757 6.383344
```

Prior T di Student con 2 gdl

Caratterizzazione della distribuzione a priori. cerco il quantile 95esimo corrispondente alla t di student con 2 gdl

```
quant <- qt(0.95,2);quant
```

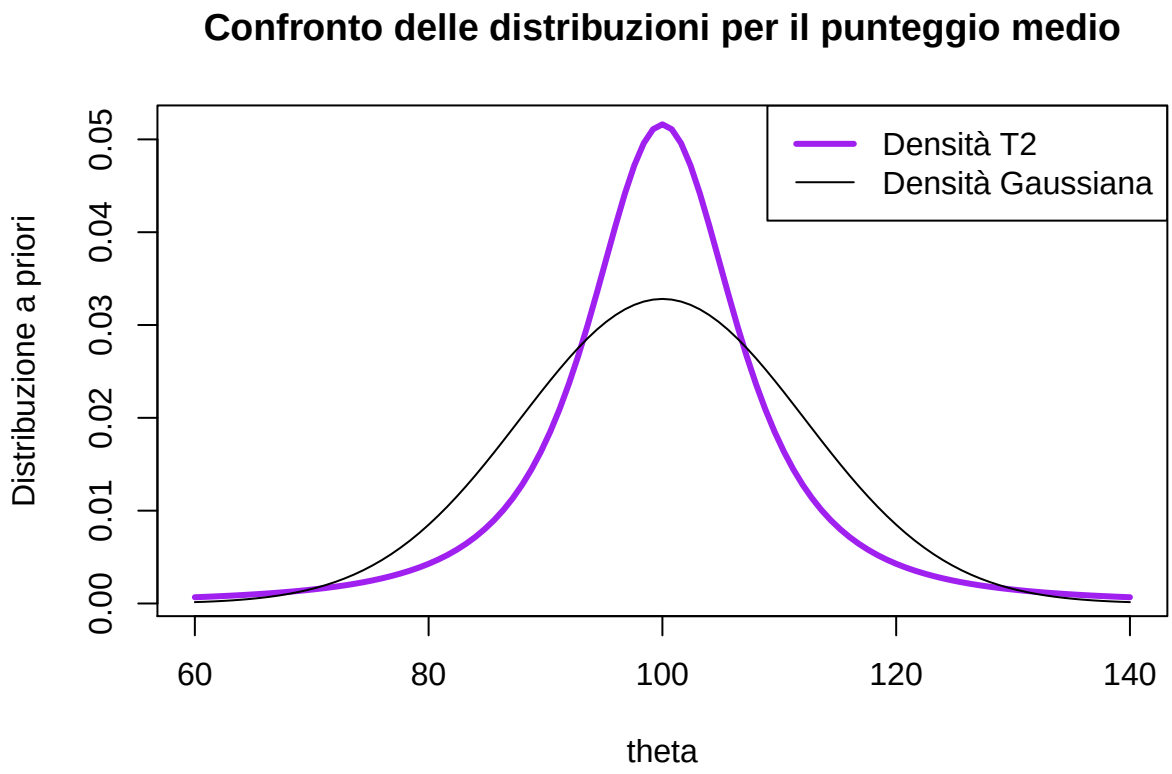
```
## [1] 2.919986
```

```
taut <- 20/quant; taut
```

```
## [1] 6.849349
```

Disegno la prior

```
curve(1/taut*dt((x-mu)/taut,2), #uso il reciproco della variabilità
      xlim=c(60,140) ,
      xlab="theta",
      ylab="Distribuzione a priori",
      main ="Confronto delle distribuzioni per il punteggio medio",
      col ='purple',
      lwd=3)
 curve(dnorm(x,mean=mu,sd=tau),
       add=TRUE,
       lwd=1)
 legend("topright",
        legend=c("Densità T2","Densità Gaussiana"),
        lwd=c(3,1),
        col=c("purple", "black"))
```



La T ha code più pesanti della normale

Verosimiglianza per un possibile range di valori di theta, il punteggio medio.

```
theta <-  seq(60, 180, length = 500)
summary(theta)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      60      90     120     120     150     180
```

```
#verosimiglianza come sigma/radice di n eprchè stiamo facendo la media campionaria
n <- 4
like <- dnorm(theta,mean=xnn,sd=sigma/sqrt(n))
summary(round(like,3) )
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000000 0.000000 0.008296 0.007000 0.053000
```

```
prior <- dt((theta - mu)/taut, 2)
summary(round(prior,3))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00100 0.00200 0.01000 0.05598 0.05625 0.35400
```

Costruisco la posteriori come prodotto tra ver e prior

```
post <-  prior * like  #calcolo post
post <- post/sum(post)  #normalizzo
posizione <- sum(theta * post); posizione #media
```

```
## [1] 108.1138
```

```
scala <- sqrt(sum(theta^2 * post) - posizione^2); scala #dispersione
```

```
## [1] 9.151818
```

La nostra distrib di student con 2 gdl ha media 108.1 e variabilità 9.15

Funzione alternativa per il calcolo della posteriori

Alternativamente posso usare questa funzione per unire tutte le operazioni! La funzione norm.t.compute, partendo da una serie di valori per theta, genera la distribuzione a posteriori e restituisce solo i parametri di scala e posizione per essa, utilizzando la prior e la verosimiglianza.

```
norm.t.compute <- function(xnn){
                  theta <- seq(60, 180, length = 500)
                  like <- dnorm(theta,mean=xnn,sd=sigma/sqrt(n))
                  prior <- dt((theta - mu)/taut, 2)
                  post <- prior * like  #combinazione di prior e verosim
                  post <- post/sum(post)  #normalizzazione
                  posizione <- sum(theta * post)  #media
          scala <- sqrt(sum(theta^2 * post) - posizione^2)  #variabilità
                  c(xnn, posizione, scala)
 }
```

Confronto tra prior Normale e Prior Student

Applico la funzione al vettore xnn dei vari punteggi medi testati per ottenerei valori di confronto

```
summ2<- t(sapply(c(110, 125, 140),norm.t.compute))
dimnames(summ2)[[2]]=c("xnn","mu1 t","tau1 t")
summ2
```

```
##      xnn    mu1 t   tau1 t
## [1,] 110 105.2921 5.841676
## [2,] 125 118.0841 7.885174
## [3,] 140 135.4134 7.973498
```

Confronto normale e student al variare della prior

```
cbind(summ1,summ2)
```

```
##      xnn      mu1     tau1 xnn    mu1 t   tau1 t
## [1,] 110 107.2439 6.383344 110 105.2921 5.841676
## [2,] 125 118.1098 6.383344 125 118.0841 7.885174
## [3,] 140 128.9757 6.383344 140 135.4134 7.973498
```

Confronto grafico

```
theta <- seq(60, 180, length=500)
mu1[3]
```

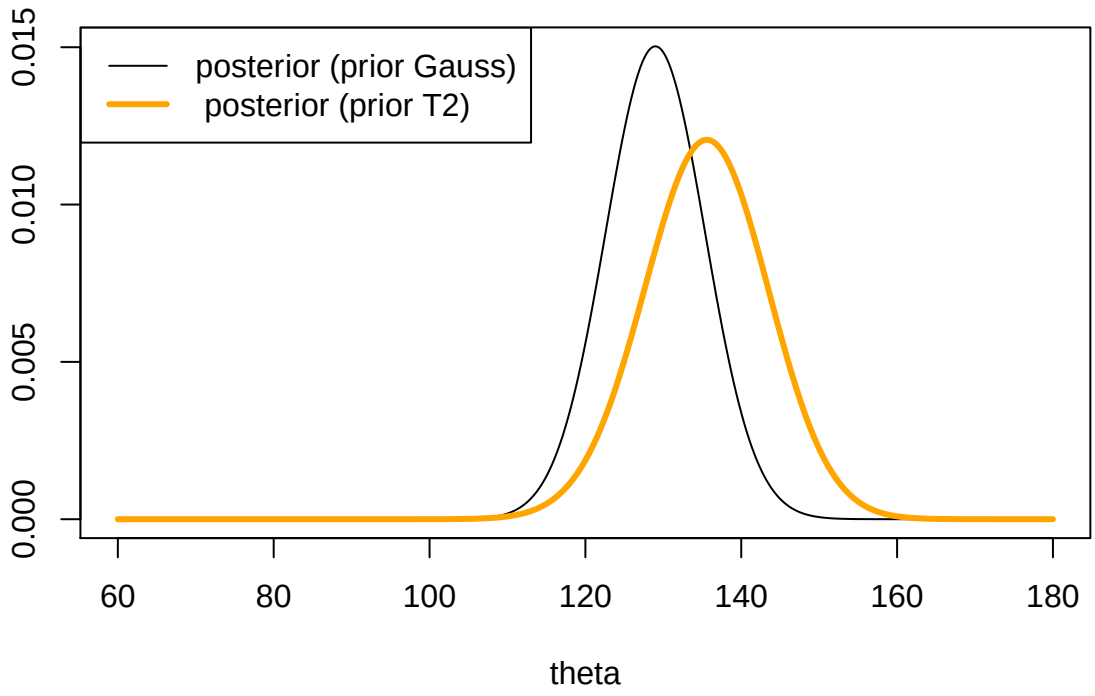
```
## [1] 128.9757
```

```
tau1
```

```
## [1] 6.383344
```

```
normpost <-dnorm(theta, mu1[3], tau1)
normpost <- normpost/sum(normpost)
```

```r
plot(theta,normpost,
      type="l",
      lwd=1,
      ylab=" ")

like <- dnorm(theta,mean=140,sd=sigma/sqrt(n))
prior <- dt((theta - mu)/taut, 2)
tpost <- prior * like / sum(prior * like)

lines(theta,tpost, col="orange", lwd=3)
legend("topleft",
       legend=c("posterior (prior Gauss)"," posterior (prior T2)"),
       lwd=c(1,3),
       col = c( "black","orange"))
```



La differenza si nota per valori lontani dalla media, cioè per outliers, per valori molto grandi del punteggio medio. Infatti differiscono soprattutto per il valore 140

Metodo Metropolis-Hastings

Esempio di applicazione del Metropolis per la distribuzione a posteriori della media campionaria

Ipotizzo un random walk per simulare i valori di theta x'= x+ epsilon supponendo epsilon come una N(0,dˆ2) con d=1

I parametri della prior sono quindi fissati e pari a mu=0 e tau=1 Per la verosimiglianza, sigma è noto e =1

```
x <- 3   #valore iniziale
mu <- 0
tau <- 2
sigma <- 1
d <- 1
```

Ora generiamo delle realizzazioni e applichiamo il rapporto R. Imposto la simulazione. Il rapporto ha: al numeratore il prodotto tra le densità corrispondenti calcolate per il valore candidato theta.p, al denominatore ha lo stesso prodotto ma calcolato per il valore precedente theta[i-1]

```
set.seed(1333)
nsim <- 10
theta <- rep(0,nsim)

#inizializzo l'algoritmo col valore iniziale
theta[1] <- x

#costruisco il ciclo simulativo
#theta.p è il candidato x'
for(i in 2:nsim){
  theta.p <- theta[i-1] + rnorm(1,0,d)  #x'= x + epsilon
  #costruisco il rapporto R = (f * g)[i] / (f*g)[i-1]
  ratio <- dnorm(x,theta.p,sigma)*dnorm(theta.p,mu,tau) /
           dnorm(x,theta[i-1],sigma)*dnorm(theta[i-1],mu,tau)
  print(round(ratio,3))

  evento <- rbinom(1,1,min(ratio,1))
  #evento successo = appartenere alla posterior
  #se evento=0 rifiuto la realizzazione, se evento=1 accetto
  #se ratio è inferiore a 1, prendo l'evento con prob data dal ratio
  #più il valore candidato si avvicina al precedente, più avrò prob alta perchè il     rapporto si avvi
  #se ratio è > 1, prendo 1 e cioè avrò prob di evento 1
  print(evento)

  theta.p[i] <- if(evento==1) theta.p else theta[i-1]

}
```

```
## [1] 0.004
## [1] 0
## [1] 1.225
## [1] 1
## [1] 0.086
## [1] 0
## [1] 0.124
## [1] 0
## [1] 0.121
## [1] 0
## [1] 0.2
## [1] 0
## [1] 0.006
## [1] 0
## [1] 0.01
## [1] 0
```

```
## [1] 1.309
## [1] 1
```

15 01 18 lez 17

Metodo Metropolis-Hastings su famiglia gaussiana

Riprende il codice precedente ma con delle modifiche e integrazioni

```
theta0 <- 3 #valore iniziale
mu <- 0    #media della prior
tau <- 2    #deviazione standard della prior
sigma <- 1   #deviazione std della verosimiglianza
d <- 1   #valore della dev std per la varianza della distribuzione del random walk
```

Per la posterior, in base alle formule della famiglia coniugata, mu1 dovrebbe essere [0+(1/(1+1/2))x3]

Ora vediamo col metodo M-H come risulta la media a posteriori mu1, ipotizzata ignota e ottenuta con simulazione. Imposto la simulazione. Ho 3 oggetti nel ciclo: - theta.p è la realizzazione candidata del random walk, distribuito come una normale con media 0 e varianza d^2 - ratio è il rapporto: verosim X prior in theta.p/ verosim X prior in theta-1 - evento accetto/rifiuto

```
nsim  <- 10
theta <- rep(0,nsim)

theta[1] <- theta0
set.seed(173)
for(i in 2:nsim){
       theta.p <- theta[i-1] + rnorm(1,0,d)   #realizzazione candidato
       ratio <- dnorm(theta0, theta.p, sigma)*
                 dnorm(theta.p, mu, tau) /
                (dnorm(theta0, theta[i-1], sigma)
                 *dnorm(theta[i-1], mu, tau))
 print(ratio)
#il ratio mi dà solo la prob di successo con cui svolgerò l'evento successo/non successo, non è in base

#l'evento riguarda la distrib di supporto per la quantità in esame
#è un altro esperimento rispetto al ratio, anzi lo utilizza ma è lui il vero esperimento che facciamo p
  evento <- rbinom(1,1,min(ratio,1))  #distrib di accettazione
  # una realizzazione da un esperimento con prob di successo min(ratio,1)
  print(evento) #vedo se evento è 1 o 0

  theta[i] <- if(evento==1) theta.p else theta[i-1]
}
```

```
## [1] 0.2265882
## [1] 0
## [1] 0.5808754
## [1] 0
## [1] 1.23536
## [1] 1
## [1] 0.8725684
## [1] 1
## [1] 1.126791
```

```
## [1] 1
## [1] 0.7768596
## [1] 1
## [1] 0.95837
## [1] 0
## [1] 1.105759
## [1] 1
## [1] 0.3801577
## [1] 0
```

```
print(theta)
```

```
##  [1] 3.000000 3.000000 3.000000 2.547716 2.889819 2.621191 1.727008
##  [8] 1.727008 1.859567 1.859567
```

In output avrò i 10 valori che mi aspetto come realizzazione accettate per la posterior. Ho 3 volte il valore iniziale (cioè ho rifiutato 3 volte il valore generato) e 7 valori realizzati, quindi ho accettato 7 volte il valore simulato

Procedura di Burn-in

Si scarta una parte di osservazioni o meglio una parte di valori generati, ad esempio la prima metà. Poi in base ad alcuni strumenti diagnostici, come dei grafici, si valutano le osservazioni rimaste.

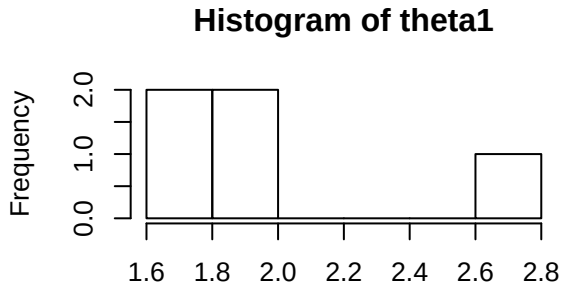
```
theta1 <-theta[-(1:(nsim/2))]; theta1 #scarto la prima metà
```

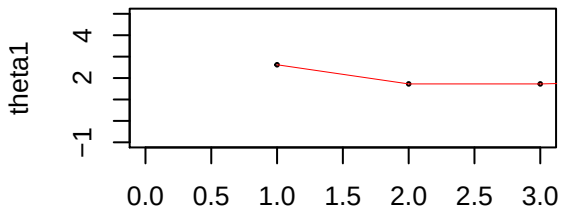
```
## [1] 2.621191 1.727008 1.727008 1.859567 1.859567
```

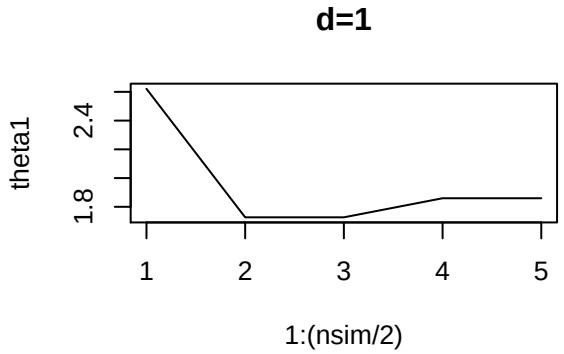
Si scartano perchè, a meno di tantissime simulazioni nell'ordine delle decine di migliaia, voglio evitare della variabilità esterna che disturba. Il Periodo di Burning è il numero di step che si eliminano per ritenere buona la simulazione. Elimino perchè non mi aspetto di aver già trovato la distribuzione di equilibrio

Grafici diagnostici per valutare la bontà della simulazione

```
par(mfrow=c(2,2))
hist(theta1)
plot(theta1,pch=1,cex=0.3,xlim=c(0,3),ylim=c(-1,5))
lines(theta1,lwd=0.5,col="red")
plot(1:(nsim/2),theta1, type ='l', main = "d=1" )
```







Nel secondo grafico ho linea di tendenza delle realizzazioni del parametro Nel terzo grafico ho sull'ascissa il valore realtivo al numero di realizzazioni per l'algoritmo, sull'ordinata ho i valori realizzati: in questo caso vedo che scende ad un valore molto basso già al secondo passo. Quello che mi aspetto è che, al crescere del numero di realizzazioni, la serie si assesti attorno ad un valore medio

Aumento il numero di simulazioni: 1 mln

```
nsim  <- 10^6
theta <- rep(0,nsim)
theta[1] <- theta0
set.seed(173)
for(i in 2:nsim){
       theta.p <- theta[i-1] + rnorm(1,0,d)
       ratio <- dnorm(theta0, theta.p, sigma)*
                 dnorm(theta.p, mu, tau) /
                (dnorm(theta0, theta[i-1], sigma)
                 *dnorm(theta[i-1], mu, tau))
  evento <- rbinom(1,1,min(ratio,1))

  theta[i] <- if(evento==1) theta.p else theta[i-1]
}
```

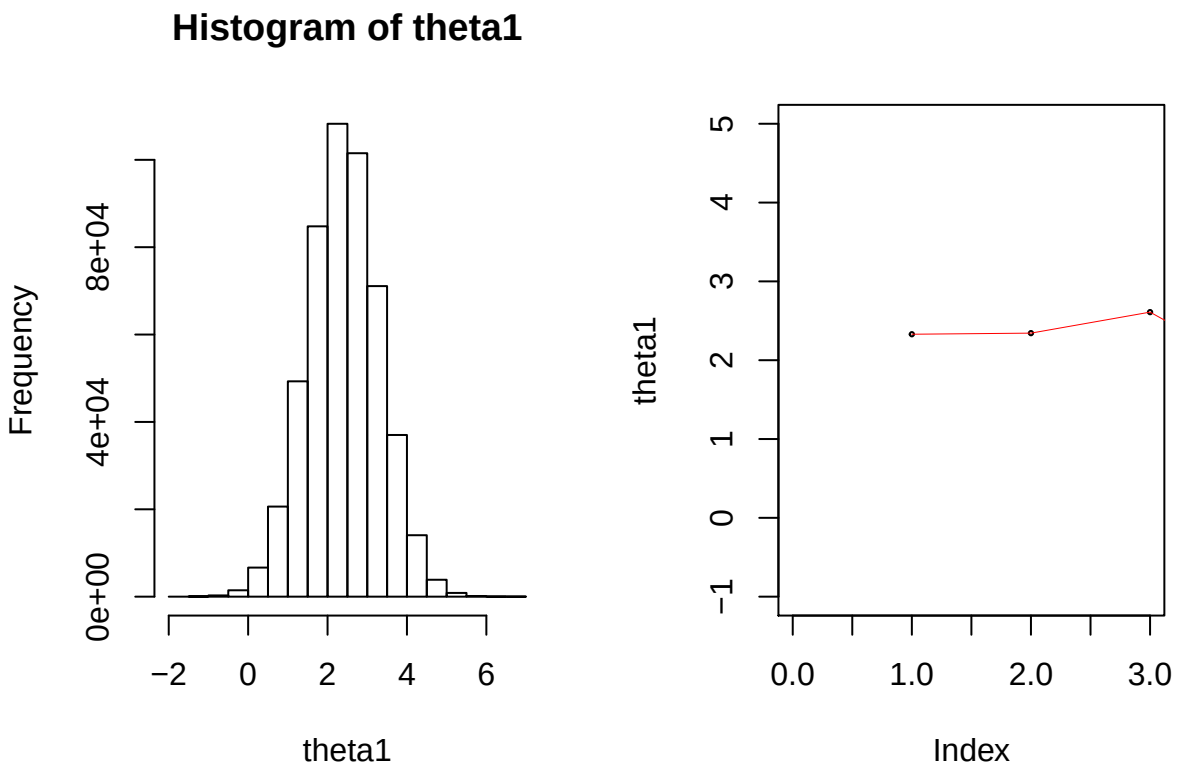
Tolgo metà simulazioni: 500k realizzazioni

```
theta1 <-theta[-(1:(nsim/2))]
summary(theta1)
```

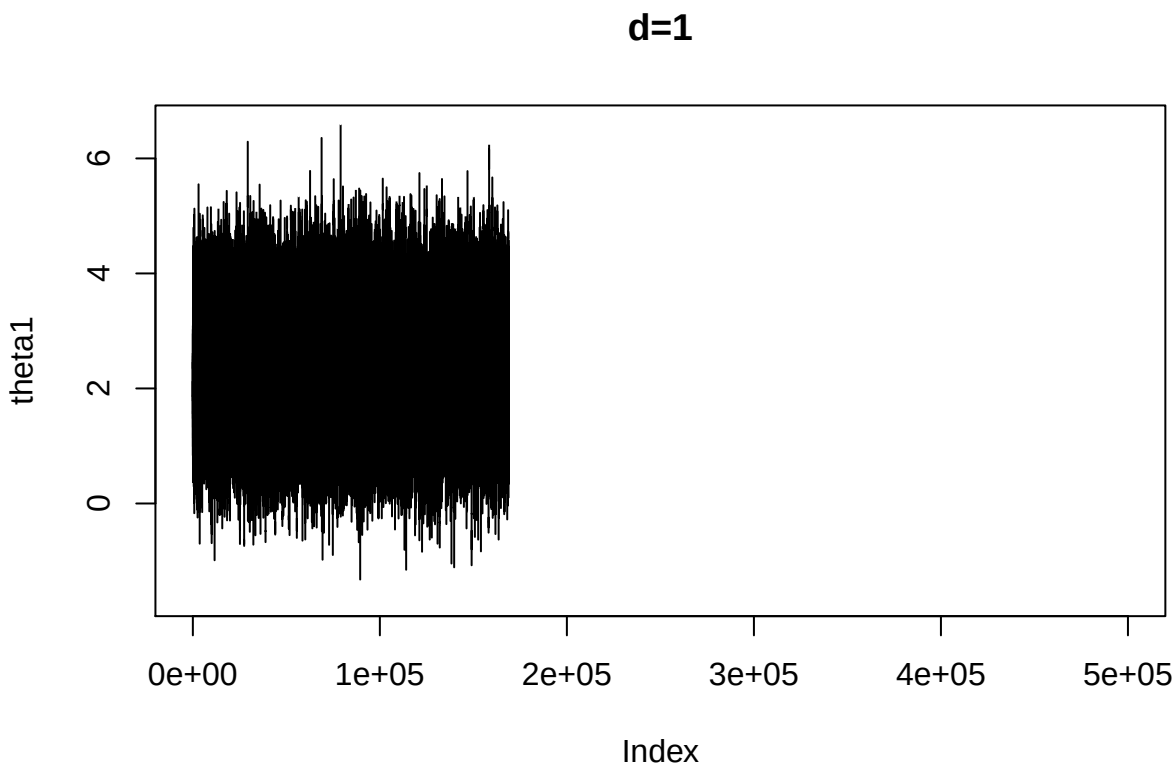
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.630   1.800   2.404   2.405   3.011   6.592
```

Grafici di diagnostica: mi aspetto che con così tante simulazioni abbia una distibuzione asintoticamente normale e simmetrica

```
par(mfrow=c(1,2))
hist(theta1)
plot(theta1,pch=1,cex=0.3,xlim=c(0,3),ylim=c(-1,5))
lines(theta1,lwd=0.5,col="red")
```




```
plot(theta1,type ='l', main = "d=1")
```



Questo grafico lo visualizziamo bene per esteso perchè serve per vedere la variabilità dei valori campionati della distribuzione approssimata a posteriori del parametro. Osservandolo vediamo che esso si muove attorno al valore medio di 2.405 per il parametro e con variabilità più o meno costante.

d=100 per la distribuzione del random walk

Cambiamo d=100 cioè aumentiamo la variabilità della distribuzione di supporto del random walk

```
theta0 <- 3
mu <- 0
tau <- 2
sigma <- 1
d <- 100
```

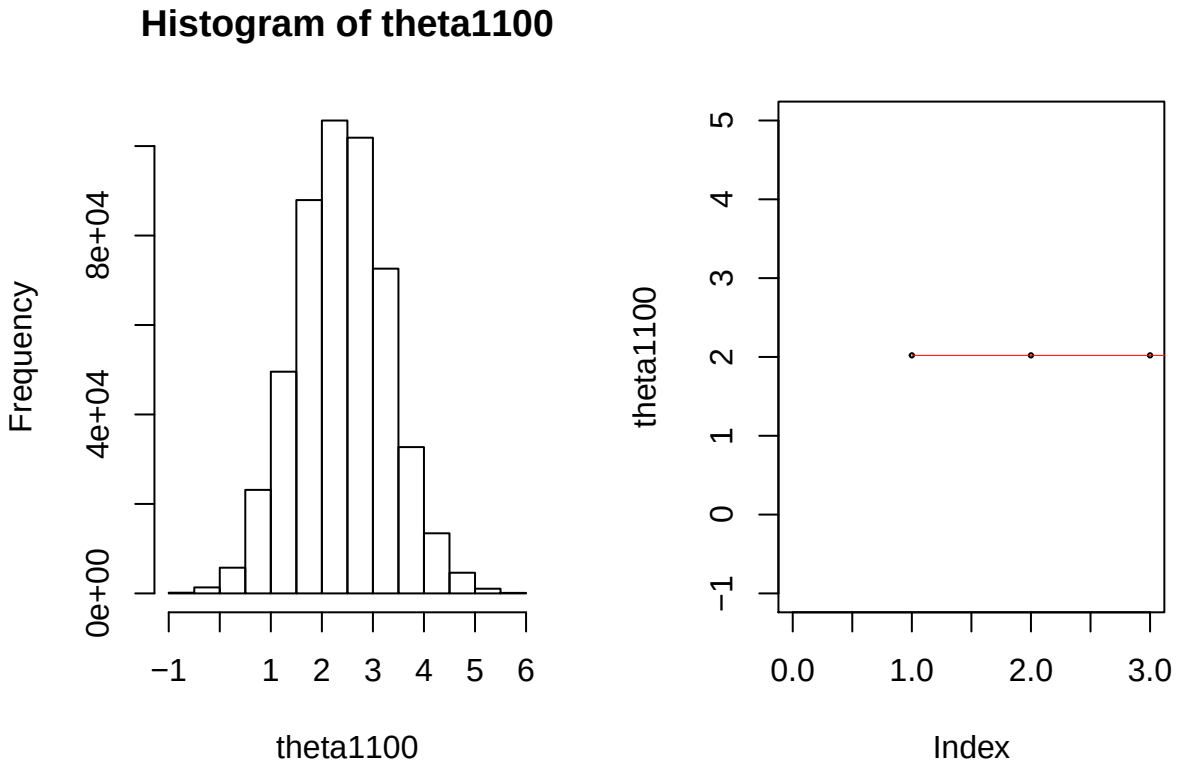
```
nsim  <- 10^6
theta100 <- rep(0,nsim)
theta100[1] <- theta0
set.seed(173)
for(i in 2:nsim){
       theta.p <- theta100[i-1] + rnorm(1,0,d)
       ratio <- dnorm(theta0, theta.p, sigma)*
                 dnorm(theta.p, mu, tau) /
                (dnorm(theta0, theta100[i-1], sigma)
                 *dnorm(theta100[i-1], mu, tau))
  evento <- rbinom(1,1,min(ratio,1))

  theta100[i] <- if(evento==1) theta.p else theta100[i-1]
```

```
}
```

```
theta1100 <-theta100[-(1:(nsim/2))]
summary(theta1100)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.8979  1.8007  2.3967  2.3916  2.9981  5.7746
```

```
par(mfrow=c(1,2))
hist(theta1100)
plot(theta1100,pch=1,cex=0.3,xlim=c(0,3),ylim=c(-1,5))
lines(theta1100,lwd=0.5,col="red")
```



Nel secondo ho una retta, per cui avendo una variabilità molto alta per il random walk, la catena non si muove molto, anzi rimane ferma per molti step rifiutando moltissimi valori. Per questo la distribuzione a posteriori simulata derivante non è da ritenersi molto significativa