Esercitazione 2

October 4, 2024

1 Esercitazione 2

Nota: Se ci sono dei parametri e non vi dico quanto valgono, potete usare qualsiasi valore voi vogliate

1.1 Importance Sampling

Assumete di avere

$$X \sim TN(\mu, \sigma^2, l, u)$$

dove $TN(\mu, \sigma^2, l, u)$ è una normale troncata, definita in $[l, u]$ con $l < u$, i.e., $X \in [l, u]$, con densità

$$f(x) = \frac{\phi(x|\mu, \sigma^2)}{\Phi(u|\mu, \sigma^2) - \Phi(l|\mu, \sigma^2)}$$

dove

$$\phi(x|\mu, \sigma^2)$$

è la densità di una normale di media μ e varianza σ^2 valutata in x, e

$$\Phi(y|\mu, \sigma^2)$$

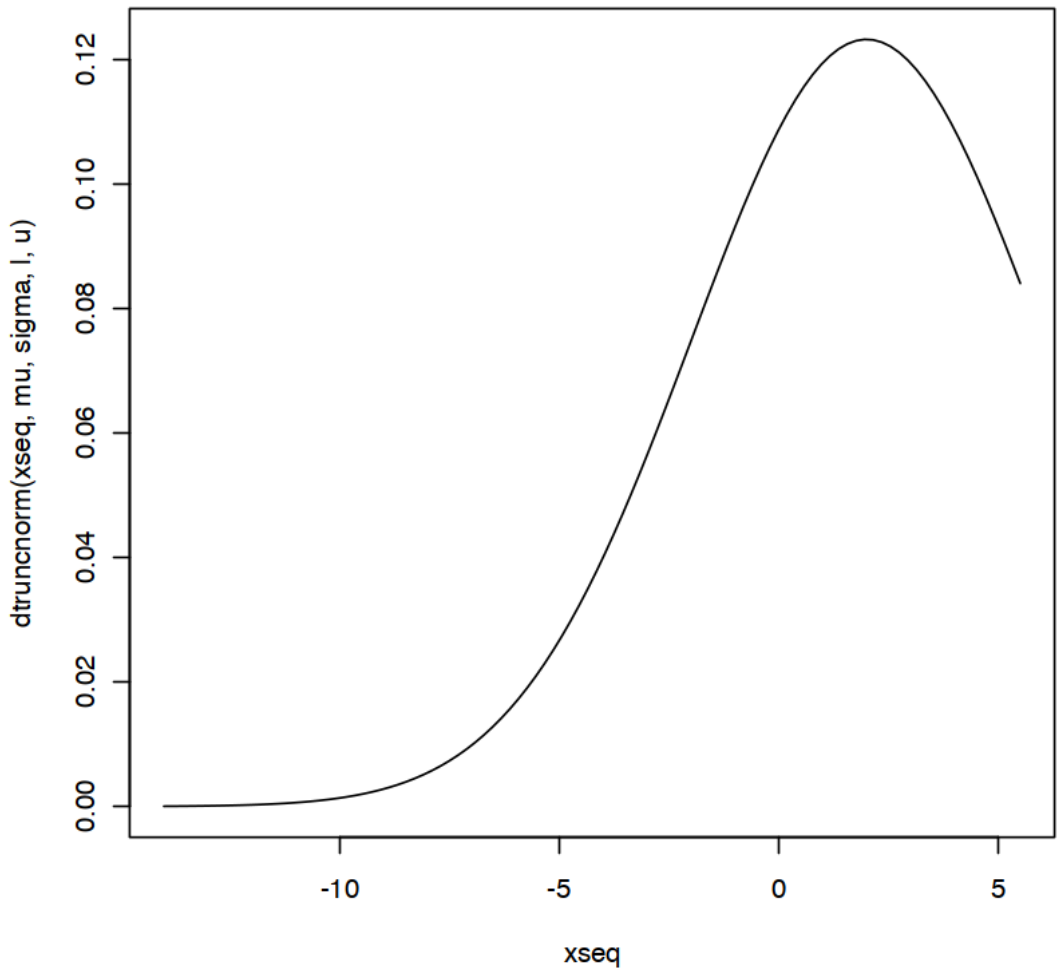
e la cumulata di una normale di media μ e varianza σ^2 valutata in y.

1. Scrivete una funzione che ne calcoli la densità, e poi mostratela graficamene
2. Usate Monte Carlo per valutare se la densità integra a 1
3. Utilizzate l'importance sampling per stimare il valore atteso di X, usando come densità "g()" un uniforme.
4. Utilizzate l'importance sampling per stimare il valore atteso di X, usando come densità "g()" una beta scalata con opportuni parametri.

La beta scalata è la distribuzione di una variabile $Z = (u - l)Y + l$, con $Y \sim B(a, b)$

[496]:
```
set.seed(1)
# punto 1
dtruncnorm = function(x, mu, sigma, l,u)
{
    pl = pnorm(l, mu,sigma)
    pu = pnorm(u, mu,sigma)

    return( dnorm(x,mu,sigma)/(pu-pl))
}
```

```r
# parametri
mu = 2
sigma=4
l = -14
u = 5.5
xseq = seq(l,u,length.out=100)

plot(xseq, dtruncnorm(xseq, mu, sigma, l,u), type="l", ylim=c(0, ␣
 ↪dtruncnorm(mu, mu, sigma, l,u)  ))
```



Per il punto 2 posso usare il fatto che il denominatore della densità è costante (se non cambio i parametri) e il numeratore non è mai maggiore di $\phi(\mu|\mu, \sigma^2)$ che è il punto massimo della densità

normale, quindi

$$M = \frac{\phi(x|\mu, \sigma^2)}{\Phi(u|\mu, \sigma^2) - \Phi(l|\mu, \sigma^2)} \geq f(x)$$

e l'integrale può essere calcolato come

$$\int_l^u \int_0^M 1_{(0,f(z)]}(w) dw dz = \int_l^u \int_0^M 1_{(0,f(z)]}(w) \frac{1}{f(z,w)} f(z,w) dw dz$$

dove $f(z, w)$ è una densità che posso scegliere, fin tanto che l'integrale non cambi. Posso per esempio definire

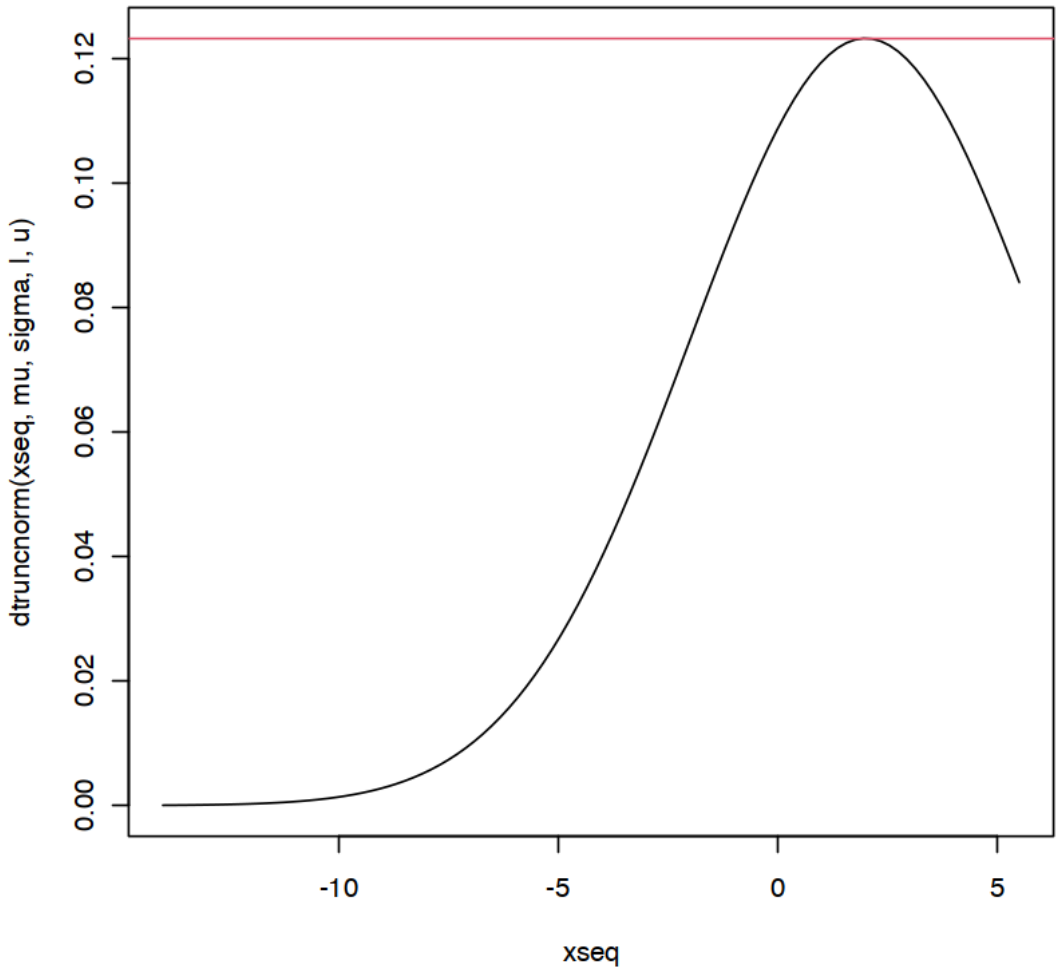
$$f(z, w) = f(z)f(w) = \frac{1}{(u-l)} \frac{1}{M}$$

assumendo indipendenza e che le due siano uniformi ($Z \sim U(l, u)$ $W \sim U(0, m)$). L'integrale è

$$\int_l^u \int_0^M 1_{(0,f(z)]}(w) \frac{1}{f(z,w)} f(z,w) dw dz \approx \frac{\sum_{i=1}^n 1_{(0,f(z_i)]}(w_i) M(b-a)}{n}$$

[497]:
```
# punto 2
M = dtruncnorm(mu, mu, sigma, l,u)

# verifico che M sia il valore massimo
plot(xseq, dtruncnorm(xseq, mu, sigma, l,u), type="l", ylim=c(0, ␣
 ↪dtruncnorm(mu, mu, sigma, l,u)  ))
abline(h = M, col=2)

# stimatore
nsim = 100000
z = runif(nsim, l,u)
w = runif(nsim, 0,M)

indicatrice  = (w <= dtruncnorm(z, mu, sigma, l,u))*1
#indicatrice
stimatore = sum(indicatrice*M*(u-l))/nsim
stimatore
```

1.00145407940228



per il 3 e il 4, voglio calcolare

$$E(X) = \int_l^u x f(x) dx = \int_l^u x \frac{f(x)}{g(x)} g(x) dx$$

approssimandolo con MC e dove g nel punto 3 è un'uniforme (tra [l,u]). Per conferma, il valore atteso della normale troncata è

$$E(X) = \mu + \frac{\phi(l|\mu,\sigma^2) - \phi(u|\mu,\sigma^2)}{\Phi(u|\mu,\sigma^2) - \Phi(l|\mu,\sigma^2)} \sigma^2$$

[498]:
```
# punto 3
nsim = 10000
xsample = runif(nsim, l,u)
```

```
stimatore = mean(  xsample*dtruncnorm(xsample, mu, sigma, l,u)/
 ↪dunif(xsample,l,u) )
stimatore
valore_vero = mu + (dnorm(l,mu,sigma) - dnorm(u,mu,sigma))/(pnorm(u, mu,sigma)␣
 ↪- pnorm(l, mu,sigma))*sigma^2
valore_vero
```

0.629572052534356

0.655820925940429

Per il punto 4 dovete calcolare la distribution di una beta scalata $Z = (u-l)Y + l$, con $Y \sim B(a,b)$. Abbiamo che

$$Y = \frac{Z-l}{u-l} = q(Z)$$

$$f_z(z) = f_Y(y(z))|q'(Z)| = f_Y(y(z))\frac{1}{u-l}$$

con $f_Y(y(z))$ la densità di una Beta.

Per l'esercizio dobbiamo calcolare l'integrale con

$$E(X) = \int_l^u x f(x)dx = \int_l^u x\frac{f(x)}{g(x)}g(x)dx$$

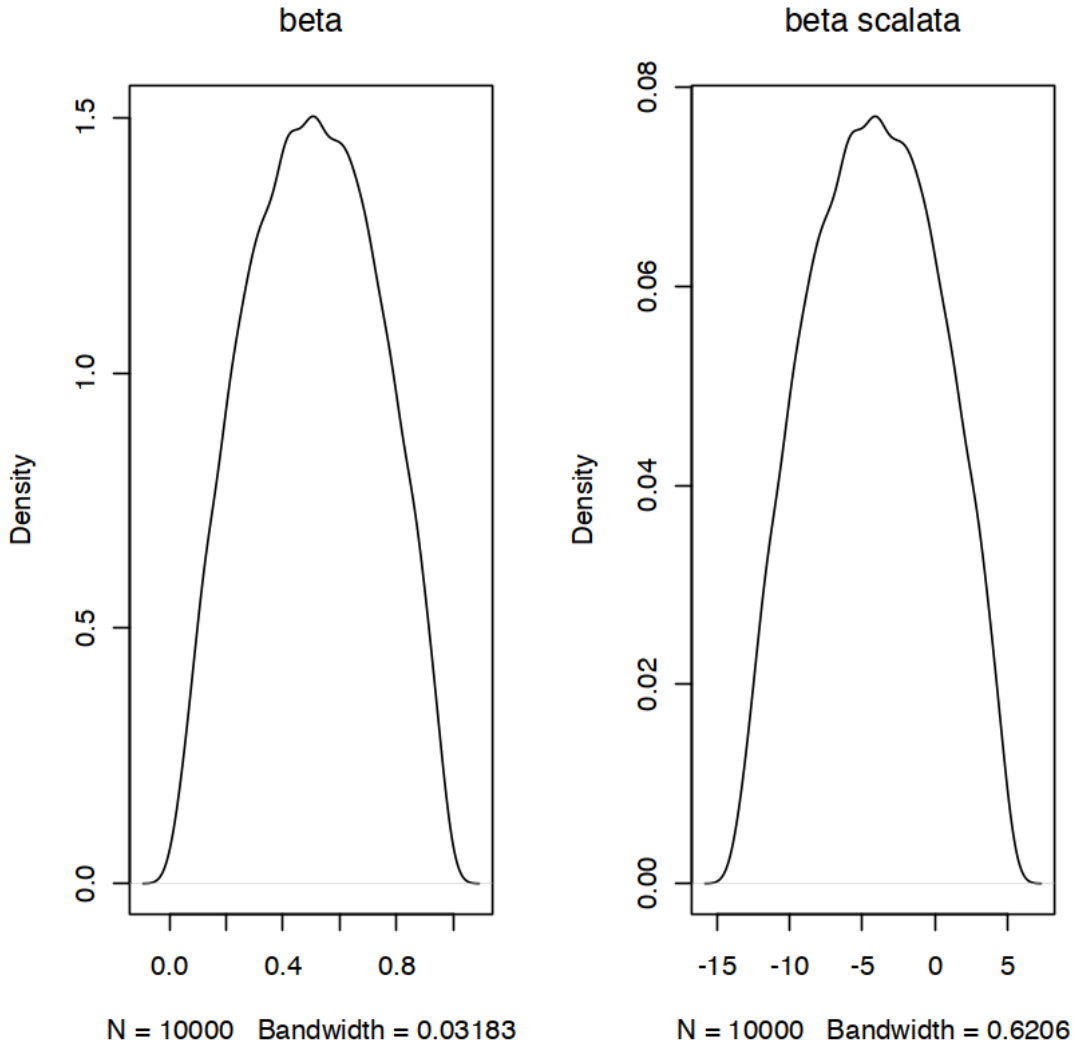
dove come $g(x)$ dobbiamo prendere la densità della beta scalata. La densità la sappiamo calcolare, ma dobbiamo anche simulare dalla beta scalata. Un campione X da una beta scalata si può ottenere come

1. simulo $Y_i \sim B(a,b)$
2. calcolo $X_i = (u-l)Y_i + l$

[499]:
```
## simulo dalla beta scalata
a = 2
b = 2
ysim = rbeta(nsim, a,b)
xsim = (u-l)*ysim +l
# vediamo i campioni della beta e della beta scalata
par(mfrow=c(1,2))
plot(density(ysim), main= "beta")
plot(density(xsim), main = "beta scalata")
par(mfrow=c(1,1))

stimatore = mean(  xsim*dtruncnorm(xsim, mu, sigma, l,u )/( dbeta(ysim,a,b)/
 ↪(u-l)  ) )
stimatore
valore_vero
```

0.727904389826087

0.655820925940429



1.2 Accept-Reject Sampling

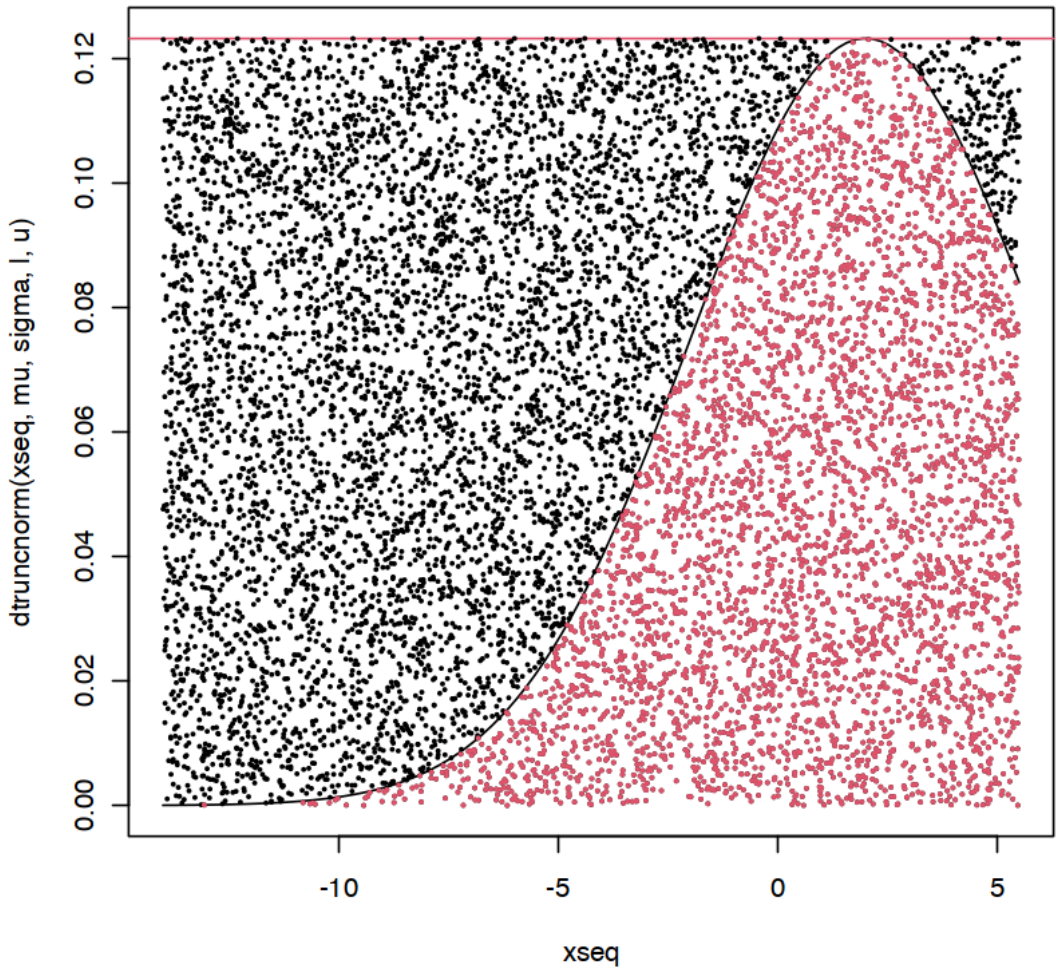
Assumete la stessa distribuzione del punto precedente

$$X \sim TN(\mu, \sigma^2, l, u)$$

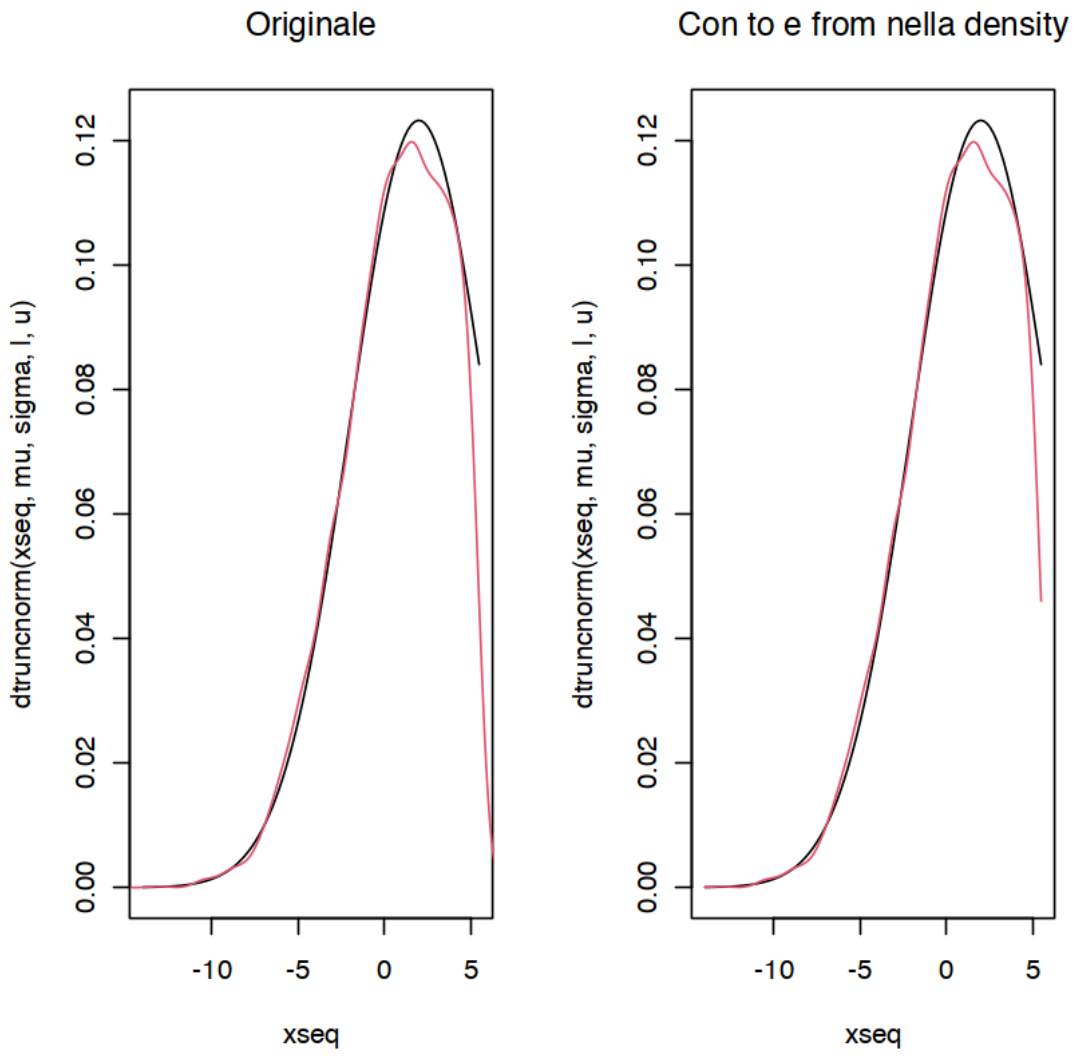
1. campionate dalla normale troncata usando l'algoritmo accept-reject. Disegnate i campioni di (y,u) colorati di nero, e i campioni (x,u) di nero.
2. verificare che i campioni seguono la distribuzione giusta
3. verificate quale è il valore del rate d'accettazione e che corrisponda a quello vero

[500]:
```
# punto 1
# Il punto 1 è quasi uguale a punto 2 del punto precedente
# definisco il massimo
```

```r
mu = 2
sigma=4
l = -14
u = 5.5
xseq = seq(l,u,length.out=100)
M = dtruncnorm(mu, mu, sigma, l,u)




# stimatore
nsim = 10000
y = runif(nsim, l,u)
u_y = runif(nsim, 0,M)

indicatrice  = (u_y <= dtruncnorm(y, mu, sigma, l,u))
#tengo solo i campioni di
x = y[indicatrice]
u_x = u_y[indicatrice]
## plotto i risultati
plot(xseq, dtruncnorm(xseq, mu, sigma, l,u), type="l", ylim=c(0, ␣
 ↪dtruncnorm(mu, mu, sigma, l,u)  ))
abline(h = M, col=2)
points(y,u_y, col=1, pch=20, cex=0.2)
points(x,u_x, col=2, pch=20, cex=0.2)
```



[501]:
```
# punto 2
par(mfrow=c(1,2))
plot(xseq, dtruncnorm(xseq, mu, sigma, l,u), type="l", ylim=c(0, ␣
 ↪dtruncnorm(mu, mu, sigma, l,u)  ), main="Originale")
lines(density(x), col=2)

plot(xseq, dtruncnorm(xseq, mu, sigma, l,u), type="l", ylim=c(0, ␣
 ↪dtruncnorm(mu, mu, sigma, l,u)  ), main="Con to e from nella density")
lines(density(x,from = l, to =u), col=2)
par(mfrow=c(1,1))
```



1.3 Marginale e congiunta

Assumiamo

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}^A \\ \mathbf{X}^B \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_A \\ \mu_B \end{pmatrix}, \begin{pmatrix} \Sigma_A & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_B \end{pmatrix} \right)$$

dove $\mathbf{X}^A$ e $\mathbf{X}^B$ sono vettori di dimensione 2. Definiamo

$$\mu = \begin{pmatrix} \mu_A \\ \mu_B \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma_A & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_B \end{pmatrix}$$

Sappiamo che

$$\mathbf{X}^A | \mathbf{X}^B \sim N_2(\mathbf{M}, \mathbf{V})$$

con

$$\mathbf{M} = \mu_A + \Sigma_{AB}\Sigma_B^{-1}(\mathbf{X}^B - \mu_B)$$

e

$$\mathbf{V} = \Sigma_A - \Sigma_{AB}\Sigma_B^{-1}\Sigma_{BA}$$

1. Simulare n campioni dalla normale multivariata
2. Simulare n campioni dalla marginale di $\mathbf{X}_A$
3. Simulare n campioni dalla normale multivariata simulando prima campioni dalla marginale di $\mathbf{X}_B$ e poi dalla condizionata di $\mathbf{X}_A|\mathbf{X}_B$
4. Far vedere che per tutte e 3 i punti precedenti i campioni di $\mathbf{X}_A$ provengono sempre dalla marginale

P.S.1 Per il punto 4 potete

1. usare la funzione smoothScatter (che è la stima di densità per una coppia di variabili),
2. oppure far vedere che la probabilità che $\mathbf{X}^A \in \mathbf{A}$ sia simile per i 3 metodi

$$P(\mathbf{X}^A \in \mathbf{A}) = \int_{\mathbb{R}^2} 1_{\mathbf{A}}(\mathbf{x}^A) f(\mathbf{x}^A) d\mathbf{x}^A$$

3. oppure calcolare la comulata

$$P(\mathbf{X}^A \in (-\inf, b] \times (-\inf, d])$$

   in un set di valori b e d equispaziati e usare il comando image

P.S.2 Mentre per μ potete prendere i valori che volete, Σ va scelta con cura visto che è una matrice simmetrica e definita positiva. Potete ottenrere una Σ valida definendo una matrice

$$\mathbf{D}$$

di dimensione 4 a rango pieno, e

$$\Sigma = \mathbf{D}\mathbf{D}^T$$

Oppure potete ottenere un campione da una $IW(\nu, \Psi)$ (Inverse Wishart) che è una distribuzione per matrici di covarianza usando il pacchecctto MCMCpack e la funzione riwish. Nell'inverse wishart $\nu \geq 4$ (dimensione di Σ) e Ψ è una matrice delle stesse dimensioni di Σ definita positiva (potete usare l'indentità)

P.S.3 se

$$\Sigma^* = \mathbf{C}\mathbf{C}^T$$

e

$$\mathbf{Y} \sim N_d(\mathbf{0}_d, \mathbf{I}_d)$$

allora

$$\mu^* + \mathbf{C}\mathbf{Y} \sim N_d(\mu^*, \Sigma^*)$$

[502]:
```
library(MCMCpack)
## parametri
mu = matrix(c(1,2,3,2), ncol=1)
sigma = riwish(4,diag(1,4))
```

```r
print("matrix Sigma")
sigma

# Punto 1
nsim = 10000
## uso cholesky per trovare C

matrixC = t(chol(sigma))
print("matrix Sigma calcolata da cholesky")
matrixC%*%t(matrixC)


xsim  = matrix(NA, ncol=4, nrow = nsim)
for(i in 1:nsim)
{
    y1 = rnorm(4,0,1)
    x1 = mu + matrixC%*%y1
    xsim[i, ] = x1
}

# prendo i campio di x_1
xa_met1 = xsim[,1:2]
smoothScatter(xa_met1)
```

```
[1] "matrix Sigma"
```

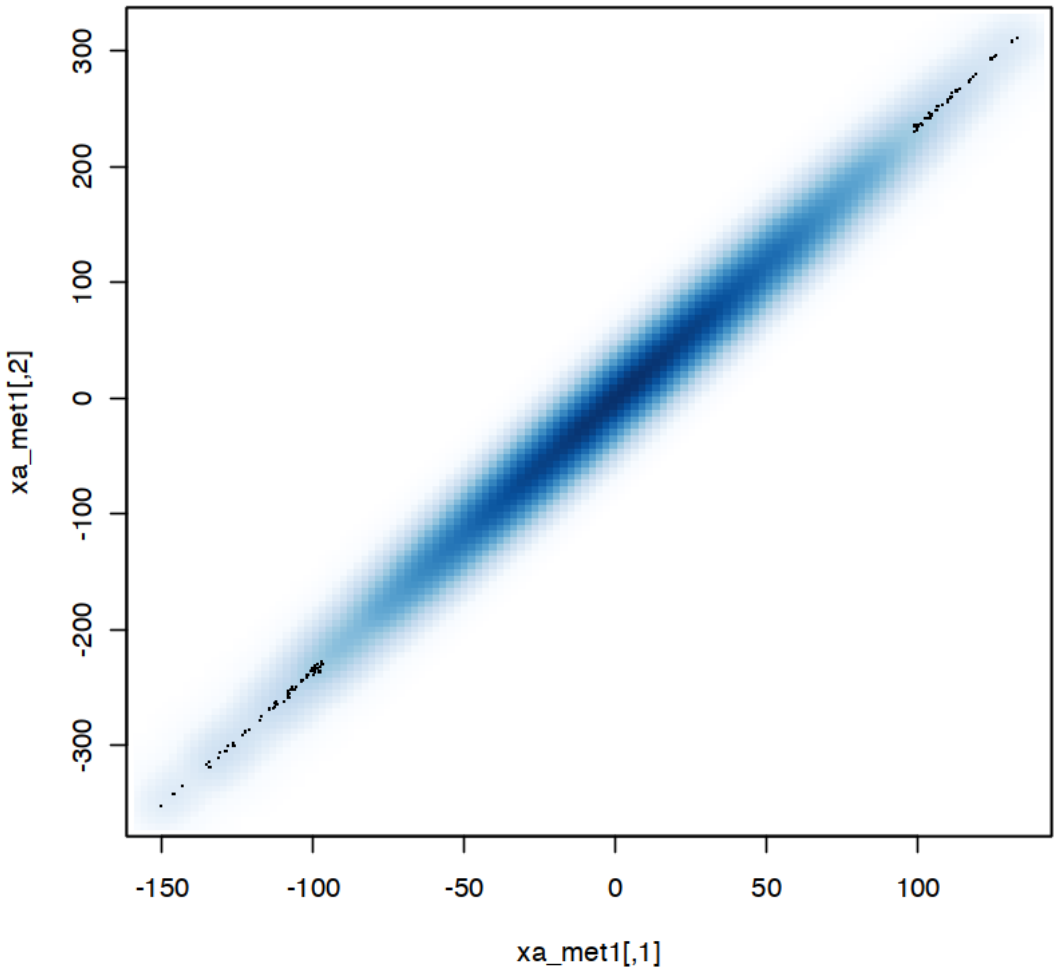
A matrix: 4 x 4 of type dbl

1505.968	3538.501	1901.286	-4978.679
3538.501	8317.481	4469.544	-11701.893
1901.286	4469.544	2402.052	-6288.033
-4978.679	-11701.893	-6288.033	16463.945

```
[1] "matrix Sigma calcolata da cholesky"
```

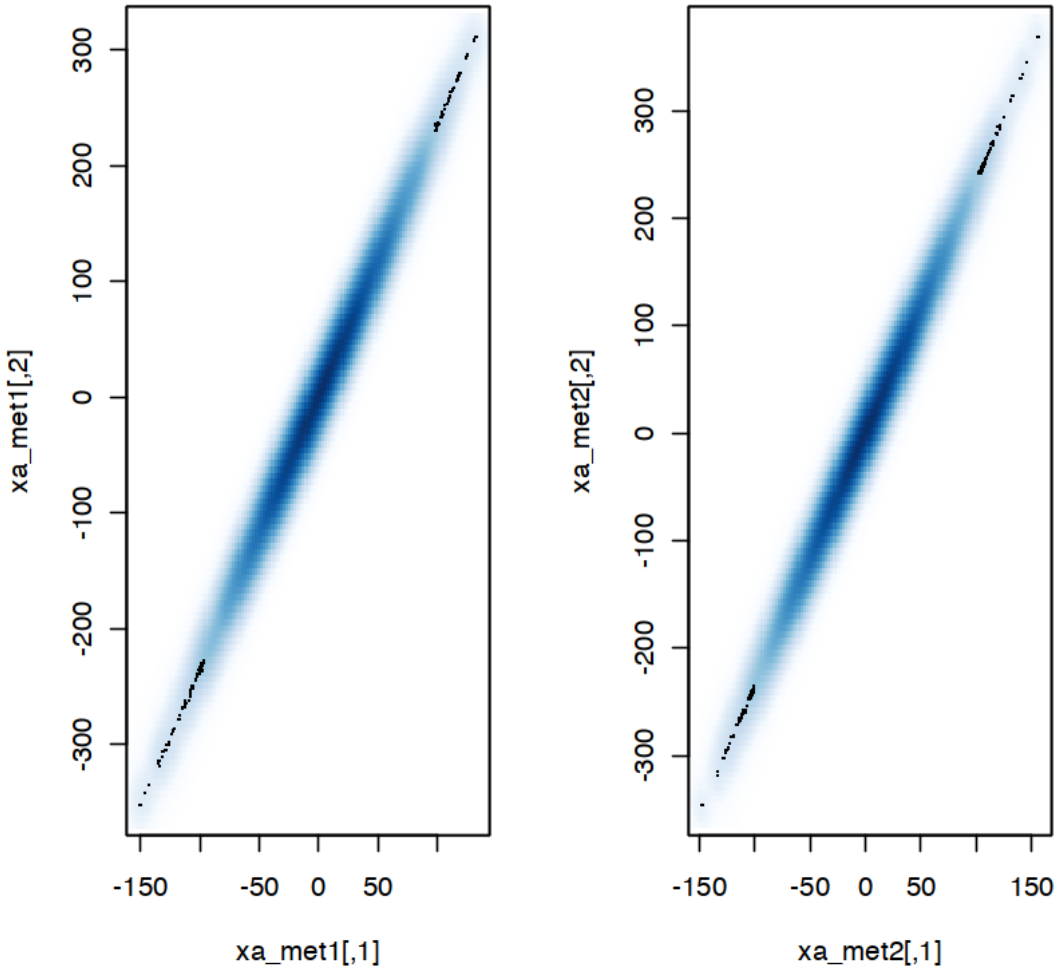
A matrix: 4 x 4 of type dbl

1505.968	3538.501	1901.286	-4978.679
3538.501	8317.481	4469.544	-11701.893
1901.286	4469.544	2402.052	-6288.033
-4978.679	-11701.893	-6288.033	16463.945



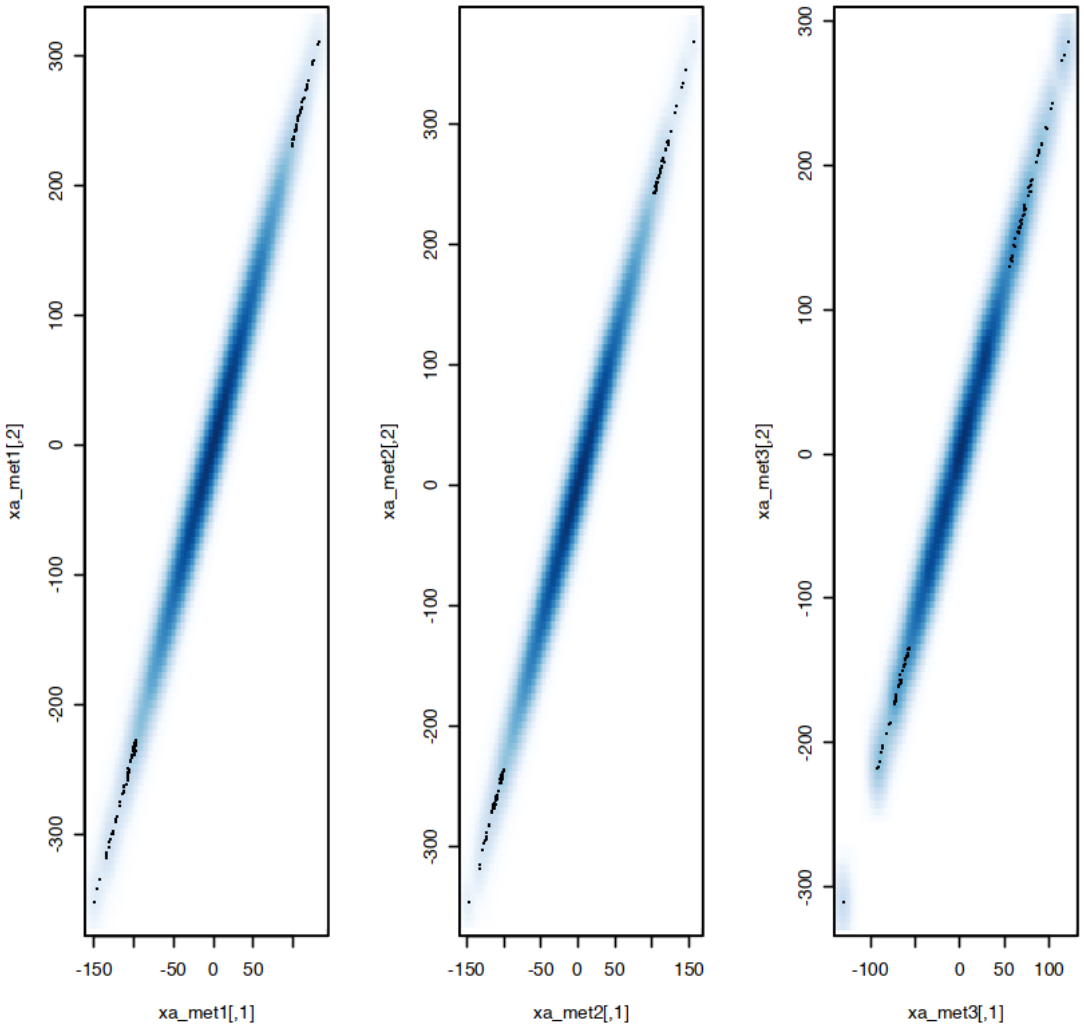
[503]:
```
# punto 2
xa_met2 = matrix(NA, nrow = nsim, ncol=2)
mua = mu[1:2]
sigmaa = sigma[1:2,1:2]
matrixCa = t(chol(sigmaa))

for(i in 1:nsim)
{
    y2 = rnorm(2,0,1)
    x2 = mua + matrixCa%*%y2
    xa_met2[i,] = x2
}
par(mfrow=c(1,2))
```

```
smoothScatter(xa_met1)
smoothScatter(xa_met2)
```



[528]:
```
# punto 3
xa_met3 = matrix(NA, nrow = nsim, ncol=2)
mua = mu[1:2]
mub = mu[3:4]

sigmaa = sigma[1:2,1:2]
sigmaab = sigma[1:2,3:4]
sigmab = sigma[3:4,3:4]

matrixCb = t(chol(sigmab))
```

```
sigmab_inv = solve(sigmab)
# varianza condizionata
matrix_var_cond =  sigmaa - sigmaab%*%sigmab_inv%*%t(sigmaab)
matrixC_var_cond = t(chol(matrix_var_cond))



for(i in 1:nsim)
{
    # marginale di b
    y3 = rnorm(2,0,1)
    xb = mub + matrixCb%*%y3

    mean_cond = mua +  sigmaab%*%sigmab_inv%*%(xb - mub)
    y4 = rnorm(2,0,1)
    xa = mean_cond + matrixC_var_cond%*%y4
    xa_met3[i,] = xa
}

par(mfrow=c(1,3))
smoothScatter(xa_met1)
smoothScatter(xa_met2)
smoothScatter(xa_met3)
```



[505]:
```
# stimo la P(X^A \in A) per diversi con $A = [a,b]\times [c,d]$

a = -3
b = 5
c = -20
d = 15
indicatrice1 = rep(NA, nsim)
indicatrice2 = rep(NA, nsim)
indicatrice3 = rep(NA, nsim)
for(i in 1:nsim)
{
    indicatrice1[i] = ( (xa_met1[i, 1]>=a) & (xa_met1[i, 1] <= b)  ) &␣
 ↪((xa_met1[i, 2]>=c) & (xa_met1[i, 2]<= d)  )
```

```
    indicatrice2[i] = ( (xa_met2[i, 1]>=a) & (xa_met2[i, 1]<= b)  ) &␣
 ↪((xa_met2[i, 2]>=c) & (xa_met2[i, 2]<= d)  )
    indicatrice3[i] = ( (xa_met3[i, 1]>=a) & (xa_met3[i, 1]<= b)  ) &␣
 ↪((xa_met3[i, 2]>=c) & (xa_met3[i, 2]<= d)  )
}
mean(indicatrice1)
mean(indicatrice2)
mean(indicatrice3)
```

0.087

0.0822

0.0828

Se prendiamo

$$a = c = -\infty$$

allora

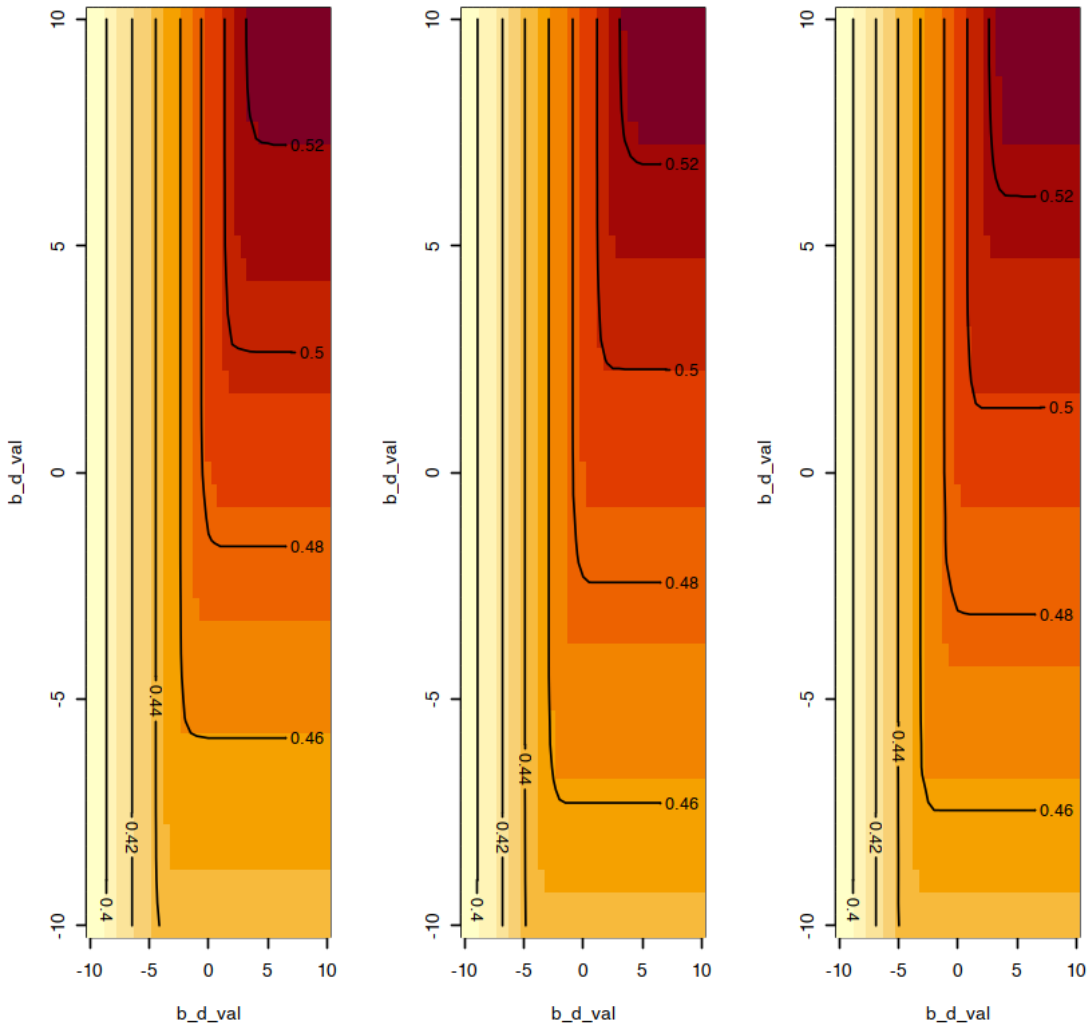
$$P(X^A \in A)$$

diventa la cumulata. Se scelgo diversi valori di b e d, che siano equispaziati, posso plottare la comulata con il comando image

[506]:
```
# scelgo i set di valori b e d
b_d_val = seq(-10,10, by = 0.5)
nval = length(b_d_val)

### matrix_cum sono le matrici che contengono la comulata
### la riga j-esima e columma r-esima contiene la comulata calcolata con b =␣
 ↪b_d_val[j] e d = b_d_val[r]
##
matrix_cum1 = matrix(NA, ncol= nval, nrow = nval)
matrix_cum2 = matrix(NA, ncol= nval, nrow = nval)
matrix_cum3 = matrix(NA, ncol= nval, nrow = nval)
a = -Inf
c = -Inf
for(j in 1:nval)
{
    b = b_d_val[j]
    for(r in 1:nval)
    {
        d = b_d_val[r]

        matrix_cum1[j,r] = mean(( (xa_met1[, 1]>=a) & (xa_met1[, 1] <= b)  ) &␣
 ↪((xa_met1[, 2]>=c) & (xa_met1[, 2]<= d)  ))
        matrix_cum2[j,r] = mean( ((xa_met2[, 1]>=a) & (xa_met2[, 1]<= b)  ) &␣
 ↪((xa_met2[, 2]>=c) & (xa_met2[, 2]<= d) ) )
        matrix_cum3[j,r] = mean(( (xa_met3[, 1]>=a) & (xa_met3[, 1]<= b)  ) &␣
 ↪((xa_met3[, 2]>=c) & (xa_met3[, 2]<= d)  ))
```

```
    }
}
```

[532]:
```
par(mfrow=c(1,3))
image(matrix_cum1, x = b_d_val, y = b_d_val)
contour(matrix_cum1, add=T, x = b_d_val, y = b_d_val)
image(matrix_cum2, x = b_d_val, y = b_d_val)
contour(matrix_cum2, add=T, x = b_d_val, y = b_d_val)
image(matrix_cum3, x = b_d_val, y = b_d_val)
contour(matrix_cum3, add=T, x = b_d_val, y = b_d_val)
par(mfrow=c(1,1))
```



1.4 Inversa Generalizzata e Discretizzazioni di variabili continue

Definite

$$c = \frac{1}{m}$$

e assumete di avere un variabile

$$X \in \left\{ \frac{0}{m} + \frac{c}{2}, \frac{1}{m} + \frac{c}{2}, \frac{2}{m} + \frac{c}{2}, \dots, \frac{m-1}{m} + \frac{c}{2} \right\} \equiv \mathcal{X}$$

discreta, che assume quindi m possibili valori. Definiamo la probabilità di X come

$$P\left(X = \frac{j}{m} + \frac{c}{2}\right) \propto f_{beta}\left(\frac{j}{m} + \frac{c}{2} | a, b\right) c$$

dove

$$f_{beta}\left(\frac{j}{m} + \frac{c}{2} | a, b\right)$$

è la densità di una beta di parametro (a, b) calcolata in $\frac{j}{m} + \frac{c}{2}$. Visto che le probabilità devono sommare a uno, devo avere per forza che

$$P\left(X = \frac{j}{m} + \frac{c}{2}\right) = \frac{f_{beta}\left(\frac{j}{m} + \frac{c}{2} | a, b\right)}{\sum_{h=0}^{m-1} f_{beta}\left(\frac{h}{m} + \frac{c}{2} | a, b\right)}$$

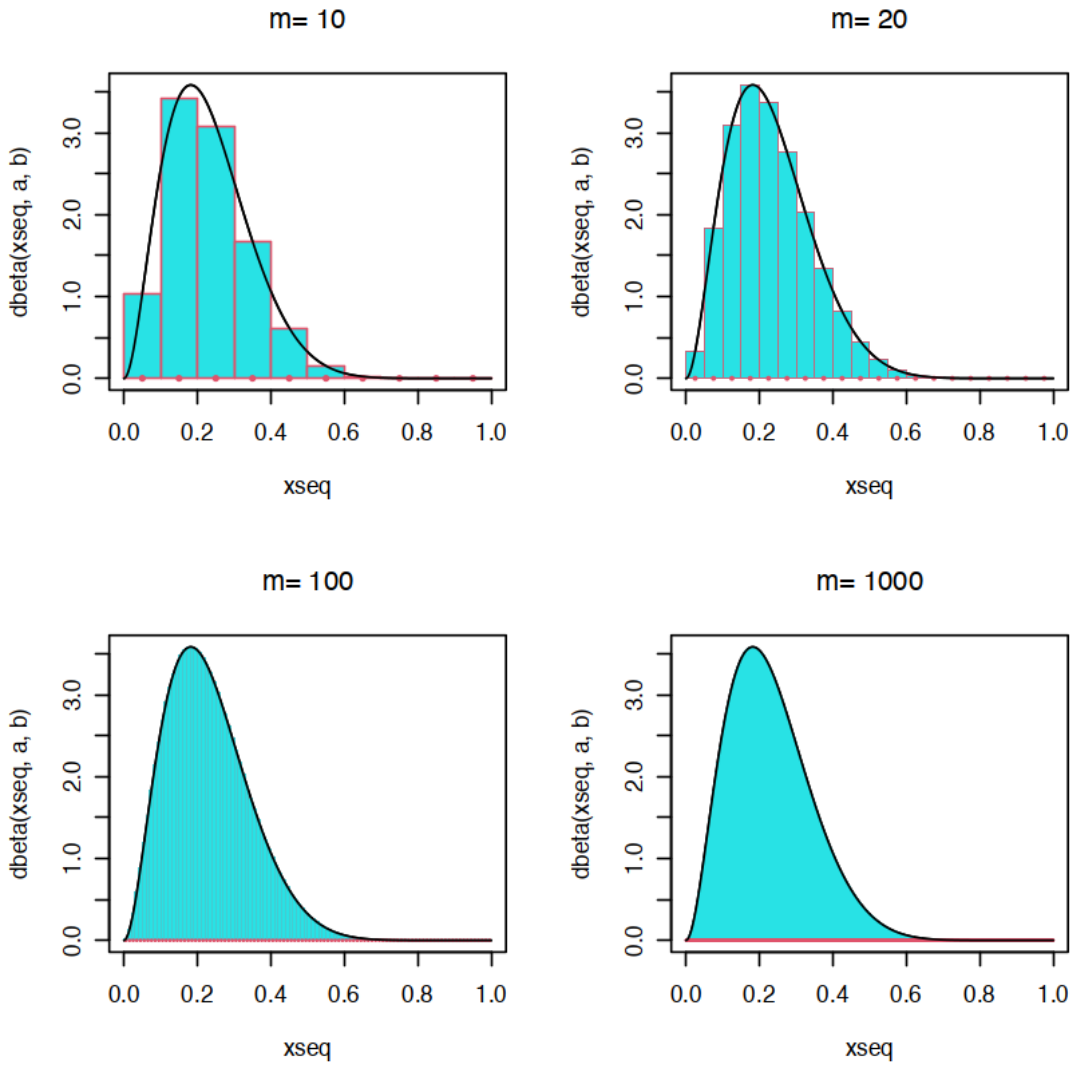
Queto è un modo per discretizzare uan variabile continua. Potete vedere più nel dettaglio cosa stiamo facendo nella figura sottostante in cui

1. La linea nera è la densità di una beta
2. I punti rossi sono i valori di $\mathcal{X}$
3. L'area di ogni rettangolo è $f(x)$ calcolata nel punto rosso associato.

Provate ad aumentare m e vedrete che la versione discreta è simile alla continua

[581]:
```
a = 3
b = 10
par(mfrow=c(2,2))
for(m in c(10,20,100,1000))
{
    c = 1/m
    xdom = c(0:(m-1))/m +c/2


    x_start = xdom - c/2
    x_end   = xdom + c/2


    prob_x = dbeta(xdom  ,a,b)
    prob_x = prob_x/sum(prob_x)
    altezza = prob_x/c
    xseq = seq(0,1, by = 0.001)
```

```
    plot(xseq, dbeta(xseq,a,b), type="l", main=paste("m=", m), lwd=1)
    rect(x_start, rep(0, m),x_end,altezza , border=2, col=5, lwd= c*10)
    points(xdom, rep(0, m), pch=20, cex= c*5, col=2)
    lines(xseq, dbeta(xseq,a,b),  col=1, lwd=1)
}
par(mfrow=c(1,1))
```



1. Disegnate la cumulata della beta, e della X nello stesso grafico (provate diversi valori di m, per esempio m = 10, m = 50, m = 1000) e notate le differenze.
2. Usando il metodo dell'uniforme e l'inversa generalizzata, simulate dalla distribuzione di X, con m piccolo (e.g $m = 5$), e verificate, tramite monte carlo, che il valore di

$$P\left(X \leq \frac{j}{m} + \frac{c}{2}\right)$$

sia effettivamente

$$\sum_{l=0}^{j}\left(\frac{f_{beta}\left(\frac{l}{m}+\frac{c}{2}|a,b\right)}{\sum_{h=0}^{m-1}f_{beta}\left(\frac{h}{m}+\frac{c}{2}|a,b\right)}\right)$$

3. fate la stessa cosa del punto 2, ma simulate usando direttamente la distribuzione di X
4. Per $m = 5$, $m = 50$ e $m = 1000$ calcolare il quantile a livello 0.1, 0.2, 0.3, ..., 0.8, 0.9, e confrontateli con quelli della beta agli stessi livelli

Per il punto 2, dovete calcolare con monte carlo

$$\int_{x} 1_{(-\infty,\frac{j}{m}+\frac{c}{2}]}(x)f(x)d\lambda(x)$$

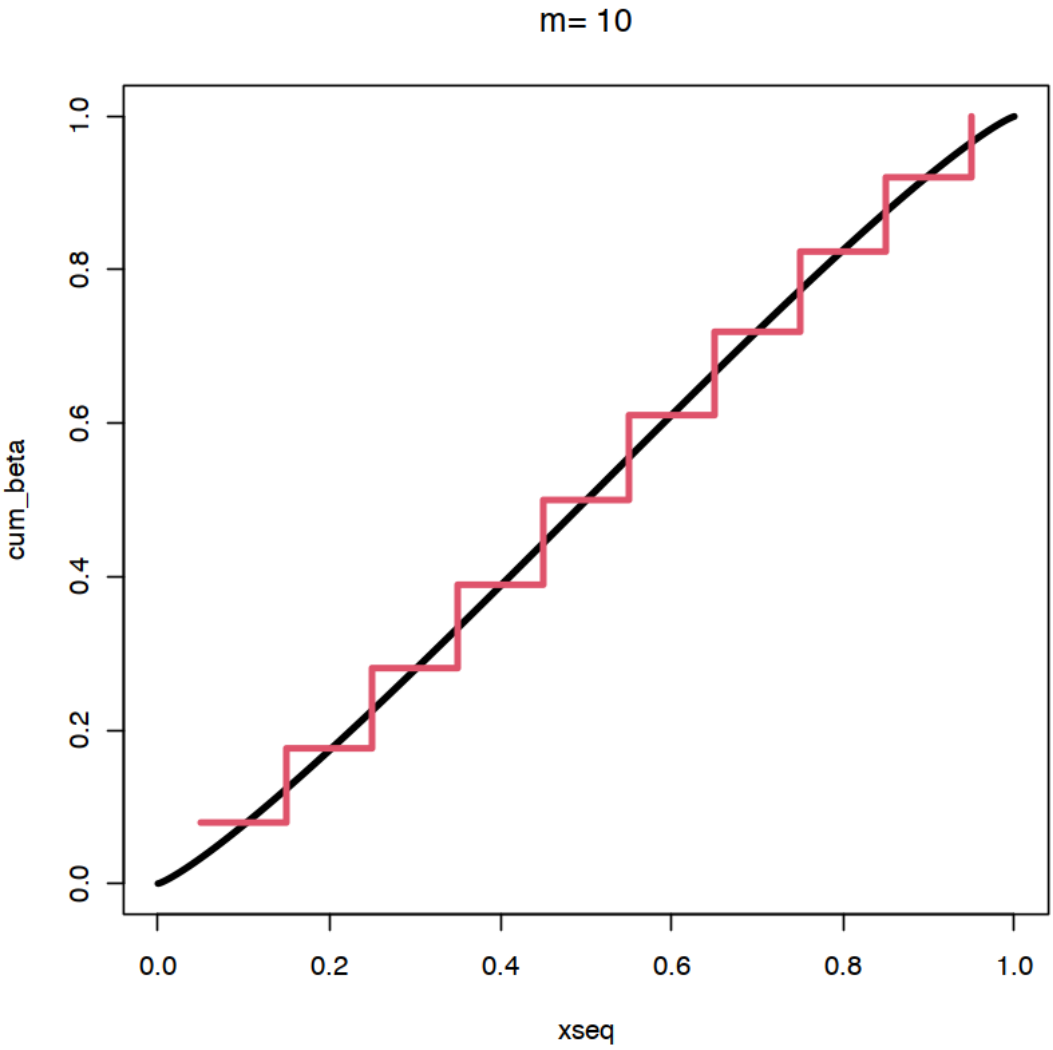
e confrontarlo con $P\left(X \leq \frac{j}{m}\right)$ per tutti i valori di j ammissibili. I campioni da $f(x)$ devono essere ottenuti con il metodo dell'inversa generalizzata

[509]:
```
# punto 1
# calcolo la densità della beta in dei valori fitti di x

a = 1.2
b = 1.2
xseq = seq(0,1, by = 0.001)
cum_beta = pbeta(xseq, a,b)


# definisco i valori del dominio di x
m = 10
c = 1/m
xdom = c(0:(m-1))/m +c/2
# calcolo i valori di P(X = x)
prob_x = dbeta(xdom  ,a,b)
prob_x = prob_x/sum(prob_x)
sum(prob_x)
cum_x1 = cumsum(prob_x)

plot(xseq, cum_beta, type="s", main=paste("m=", m), lwd=3)
lines(xdom,cum_x1, pch=20, cex=2, col=2 ,type="s", lwd=3)


# definisco i valori del dominio di x
m = 50
```

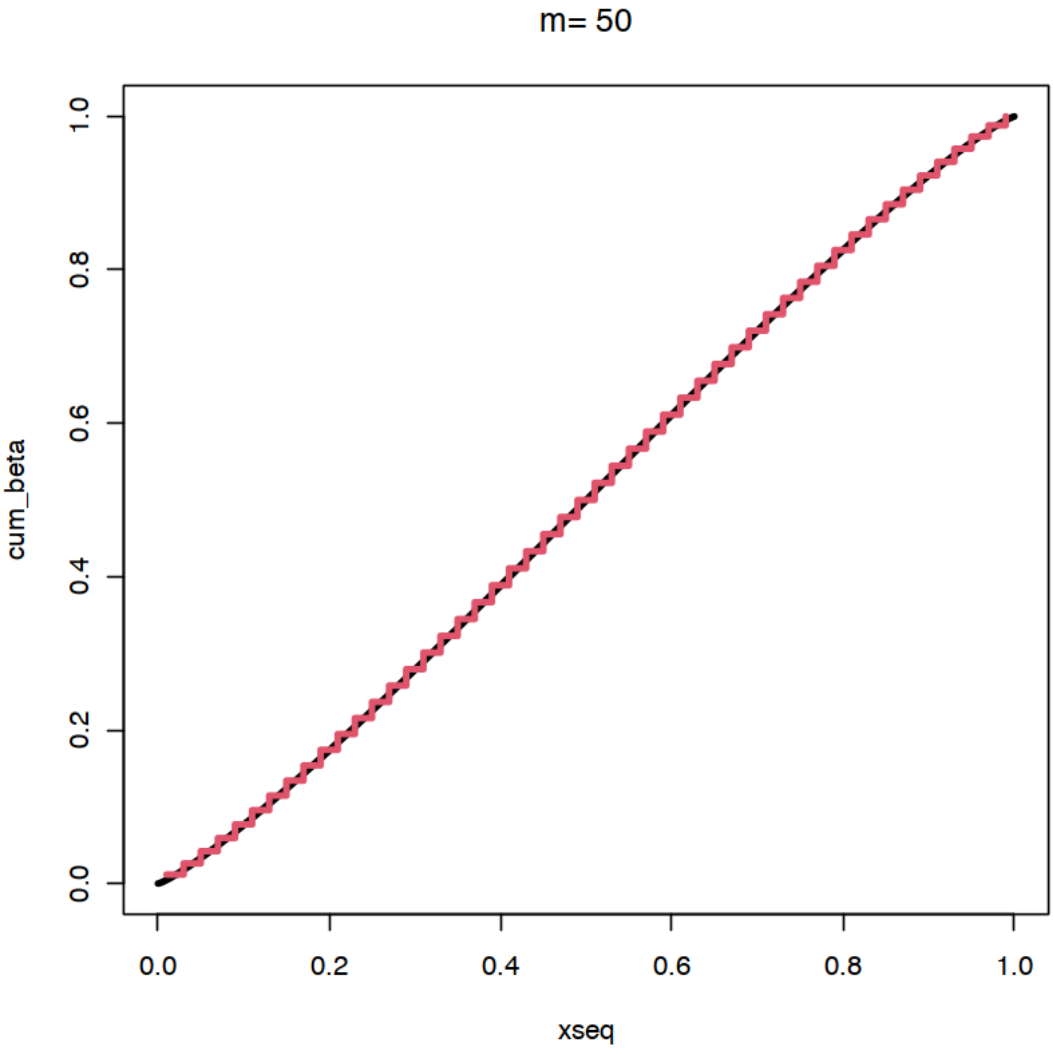
```r
c = 1/m
xdom = c(0:(m-1))/m+c/2
# calcolo i valori di P(X = x)
prob_x = dbeta(xdom  ,a,b)
prob_x = prob_x/sum(prob_x)
sum(prob_x)
cum_x1 = cumsum(prob_x)

plot(xseq, cum_beta, type="s", main=paste("m=", m), lwd=3)
lines(xdom,cum_x1, pch=20, cex=2, col=2 ,type="s", lwd=3)



# definisco i valori del dominio di x
m = 1000
c = 1/m
xdom = c(0:(m-1))/m+c/2
# calcolo i valori di P(X = x)
prob_x = dbeta(xdom  ,a,b)
prob_x = prob_x/sum(prob_x)
sum(prob_x)
cum_x1 = cumsum(prob_x)

plot(xseq, cum_beta, type="s", main=paste("m=", m), lwd=3)
lines(xdom,cum_x1, pch=20, cex=2, col=2 ,type="s", lwd=3)
```

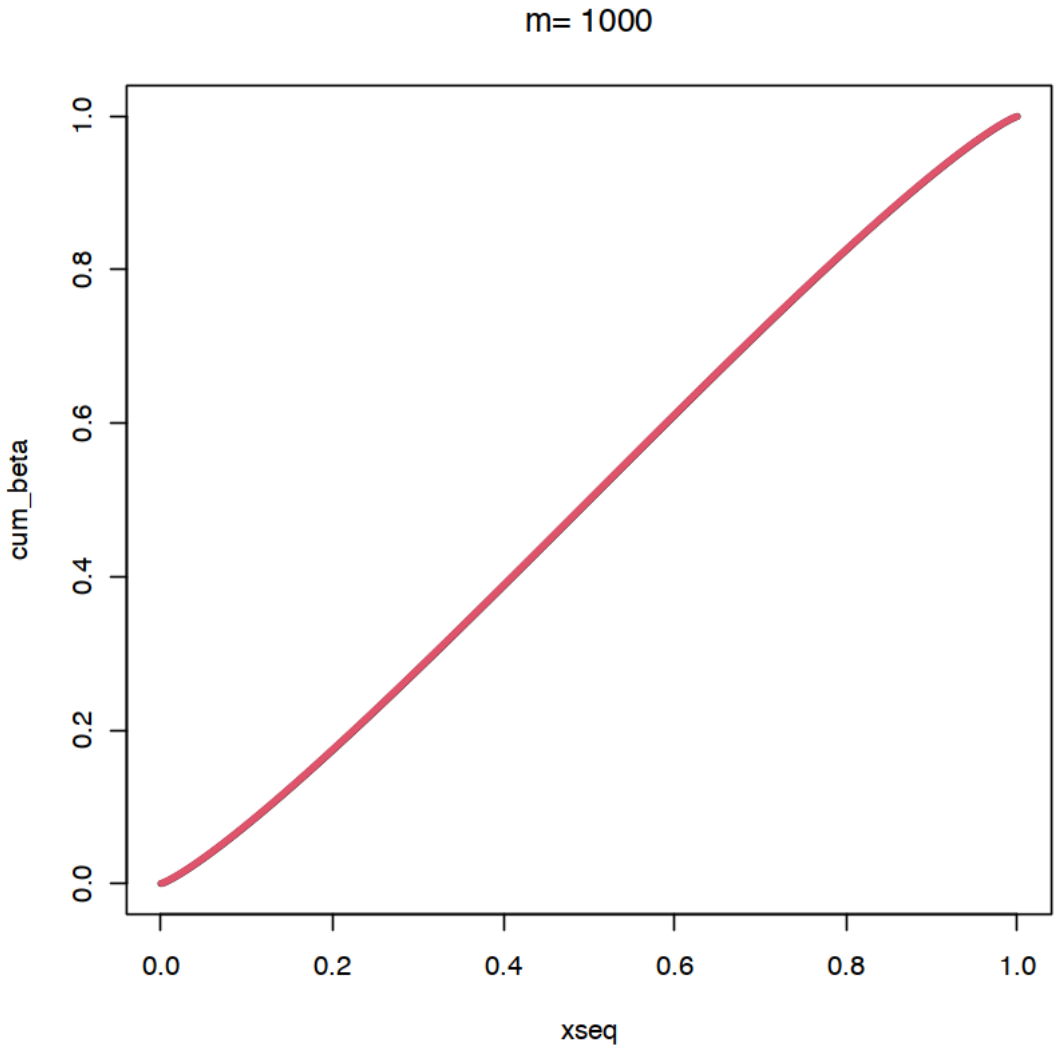
1

1



0.999999999999999


m= 50
cum_beta
0.0
0.2
0.4
0.6
0.8
1.0
xseq
0.0
0.2
0.4
0.6
0.8
1.0

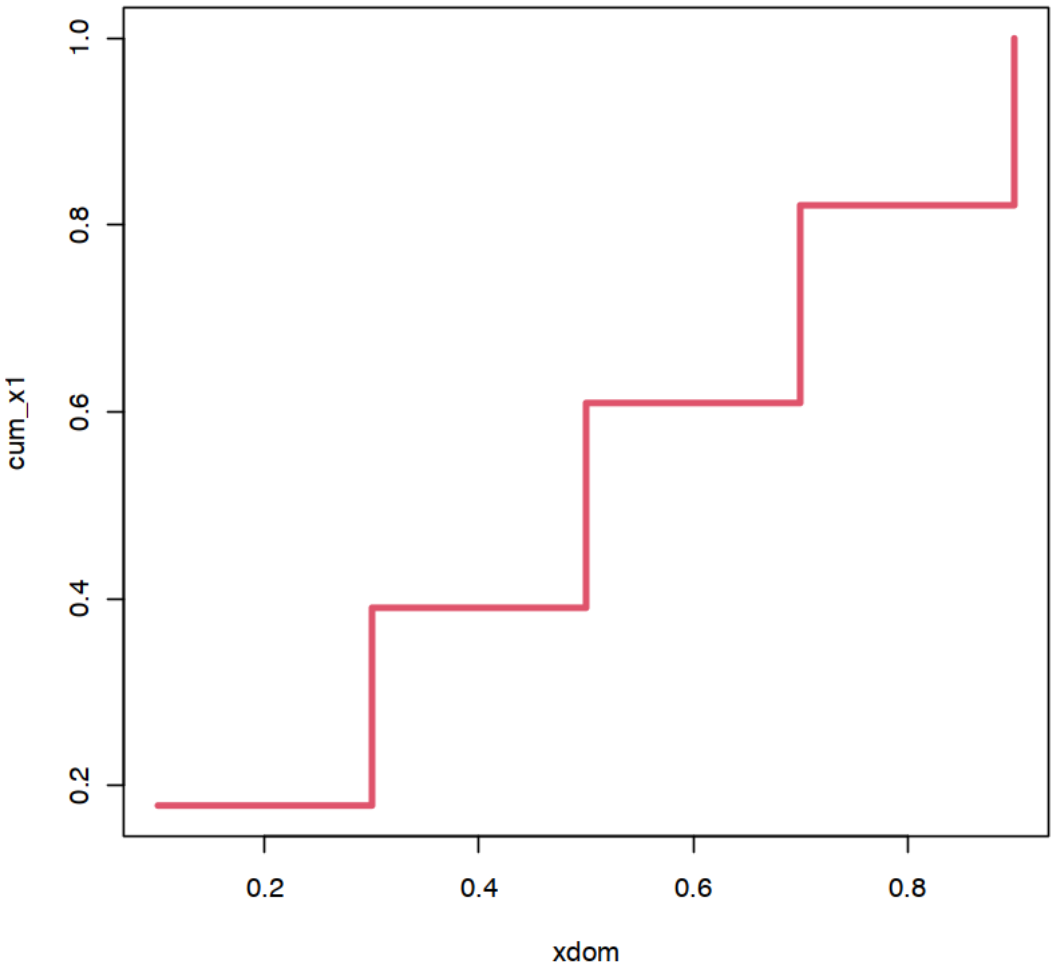


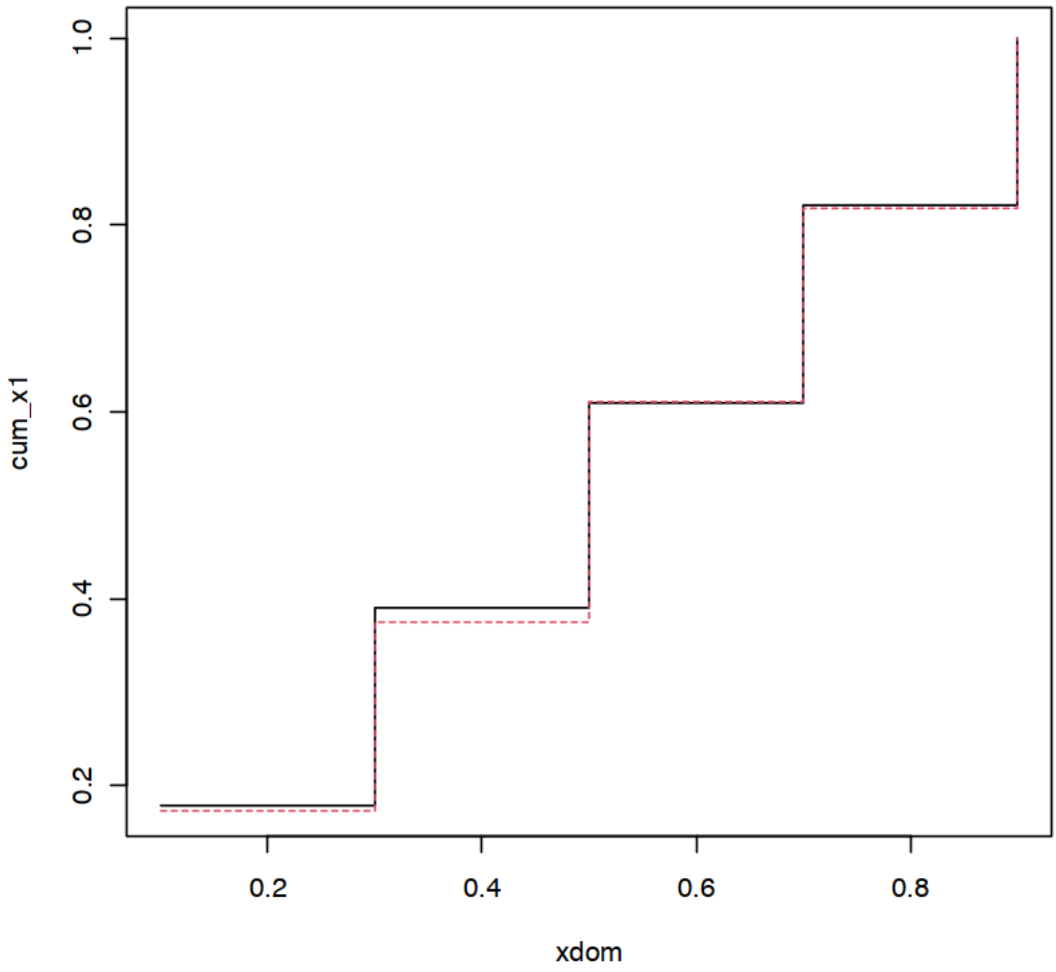
Notete come quando i punti m sono parecchi, non c'è quasi differenza tra le due comulate

[510]:
```
# Punto 2
# calcolo la f  con m = 10
m = 5
c = 1/m
xdom = c(0:(m-1))/m +c/2
# calcolo i valori di P(X = x)
prob_x = dbeta(xdom  ,a,b)
prob_x = prob_x/sum(prob_x)
sum(prob_x)
cum_x1 = cumsum(prob_x)
```

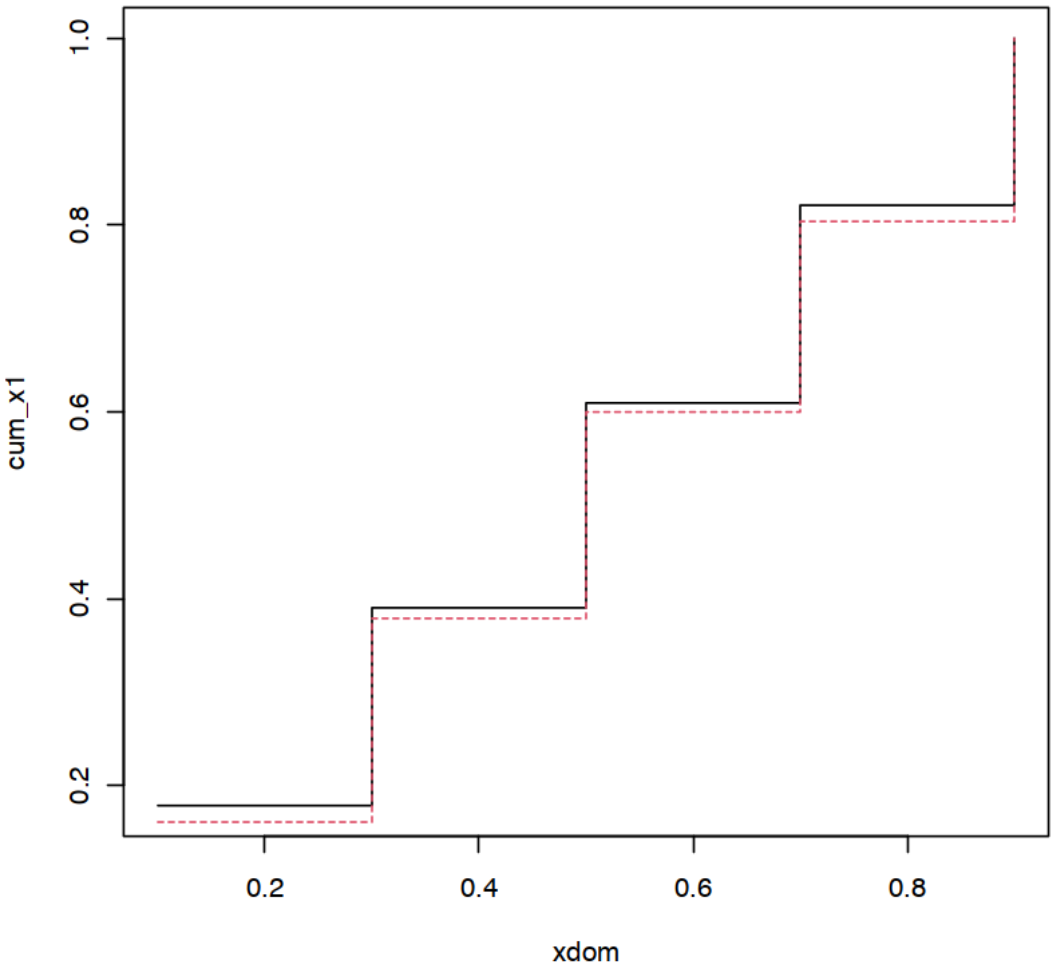
```r
plot(xdom,cum_x1, pch=20, cex=2, col=2 ,type="s", lwd=3)


### simulo nsim campioni da un'uniforme e poi valuto la F^{-1}
nsim = 1000
u = runif(nsim, 0,1)
xsamp = rep(NA, nsim)
for(isim in 1:nsim)
{
    w = which(cum_x1>=u[isim])[1]
    xsamp[isim] = xdom[w]
}

## comfronto i valori stimati con i valori veri
dens_stimata = rep(NA, m)
for(im in 1:m)
{
    dens_stimata[im] = mean(xsamp == xdom[im])
}

plot(xdom, cum_x1, type="s")
lines(xdom, cumsum(dens_stimata), type="s", col=2, lty = 2)
```

1


cum_x1
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
xdom

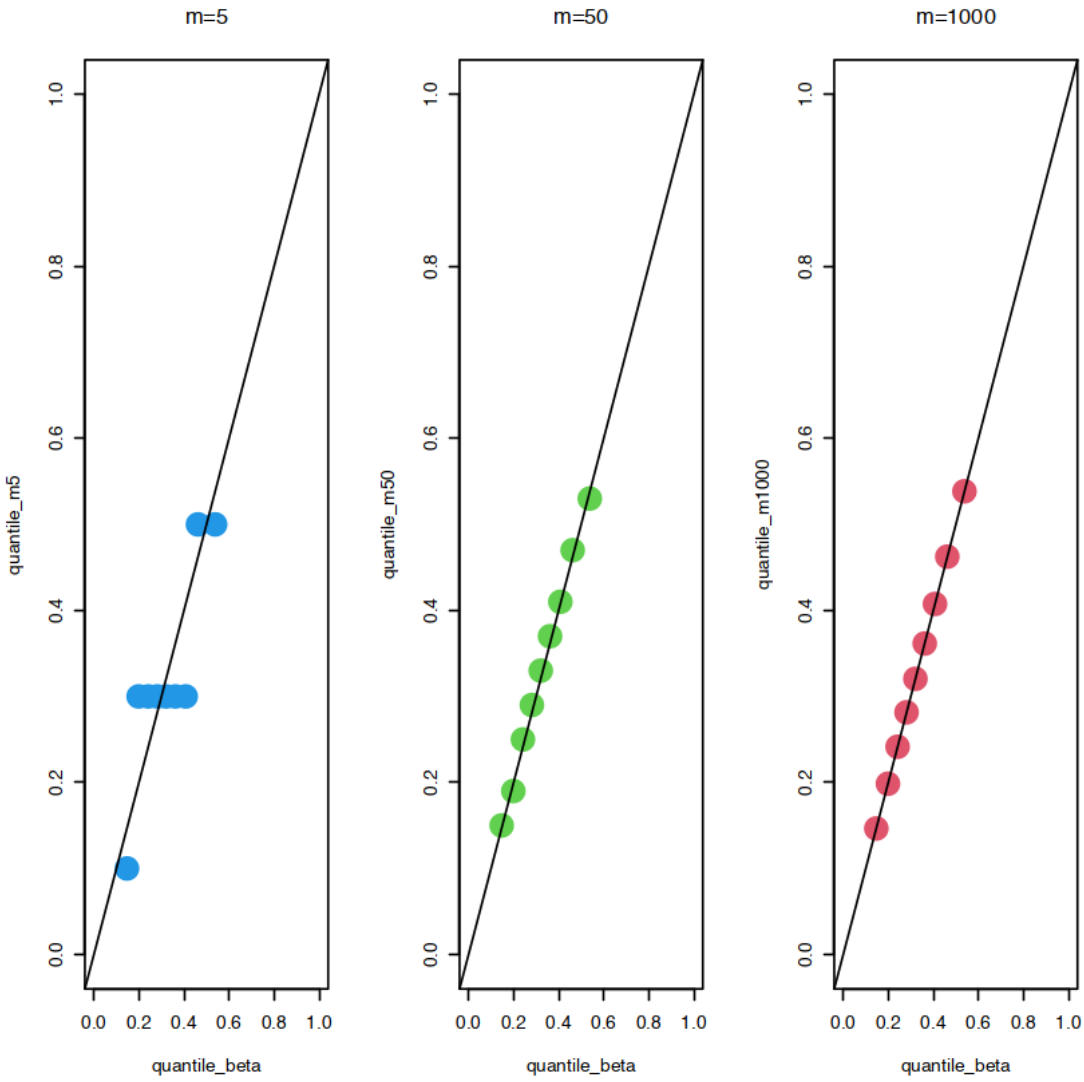


[511]:
```
# Punto 4
# per simulare da una discreta, con un valore prestabilito di probabilitò,␣
 ↪potete usare sample

xsamp = sample(xdom, nsim, prob = prob_x, replace=T)
dens_stimata = rep(NA, m)
for(im in 1:m)
{
    dens_stimata[im] = mean(xsamp == xdom[im])
    }
plot(xdom, cum_x1, type="s")
lines(xdom, cumsum(dens_stimata), type="s", col=2, lty = 2)
```



[527]:
```
### punto 4
# definisco i vettori che conterrano i quantili
vec_quantili = seq(0.1,0.9 ,by=0.1)
quantile_m5 = rep(NA, 9)
quantile_m50 = rep(NA, 9)
quantile_m1000 = rep(NA, 9)
quantile_beta = rep(NA, 9)
for(m in c(5,50,1000))
{
    c = 1/m
    xdom = c(0:(m-1))/m +c/2
    prob_x = dbeta(xdom  ,a,b)
    prob_x = prob_x/sum(prob_x)
```

```
    cum_x = cumsum(prob_x)
    for(i in 1:9)
    {
        w = which(cum_x>=vec_quantili[i])[1]
        cum_x = cumsum(prob_x)
        if(m == 5) quantile_m5[i] = xdom[w]
        if(m == 50) quantile_m50[i] = xdom[w]
        if(m == 1000) quantile_m1000[i] = xdom[w]

    }
}
quantile_beta = qbeta(vec_quantili,a,b)
par(mfrow=c(1,3))
plot(quantile_beta,quantile_m5, pch=20, col= 4, ylim=c(0,1), xlim=c(0,1),␣
 ↪cex=4, main="m=5")
abline(a=0,b=1)
plot(quantile_beta,quantile_m50, pch=20, col= 3, ylim=c(0,1), xlim=c(0,1),␣
 ↪cex=4, main="m=50")
abline(a=0,b=1)
plot(quantile_beta,quantile_m1000, pch=20, col= 2, ylim=c(0,1), xlim=c(0,1),␣
 ↪cex=4, main="m=1000")
abline(a=0,b=1)



par(mfrow=c(1,1))
```


m=5
m=50
m=1000
quantile_m5
quantile_m50
quantile_m1000
quantile_beta
quantile_beta
quantile_beta
0.0
0.2
0.4
0.6
0.8
1.0