Lab 1 con soluzioni

October 4, 2025

1 Esercitazione 1

1.1 Qualche risultato che vi serve per gli esercizi

Il Monte Carlo vale anche per funzioni di più variabile:

$$E(h(X,Y)) = \int_{\mathcal{X}} \int_{\mathcal{Y}} h(x,y) f(x,y) d\lambda(y) d\lambda(x) \approx \frac{\sum_{i=1}^{n} h(x_i, y_i)}{n}$$

e anche nel caso in cui $h(x,y) = h^*(x)$:

$$E(h^*(X)) = \int_{\mathcal{X}} h^*(x) f(x) d\lambda(x) = \int_{\mathcal{X}} \int_{\mathcal{Y}} h(x,y) f(x,y) d\lambda(y) d\lambda(x) \approx \frac{\sum_{i=1}^{n} h(x_i, y_i)}{n} = \frac{\sum_{i=1}^{n} h^*(x_i)}{n}$$

Potete simulare da una congiunta

$$f(x^1, x^2, \dots, x^p) = f(x^1) f(x^2|x^1) f(x^3|x^2, x^1) \dots f(x^p|x^{p-1}, x^{p-2}, \dots, x^2, x^1)$$

1. simulando x^1 da $f(x^1)$,
2. poi x^2 da $f(x^2|x^1)$,
3. poi x^3 da $f(x^3|x^2, x^1)$
4. e poi ….,
5. e x^p da $f(x^p|x^{p-1}, x^{p-2}, \dots, x^2, x^1)$

1.2 Stime di densità

Dato un campione $(x_1, \dots x_n)$ potete ottenere una stima della densità sottostante in un punto a come

$$\hat{f}(a) = \frac{\sum_{i=1}^{n} k(x_i - a)}{n} \approx \int_{\mathcal{X}} k(x-a) f(x) d\lambda(x)$$

dove $k(x, a)$ è un kernel, che ha le seguenti proprietà

1. $k(x-a) > 0$ per tutti
2. simmetrico rispetto a zero

Fate attenzione che $\hat{f}(a)$ non è la stima monte carlo della densità, ma la stima montecarlo di $\int_{\mathcal{X}} k(x-a) f(x) d\lambda(x)$. Questa la si usa per le volte in cui non si può simulare e si hanno un set fisso di dati

Questo è quello che fa il comando density() di R. L'integrale è di fatto una convoluzione tra $f(x)$ e il kernel $ k(x - a)$.

Adesso assumete che $X \sim G(10, 10)$, simulate $n = 10$ osservazioni e stimate con il metodo del kernel la densità sottostante assumendo un kernel gaussiano

$$k(x_i - a) = (2\pi\sigma^2)^{-0.5} \exp(-\frac{1}{2\sigma^2}(x_i - a)^2)$$

(il kernel è la densità di una normale con media a, valutata in x_i) vedete le differenze se cambiate il valore di $\sigma \in \{0.01, 0.05, 0.1\}$ e confrontatelo graficamento con la vera densità

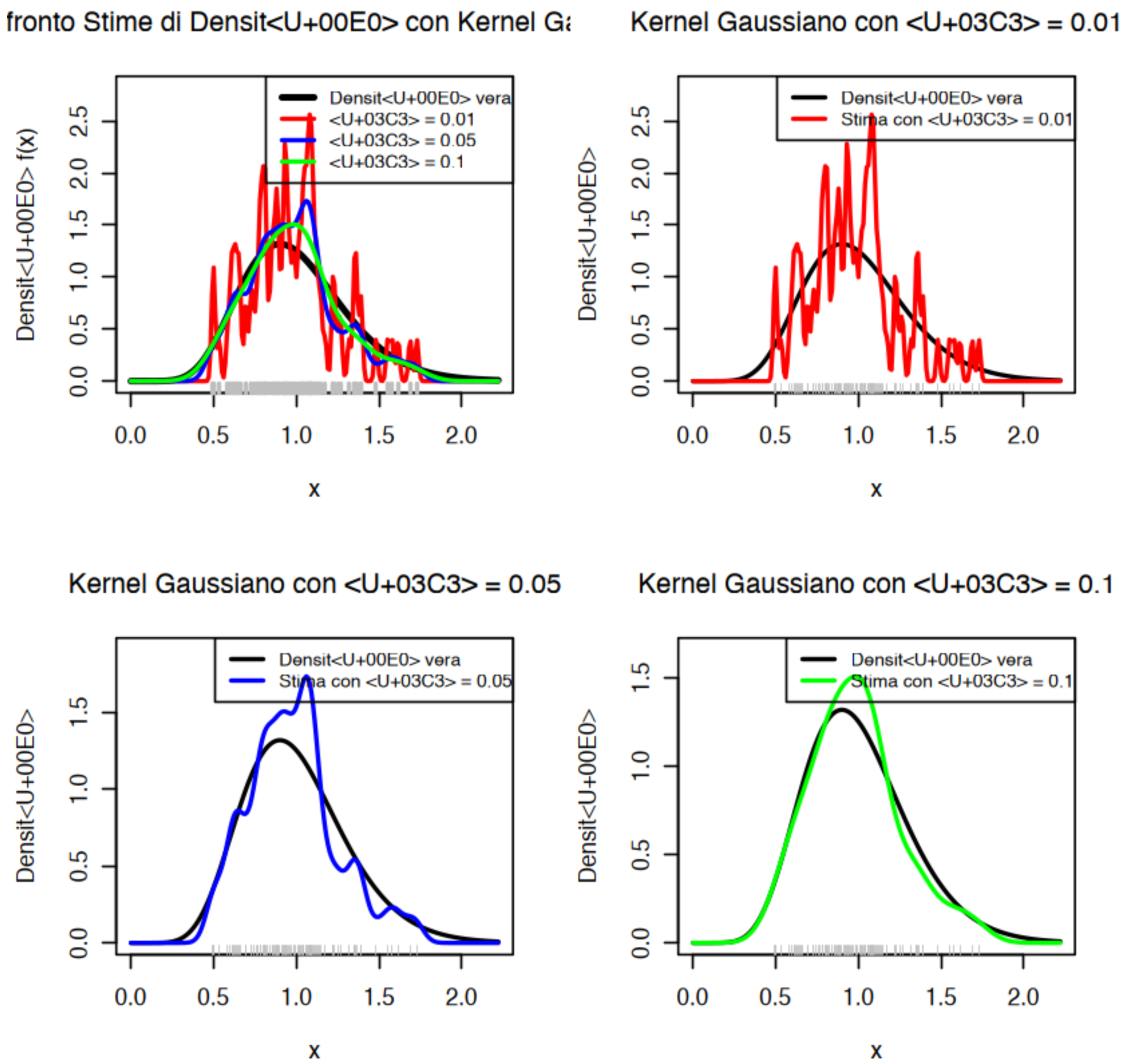
1.2.1 Soluzione

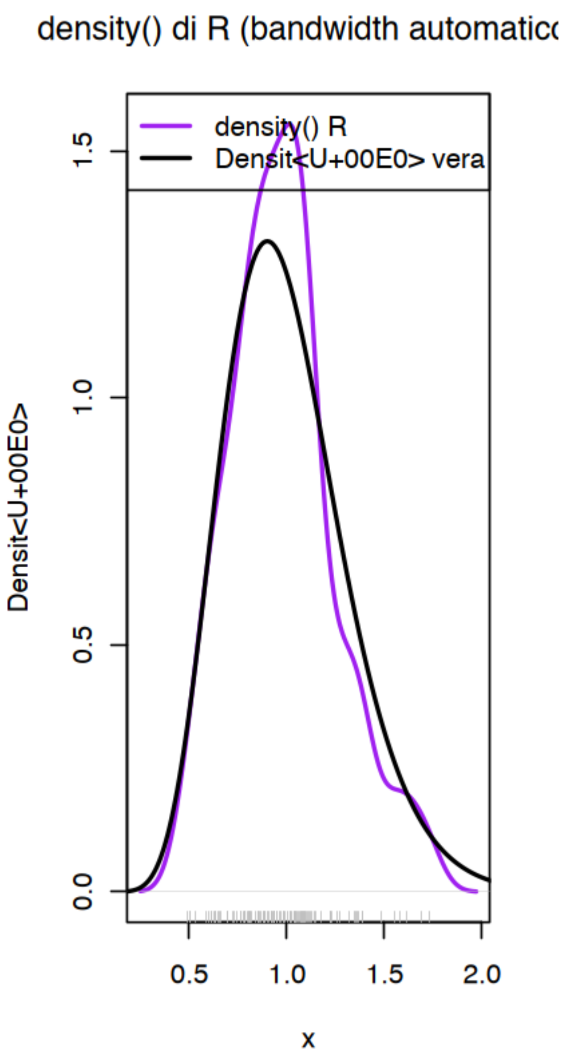
[53]:
```
# ============================================================================
# STIMA DI DENSITÀ CON KERNEL GAUSSIANO
# ============================================================================

# Parametri della distribuzione Gamma di riferimento
a <- 10    # Parametro di forma
b <- 10    # Parametro di scala
n <- 100   # Dimensione del campione

# Generazione del campione dalla distribuzione Gamma
set.seed(123)  # Per riproducibilità
x <- rgamma(n, shape = a, rate = b)

cat("=== INFORMAZIONI SUL CAMPIONE ===\n")
cat("Parametri Gamma: shape =", a, ", rate =", b, "\n")
cat("Dimensione campione:", n, "\n")
cat("Media teorica:", a/b, "| Media campionaria:", round(mean(x), 4), "\n")
cat("Varianza teorica:", a/b^2, "| Varianza campionaria:", round(var(x), 4),␣
 ↪"\n\n")

# Parametri del kernel gaussiano (bandwidth)
sigma_vals <- c(0.01, 0.05, 0.1)
sigma_names <- paste("  =", sigma_vals)

# Griglia di valutazione per la densità
x_min <- max(0, min(x) - 0.5)
x_max <- max(x) + 0.5
xseq <- seq(x_min, x_max, by = 0.01)

# Calcolo delle stime di densità per ogni valore di sigma
cat("=== CALCOLO STIME DI DENSITÀ ===\n")
densita_stimata <- matrix(NA, nrow = length(xseq), ncol = length(sigma_vals))

for(j in 1:length(sigma_vals)) {
    sigma <- sigma_vals[j]
    cat("Calcolando stima con   =", sigma, "...\n")

    # Stima di densità: f(a) = (1/n) * Σ k(x_i - a)
```

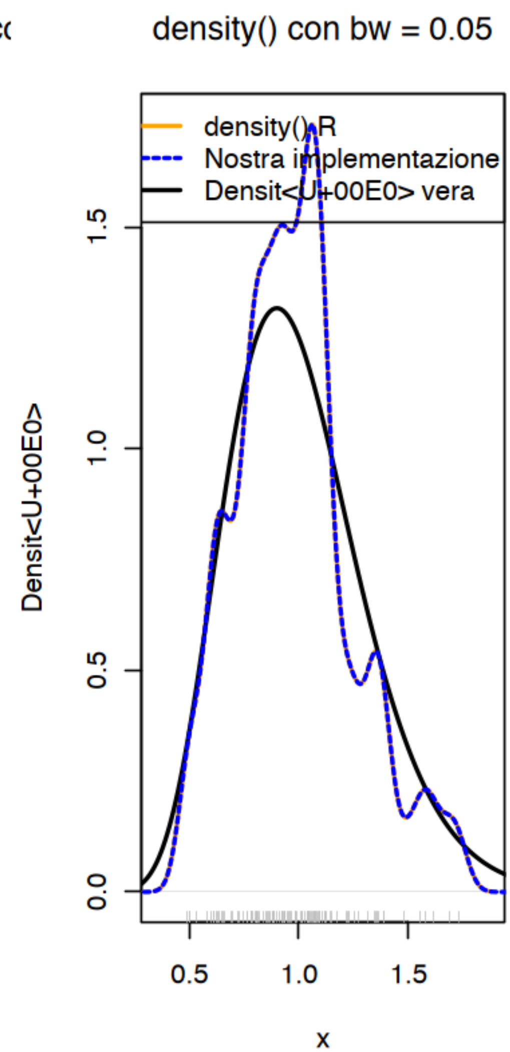
```r
    for(i in 1:length(xseq)) {
        a_val <- xseq[i]
        # Kernel gaussiano: k(x_i - a) =  ((x_i - a)/) /
        densita_stimata[i, j] <- mean(dnorm(x, mean = a_val, sd = sigma))
    }
}

# Densità teorica per confronto
densita_teorica <- dgamma(xseq, shape = a, rate = b)

# VISUALIZZAZIONE AVANZATA
par(mfrow = c(2, 2))

# Grafico principale: confronto di tutte le stime
plot(xseq, densita_teorica, type = "l", lwd = 3, col = "black",
     main = "Confronto Stime di Densità con Kernel Gaussiano",
     xlab = "x", ylab = "Densità f(x)",
     ylim = c(0, max(c(densita_teorica, densita_stimata)) * 1.1))

colors <- c("red", "blue", "green")
for(j in 1:length(sigma_vals)) {
    lines(xseq, densita_stimata[, j], col = colors[j], lwd = 2)
}

# Aggiungo i punti del campione
rug(x, col = "gray", lwd = 2)

legend("topright",
       legend = c("Densità vera", sigma_names),
       col = c("black", colors),
       lwd = c(3, rep(2, length(sigma_vals))),
       cex = 0.8)

# Grafici individuali per ogni sigma
for(j in 1:length(sigma_vals)) {
    plot(xseq, densita_teorica, type = "l", lwd = 2, col = "black",
         main = paste("Kernel Gaussiano con", sigma_names[j]),
         xlab = "x", ylab = "Densità",
         ylim = c(0, max(densita_teorica, densita_stimata[, j]) * 1.1))

    lines(xseq, densita_stimata[, j], col = colors[j], lwd = 2)
    rug(x, col = "gray")

    legend("topright",
           legend = c("Densità vera", paste("Stima con", sigma_names[j])),
           col = c("black", colors[j]), lwd = 2, cex = 0.8)
}
```

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

# CONFRONTO CON density() DI R
cat("\n=== CONFRONTO CON density() DI R ===\n")
par(mfrow = c(1, 2))

# Usando density() con bandwidth automatico
dens_auto <- density(x)
plot(dens_auto, main = "density() di R (bandwidth automatico)",
     xlab = "x", ylab = "Densità", lwd = 2, col = "purple")
lines(xseq, densita_teorica, col = "black", lwd = 2)
rug(x, col = "gray")
legend("topright", c("density() R", "Densità vera"),
       col = c("purple", "black"), lwd = 2)

cat("Bandwidth automatico di R:", round(dens_auto$bw, 4), "\n")

# Usando density() con bandwidth manuale (quello intermedio)
dens_manual <- density(x, bw = sigma_vals[2])
plot(dens_manual, main = paste("density() con bw =", sigma_vals[2]),
     xlab = "x", ylab = "Densità", lwd = 2, col = "orange")
lines(xseq, densita_teorica, col = "black", lwd = 2)
lines(xseq, densita_stimata[, 2], col = "blue", lwd = 2, lty = 2)
rug(x, col = "gray")
legend("topright", c("density() R", "Nostra implementazione", "Densità vera"),
       col = c("orange", "blue", "black"), lwd = 2, lty = c(1, 2, 1))

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

```
=== INFORMAZIONI SUL CAMPIONE ===
Parametri Gamma: shape = 10 , rate = 10
Dimensione campione: 100
Media teorica: 1 | Media campionaria: 0.9785
Varianza teorica: 0.1 | Varianza campionaria: 0.072

=== CALCOLO STIME DI DENSIT<U+00C0> ===
Calcolando stima con <U+03C3> = 0.01 …
Calcolando stima con <U+03C3> = 0.05 …
Calcolando stima con <U+03C3> = 0.1 …

=== CONFRONTO CON density() DI R ===
Bandwidth automatico di R: 0.0807
```


fronto Stime di Densit<U+00E0> con Kernel Ga
Densit<U+00E0> vera
<U+03C3> = 0.01
<U+03C3> = 0.05
<U+03C3> = 0.1
Densit<U+00E0> f(x)
x
Kernel Gaussiano con <U+03C3> = 0.01
Densit<U+00E0> vera
Stima con <U+03C3> = 0.01
Densit<U+00E0>
x
Kernel Gaussiano con <U+03C3> = 0.05
Densit<U+00E0> vera
Stima con <U+03C3> = 0.05
Densit<U+00E0>
x
Kernel Gaussiano con <U+03C3> = 0.1
Densit<U+00E0> vera
Stima con <U+03C3> = 0.1
Densit<U+00E0>
x





1.3 Teorema del limite centrale

Assumete che $X \sim P(\lambda)$ e definite

$$\bar{X}_n = \sum_{i=1}^{n} X_i/n$$

Verificate se la distribuzione di

$$Z = \frac{\bar{X}_n - E(X)}{\sqrt{\frac{Var(X)}{n}}}$$

segue approssimativamente un $N(0,1)$ con $n = 2, 100, 10000$.

Provate con $\lambda = 0.1$ e $\lambda = 10$, e plottate le distribuzione di Z usando un istogramma e il comando density()

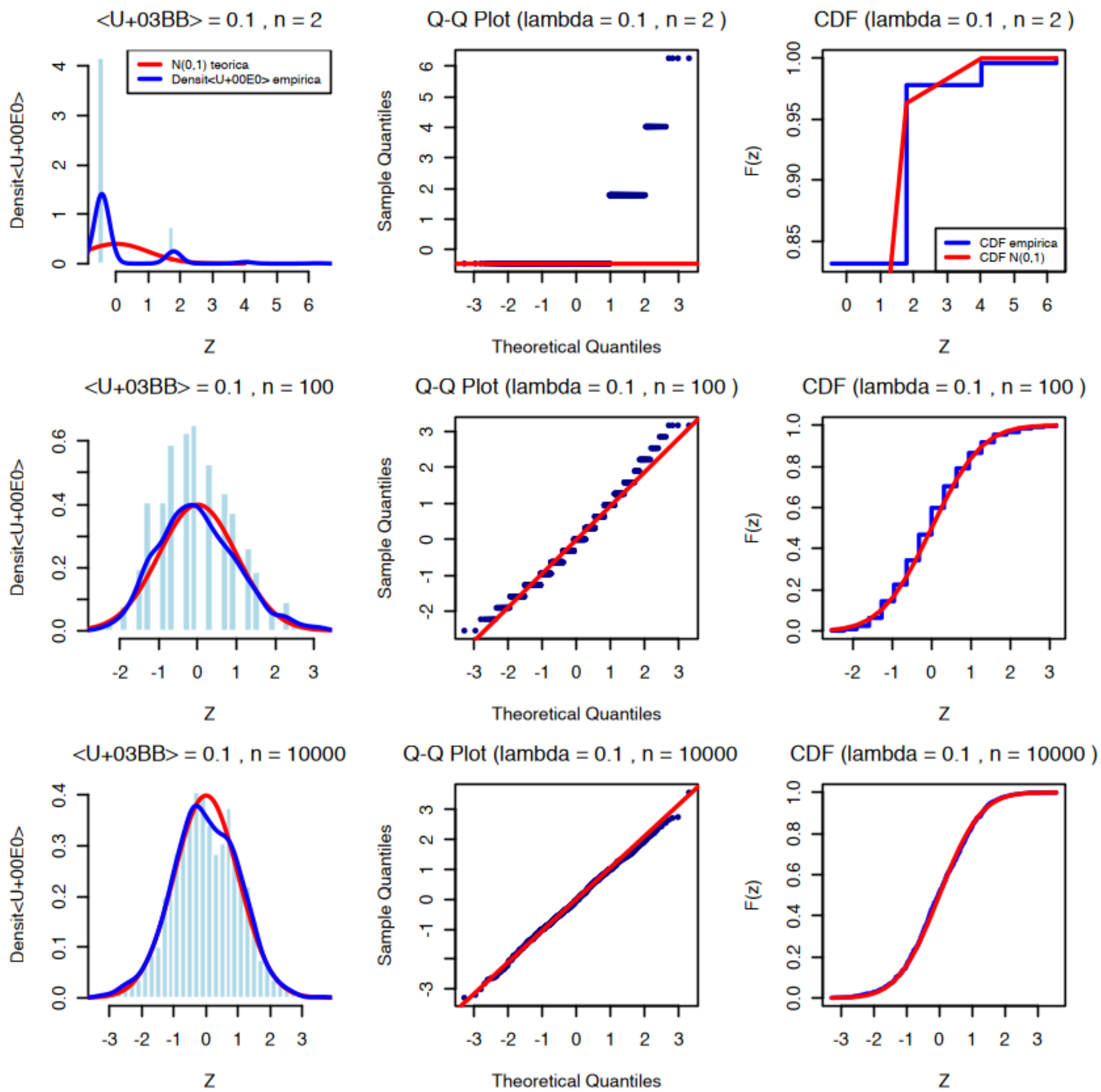
[54]:

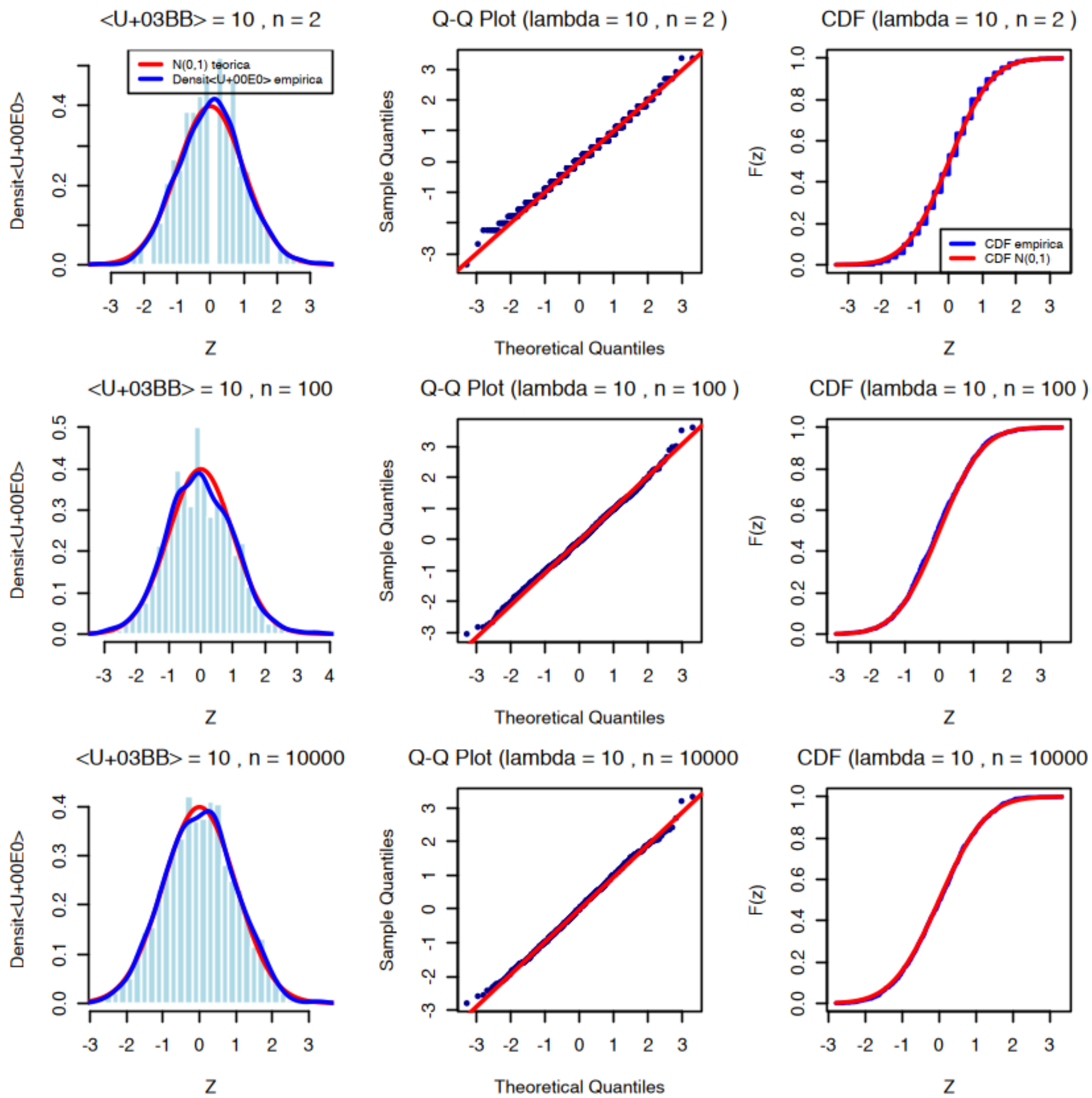
```
# ==============================================================================
# VERIFICA DEL TEOREMA DEL LIMITE CENTRALE
# ==============================================================================

# Parametri della simulazione
nsim <- 1000      # Numero di simulazioni per ogni scenario
n_values <- c(2, 100, 10000)  # Diverse dimensioni campionarie
lambda_values <- c(0.1, 10)   # Diversi parametri della Poisson

cat("=== TEOREMA DEL LIMITE CENTRALE ===\n")


# Generazione anticipata di tutti i campioni necessari
max_n <- max(n_values)
set.seed(123)

# Pre-generazione di campioni per efficienza
all_samples <- list()
for(i in 1:length(lambda_values)) {
    lambda <- lambda_values[i]
    # Matrice nsim × max_n
    all_samples[[i]] <- matrix(rpois(max_n * nsim, lambda), nrow = max_n, ncol = nsim)
}


# Analisi per ogni combinazione di parametri
for(lambda in lambda_values) {
    lambda_idx <- which(lambda_values == lambda)
    samples <- all_samples[[lambda_idx]]



    # Setup per grafici
    par(mfrow = c(3, 3), mar = c(4, 4, 3, 2))

    for(j in 1:length(n_values)) {
        n <- n_values[j]

        # Estrazione dei campioni di dimensione n
        x_samples <- samples[1:n, , drop = FALSE]

        # Calcolo delle medie campionarie
        x_bar <- colMeans(x_samples)

        # Standardizzazione: Z = (X - ) / _X
        # Dove _X = √(Var(X)/n) = √( /n)
```

```r
z <- (x_bar - lambda) / sqrt(lambda / n)



# Statistiche descrittive
z_mean <- mean(z)
z_var <- var(z)
z_sd <- sd(z)





# GRAFICO 1: Istogramma con sovrapposizione normale
hist(z, breaks = 30, freq = FALSE,
     main = paste("  =", lambda, ", n =", n),
     xlab = "Z", ylab = "Densità",
     col = "lightblue", border = "white")

# Sovrapposizione della densità normale teorica
z_seq <- seq(-4, 4, length.out = 200)
lines(z_seq, dnorm(z_seq, 0, 1), col = "red", lwd = 2)

# Sovrapposizione della densità empirica
lines(density(z), col = "blue", lwd = 2)

if(j == 1) {  # Legenda solo nel primo grafico
    legend("topright",
           c("N(0,1) teorica", "Densità empirica"),
           col = c("red", "blue"), lwd = 2, cex = 0.7)
}

# GRAFICO 2: Q-Q plot
qqnorm(z, main = paste("Q-Q Plot (lambda =", lambda, ", n =", n, ")"),
       pch = 20, cex = 0.5, col = "darkblue")
qqline(z, col = "red", lwd = 2)

# GRAFICO 3: Funzione di ripartizione empirica vs teorica
z_sorted <- sort(z)
empirical_cdf <- ecdf(z_sorted)

plot(z_sorted, empirical_cdf(z_sorted), type = "s",
     main = paste("CDF (lambda =", lambda, ", n =", n, ")"),
     xlab = "Z", ylab = "F(z)", col = "blue", lwd = 2)
lines(z_sorted, pnorm(z_sorted, 0, 1), col = "red", lwd = 2)

if(j == 1) {  # Legenda solo nel primo grafico
```

```
            legend("bottomright",
                   c("CDF empirica", "CDF N(0,1)"),
                   col = c("blue", "red"), lwd = 2, cex = 0.7)
        }
    }

    par(mfrow = c(1, 1))
    cat("----------------------------------------\n\n")
}
```

```
=== TEOREMA DEL LIMITE CENTRALE ===
----------------------------------------
```





1.4 Beta binomiale

Assumete che

$$X|p \sim Bin(m, p) \qquad p \sim Beta(a, b)$$

con m intero, $p \in (0, 1)$ e $X \in \{0, 1, ... m\}$.

è noto che la marginale di X è una beta-binomiale di parametri (m,a,b)

$$X \sim BetaBin(m, a, b)$$

che ha pmf

$$f(x) = P(X = x) = \binom{m}{x} \frac{B(a + x, b + n - x)}{B(a, b)}$$

dove il coefficiente binomiale è

$$\binom{m}{x} = \frac{m!}{x!(m - x)!}$$

Inoltre sappiamo che

$$E(X) = m\frac{a}{a + b}$$

mentre

$$E(X|p) = mp$$

1. Stimare i valori di $f(x)$ nei punti $\{0, 1, \dots, m\}$ usando la distribuzione condizionate di $X|p$.
2. Scrivete una funzione che calcoli la marginale e verificate graficamente che la f() vera e quella stimata siano simili
3. stimare il valore atteso di X sia usando

   $$E(X) = \int x f(x) d\lambda(x)$$

   sia

   $$E_p(E_x(X|p)) = E(X) = \int E_x(X|p) f(p) d\lambda(p)$$

   e verificare quale dei due è più variabile usando $n = 10$ e $n = 1000$.

Per la simulazione di una beta-binomiale usate il pacchetto TailRank, che va installato usando install.packages("TailRank"). Se il pacchetto Biobase vi da problemi, usate i comandi

install.packages("BiocManager")

BiocManager::install("Biobase")

Alternativamente, potete vedere nell'esercizio "Congiunte e marginali" che se simulate p_i dalla sua marginale $B(a, b)$, e poi x_i da $Binom(m, p)$, allora x_i è un **campione dalla marginale BetaBinomial**

Stima varianza

Per la stima della varianza di

$$\hat{E}(h(Y)) = \frac{\sum_{i=1}^{n} h(y_i)}{n}$$

avete due possibilità.

Come prima possibilità potete scrivere

$$Var(\hat{E}(h(Y))) = \frac{\sum_{i=1}^{n} Var(h(y_i))}{n^2} = \frac{Var(h(y))}{n}$$

e stimare $Var(h(y))$ con

$$Var(h(y)) \approx \frac{\sum_{i=1}^{n^*} (h(y_i^*) - E(h(y^*)))^2}{n^*}$$

dove ho utilizzato lo $*$ per indicare che potete usare o no i gli stessi campioni usati per calcolare $\hat{E}(h(Y))$ oppure prenderne dei nuovi. Inoltre non conoscete $E(h(y^*))$ e dovete stimarlo

La seconda opzione è di definire

$$W = \hat{E}(h(Y))$$

e calcolare

$$Var(W) \approx \frac{\sum_{i=1}^{n^{**}} (W_i^{**} - E(W))^2}{n^{**}}$$

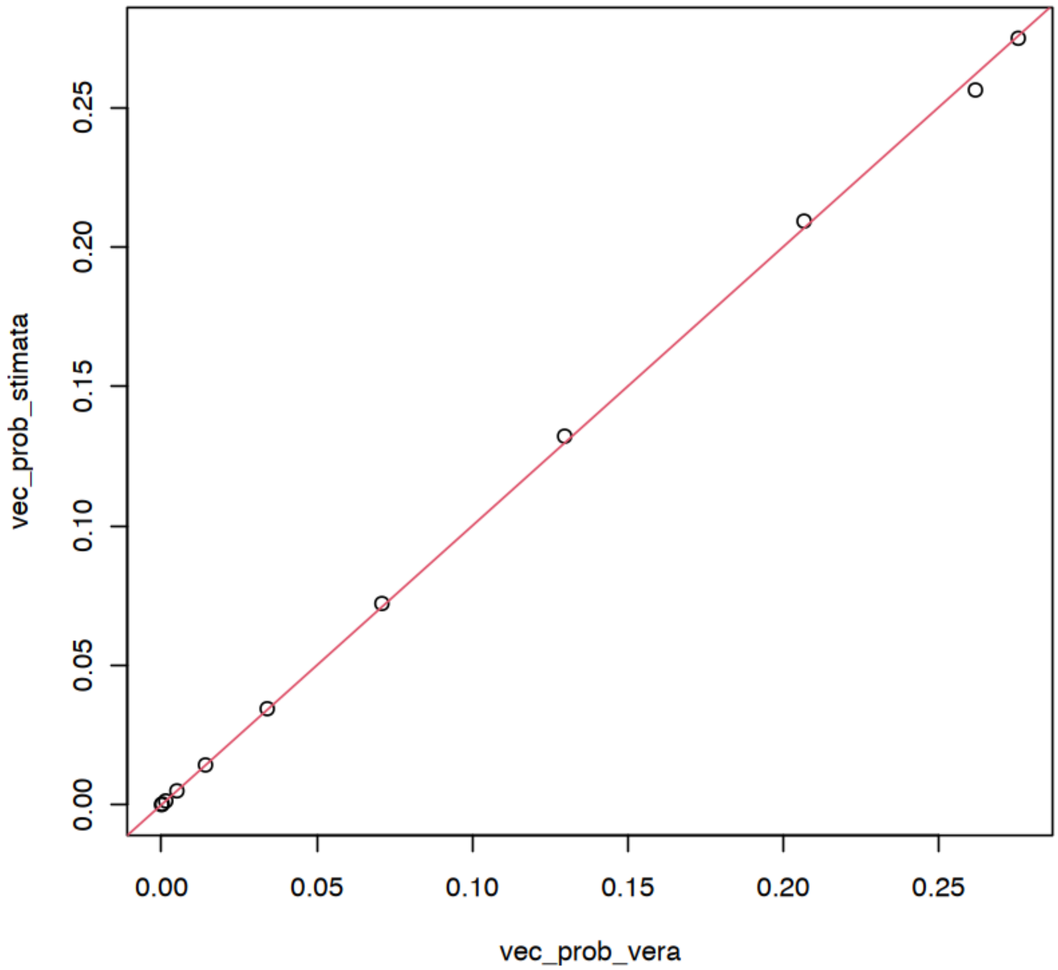
quindi usando campioni di W invece che di X. Anche qui dovete stimare $E(W)$

1.4.1 Soluzione

[63]:
```
# ==============================================================================
# DISTRIBUZIONE BETA-BINOMIALE
# ==============================================================================

# Funzione per calcolare la densità della Beta-Binomiale
# Versione numericamente stabile usando log-densità
dbetabin <- function(x, m, a, b, log = FALSE) {
    # Controlli di validità
    if(any(x < 0 | x > m | !is.finite(x))) {
        warning("x deve essere tra 0 e m")
        return(rep(NaN, length(x)))
    }
    if(m <= 0 | a <= 0 | b <= 0) {
        stop("I parametri m, a, b devono essere positivi")
    }

    # Calcolo in log-scala per stabilità numerica
    log_prob <- lchoose(m, x) + lbeta(a + x, b + m - x) - lbeta(a, b)

    if(log) return(log_prob) else return(exp(log_prob))
}

# Parametri del modello
a <- 10   # Parametro  della Beta
b <- 2    # Parametro  della Beta
m <- 10   # Numero di prove Binomiali
n <- 1000 # Dimensione campione per simulazione

# Media e varianza teoriche
media_teorica <- m * a / (a + b)
varianza_teorica <- m * a * b * (a + b + m) / ((a + b)^2 * (a + b + 1))

cat("=== INFORMAZIONI SUL MODELLO ===\n")
cat("Parametri: m =", m, ", a =", a, ", b =", b, "\n")
cat("Media teorica di X ~ BetaBin(m,a,b):", round(media_teorica, 4), "\n")
cat("Varianza teorica:", round(varianza_teorica, 4), "\n\n")

# PUNTO 1: Stima delle probabilità usando la distribuzione condizionata
```

```r
cat("=== PUNTO 1: STIMA VIA DISTRIBUZIONE CONDIZIONATA ===\n")

# Griglia di valori per cui calcolare P(X = k)
xvec <- 0:m
vec_prob_vera <- numeric(length(xvec))
vec_prob_stimata <- numeric(length(xvec))

# Simulazione da p ~ Beta(a,b)
set.seed(123)
p_samples <- rbeta(n, a, b)

# Calcolo delle probabilità stimate e vere
for(i in 1:length(xvec)) {
    k <- xvec[i]
    # Probabilità vera dalla Beta-Binomiale
    vec_prob_vera[i] <- dbetabin(k, m, a, b)
    # Stima Monte Carlo: E[P(X=k|p)] = E[dbinom(k, m, p)]
    vec_prob_stimata[i] <- mean(dbinom(k, m, prob = p_samples))
}

# Verifica somme (devono essere   1)
sum_vera <- sum(vec_prob_vera)
sum_stimata <- sum(vec_prob_stimata)

cat("Verifica:\n")
cat("  Somma probabilità vere:", round(sum_vera, 6), "\n")
cat("  Somma probabilità stimate:", round(sum_stimata, 6), "\n\n")

plot(vec_prob_vera, vec_prob_stimata)
abline(a=0, b=1, col=2)
```

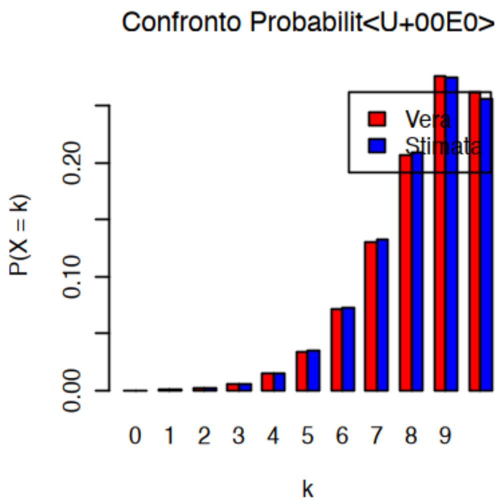
```
=== INFORMAZIONI SUL MODELLO ===
Parametri: m = 10 , a = 10 , b = 2
Media teorica di X ~ BetaBin(m,a,b): 8.3333
Varianza teorica: 2.3504

=== PUNTO 1: STIMA VIA DISTRIBUZIONE CONDIZIONATA ===
Verifica:
  Somma probabilit<U+00E0> vere: 1
  Somma probabilit<U+00E0> stimate: 1
```

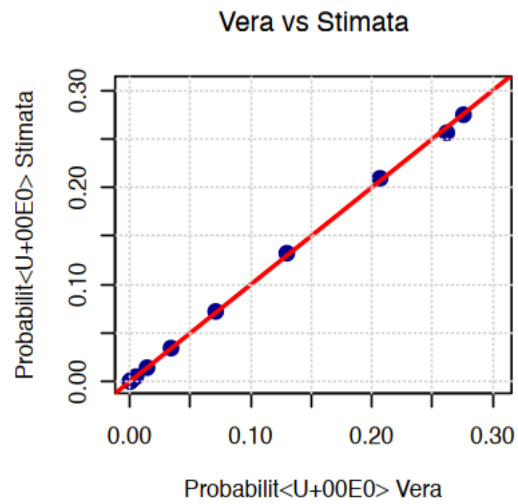


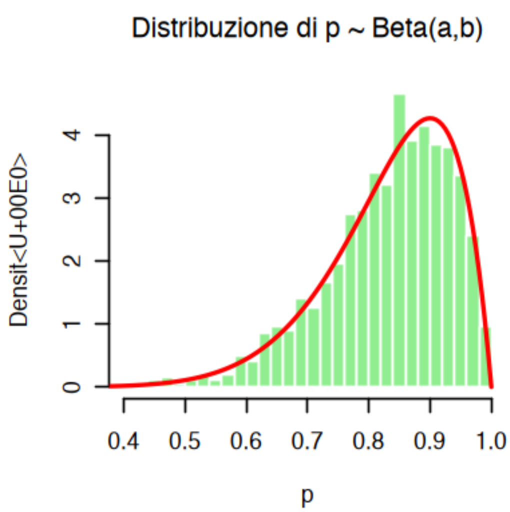
[64]:
```
# PUNTO 2: Visualizzazione e confronto
cat("=== PUNTO 2: CONFRONTO GRAFICO ===\n")

# Tabella di confronto
comparison_table <- data.frame(
    k = xvec,
    Prob_Vera = round(vec_prob_vera, 6),
    Prob_Stimata = round(vec_prob_stimata, 6),
    Errore_Abs = round(abs(vec_prob_vera - vec_prob_stimata), 6)
)

print(comparison_table)
```

```r
# Visualizzazione grafica migliorata
par(mfrow = c(2, 2))

# Grafico 1: Confronto barplot
barplot(rbind(vec_prob_vera, vec_prob_stimata),
        beside = TRUE, names.arg = xvec,
        col = c("red", "blue"),
        main = "Confronto Probabilità",
        xlab = "k", ylab = "P(X = k)",
        legend.text = c("Vera", "Stimata"))

# Grafico 2: Scatter plot
plot(vec_prob_vera, vec_prob_stimata,
     pch = 20, cex = 2, col = "darkblue",
     main = "Vera vs Stimata",
     xlab = "Probabilità Vera", ylab = "Probabilità Stimata",
     xlim = c(0, max(vec_prob_vera) * 1.1),
     ylim = c(0, max(vec_prob_stimata) * 1.1))
abline(a = 0, b = 1, col = "red", lwd = 2)
grid()



# Grafico 4: Distribuzione dei campioni p
hist(p_samples, breaks = 30, freq = FALSE,
     col = "lightgreen", border = "white",
     main = "Distribuzione di p ~ Beta(a,b)",
     xlab = "p", ylab = "Densità")

# Sovrapposizione densità teorica
p_seq <- seq(0, 1, length.out = 200)
lines(p_seq, dbeta(p_seq, a, b), col = "red", lwd = 2)

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

```
=== PUNTO 2: CONFRONTO GRAFICO ===
    k Prob_Vera Prob_Stimata Errore_Abs
1   0  0.000031     0.000027   0.000004
2   1  0.000284     0.000270   0.000014
3   2  0.001403     0.001388   0.000016
4   3  0.004990     0.004988   0.000002
5   4  0.014190     0.014260   0.000070
6   5  0.034056     0.034437   0.000381
7   6  0.070949     0.072153   0.001203
8   7  0.129736     0.132141   0.002405
9   8  0.206767     0.209254   0.002487
10  9  0.275689     0.274815   0.000874
11 10  0.261905     0.256267   0.005637
```







[70]:
```
# ============================================================================
# PUNTO 3: CONFRONTO METODI DI STIMA DEL VALORE ATTESO
# ============================================================================

library(TailRank)

# Funzione alternativa per simulare da Beta-Binomiale
rbb_custom <- function(n, m, a, b) {
    p_vals <- rbeta(n, a, b)
    x_vals <- numeric(n)
    for(i in 1:n) {
        x_vals[i] <- rbinom(1, m, p_vals[i])
    }
```

```
    return(x_vals)
}

# Selezione della funzione da usare
if(exists("tailrank_available") && tailrank_available && exists("rbb")) {
    rbb_func <- rbb
    cat("Usando rbb() da TailRank\n")
} else {
    rbb_func <- rbb_custom
    cat("Usando simulazione custom\n")
}

cat("\n=== CONFRONTO TRA DUE METODI DI STIMA ===\n")
cat("Metodo 1: E[X] tramite simulazione diretta dalla marginale\n")
cat("Metodo 2: E[X] = E[E[X|p]] = E[mp] tramite distribuzione condizionata\n\n")

# Parametri per l'analisi della varianza
nsim <- 1000  # Numero di repliche
n_vals <- c(10, 100, 1000)  # Diverse dimensioni campionarie

# Inizializzazione dataframe risultati
risultati_varianza <- data.frame(
    n = integer(),
    Media_Marg = numeric(),
    Var_Marg = numeric(),
    Media_Cond = numeric(),
    Var_Cond = numeric(),
    Ratio_Var = numeric()
)

set.seed(456)  # Per riproducibilità

for(n_curr in n_vals) {
    cat("Analisi con n =", n_curr, "...\n")

    meanx_marg <- numeric(nsim)
    meanx_cond <- numeric(nsim)

    for(isim in 1:nsim) {
        # Metodo 1: Simulazione diretta dalla marginale
        x_marginale <- rbb_func(n_curr, m, a, b)
        meanx_marg[isim] <- mean(x_marginale)

        # Metodo 2: Via distribuzione condizionata
        p_cond <- rbeta(n_curr, a, b)
        meanx_cond[isim] <- mean(m * p_cond)  # E[X|p] = mp
    }
```

```r
    # Calcolo statistiche
    media_marg <- mean(meanx_marg)
    var_marg <- var(meanx_marg)
    media_cond <- mean(meanx_cond)
    var_cond <- var(meanx_cond)
    ratio_var <- var_marg / var_cond

    # Memorizzazione risultati
    risultati_varianza <- rbind(risultati_varianza, data.frame(
        n = n_curr,
        Media_Marg = round(media_marg, 4),
        Var_Marg = round(var_marg, 6),
        Media_Cond = round(media_cond, 4),
        Var_Cond = round(var_cond, 6),
        Ratio_Var = round(ratio_var, 4)
    ))

    cat("  Media marginale:", round(media_marg, 4), "\n")
    cat("  Varianza marginale:", round(var_marg, 6), "\n")
    cat("  Media condizionata:", round(media_cond, 4), "\n")
    cat("  Varianza condizionata:", round(var_cond, 6), "\n")
    cat("  Rapporto varianze:", round(ratio_var, 4), "\n\n")
}

cat("=== TABELLA RIASSUNTIVA ===\n")
print(risultati_varianza)
```

```
Usando rbb() da TailRank

=== CONFRONTO TRA DUE METODI DI STIMA ===
Metodo 1: E[X] tramite simulazione diretta dalla marginale
Metodo 2: E[X] = E[E[X|p]] = E[mp] tramite distribuzione condizionata


=== CONFRONTO TRA DUE METODI DI STIMA ===
Metodo 1: E[X] tramite simulazione diretta dalla marginale
Metodo 2: E[X] = E[E[X|p]] = E[mp] tramite distribuzione condizionata

Analisi con n = 10 …
  Media marginale: 8.3254
  Varianza marginale: 0.256591
  Media condizionata: 8.3216
  Varianza condizionata: 0.109322
  Rapporto varianze: 2.3471

Analisi con n = 100 …
  Media marginale: 8.3374
```

```
  Varianza marginale: 0.025621
  Media condizionata: 8.3273
  Varianza condizionata: 0.01048
  Rapporto varianze: 2.4448

Analisi con n = 1000 …
  Media marginale: 8.333
  Varianza marginale: 0.002388
  Media condizionata: 8.3334
  Varianza condizionata: 0.001075
  Rapporto varianze: 2.2223

=== TABELLA RIASSUNTIVA ===
     n Media_Marg Var_Marg Media_Cond Var_Cond Ratio_Var
1   10     8.3254 0.256591     8.3216 0.109322    2.3471
2  100     8.3374 0.025621     8.3273 0.010480    2.4448
3 1000     8.3330 0.002388     8.3334 0.001075    2.2223
```

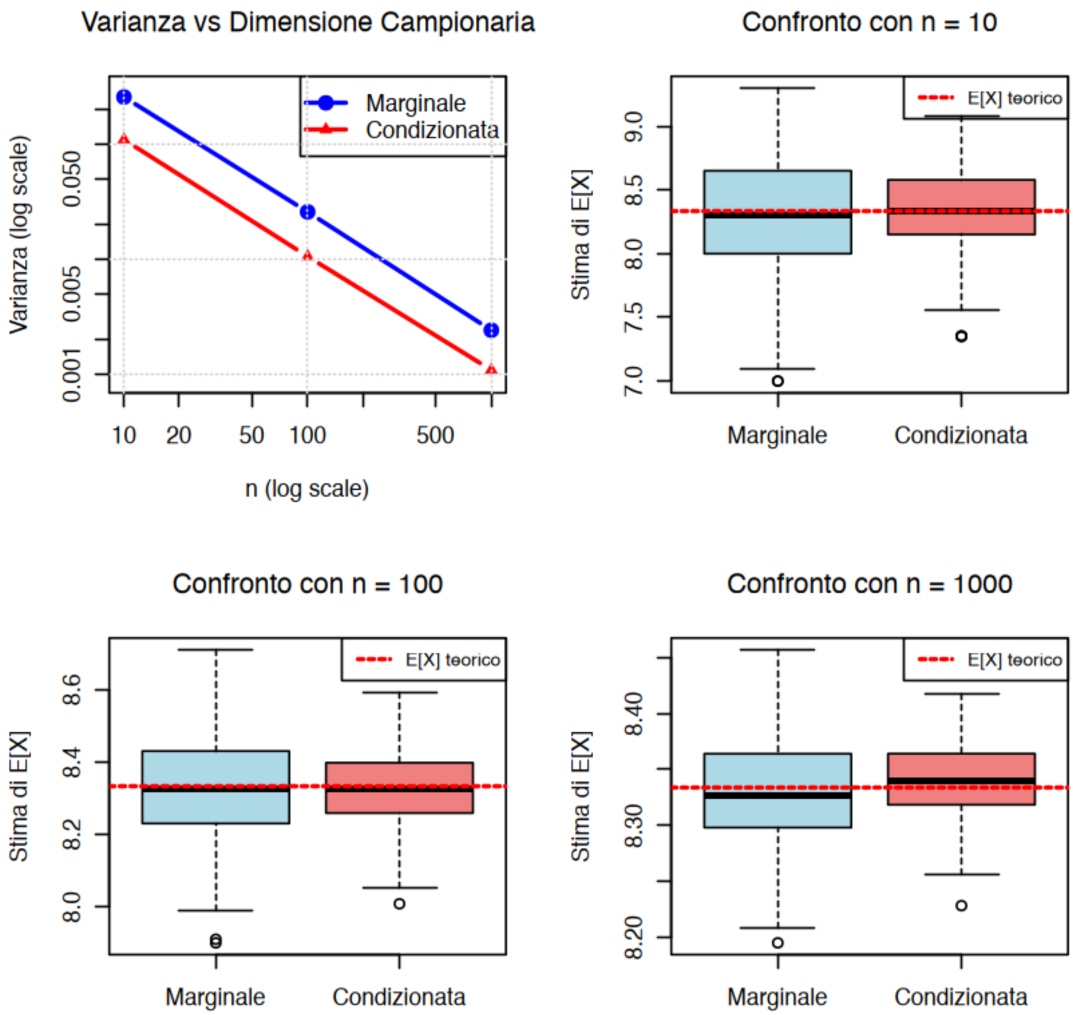

```
=== TABELLA RIASSUNTIVA ===
     n Media_Marg Var_Marg Media_Cond Var_Cond Ratio_Var
1   10     8.3254 0.256591     8.3216 0.109322    2.3471
2  100     8.3374 0.025621     8.3273 0.010480    2.4448
3 1000     8.3330 0.002388     8.3334 0.001075    2.2223
```


[74]:

```
# Visualizzazione avanzata dei risultati
cat("\n=== VISUALIZZAZIONE RISULTATI ===\n")

# Setup grafico principale
par(mfrow = c(2, 2))

# Grafico 1: Varianze vs dimensione campionaria (log-log)
plot(risultati_varianza$n, risultati_varianza$Var_Marg,
     log = "xy",
     type = "b", pch = 19, col = "blue", lwd = 2,
     main = "Varianza vs Dimensione Campionaria",
     xlab = "n (log scale)", ylab = "Varianza (log scale)",
     ylim = c(min(c(risultati_varianza$Var_Marg, risultati_varianza$Var_Cond))␣
 ↪* 0.8,
              max(c(risultati_varianza$Var_Marg, risultati_varianza$Var_Cond))␣
 ↪* 1.2))

lines(risultati_varianza$n, risultati_varianza$Var_Cond,
      type = "b", pch = 17, col = "red", lwd = 2)

legend("topright", c("Marginale", "Condizionata"),
       col = c("blue", "red"), pch = c(19, 17), lwd = 2)
grid()
```

```r
# Grafici 2-4: Box plot per ogni valore di n
for(i in 1:length(n_vals)) {
    n_curr <- n_vals[i]

    # Ri-simulazione per la visualizzazione (con seed fisso)
    set.seed(456 + i)

    meanx_marg <- numeric(200)  # Campione più piccolo per visualizzazione
    meanx_cond <- numeric(200)

    for(isim in 1:200) {
        meanx_marg[isim] <- mean(rbb_func(n_curr, m, a, b))
        p_temp <- rbeta(n_curr, a, b)
        meanx_cond[isim] <- mean(m * p_temp)
    }

    # Box plot per confronto
    boxplot(list(Marginale = meanx_marg, Condizionata = meanx_cond),
            main = paste("Confronto con n =", n_curr),
            ylab = "Stima di E[X]",
            col = c("lightblue", "lightcoral"))

    abline(h = media_teorica, col = "red", lwd = 2, lty = 2)
    legend("topright", "E[X] teorico", col = "red", lwd = 2, lty = 2, cex = 0.8)
}

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

```
=== VISUALIZZAZIONE RISULTATI ===
```



[78]:
```
# ============================================================================
# ANALISI DETTAGLIATA DELLE VARIANZE - DUE METODI
# ============================================================================

cat("\n=== ANALISI VARIANZA: DUE METODI A CONFRONTO ===\n")
cat("Metodo 1: Var(h(X)) / n dove h(X) = X\n")
cat("Metodo 2: Var(stimatore) calcolata empiricamente\n\n")

# Parametri per confronto dettagliato
n_analisi <- 1000
nstarstar <- 500  # Numero di repliche per metodo 2

# Analisi teorica
```

```r
var_teorica_betabin <- varianza_teorica
var_teorica_p <- a * b / ((a + b)^2 * (a + b + 1))
var_teorica_mp <- m^2 * var_teorica_p

cat("=== VERIFICA TEORICA ===\n")
cat("Var(p) teorica:", round(var_teorica_p, 6), "\n")
cat("Var(mp) teorica:", round(var_teorica_mp, 4), "\n")
cat("Var(X) teorica (Beta-Bin):", round(var_teorica_betabin, 4), "\n")
cat("Rapporto Var(X)/Var(mp):", round(var_teorica_betabin/var_teorica_mp, 4), "\n\n")

## METODO 1 - MARGINALE
cat("--- MARGINALE ---\n")

# Campione per stima della varianza di X
set.seed(789)
xstar_marg <- rbb_func(n_analisi, m, a, b)
ex_marg <- mean(xstar_marg)
var_x_marg <- var(xstar_marg)  # Var(X)
stimatore1_marg <- var_x_marg / n_analisi  # Var(X̄) = Var(X)/n

cat("Metodo 1 - Var(X̄) = Var(X)/n:\n")
cat("  Var(X) stimata:", round(var_x_marg, 4), "\n")
cat("  Var(X̄) = Var(X)/n:", round(stimatore1_marg, 6), "\n")

# METODO 2 - Varianza empirica dello stimatore
vec_mean_marg <- numeric(nstarstar)
for(isim in 1:nstarstar) {
    xtemp <- rbb_func(n_analisi, m, a, b)
    vec_mean_marg[isim] <- mean(xtemp)
}
stimatore2_marg <- var(vec_mean_marg)

cat("Metodo 2 - Var empirica di", nstarstar, "stime:\n")
cat("  Var(X̄) empirica:", round(stimatore2_marg, 6), "\n")
cat("  Rapporto Metodo1/Metodo2:", round(stimatore1_marg/stimatore2_marg, 4), "\n\n")

## METODO 1 - CONDIZIONATA
cat("--- CONDIZIONATA ---\n")

# Per il metodo condizionata: h(p) = mp
set.seed(790)
p_star <- rbeta(n_analisi, a, b)
cond_mean <- m * p_star  # h(p) = mp
ex_cond <- mean(cond_mean)
var_h_cond <- var(cond_mean)  # Var(h(p)) = Var(mp) = m²Var(p)
```

```r
stimatore1_cond <- var_h_cond / n_analisi

cat("Metodo 1 - Var(E[X|p]) = Var(mp):\n")
cat("  Var(mp) stimata:", round(var_h_cond, 4), "\n")
cat("  Var(media) = Var(mp)/n:", round(stimatore1_cond, 6), "\n")

# METODO 2 - Varianza empirica dello stimatore
vec_mean_cond <- numeric(nstarstar)
for(isim in 1:nstarstar) {
    p_temp <- rbeta(n_analisi, a, b)
    vec_mean_cond[isim] <- mean(m * p_temp)
}
stimatore2_cond <- var(vec_mean_cond)

cat("Metodo 2 - Var empirica di", nstarstar, "stime:\n")
cat("  Var empirica:", round(stimatore2_cond, 6), "\n")
cat("  Rapporto Metodo1/Metodo2:", round(stimatore1_cond/stimatore2_cond, 4), "\n\n")

# CONFRONTO FINALE
cat("=== CONFRONTO FINALE ===\n")
cat("Varianza Marginale (Metodo 1):", round(stimatore1_marg, 6), "\n")
cat("Varianza Condizionata (Metodo 1):", round(stimatore1_cond, 6), "\n")
cat("Rapporto Marg/Cond:", round(stimatore1_marg/stimatore1_cond, 4), "\n\n")
```

```
=== ANALISI VARIANZA: DUE METODI A CONFRONTO ===
Metodo 1: Var(h(X)) / n dove h(X) = X
Metodo 2: Var(stimatore) calcolata empiricamente

=== VERIFICA TEORICA ===
Var(p) teorica: 0.010684
Var(mp) teorica: 1.0684
Var(X) teorica (Beta-Bin): 2.3504
Rapporto Var(X)/Var(mp): 2.2

--- MARGINALE ---
Metodo 1: Var(h(X)) / n dove h(X) = X
Metodo 2: Var(stimatore) calcolata empiricamente

=== VERIFICA TEORICA ===
Var(p) teorica: 0.010684
Var(mp) teorica: 1.0684
Var(X) teorica (Beta-Bin): 2.3504
Rapporto Var(X)/Var(mp): 2.2

--- MARGINALE ---
Metodo 1 - Var(X<U+0304>) = Var(X)/n:
```

```
  Var(X) stimata: 2.3825
  Var(X<U+0304>) = Var(X)/n: 0.002383
Metodo 1 - Var(X<U+0304>) = Var(X)/n:
  Var(X) stimata: 2.3825
  Var(X<U+0304>) = Var(X)/n: 0.002383
Metodo 2 - Var empirica di 500 stime:
  Var(X<U+0304>) empirica: 0.002307
  Rapporto Metodo1/Metodo2: 1.0329

--- CONDIZIONATA ---
Metodo 1 - Var(E[X|p]) = Var(mp):
  Var(mp) stimata: 1.0024
  Var(media) = Var(mp)/n: 0.001002
  Var(X<U+0304>) empirica: 0.002307
  Rapporto Metodo1/Metodo2: 1.0329

--- CONDIZIONATA ---
Metodo 1 - Var(E[X|p]) = Var(mp):
  Var(mp) stimata: 1.0024
  Var(media) = Var(mp)/n: 0.001002
Metodo 2 - Var empirica di 500 stime:
  Var empirica: 0.001093
  Rapporto Metodo1/Metodo2: 0.9175

=== CONFRONTO FINALE ===
Varianza Marginale (Metodo 1): 0.002383
Varianza Condizionata (Metodo 1): 0.001002
Rapporto Marg/Cond: 2.3768
```

1.5 Congiunte e marginali

Come nel punto precedente, assumete che

$$X|p \sim Bin(m,p) \qquad p \sim Beta(a,b)$$

Simulate n campioni dalla congiunta

$$f(x,p) = f(x|p)f(p)$$

simulando $p_i \sim Beta(a,b)$ e poi $X_i|p_i \sim Bin(m,p_i)$

Verificate che i campioni $(x_1,\dots,x_n)$ provengono dalla marginale di X (questo vi da anche un modo facile per simualre da una marginale quando si conoscono le opportune condizionate). Potete usare il seguente risultato

$$P(X \in A) = \int_A f(x)\lambda(x) = \int_{\mathcal{X}} 1_A(x) f(x) d\lambda(x) = \int_{\mathcal{X}}\int_p 1_A(x) f(x,p) d\lambda(p) d\lambda(x)$$

per stimare la $P(X=a)$

1.5.1 Soluzione

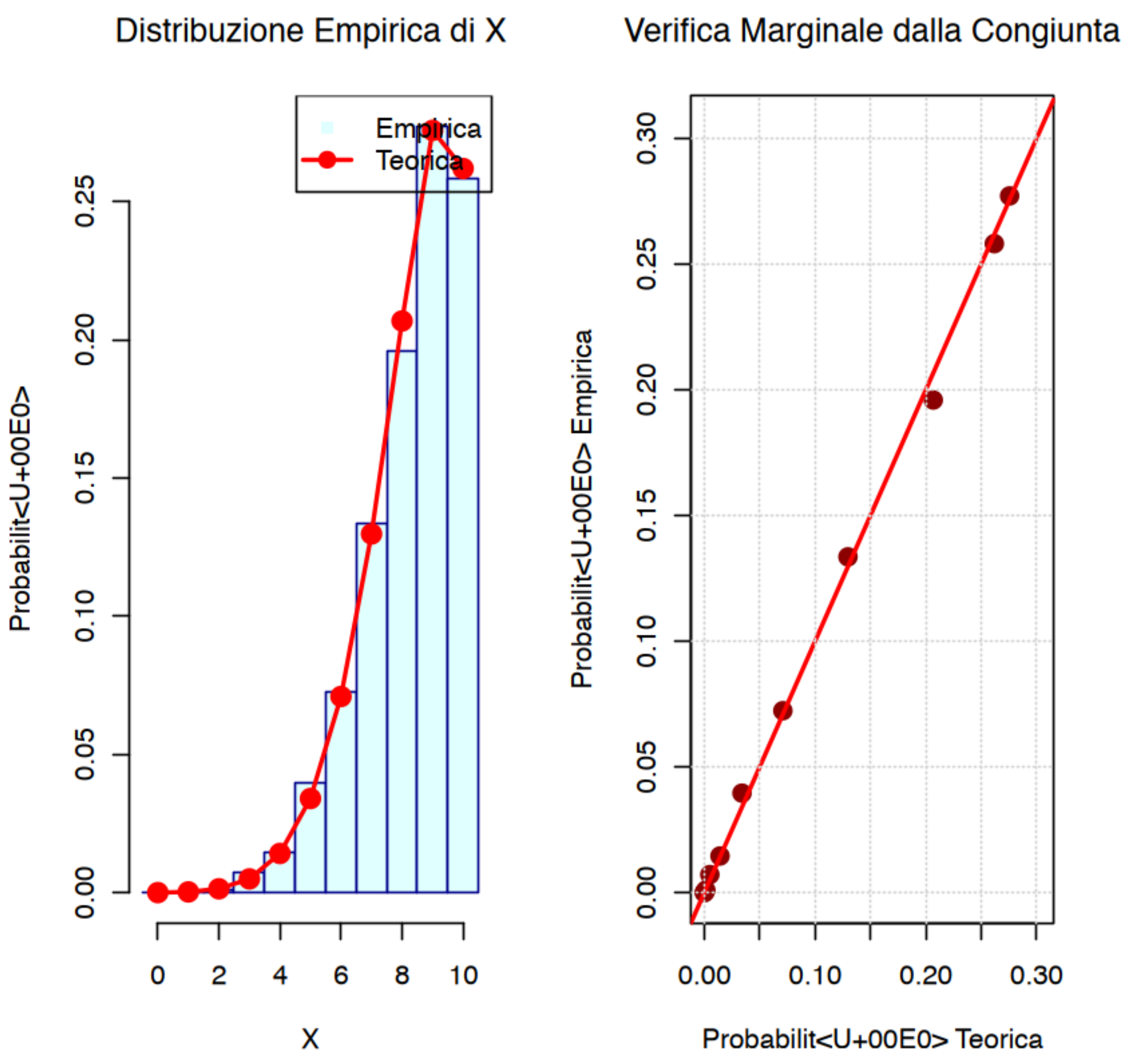
[81]:
```r
# ==============================================================================
# VERIFICA TRAMITE SIMULAZIONE DALLA CONGIUNTA
# ==============================================================================

cat("\n=== VERIFICA: SIMULAZIONE DALLA CONGIUNTA ===\n")
cat("Simulando (p,X) dalla congiunta f(x,p) = f(x|p)f(p)\n")
cat("e verificando che i campioni X seguono la marginale\n\n")

# Gestione della funzione dbb (densità beta-binomiale)
if(exists("tailrank_available") && tailrank_available) {
    tryCatch({
        # Test se dbb è disponibile da TailRank
        test_dbb <- dbb(0, m, a, b)
        dbb_available <- TRUE
        cat("Usando dbb() da TailRank\n")
    }, error = function(e) {
        dbb_available <- FALSE
    })
} else {
    dbb_available <- FALSE
}

if(!dbb_available) {
    # Usa la nostra implementazione
    dbb <- dbetabin
    cat("Usando implementazione custom di dbetabin\n")
}

# Simulazione dalla congiunta
n_congiunta <- 5000
set.seed(800)

# Simulo p ~ Beta(a,b) poi X|p ~ Binomial(m,p)
p_campioni <- rbeta(n_congiunta, a, b)
x_campioni <- numeric(n_congiunta)

for(i in 1:n_congiunta) {
    x_campioni[i] <- rbinom(1, m, p_campioni[i])
}

# Calcolo delle probabilità empiriche e teoriche
stima_marginale <- numeric(m + 1)
densita_marginale <- numeric(m + 1)

for(k in 0:m) {
    # Probabilità empirica: P(X = k)
```

```r
    stima_marginale[k + 1] <- mean(x_campioni == k)
    # Probabilità teorica dalla Beta-Binomiale
    densita_marginale[k + 1] <- dbb(k, m, a, b)
}

# Visualizzazione
par(mfrow = c(1, 2))

# Istogramma dei campioni X con sovrapposizione teorica
hist(x_campioni, breaks = -0.5:(m+0.5), freq = FALSE,
     col = "lightcyan", border = "darkblue",
     main = "Distribuzione Empirica di X",
     xlab = "X", ylab = "Probabilità")

# Sovrapposizione delle probabilità teoriche
points(0:m, densita_marginale, pch = 19, col = "red", cex = 1.5)
lines(0:m, densita_marginale, col = "red", lwd = 2)
legend("topright", c("Empirica", "Teorica"),
       col = c("lightcyan", "red"),
       pch = c(15, 19), lwd = c(NA, 2))

# Scatter plot: empirica vs teorica
plot(densita_marginale, stima_marginale,
     pch = 20, cex = 2, col = "darkred",
     main = "Verifica Marginale dalla Congiunta",
     xlab = "Probabilità Teorica",
     ylab = "Probabilità Empirica",
     xlim = c(0, max(densita_marginale) * 1.1),
     ylim = c(0, max(stima_marginale) * 1.1))
abline(a = 0, b = 1, col = "red", lwd = 2)
grid()

par(mfrow = c(1, 1))

# Tabella di confronto
comparison_congiunta <- data.frame(
    k = 0:m,
    Empirica = round(stima_marginale, 4),
    Teorica = round(densita_marginale, 4),
    Errore = round(abs(stima_marginale - densita_marginale), 4)
)

print(comparison_congiunta)

cat("\n=== CONCLUSIONI ESERCIZIO BETA-BINOMIALE ===\n")
cat("1. La simulazione dalla congiunta produce campioni dalla marginale␣
 ↪corretta\n")
```

```r
cat("2. I due metodi di stima di E[X] convergono al valore teorico\n")
cat("3. La varianza del metodo condizionato è generalmente diversa da quella marginale\n")
cat("4. Il metodo di simulazione dalla congiunta è più efficiente computazionalmente\n")
```

```
=== VERIFICA: SIMULAZIONE DALLA CONGIUNTA ===
Simulando (p,X) dalla congiunta f(x,p) = f(x|p)f(p)
e verificando che i campioni X seguono la marginale

Usando dbb() da TailRank
Simulando (p,X) dalla congiunta f(x,p) = f(x|p)f(p)
e verificando che i campioni X seguono la marginale

Usando dbb() da TailRank
    k Empirica Teorica Errore
1   0   0.0000  0.0000 0.0000
2   1   0.0002  0.0003 0.0001
3   2   0.0010  0.0014 0.0004
4   3   0.0072  0.0050 0.0022
5   4   0.0146  0.0142 0.0004
6   5   0.0396  0.0341 0.0055
7   6   0.0724  0.0709 0.0015
8   7   0.1336  0.1297 0.0039
9   8   0.1960  0.2068 0.0108
10  9   0.2772  0.2757 0.0015
11 10   0.2582  0.2619 0.0037

=== CONCLUSIONI ESERCIZIO BETA-BINOMIALE ===
1. La simulazione dalla congiunta produce campioni dalla marginale corretta
2. I due metodi di stima di E[X] convergono al valore teorico
3. La varianza del metodo condizionato <U+00E8> generalmente diversa da quella
marginale
4. Il metodo di simulazione dalla congiunta <U+00E8> pi<U+00F9> efficiente
computazionalmente
    k Empirica Teorica Errore
1   0   0.0000  0.0000 0.0000
2   1   0.0002  0.0003 0.0001
3   2   0.0010  0.0014 0.0004
4   3   0.0072  0.0050 0.0022
5   4   0.0146  0.0142 0.0004
6   5   0.0396  0.0341 0.0055
7   6   0.0724  0.0709 0.0015
8   7   0.1336  0.1297 0.0039
9   8   0.1960  0.2068 0.0108
10  9   0.2772  0.2757 0.0015
11 10   0.2582  0.2619 0.0037
```

```
=== CONCLUSIONI ESERCIZIO BETA-BINOMIALE ===
1. La simulazione dalla congiunta produce campioni dalla marginale corretta
2. I due metodi di stima di E[X] convergono al valore teorico
3. La varianza del metodo condizionato <U+00E8> generalmente diversa da quella
marginale
4. Il metodo di simulazione dalla congiunta <U+00E8> pi<U+00F9> efficiente
computazionalmente
```



1.6 Area del cerchio

Sappiamo che un cerchio di raggio unitario nel piano (x, y) ha area

$$2\int_{-1}^{1}\sqrt{1-x^2}dx = 4\int_{0}^{1}\sqrt{1-x^2}dx = \pi$$

Usando Monte Carlo, stimare il valore di π assumendo

1. $x \sim U(-1, 1)$
2. $x \sim B(a, b)$
3. $x \sim U(-1, 1)$ e $y \sim U(0, 1)$

Per risolvere i punti 1 e 2, notate che potete scrivere gli integrali come

$$\int_X g(x)dx = \int_X \frac{g(x)}{f(x)}f(x)dx$$

dove $f(x)$ è una densità che assume valori diversi da zero in corrispondenza dei punti in cui $g(x)$ è diversa da zero, tranne che su un insieme di misura nulla.

Per il punto 3 invece ricordatevi che il Monte Carlo vale anche per funzioni di più variabile

[83]:
```
# ==============================================================================
# STIMA DI   TRAMITE MONTE CARLO - TRE METODI
# ==============================================================================

cat("=== STIMA DI   CON MONTE CARLO ===\n")
cat("Obiettivo: stimare   usando l'area del cerchio unitario\n")
cat("Area teorica =  , quindi stimando l'area otteniamo  \n\n")

# Parametri per l'analisi
n_values <- c(1000, 10000, 100000)  # Diverse dimensioni campionarie
n_replications <- 50               # Numero di repliche per analisi variabilità

# Storage per risultati
risultati_pi <- data.frame(
    Metodo = character(),
    n = integer(),
    Stima_Pi = numeric(),
    Errore_Abs = numeric(),
    Errore_Rel_Perc = numeric(),
    Varianza = numeric()
)

# ==============================================================================
# METODO 1: INTEGRAZIONE CON UNIFORME x   [-1,1]
# ==============================================================================

cat("=== METODO 1: UNIFORME SU [-1,1] ===\n")
cat("Calcolo:   = 4 ¹ √(1-x²) dx = 2  ¹ √(1-x²) dx\n")
cat("Monte Carlo: E[4√(1-X²)] con X ~ U(-1,1)\n\n")
```

```r
metodo1_stime <- matrix(NA, nrow = length(n_values), ncol = n_replications)

for(i in 1:length(n_values)) {
    n <- n_values[i]
    cat("n =", n, "...\n")

    set.seed(100 + i)  # Per riproducibilità
    stime_replica <- numeric(n_replications)

    for(rep in 1:n_replications) {
        x <- runif(n, -1, 1)
        # h(x) = 4√(1-x²) per x   [-1,1]
        stime_replica[rep] <- mean(4 * sqrt(1 - x^2))
    }

    metodo1_stime[i, ] <- stime_replica
    media_stima <- mean(stime_replica)
    var_stima <- var(stime_replica)
    errore_abs <- abs(media_stima - pi)
    errore_rel <- (errore_abs / pi) * 100

    cat("  Stima  :", round(media_stima, 6), "\n")
    cat("  Errore assoluto:", round(errore_abs, 6), "\n")
    cat("  Errore relativo:", round(errore_rel, 4), "%\n")
    cat("  Varianza:", round(var_stima, 6), "\n\n")

    risultati_pi <- rbind(risultati_pi, data.frame(
        Metodo = "Uniforme [-1,1]",
        n = n,
        Stima_Pi = round(media_stima, 6),
        Errore_Abs = round(errore_abs, 6),
        Errore_Rel_Perc = round(errore_rel, 4),
        Varianza = round(var_stima, 6)
    ))
}

# ============================================================================
# METODO 2: INTEGRAZIONE CON BETA
# ============================================================================

cat("=== METODO 2: BETA SU [0,1] ===\n")
cat("Trasformazione: x   [0,1], y = 2x-1   [-1,1]\n")
cat("Integral:   = 4 ¹ √(1-(2x-1)²) dx\n")
cat("Monte Carlo: E[4√(1-(2X-1)²)/f_Beta(X)] con X ~ Beta(a,b)\n\n")

# Parametri della Beta
```

```r
a_beta <- 2
b_beta <- 3
cat("Parametri Beta: a =", a_beta, ", b =", b_beta, "\n\n")

metodo2_stime <- matrix(NA, nrow = length(n_values), ncol = n_replications)

for(i in 1:length(n_values)) {
    n <- n_values[i]
    cat("n =", n, "...\n")

    set.seed(200 + i)
    stime_replica <- numeric(n_replications)

    for(rep in 1:n_replications) {
        x <- rbeta(n, a_beta, b_beta)
        y <- 2*x - 1  # Trasformazione a [-1,1]
        # h(x) = 4√(1-y²) / f_Beta(x) dove f_Beta(x) è la densità Beta
        stime_replica[rep] <- mean(4 * sqrt(1 - y^2) / dbeta(x, a_beta, b_beta))
    }

    metodo2_stime[i, ] <- stime_replica
    media_stima <- mean(stime_replica)
    var_stima <- var(stime_replica)
    errore_abs <- abs(media_stima - pi)
    errore_rel <- (errore_abs / pi) * 100

    cat("  Stima  :", round(media_stima, 6), "\n")
    cat("  Errore assoluto:", round(errore_abs, 6), "\n")
    cat("  Errore relativo:", round(errore_rel, 4), "%\n")
    cat("  Varianza:", round(var_stima, 6), "\n\n")

    risultati_pi <- rbind(risultati_pi, data.frame(
        Metodo = paste0("Beta(", a_beta, ",", b_beta, ")"),
        n = n,
        Stima_Pi = round(media_stima, 6),
        Errore_Abs = round(errore_abs, 6),
        Errore_Rel_Perc = round(errore_rel, 4),
        Varianza = round(var_stima, 6)
    ))
}

# =============================================================================
# METODO 3: METODO HIT-OR-MISS (GEOMETRICO)
# =============================================================================

cat("=== METODO 3: HIT-OR-MISS GEOMETRICO ===\n")
```

```
cat("Area cerchio =   = 4 × P((X,Y) dentro cerchio unitario)\n")
cat("Con (X,Y) ~ U([-1,1] × [0,1])\n")
cat("Condizione: X² + Y²   1\n\n")

metodo3_stime <- matrix(NA, nrow = length(n_values), ncol = n_replications)

for(i in 1:length(n_values)) {
    n <- n_values[i]
    cat("n =", n, "...\n")

    set.seed(300 + i)
    stime_replica <- numeric(n_replications)

    for(rep in 1:n_replications) {
        x <- runif(n, -1, 1)  # X ~ U(-1,1)
        y <- runif(n, 0, 1)   # Y ~ U(0,1)

        # Indicatrice: 1 se (x,y) è dentro il cerchio
        dentro_cerchio <- (x^2 + y^2) <= 1

        # Area = 4 × P(dentro cerchio) = 4 × (# successi / n)
        stime_replica[rep] <- 4 * mean(dentro_cerchio)
    }

    metodo3_stime[i, ] <- stime_replica
    media_stima <- mean(stime_replica)
    var_stima <- var(stime_replica)
    errore_abs <- abs(media_stima - pi)
    errore_rel <- (errore_abs / pi) * 100

    cat("  Stima  :", round(media_stima, 6), "\n")
    cat("  Errore assoluto:", round(errore_abs, 6), "\n")
    cat("  Errore relativo:", round(errore_rel, 4), "%\n")
    cat("  Varianza:", round(var_stima, 6), "\n\n")

    risultati_pi <- rbind(risultati_pi, data.frame(
        Metodo = "Hit-or-Miss",
        n = n,
        Stima_Pi = round(media_stima, 6),
        Errore_Abs = round(errore_abs, 6),
        Errore_Rel_Perc = round(errore_rel, 4),
        Varianza = round(var_stima, 6)
    ))
}

# ==============================================================================
# VISUALIZZAZIONE E CONFRONTO
```

```r
# ============================================================================

cat("=== TABELLA RIASSUNTIVA ===\n")
print(risultati_pi)

# Visualizzazione grafica avanzata
par(mfrow = c(2, 2))

# Grafico 1: Convergenza per ogni metodo
colors <- c("blue", "red", "green")
metodi_nomi <- c("Uniforme [-1,1]", paste0("Beta(", a_beta, ",", b_beta, ")"), "Hit-or-Miss")

plot(NULL, xlim = c(min(n_values), max(n_values)), ylim = c(2.8, 3.4),
     log = "x", main = "Convergenza delle Stime di  ",
     xlab = "n (log scale)", ylab = "Stima di  ")
abline(h = pi, col = "black", lwd = 2, lty = 2)

for(i in 1:3) {
    subset_data <- risultati_pi[risultati_pi$Metodo == metodi_nomi[i], ]
    lines(subset_data$n, subset_data$Stima_Pi,
          col = colors[i], lwd = 2, type = "b", pch = 19)
}

legend("topright", c("  teorico", metodi_nomi),
       col = c("black", colors), lwd = 2, lty = c(2, rep(1, 3)))

# Grafico 2: Errore relativo
plot(NULL, xlim = c(min(n_values), max(n_values)),
     ylim = c(0, max(risultati_pi$Errore_Rel_Perc)),
     log = "x", main = "Errore Relativo %",
     xlab = "n (log scale)", ylab = "Errore Relativo (%)")

for(i in 1:3) {
    subset_data <- risultati_pi[risultati_pi$Metodo == metodi_nomi[i], ]
    lines(subset_data$n, subset_data$Errore_Rel_Perc,
          col = colors[i], lwd = 2, type = "b", pch = 19)
}
legend("topright", metodi_nomi, col = colors, lwd = 2)

# Grafico 3: Varianza
plot(NULL, xlim = c(min(n_values), max(n_values)),
     ylim = c(min(risultati_pi$Varianza), max(risultati_pi$Varianza)),
     log = "xy", main = "Varianza delle Stime",
     xlab = "n (log scale)", ylab = "Varianza (log scale)")

for(i in 1:3) {
```

```r
    subset_data <- risultati_pi[risultati_pi$Metodo == metodi_nomi[i], ]
    lines(subset_data$n, subset_data$Varianza,
          col = colors[i], lwd = 2, type = "b", pch = 19)
}
legend("topright", metodi_nomi, col = colors, lwd = 2)

# Grafico 4: Visualizzazione geometrica del Metodo 3
n_vis <- 1000
set.seed(999)
x_vis <- runif(n_vis, -1, 1)
y_vis <- runif(n_vis, 0, 1)
dentro <- (x_vis^2 + y_vis^2) <= 1

plot(x_vis, y_vis, pch = 20, cex = 0.5,
     col = ifelse(dentro, "red", "blue"),
     main = "Metodo Hit-or-Miss",
     xlab = "X", ylab = "Y",
     xlim = c(-1, 1), ylim = c(0, 1))

# Disegno il quarto di cerchio
theta <- seq(0, pi/2, length.out = 100)
x_circle <- cos(theta)
y_circle <- sin(theta)
lines(x_circle, y_circle, col = "black", lwd = 2)

# Completamento del contorno
lines(c(-1, 1), c(0, 0), col = "black", lwd = 1)  # Asse x
lines(c(-1, -1), c(0, 1), col = "black", lwd = 1)  # Lato sinistro
lines(c(1, 1), c(0, 1), col = "black", lwd = 1)    # Lato destro
lines(c(-1, 1), c(1, 1), col = "black", lwd = 1)   # Lato superiore

stima_vis <- 4 * mean(dentro)
legend("topright",
       c(paste("Dentro cerchio:", sum(dentro)),
         paste("Fuori cerchio:", sum(!dentro)),
         paste("Stima  :", round(stima_vis, 4))),
       col = c("red", "blue", "black"), pch = c(20, 20, NA))

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

```
=== STIMA DI <U+03C0> CON MONTE CARLO ===
Obiettivo: stimare <U+03C0> usando l'area del cerchio unitario
Area teorica = <U+03C0>, quindi stimando l'area otteniamo <U+03C0>

=== METODO 1: UNIFORME SU [-1,1] ===
Calcolo: <U+03C0> = 4<U+222B><U+2080><U+00B9> <U+221A>(1-x<U+00B2>) dx =
2<U+222B><U+208B><U+2081><U+00B9> <U+221A>(1-x<U+00B2>) dx
Monte Carlo: E[4<U+221A>(1-X<U+00B2>)] con X ~ U(-1,1)
```

```
Obiettivo: stimare <U+03C0> usando l'area del cerchio unitario
Area teorica = <U+03C0>, quindi stimando l'area otteniamo <U+03C0>

=== METODO 1: UNIFORME SU [-1,1] ===
Calcolo: <U+03C0> = 4<U+222B><U+2080><U+00B9> <U+221A>(1-x<U+00B2>) dx =
2<U+222B><U+208B><U+2081><U+00B9> <U+221A>(1-x<U+00B2>) dx
Monte Carlo: E[4<U+221A>(1-X<U+00B2>)] con X ~ U(-1,1)

n = 1000 …
  Stima <U+03C0>: 3.143709
  Errore assoluto: 0.002116
  Errore relativo: 0.0674 %
  Varianza: 0.000821

n = 10000 …
  Stima <U+03C0>: 3.140155
  Errore assoluto: 0.001438
  Errore relativo: 0.0458 %
  Varianza: 7.8e-05

n = 1e+05 …
  Stima <U+03C0>: 3.141878
  Errore assoluto: 0.000285
  Errore relativo: 0.0091 %
  Varianza: 8e-06

=== METODO 2: BETA SU [0,1] ===
Trasformazione: x <U+2208> [0,1], y = 2x-1 <U+2208> [-1,1]
Integral: <U+03C0> = 4<U+222B><U+2080><U+00B9> <U+221A>(1-(2x-1)<U+00B2>) dx
Monte Carlo: E[4<U+221A>(1-(2X-1)<U+00B2>)/f_Beta(X)] con X ~ Beta(a,b)

Parametri Beta: a = 2 , b = 3

=== METODO 2: BETA SU [0,1] ===
Trasformazione: x <U+2208> [0,1], y = 2x-1 <U+2208> [-1,1]
Integral: <U+03C0> = 4<U+222B><U+2080><U+00B9> <U+221A>(1-(2x-1)<U+00B2>) dx
Monte Carlo: E[4<U+221A>(1-(2X-1)<U+00B2>)/f_Beta(X)] con X ~ Beta(a,b)

Parametri Beta: a = 2 , b = 3

n = 1000 …
  Stima <U+03C0>: 3.142515
  Errore assoluto: 0.000922
  Errore relativo: 0.0293 %
  Varianza: 0.024605

n = 10000 …
```

```
  Stima <U+03C0>: 3.136127
  Errore assoluto: 0.005466
  Errore relativo: 0.174 %
  Varianza: 0.00173

n = 1e+05 …
  Stima <U+03C0>: 3.143936
  Errore assoluto: 0.002343
  Errore relativo: 0.0746 %
  Varianza: 0.000269

=== METODO 3: HIT-OR-MISS GEOMETRICO ===
Area cerchio = <U+03C0> = 4 <U+00D7> P((X,Y) dentro cerchio unitario)
Con (X,Y) ~ U([-1,1] <U+00D7> [0,1])
Condizione: X<U+00B2> + Y<U+00B2> <U+2264> 1

n = 1000 …
  Stima <U+03C0>: 3.142515
  Errore assoluto: 0.000922
  Errore relativo: 0.0293 %
  Varianza: 0.024605

n = 10000 …
  Stima <U+03C0>: 3.136127
  Errore assoluto: 0.005466
  Errore relativo: 0.174 %
  Varianza: 0.00173

n = 1e+05 …
  Stima <U+03C0>: 3.143936
  Errore assoluto: 0.002343
  Errore relativo: 0.0746 %
  Varianza: 0.000269

=== METODO 3: HIT-OR-MISS GEOMETRICO ===
Area cerchio = <U+03C0> = 4 <U+00D7> P((X,Y) dentro cerchio unitario)
Con (X,Y) ~ U([-1,1] <U+00D7> [0,1])
Condizione: X<U+00B2> + Y<U+00B2> <U+2264> 1

n = 1000 …
  Stima <U+03C0>: 3.1436
  Errore assoluto: 0.002007
  Errore relativo: 0.0639 %
  Varianza: 0.002488

n = 10000 …
  Stima <U+03C0>: 3.141192
  Errore assoluto: 0.000401
```

```
  Errore relativo: 0.0128 %
  Varianza: 0.000337

n = 1e+05 …
  Stima <U+03C0>: 3.142972
  Errore assoluto: 0.001379
  Errore relativo: 0.0439 %
  Varianza: 2.3e-05

=== TABELLA RIASSUNTIVA ===
           Metodo     n Stima_Pi Errore_Abs Errore_Rel_Perc Varianza
1 Uniforme [-1,1] 1e+03 3.143709   0.002116          0.0674 0.000821
2 Uniforme [-1,1] 1e+04 3.140155   0.001438          0.0458 0.000078
3 Uniforme [-1,1] 1e+05 3.141878   0.000285          0.0091 0.000008
4       Beta(2,3) 1e+03 3.142515   0.000922          0.0293 0.024605
5       Beta(2,3) 1e+04 3.136127   0.005466          0.1740 0.001730
6       Beta(2,3) 1e+05 3.143936   0.002343          0.0746 0.000269
7     Hit-or-Miss 1e+03 3.143600   0.002007          0.0639 0.002488
8     Hit-or-Miss 1e+04 3.141192   0.000401          0.0128 0.000337
9     Hit-or-Miss 1e+05 3.142972   0.001379          0.0439 0.000023
n = 1000 …
  Stima <U+03C0>: 3.1436
  Errore assoluto: 0.002007
  Errore relativo: 0.0639 %
  Varianza: 0.002488

n = 10000 …
  Stima <U+03C0>: 3.141192
  Errore assoluto: 0.000401
  Errore relativo: 0.0128 %
  Varianza: 0.000337

n = 1e+05 …
  Stima <U+03C0>: 3.142972
  Errore assoluto: 0.001379
  Errore relativo: 0.0439 %
  Varianza: 2.3e-05

=== TABELLA RIASSUNTIVA ===
           Metodo     n Stima_Pi Errore_Abs Errore_Rel_Perc Varianza
1 Uniforme [-1,1] 1e+03 3.143709   0.002116          0.0674 0.000821
2 Uniforme [-1,1] 1e+04 3.140155   0.001438          0.0458 0.000078
3 Uniforme [-1,1] 1e+05 3.141878   0.000285          0.0091 0.000008
4       Beta(2,3) 1e+03 3.142515   0.000922          0.0293 0.024605
5       Beta(2,3) 1e+04 3.136127   0.005466          0.1740 0.001730
6       Beta(2,3) 1e+05 3.143936   0.002343          0.0746 0.000269
7     Hit-or-Miss 1e+03 3.143600   0.002007          0.0639 0.002488
8     Hit-or-Miss 1e+04 3.141192   0.000401          0.0128 0.000337
```

```
9      Hit-or-Miss 1e+05 3.142972    0.001379           0.0439 0.000023
```