Università degli studi di Milano–Bicocca
Scuola di Economia e Statistica

Corso di Laurea in
Scienze Statistiche ed Economiche

Metodi di campionamento per distribuzioni univariate

Relatore: Dott. Tommaso Rigon

Tesi di laurea di:
Riccardo Cicuttin
Matricola N. 879735

Anno Accademico 2023/2024

Indice

Capitolo 1

Generazione di numeri (pseudo) casuali

1.1 Introduzione ai metodi Monte Carlo

Con l'espressione metodi Monte Carlo ci si riferisce ad un insieme di tecniche numeriche basate sull'utilizzo di numeri generati casualmente.
L'impiego di queste tecniche risulta molto vasto, possono essere infatti utilizzate per simulare fenomeni complessi, ridurre l'utilizzo di risorse di calcolo ed in particolare modo per fornire soluzioni numeriche a problemi che non dispongono di soluzioni analitiche.
L'idea dietro questi metodi è che utilizzando un campione di valori che riflettano il comportamento del fenomeno in esame, si possano ottenere informazioni su questo studiando le proprietà statistiche del campione. Dal momento quindi che non forniscono risoluzioni dirette ed esatte, ma utilizzano campioni generati casualmente per riprodurre il sistema oggetto di studio, i metodi Monte Carlo sono delle tecniche simulative. Facendo impiego di numeri casuali, necessariamente anche i risultati che producono lo sono, i quali quindi non assumendo valori deterministici risultano essere per loro natura delle approssimazioni.

In modo da comprendere il ruolo fondamentale svolto dai numeri generati casualmente in questi metodi, può risultare opportuno fornire un esempio di uno dei loro impieghi più comuni, ovvero la risoluzione numerica di integrali non risolvibili analiticamente. Essendo in un contesto statistico, si può considerare la risoluzione di un integrale che fornisca il valore atteso di una trasformazione $g : \mathbb{R} \to \mathbb{R}$ di una variabile aleatoria $X \sim f(x)$ con supporto $\mathcal{D}$:

$$\int_{\mathcal{D}} g(x)f(x)\mathrm{d}x = \mathbb{E}[g(X)].$$

Utilizzando l'integrazione Monte Carlo, si genera un campione $X_1, \ldots, X_N$ di valori i.i.d. da una densità $f(x)$, si calcolano i valori $g(X_1), \ldots, g(X_N)$ e se ne effettua la media campionaria

$$\frac{1}{N} \sum_{i=1}^{N} g\left(X_i\right) \approx \mathbb{E}[g(X)],$$

la quale fornisce una approssimazione Monte Carlo dell'integrale analizzato.
Infatti per la legge dei grandi numeri questa quantità, facendo tendere ad infinito la numerosità del campione, converge al valore effettivo del valore atteso:

$$\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} g\left(X_i\right) = \mathbb{E}[g(X)].$$

Dove l'uguaglianza viene detta "quasi certa", ovvero vale:

$$P\left(\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} g\left(X_i\right) = \mathbb{E}[g(X)] \right) = 1.$$

Già da questo semplice esempio emerge l'importanza cruciale della generazione di un campione casuale che rifletta nel modo più fedele possibile la reale distribuzione della variabile aleatoria considerata. Inoltre, si nota anche come la qualità dell'approssimazione dipenda da quanto più sia elevata la numerosità del campione.
Questi aspetti sono comuni a tutti gli utilizzi di simulazioni Monte Carlo, per questa ragione si ha avuto la necessità di elaborare tecniche che garantissero di ottenere valori in linea con le distribuzioni di interesse, facendo possibilmente utilizzo di una limitata quantità di risorse computazionali.

1.2 Generare campioni casuali da una distribuzione

1.2.1 Numeri casuali e numeri pseudo casuali

La questione della generazione di numeri casuali appare centrale nello studio ed utilizzo dei metodi Monte Carlo. Ognuno di questi si basa infatti sulla disponibilità di un numero arbitrariamente elevato di realizzazioni delle variabili casuali i.i.d. $X_1, \ldots, X_N$ distribuite concordemente ad una distribuzione f.
Attualmente, la generazione di numeri casuali viene effettuata da computer, i quali sono per loro natura macchine deterministiche. In questo caso quindi, se si volessero ottenere numeri *effettivamente* casuali, lo si potrebbe fare soltanto utilizzando un

fenomeno fisico esterno come fonte di casualità. Tuttavia, considerando che i metodi Monte Carlo impiegano quantità anche estremamente elevate di numeri casuali, questo tipo di soluzioni risultano tendenzialmente molto costose ed esigenti in termini di risorse computazionali. Per questa ragione sono dovute essere sviluppate delle alternative più convenienti, ossia algoritmi che potessero produrre queste serie di numeri utilizzando soltanto le risorse offerte da un computer. Essendo implementati su una macchina deterministica, questi algoritmi producono necessariamente valori deterministici, i quali sono per questa ragione chiamati numeri "pseudo casuali".
In questo contesto, una sequenza di numeri generata con queste tecniche, può essere utilizzata come se fosse un campione casuale nel caso in cui, avendo generato $X_1, \ldots, X_N$, la conoscenza dei soli valori di X_N o di $X_1, \ldots, X_N$ non fornisca alcuna informazione utile per ottenere il valore di X_{N+1}.
Questo tipo di algoritmi viene detto "generatore di numeri pseudo casuali" e le caratteristiche che lo definiscono permettono di comprendere le motivazioni per cui i numeri che produce sono a tutti gli effetti deterministici.
L'Ecuyer (1994) fornisce la seguente definizione di generatore di numeri pseudo casuali:

Definizione 1.1 (Generatore di numeri pseudo casuali)**.** Un generatore di numeri pseudo casuali è definito dalla struttura $\mathcal{G} = (S, s_0, T, U, G)$, dove S è un insieme finito di stati, $s_0 \in S$ è lo stato iniziale, $T : S \to S$ è la funzione di transizione, U è l'insieme finito degli output, e $G : S \to U$ è la funzione di output.

Il generatore opera nel modo seguente: parte dallo stato iniziale s_0 (detto seme) e pone $u_0 = G(s_0)$. Successivamente, per $i = 1, 2, \ldots$, siano $s_i = T(s_{i-1})$ e $u_i = G(s_i)$. Si assume che T e G siano facilmente calcolabili. La sequenza (u_i) è l'output del generatore e i valori u_i sono le osservazioni.
Essendo il valore iniziale s_0 dato, l'ouput finale del generatore (u_i) risulta sempre invariabilmente lo stesso per ogni dato valore del seme: è per questo motivo che i valori ottenuti sono a tutti gli effetti deterministici e soprattutto riproducibili.
Inoltre, essendo l'insieme S finito, il generatore dovrà necessariamente dover rivisitare uno stato già visitato in precedenza, per cui produrrà nuovamente una serie di numeri già generata: per loro natura dunque questi algoritmi sono periodici, dove con periodo viene identificato il minor numero intero di passi compiuto per tornare a rivisitare uno stesso stato. Noto questo aspetto, risulta quindi evidente come un buon generatore di numeri pseudo casuali debba possedere il periodo più ampio possibile, date le esigenze di moli elevate di numeri casuali per l'applicazione dei metodi Monte Carlo.

La caratteristica di maggiore rilevanza per un generatore di numeri casuali è tuttavia che la sequenza che questi producano si comporti come se provenisse effettivamente da una sequenza di variabili casuali i.i.d. distribuita uniformemente su U. Sempre L'Ecuyer (1994) afferma infatti come l'insieme U sia solitamente un insieme di interi della forma $\{0, \ldots, m-1\}$ o un insieme finito di valori tra 0 ed 1 che approssimi la distribuzione $\mathcal{U}(0,1)$. Questo ultimo aspetto in particolare risulta di cruciale importanza nello sviluppo di algoritmi che permettano di campionare valori casuali da qualsiasi distribuzione.

1.2.2 Campionamento da $\mathcal{U}(0,1)$

La possibilità di generare numeri pseudo casuali che simulino un campione estratto da una variabile casuale $\mathcal{U}(0,1)$ è a tutti gli effetti ciò che permette di generare successivamente un campione da qualsiasi altra distribuzione.
Dal momento che, come descritto, per ottenere questi valori vengono utilizzati algoritmi deterministici, in modo da poter poi procedere ad ottenere campioni non uniformi, deve necessariamente essere effettuata la seguente assunzione, fornita da Devroye (1986): "esiste un generatore perfetto di numeri pseudo casuali uniformi su (0,1), ossia un generatore capace di produrre una sequenza $\{U_1, U_2, \ldots\}$ di variabili casuali con distribuzione uniforme su (0,1)".
Questa assunzione risulta essere il blocco fondamentale su cui si basa la generazione di campioni da qualsiasi distribuzione. Nella pratica si tratta in ogni caso di una assunzione forte, e la validità di questi generatori in termini di qualità delle sequenze generate si riduce agli aspetti testabili di queste, dunque alla verifica che la sequenza $U_1, \ldots, U_n$ conduca all'accettazione dell'ipotesi:

$$\mathcal{H}_0 : U_1, \ldots, U_n \text{ sono i.i.d. a } \mathcal{U}(0,1).$$

Attualmente esistono numerosi algoritmi che soddisfano questa condizione, per cui nonostante l'assunzione iniziale effettuata risulti poco realistica, in ogni software è già stato implementato un generatore che rispetti le caratteristiche desiderate. In particolare, nelle analisi successive, viene utilizzato il software R ed il suo generatore di numeri pseudo casuali.
I valori ottenuti con l'impiego di questi generatori svolgono un ruolo di primaria importanza in quanto è attraverso la loro manipolazione che è possibile campionare dei valori da qualsiasi altra distribuzione che abbia come supporto un sottoinsieme $\mathcal{D} \subseteq \mathbb{R}$.

1.2.3 Campionamento da distribuzioni non uniformi

Considerando un contesto unidimensionale, con campionamento da distribuzioni non uniformi si intende, data una variabile casuale $X \sim f(x)$ con supporto $\mathcal{D} \subseteq \mathbb{R}$, la generazione di una sequenza di numeri reali che simuli una sequenza di realizzazioni di una serie di variabili casuali $X_1, \ldots, X_N$ che siano i.i.d. a $X \sim f(x)$.
Per poter svolgere questo compito, sono stati elaborati svariati metodi di campionamento, i quali possono essere definiti sostanzialmente come algoritmi che trasformano uno o più valori pseudo casuali uniformi in una osservazione della variabile casuale di interesse. Gli algoritmi di generazione di numeri casuali da distribuzioni non uniformi sono quindi tutti costruiti a partire da algoritmi di generazione di numeri casuali da distribuzioni uniformi.
Questi algoritmi si suddividono in due macro categorie: algoritmi diretti ed indiretti, classificati in base al modo in cui i campioni sono generati dalla distribuzione di probabilità considerata.

Metodi di campionamento diretti

I metodi di campionamento diretti indicano tutti quei metodi che generano un campione casuale dalla distribuzione di interesse applicando una trasformazione che mappa direttamente uno o più valori pseudo casuali uniformi in una realizzazione della variabile casuale di interesse, senza fare uso di approssimazioni o test. Questi metodi in generale sfruttano le proprietà analitiche delle funzioni di densità delle distribuzioni considerate o le relazioni che intercorrono tra le diverse distribuzioni, per cui molti di questi metodi diretti sono specifici per le distribuzioni considerate.
In questa categoria, il metodo più noto e sicuramente il più utilizzato nei contesti in cui esso è applicabile, risulta essere il metodo dell'inversione. Nonostante esistano numerosi metodi diretti, questo è il più esemplificativo per quanto riguarda l'utilizzo delle proprietà analitiche delle funzioni di densità per il campionamento e la relazione che intercorre tra i metodi di campionamento e la generazione di valori pseudo casuali uniformi in (0,1).
Questo metodo viene descritto ad esempio in Robert & Casella (2004) facendo uso della seguente definizione.

Definizione 1.2. Data una variabile casuale continua X, con funzione di densità $f(x)$ e funzione di ripartizione $F(x) = \mathrm{P}\{X \leq x\} = \int_{-\infty}^{x} f(v)dv$, sempre monotona

crescente, la sua inversa generalizzata viene definita come

$$F^-(u) = \inf\{x : F(x) \geq u\}. \tag{1.1}$$

Questa definizione viene impiegata per ottenere la definizione di una qualsiasi variabile casuale univariata come trasformazione di una variabile casuale $\mathcal{U}(0,1)$.

Teorema 1.1. *Siano date la variabile casuale $U \sim \mathcal{U}(0,1)$ e la variabile casuale X con funzione di ripartizione $F(x)$. La variabile casuale $F^-(U)$ possiede distribuzione F.*

Dimostrazione. *$\forall u \in (0,1)$ e $\forall x \in F^-((0,1))$, l'inversa generalizzata soddisfa*

$$F\left(F^-(u)\right) \geq u \quad e \quad F^-(F(x)) \leq x.$$

Da cui si ottiene

$$\left\{(u,x) : F^-(u) \leq x\right\} = \{(u,x) : F(x) \geq u\}. \tag{1.2}$$

ed inoltre

$$P\left(F^-(U) \leq x\right) = P(U \leq F(x)) = F(x). \tag{1.3}$$

Da questo teorema discende che, in modo da generare un campione x da una variabile casuale $X \sim f$, è sufficiente generare un campione u da $U \sim \mathcal{U}(0,1)$ e successivamente effettuare la trasformazione $x = F^-(u)$.

In questo contesto quindi il campionamento da una qualsiasi distribuzione univariata viene permessa da una trasformazione deterministica di valori pseudo casuali uniformi. A livello intuitivo, ciò che viene fatto è generare ordini casuali di quantili, i quali sono poi utilizzati per ottenere i corrispondenti quantili della distribuzione di interesse, che costituiscono appunto il campione che si voleva generare.

Questo metodo dunque risulta di elevata efficacia, tuttavia presuppone che siano note le forme analitiche di F e F^-, ed inoltre che il calcolo di $F^-(u)$ sia effettuabile in tempi ragionevoli e con un limitato dispendio di risorse. Queste richieste tuttavia non sono spesso soddisfatte a livello pratico, per cui a meno delle situazioni in cui siano disponibili altri metodi diretti specifici per il caso trattato, si necessita di tecniche di campionamento alternative.

Capitolo 2

Metodi di campionamento indiretti

Esistono diverse distribuzioni univariate per le quali il metodo dell'inversione o altri tipi di trasformazioni non sono applicabili o risultano inefficienti. In questi contesti, non risulta possibile sfruttare direttamente le proprietà delle distribuzioni di interesse per campionare, ma si ricorre a metodi che utilizzino come informazione soltanto la forma funzionale della funzione di densità della variabile casuale di interesse, a meno di una costante moltiplicativa. Per questo motivo tali metodi, comprendenti una vasta gamma di algoritmi di campionamento, sono detti indiretti. Di questi ne verranno analizzati alcuni tra i maggiormente utilizzati, i quali, come anche diversi altri, sono basati su un risultato che ancora una volta stabilisce un legame fondamentale tra il campionamento da distribuzioni uniformi e non. Questo risultato è stato formalizzato nel seguente teorema.

Teorema 2.1 (Teorema fondamentale della simulazione)**.** *Campionare da una variabile casuale univariata con funzione di densità $f(x) \propto p(x)$ equivale a campionare dalla distribuzione uniforme*

$$(X, U) \sim \mathcal{U}\left\{(x, u) \in \mathbb{R}^2 : 0 \leq u \leq p(x)\right\}.$$

La variabile casuale X ha distribuzione marginale con funzione di densità $f(x)$.

L'importanza di questo teorema è dovuta al fatto che data la distribuzione congiunta (X, U), se si ha come distribuzione marginale di questa $X \sim f(x)$, generando un vettore casuale (x', u') dalla regione $\{(x, u) \in \mathbb{R}^2 : 0 \leq u \leq p(x)\}$, il valore x' costituisce una realizzazione della variabile casuale di interesse $X \sim f(x)$, ottenuto utilizzando la funzione p soltanto per il calcolo dei valori $p(x)$.
Quanto affermato fornisce la base teorica su cui si fondano i metodi analizzati di seguito.

2.1 Metodo accettazione-rifiuto

Il metodo accettazione-rifiuto (*rejection sampling*) rappresenta una tecnica di campionamento universale, in quanto potenzialmente potrebbe essere applicata per ottenere valori casuali da qualsiasi distribuzione. Questa sua caratteristica rende infatti questo metodo uno dei più versatili ed utilizzati, anche grazie al fatto che richieda soltanto la conoscenza della funzione di densità della variabile casuale di interesse a meno di una costante moltiplicativa, aspetto che torna particolarmente utile in diversi contesti. L'idea su cui si basa è l'individuare una funzione maggiorante la funzione di densità di interesse da cui è più semplice generare un campione, dal quale poi si seleziona un sottocampione distribuito concordemente alla distribuzione da cui si vuole campionare, in base al soddisfacimento di una condizione.

Funzionamento dell'algoritmo

Si considerano la variabile casuale X con funzione di densità $f(x) \propto p(x)$ e supporto $\mathcal{D} \subseteq \mathbb{R}$ da cui si vuole campionare, ed una funzione di densità strumentale $\pi(x)$ definita a meno di una costante moltiplicativa, $\pi(x) \propto \bar{\pi}(x)$. Successivamente, si sceglie una costante M tale che la funzione $M\bar{\pi}(x)$ sia maggiorante per $p(x)$, per cui deve valere che

$$M\bar{\pi}(x) \geq p(x) \ \ ,\forall x \in \mathcal{D}. \tag{2.1}$$

Si campiona quindi un valore x' da $\pi(x)$ ed uno u' da $\mathcal{U}(0,1)$. Il valore campionato x' viene accettato come realizzazione di X se vale la disuguaglianza $u' \leq \frac{p(x')}{M\bar{\pi}(x')}$, altrimenti viene rifiutato.
Questa tecnica permette di ottenere campioni di numerosità N implementando quanto descritto nell'algoritmo 1.

Algoritmo 1: Rejection Sampling

Input : Funzione di densità obiettivo (anche non normalizzata) $p(x)$,
funzione di densità strumentale (anche non normalizzata) $\bar{\pi}(x)$,
costante M tale che $p(x) \leq M\bar{\pi}(x)\ \forall x$,
numerosità del campione N.

Output : Campione $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ da $f(x) \propto p(x)$

$i \leftarrow 1$
while $i \leq N$ **do**
  Generare $x' \sim \pi(x)$
  Generare $u' \sim \mathcal{U}(0,1)$
  if $u' \leq \frac{p(x')}{M\bar{\pi}(x')}$ **then**
    $x^{(i)} \leftarrow x'$
    $i \leftarrow i + 1$
  end
  else
    Rifiutare x'
  end
end

Il campione $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ ottenuto dai valori accettati dall'algoritmo si distribuisce secondo $f(x)$, risultato su cui si basa ogni algoritmo accettazione-rifiuto e formalizzato in Martino et al. (2018) nel seguente teorema.

Teorema 2.2. *Siano le variabili casuali X_1 e X_2 con funzioni di densità rispettivamente $\pi(x) \propto \bar{\pi}(x)$ ed $f(x) \propto p(x)$ definite sul supporto $\mathcal{D}$, e sia $U \sim \mathcal{U}(0,1)$. Se esiste un valore M tale che $M \geq \frac{p(x)}{\bar{\pi}(x)}\ \forall x \in \mathcal{D}$, allora*

$$\mathrm{P}\left\{X_1 \leq y \,\middle|\, U \leq \frac{p(X_1)}{M\bar{\pi}(X_1)}\right\} = \mathrm{P}\{X_2 \leq y\} \tag{2.2}$$

Dimostrazione. Assumendo, senza perdita di generalità, che $\mathcal{D} = \mathbb{R}$ e considerando che $U \sim \mathcal{U}(0,1)$ e $X_1 \sim \pi(x)$, con $\pi(x) \propto \bar{\pi}(x)$, si può scrivere

$$\begin{aligned}\mathrm{P}\left\{X_1 \leq y \,\middle|\, U \leq \frac{p(X_1)}{M\bar{\pi}(X_1)}\right\} &= \frac{\mathrm{P}\left\{X_1 \leq y, U \leq \frac{p(X_1)}{M\bar{\pi}(X_1)}\right\}}{\mathrm{P}\left\{U \leq \frac{p(X_1)}{M\bar{\pi}(X_1)}\right\}} \\ &= \frac{\int_{-\infty}^{y}\int_{0}^{\frac{p(x)}{M\bar{\pi}(x)}} \bar{\pi}(x)dudx}{\int_{-\infty}^{+\infty}\int_{0}^{\frac{p(x)}{M\bar{\pi}(x)}} \bar{\pi}(x)dudx}.\end{aligned}$$

Integrando rispetto a u si ottiene

$$\frac{\int_{-\infty}^{y} \frac{p(x)}{M\bar{\pi}(x)}\bar{\pi}(x)dx}{\int_{-\infty}^{+\infty} \frac{p(x)}{M\bar{\pi}(x)}\bar{\pi}(x)dx} = \frac{\int_{-\infty}^{y} p(x)dx}{\int_{-\infty}^{+\infty} p(x)dx} = \mathrm{P}\left\{X_1 \leq y \,\middle|\, U \leq \frac{p\left(X_1\right)}{M\bar{\pi}\left(X_1\right)}\right\}.$$

Inoltre, valendo $f(x) \propto p(x)$*, ossia* $f(x) = \frac{1}{c}p(x)$ *con* $c = \int_{-\infty}^{+\infty} p(x)dx$*, si può riformulare l'espressione come*

$$\mathrm{P}\left\{X_1 \leq y \,\middle|\, U \leq \frac{p\left(X_1\right)}{M\bar{\pi}\left(X_1\right)}\right\} = \frac{\int_{-\infty}^{y} cf(x)dx}{c} = \int_{-\infty}^{y} f(x)dx.$$

Dal momento che la variabile casuale X_2 *ha densità* $f(x)$*, vale che*

$$\mathrm{P}\left\{X_1 \leq y \,\middle|\, U \leq \frac{p\left(X_1\right)}{M\bar{\pi}\left(X_1\right)}\right\} = \mathrm{P}\left\{X_2 \leq y\right\} = \int_{-\infty}^{y} f(x)dx.$$

Per cui la tesi è verificata.

Questo risultato stabilisce il motivo per cui questo algoritmo funzioni, tuttavia un aspetto fondamentale da considerare è quanti campioni debbano essere rifiutati per poterne accettare uno. Questa questione dipende direttamente da quanto precisamente la funzione $M\bar{\pi}(x)$ riesca ad approssimare la funzione $p(x)$.
La seguente rappresentazione grafica permette di comprendere come la scelta di $M\bar{\pi}(x)$ sia in effetti cruciale per la progettazione di questi algoritmi.

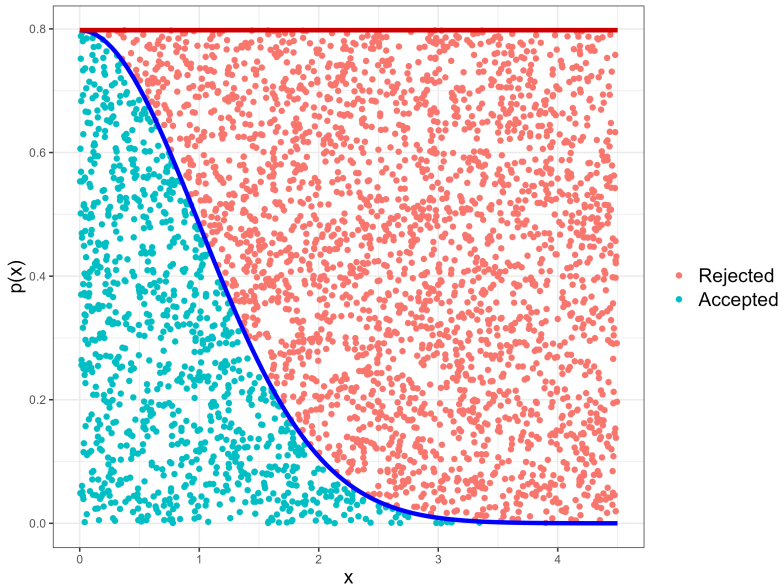

Figura 2.1: Rejection sampling sulla distribuzione $Half-Normal(0,1)$, con distribuzione strumentale $\mathcal{U}(0,4.5)$.

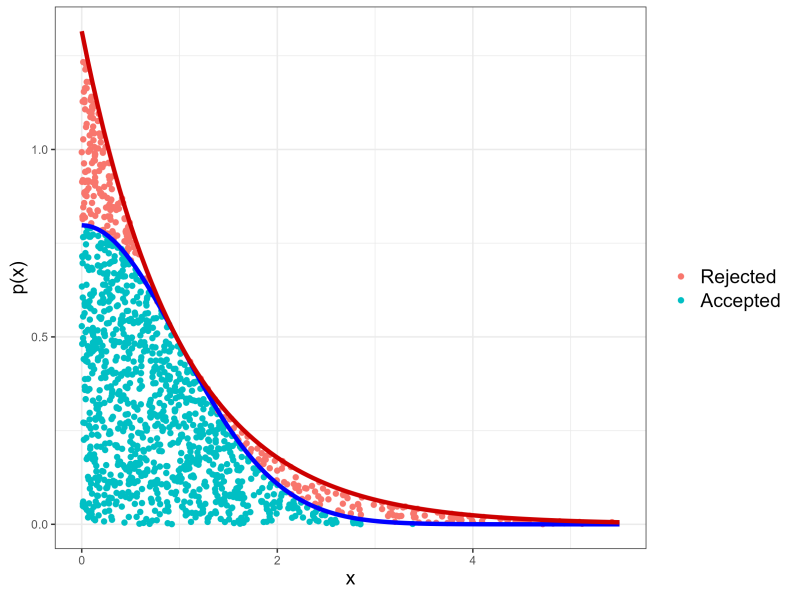

Figura 2.2: Rejection sampling sulla distribuzione $Half-Normal(0,1)$, con distribuzione strumentale $Exp(1)$.

L'interpretazione geometrica del metodo è quindi la seguente: si campiona uniformemente dalla regione bidimensionale

$$\left\{(x,u) \in \mathbb{R}^2 : 0 \leq u \leq M\bar{\pi}(x)\right\},$$

e si accettano soltanto i campioni che ricadono nella regione

$$\left\{(x,u) \in \mathbb{R}^2 : 0 \leq u \leq p(x)\right\}.$$

Le ascisse dei punti ottenuti con questo procedimento si distribuiscono in base alla densità $f(x)$ per il risultato fornito dal Teorema 2.1, le quali costituiscono quindi il campione dalla distribuzione di interesse.
Risulta dunque evidente come l'efficienza del campionamento sia dipendente dalla differenza che intercorre tra i valori assunti dalle due funzioni, dove con efficienza si intende quanti campioni siano rifiutati rispetto a quelli accettati.
Questo aspetto viene fornito in Martino et al. (2018) nella seguente definizione.

Definizione 2.1 (Probabilità di accettazione e di rifiuto)**.** Siano dati $U \sim \mathcal{U}(0,1)$ e la variabile casuale X con funzione di densità non normalizzata $\bar{\pi}(x)$ sul supporto $\mathcal{D}$. Se vale $M\bar{\pi}(x) \geq p(x)$, si definiscono la probabilità di accettazione $\hat{a}$ e la probabilità

di rifiuto $\hat{r}$ rispettivamente come

$$\hat{a} = \mathrm{P}\left\{U \leq \frac{p\,(X)}{M\bar{\pi}\,(X)}\right\} = \frac{c}{Mc_\pi}, \tag{2.3}$$

e

$$\begin{aligned} \hat{r} &= 1 - \hat{a} = 1 - \frac{c}{Mc_\pi} = \frac{Mc_\pi - c}{Mc_\pi} \\ &= \frac{1}{Mc_\pi}\left(\int_D M\bar{\pi}(x)dx - \int_D p(x)dx\right) \\ &= \frac{1}{Mc_\pi}\left(\int_D |M\bar{\pi}(x) - p(x)|dx\right) = \frac{1}{Mc_\pi} d(M\bar{\pi}, p). \end{aligned} \tag{2.4}$$

Dove $c = \int_{\mathcal{D}} p(x)dx$ e $c_\pi = \int_{\mathcal{D}} \bar{\pi}(x)dx$.
La funzione $d(M\bar{\pi}, p)$ indica la distanza L_1 tra $M\bar{\pi}(x)$ e $p(x)$.

Quanto affermato significa che la probabilità di rifiuto sia funzione crescente della distanza tra le due funzioni di densità non normalizzate. Questo aspetto può rappresentare un problema: nel caso in cui non si riesca ad individuare una funzione strumentale che approssimi adeguatamente la funzione target, l'algoritmo produrrà un numero elevato di valori rifiutati, andando quindi ad abbassare l'efficienza della procedura.
Per questa ragione sono stati proposti metodi di campionamento accettazione-rifiuto più elaborati, i quali sfruttano i risultati forniti dalla procedura standard ma permettendo di ottenere performance migliori.

2.2 Metodo accettazione-rifiuto adattivo (ARS)

Una delle maggiori difficoltà che si incontrano nella progettazione di un algoritmo accettazione-rifiuto è l'individuazione di una funzione di densità strumentale da cui sia semplice campionare e che approssimi adeguatamente la funzione target $p(x)$. Il metodo accettazione rifiuto adattivo (*adaptive rejection sampling*, ARS), proposto da Gilks & Wild (1992), propone una soluzione a questa problematica, sfruttando i campioni rifiutati dalla procedura standard per ottenere progressivamente una migliore approssimazione della funzione $p(x)$ e di conseguenza una sempre maggiore probabilità di accettazione.
Nonostante rappresenti una procedura maggiormente efficiente, necessita il soddisfacimento di una condizione, la quale ne limita le possibilità di applicazione. Per poter utilizzare il metodo ARS si richiede infatti che la variabile casuale da cui si vuole

campionare possieda funzione di densità $f(x) \propto p(x) \geq 0$, con supporto $\mathcal{D} \subseteq \mathbb{R}$, tale che la funzione $h(x) = \log[p(x)]$, con $x \in \mathcal{D}$, sia concava.

Funzionamento dell'algoritmo

Si considerano la variabile casuale X con funzione di densità $f(x) \propto p(x)$ e supporto $\mathcal{D} \subseteq \mathbb{R}$ da cui si vuole campionare.
In modo da poter costruire la funzione di densità strumentale, si definisce un insieme di punti di supporto $\mathcal{S}_t = \{s_1, \ldots, s_{m_t}\}$ dove $s_j \in \mathcal{D}, j = 1, \ldots, m_t$ ed il valore $t \in \mathbb{N}$ indica l'iterazione corrente ($t = 0, 1, 2, \ldots$). Il numero dei punti m_t aumenta se nell'iterazione t dell'algoritmo viene rifiutato il campione estratto, altrimenti rimane invariato.
Questo insieme di punti viene utilizzato per costruire una sequenza di funzioni non negative, $\{\bar{\pi}_t(x)\}_{t=0}^{+\infty}$, tali che:
1. $\bar{\pi}_t(x) \geq p(x), \quad \forall x \in \mathcal{D}, \quad \forall t \in \mathbb{N}$.
2. Risulta possibile campionare esattamente da $\pi_t(x) \propto \bar{\pi}_t(x), \quad \forall t \in \mathbb{N}$.
3. Se $|\mathcal{S}_t| = m_t \to \infty$, allora vale che $\lim_{m_t \to \infty} \bar{\pi}_t(x) = p(x)$ in $\mathcal{D}$.

Nel caso in cui queste condizioni siano verificate ed esista un metodo per la costruzione di $\{\bar{\pi}_t(x)\}_{t=0}^{+\infty}$, il metodo ARS permette di ottenere campioni di numerosità N dalla distribuzione di interesse implementando l'algoritmo seguente.

Algoritmo 2: Adaptive Rejection Sampling

Input : Insieme dei punti di supporto iniziali $\mathcal{S}_0 = \{s_1, \ldots, s_{m_0}\}$ con $s_i \in \mathcal{D}$, $i = 1, \ldots, m_0$,
supporto $\mathcal{D}$,
funzione di densità target non normalizzata $p(x)$,
numerosità del campione N.

Output : Campione $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ da$f(x) \propto p(x)$.

```
t ← 0
i ← 1
while i ≤ N do
    Costruire π_t(x) ∝ π̄_t(x) utilizzando l'insieme di punti S_t
    Generare x' ∼ π_t(x)
    Generare u' ∼ U(0, 1)
    if u' ≤ p(x') / π̄_t(x') then
        x^(i) ← x'
        i ← i + 1
        S_{t+1} ← S_t
    end
    else
        Rifiutare x'
        S_{t+1} ← S_t ∪ {x'}
    end
    t ← t + 1
end
```

Chiaramente anche in questo caso il campione $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ ottenuto dai valori accettati dall'algoritmo si distribuisce secondo $f(x)$ per il risultato fornito dal Teorema 2.2.

Questa formulazione fornisce la struttura generale dell'algoritmo, tuttavia prima della sua implementazione deve essere chiarito come costruire progressivamente con le diverse iterazioni la funzione di densità strumentale e come campionare da essa.

Scelta dei punti di supporto iniziali

Il numero minimo di punti di supporto iniziali è $m_0 = 2$, $\mathcal{S}_0 = \{s_1, s_2\}$, con $s_1 < s_2$. Per ottenere una funzione $\bar{\pi}_0$ appropriata, la derivata prima di $h(x) = \log[p(x)]$ deve

avere segni opposti quando valutata in s_1 e s_2, in modo tale che la moda di $p(x)$ sia contenuta in $[s_1, s_2]$.
Normalmente, i punti di supporto iniziali sono scelti manualmente utilizzando le informazioni di cui si dispone sulla funzione di densità target, tuttavia una procedura simile può risultare lenta in contesti in cui devono essere estratti campioni sequenzialmente da distribuzioni diverse tra loro. Nonostante la scelta manuale dei punti di supporto iniziali possa comunque rappresentare una valida alternativa, può risultare utile anche l'utilizzo di una procedura automatica che faccia utilizzo di una quantità limitata di informazioni sulla distribuzione. Una soluzione di questo tipo viene proposta da James (2024), la quale si basa su tecniche di approssimazione numerica. Si tratta di un algoritmo iterativo per l'individuazione dei punti di supporto iniziali in un contesto di ARS standard, il quale richiede oltre alla funzione target soltanto la conoscenza di un punto appartenente alla regione ad elevata densità della distribuzione. Tuttavia l'impiego di questa tecnica rende il campionamento più pesante dal punto di vista computazionale, pur offrendo il vantaggio di renderlo più automatico. Nell'implementazione di questi algoritmi, va quindi sempre considerato una sorta di compromesso tra quanto li si voglia ottenere ampiamente applicabili, quanta informazione debbano ricevere in input e quanto li si voglia efficienti in termini di tempi di esecuzione.

Costruzione della funzione di densità strumentale

L'obiettivo è la costruzione di una funzione dominante per $h(x) = \log[p(x)]$ localmente, su appositi intervalli in cui il supporto $\mathcal{D}$ viene partizionato. Per effettuare questa procedura, si utilizza l'insieme dei punti di supporto $\mathcal{S}_t = \{s_1, \ldots, s_{m_t}\} \subset \mathcal{D}$, dove i punti ad ogni iterazione t sono in ordine crescente, $s_1 < \ldots < s_{m_t}$. Ad ogni iterazione t vengono quindi calcolate le rette tangenti ad $h(x)$ in ognuno dei punti $s_j \in \mathcal{S}_t$, definite dalle equazioni $w_j(x) = a_j + b_j x$. Per ognuna di queste, in modo da costruire la funzione maggiorante per $h(x)$, viene considerato soltanto il segmento che comprende i valori minori tra tutte le rette su tutto il supporto, andando ad ottenere la funzione

$$W_t(x) = \min\{w_1(x), \ldots, w_{m_t}(x)\} \geq h(x), \quad \forall x \in \mathcal{D}. \tag{2.5}$$

La disuguaglianza $W_t(x) \geq h(x)$ vale come conseguenza della concavità di $h(x)$, dal momento che per definizione una funzione concava risulta assumere in ogni punto del suo dominio valori minori o al più uguali di quelli assunti dalle sue rette tangenti.

Essendo la trasformazione esponenziale monotona, discende quindi che la funzione definita come $\bar{\pi}_t(x) = \exp[W_t(x)]$ è una funzione maggiorante per $p(x)$, per cui vale

$$\bar{\pi}_t(x) = \exp[W_t(x)] \geq p(x) = \exp[h(x)], \quad \forall x \in \mathcal{D}. \tag{2.6}$$

Utilizzando $W_t(x)$, la funzione di densità strumentale si ottiene quindi come

$$\pi_t(x) = c_{\pi_t} \exp[W_t(x)] \propto \bar{\pi}_t(x), \tag{2.7}$$

dove $c_{\pi_t} = \left(\int_{\mathcal{D}} \exp\left(W_t(x)\right) dx\right)^{-1} = \left(\int_{\mathcal{D}} \min\left\{\exp(a_1 + b_1 x), \ldots, \exp(a_{m_t} + b_{m_t} x)\right\} dx\right)^{-1}$ è la costante di normalizzazione, per cui vale $\pi_t(x) = c_t \bar{\pi}_t(x)$.
Il maggior vantaggio che offre questo metodo è il progressivo miglioramento della funzione $\bar{\pi}_t(x)$ descritta, la quale infatti risulta indicizzata da t: essa infatti può essere aggiornata nel corso delle diverse iterazioni. Come indicato nell'algoritmo 2 questo processo è consentito facendo uso dell'informazione fornita dal rifiuto di un campione x': se questo non viene accettato, significa che in corrispondenza di x' la distanza tra $p(x)$ e $\bar{\pi}_t(x)$ è elevata, per cui questa informazione locale può essere utilizzata per ottenere una funzione di densità strumentale che approssimi meglio la funzione di densità obiettivo. Ogni valore rifiutato viene infatti incluso nell'insieme dei punti di supporto, il quale viene in questi casi aggiornato, ponendo $\mathcal{S}_{t+1} = S_t \cup \{x'\}$ ed $m_{t+1} = m_t + 1$. Effettuata questa operazione, viene ripetuta la procedura descritta in precedenza, calcolando quindi la nuova funzione $\bar{\pi}_{t+1}(x) = \exp[W_{t+1}(x)]$, la quale fornisce una migliore approssimazione di $p(x)$ rispetto alla precedente.
La seguente rappresentazione grafica fornisce un'intuizione del funzionamento di questo metodo.

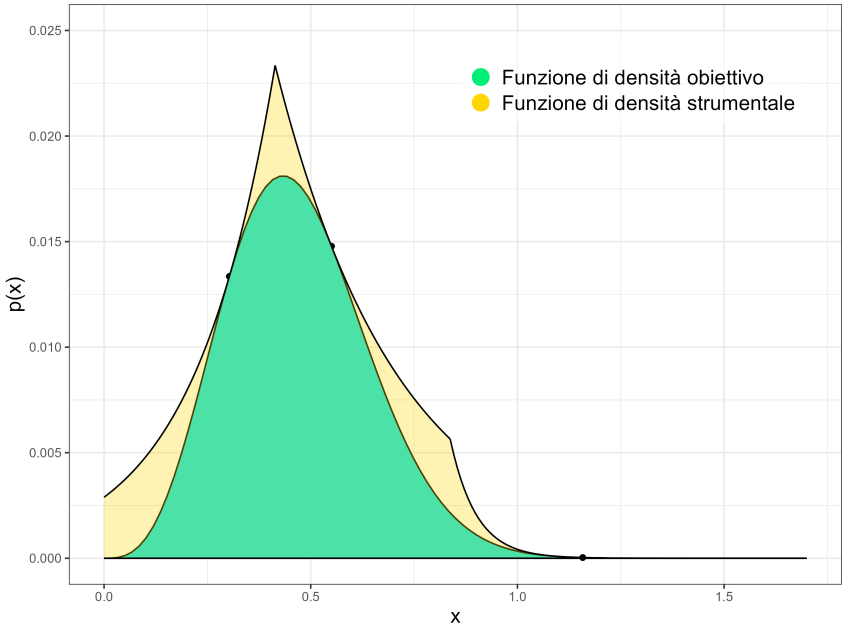

Figura 2.3: ARS sulla distribuzione $\sqrt{Gamma(1.5, 2)}$. $N = 2$, $m_t = 3$.

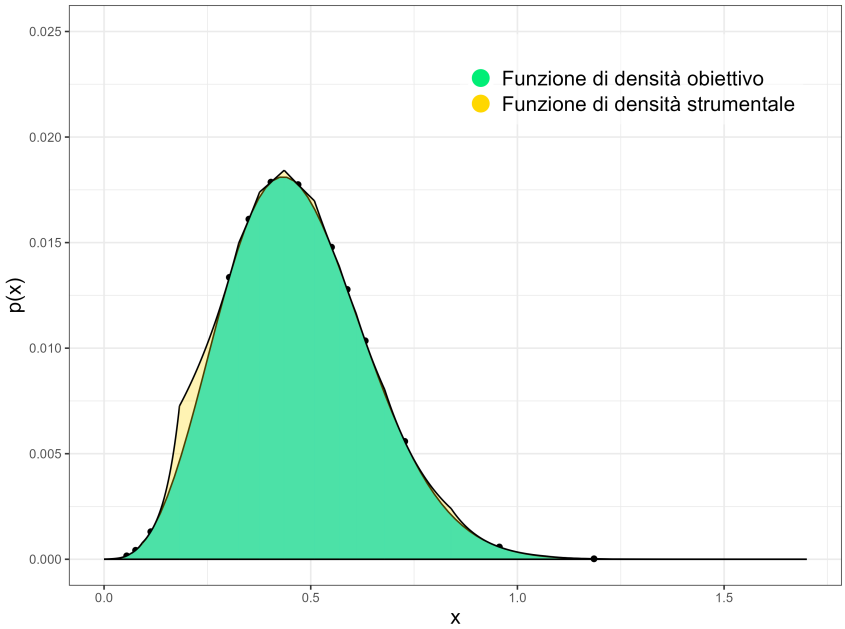

Figura 2.4: ARS sulla distribuzione $\sqrt{Gamma(1.5, 2)}$. $N = 200$, $m_t = 13$.

Risulta evidente come la distanza tra le due funzioni si riduca progressivamente al crescere del numero di campioni estratti, conducendo di conseguenza ad una sempre maggiore probabilità di accettazione.

Campionamento dalla funzione di densità strumentale

In modo da poter campionare dalla funzione di densità strumentale, si necessita inanzitutto del calcolo dei punti di intersezione tra le rette consecutive tangenti ad $h(x)$, ovvero l'individuazione delle ascisse ξ_j, $j = 1, \ldots, m_t - 1$ tali che $w_j(\xi_j) = w_{j+1}(\xi_j), j = 1, \ldots, m_t - 1$. Questi punti sono utilizzati per definire gli intervalli $\mathcal{I}_0 = (-\infty, \xi_1]$, $\mathcal{I}_j = (\xi_j, \xi_{j+1}]$ per $j = 1, \ldots, m_t - 1$, $\mathcal{I}_{m_t} = (\xi_{m_t}, +\infty)$ in cui viene partizionato $\mathcal{D}$. In ognuno di questi intervalli viene calcolata l'area sottostante alla corrispondente componente della funzione $\bar{\pi}_t(x)$ svolgendo l'integrale

$$\eta_j = \int_{\mathcal{I}_j} \bar{\pi}_t(x)dx = \int_{\mathcal{I}_j} \exp(a_j + b_j x)dx, \quad j = 0, \ldots, m_t.$$

I valori ottenuti sono successivamente normalizzati

$$\bar{\eta}_j = \frac{\eta_j}{\sum_{j=0}^{m_t} \eta_j} = \frac{\eta_j}{c_{\pi_t}},$$

dove $\sum_{k=0}^{m_t} \eta_k = c_{\pi_t} = \left(\int_{\mathcal{D}} \exp[(W_t(x)]dx\right)^{-1}$ è sempre la costante di normalizzazione di $\bar{\pi}_t(x)$. I valori $\bar{\eta}_j$, $j = 0, \ldots, m_t$, costituiscono i pesi normalizzati assegnati ad ogni componente di $\bar{\pi}_t(x)$, ossia le probabilità di trovarsi nell'intervallo $\mathcal{I}_j$ comprendente la componente $\exp(a_j + b_j x)$ di $\bar{\pi}_t(x)$, in base al peso rappresentato dall'area $\int_{\mathcal{I}_j} \exp(a_j + b_j x)dx$ sull'area totale $\int_{\mathcal{D}} \exp[W_t(x)]dx$.
Per poter campionare da $\pi_t(x)$, si seguono quindi due passaggi:

1. Si estrae casualmente un intervallo $\mathcal{I}_j$, $j = 0, \ldots, m_t$, concordemente ai pesi $\bar{\eta}_j$, $j = 0, \ldots, m_t$, posseduti da ognuno.

2. Si campiona casualmente da $\bar{\pi}_t(x)$ sull'intervallo estratto $\mathcal{I}_j$.

Il campionamento da $\bar{\pi}_t(x)$ in $\mathcal{I}_j$ viene effettuato con il metodo dell'inversione, tuttavia risulta utile chiarire alcuni aspetti per illustrare la procedura in questo determinato contesto.
Inanzitutto, deve essere fornita l'espressione della funzione di ripartizione della distribuzione, definita come

$$\begin{aligned} F_{\pi_t}(x) &= c_{\pi_t} \int_{-\infty}^{x} \pi_t(v)dv \\ &= c_{\pi_t} \int_{-\infty}^{x} \min\left\{\exp(a_1 + b_1 v), \ldots, \exp(a_{m_t} + b_{m_t} v)\right\} dv. \end{aligned} \tag{2.8}$$

Nel momento in cui viene estratto un intervallo $\mathcal{I}_j$, si ottiene per definizione $\min\{\exp(a_1 + b_1 x), \ldots, \exp(a_{m_t} + b_{m_t} x)\} = \exp(a_j + b_j x)$, tuttavia campionare dalla distribuzione con funzione di ripartizione $c_{\pi_t} \int_{-\infty}^{x} \exp(a_j + b_j v) dv$ non risulterebbe corretto, in quanto si vuole campionare soltanto all'interno dell'intervallo estratto. Per poterlo fare risulta utile introdurre la seguente definizione.

Definizione 2.2 (Distribuzione troncata)**.** Sia data una variabile casuale X sul supporto $\mathcal{D} \subseteq \mathbb{R}$, con funzione di densità $f_X(x)$ e funzione di ripartizione $F_X(x)$. La variabile casuale X_T ottenuta troncando la distribuzione X sull'intervallo $[a, b] \subset \mathcal{D}$ possiede funzione di densità e funzione di ripartizione definite rispettivamente da

$$f_{X_T}(y) = \begin{cases} \frac{f_X(x)}{F_X(b) - F_X(a)} & \text{se } a \leq x \leq b \\ 0 & \text{altrimenti.} \end{cases} \tag{2.9}$$

$$F_{X_T}(x) = \begin{cases} 0 & \text{se } x \leq a \\ \frac{F_X(x) - F_X(a)}{F_X(b) - F_X(a)} & \text{se } a < x \leq b \\ 1 & \text{se } x > b, a < b. \end{cases} \tag{2.10}$$

Nel caso specifico trattato, si vuole troncare la distribuzione con funzione di ripartizione $F_j(x) = c_{\pi_t} \int_{-\infty}^{x} \exp(a_j + b_j v) dv$ sull'intervallo $\mathcal{I}_j = (\xi_j, \xi_{j+1}]$.
Svolgendo alcuni calcoli si ottiene quindi la funzione

$$\begin{aligned} F_{\mathcal{I}_j}(x) &= \frac{F_j(x) - F_j(\xi_j)}{F_j(\xi_{j+1}) - F_j(\xi_j)} \\ &= \frac{c_{\pi_t} \int_{-\infty}^{x} \exp(a_j + b_j v) dv - c_{\pi_t} \int_{-\infty}^{\xi_j} \exp(a_j + b_j v) dv}{c_{\pi_t} \int_{-\infty}^{\xi_{j+1}} \exp(a_j + b_j v) dv - c_{\pi_t} \int_{-\infty}^{\xi_j} \exp(a_j + b_j v) dv} \\ &= \frac{\int_{\xi_j}^{x} \exp(a_j + b_j v) dv}{\int_{\xi_j}^{\xi_{j+1}} \exp(a_j + b_j v) dv} \\ &= \frac{\exp\left[b_j (x - \xi_j)\right] - 1}{\exp\left[b_j (\xi_{j+1} - \xi_j)\right] - 1}. \end{aligned} \tag{2.11}$$

Si nota come il valore di c_{π_t} non sia necessario per il calcolo di $F_{\mathcal{I}_j}(x)$.
La funzione così individuata si presta all'utilizzo del metodo dell'inversione, utilizzandone l'inversa definita da

$$F_{\mathcal{I}_j}{}^{-1}(p) = \frac{1}{b_j} \log\left\{p \exp\left[b_j (\xi_{j+1} - \xi_j)\right] - p + 1\right\} + \xi_j, \tag{2.12}$$

questo conclude la presentazione del metodo ARS.

2.3 Metodo del rapporto di uniformi

Il rapporto di uniformi (*ratio of uniforms*, ROU) è un metodo di campionamento basato sulla trasformazione della regione bidimensionale $\mathcal{A}_0$ sottostante alla funzione di densità target $f(x) \propto p(x)$, in modo da ottenere la nuova regione $\mathcal{A}$. Utilizzando questa tecnica, si campiona uniformemente da $\mathcal{A}$ per ottenere campioni dalla distribuzione di interesse.
Questo metodo si rivela utile nel caso in cui effettuare un campionamento uniforme da $\mathcal{A}$ sia più semplice rispetto a farlo direttamente da $f(x)$. Si dimostra inoltre vantaggioso nei casi in cui la regione $\mathcal{A}_0$ non è limitata e l'individuazione di una funzione dominante per $f(x)$ sia difficoltosa. In queste situazioni, non risulta possibile utilizzare una tecnica di *rejection sampling* standard che impieghi una distribuzione strumentale uniforme, a meno di troncare la distribuzione di interesse ed andando quindi a campionare da una distribuzione di fatto diversa dalla target. Effettuando invece la trasformazione che produce $\mathcal{A}$, si può ottenere una regione limitata, la quale può quindi essere racchiusa all'interno di un rettangolo nel nuovo spazio di riferimento, da cui poi si campiona uniformemente. Per questa ragione i casi di interesse risultano essere quelli in cui la regione $\mathcal{A}$ è limitata, della quale pur non conoscendone esplicitamente il confine può essere stabilito se un campione vi appartenga o meno sfruttando i risultati che vengono illustrati di seguito.
Il metodo del rapporto di uniformi venne introdotto da Kinderman & Monahan (1977); il risultato sui cui esso si basa è riportato con la formulazione più recente di Martino et al. (2018) nel seguente teorema.

Teorema 2.3. *Sia data la funzione di densità $p(x) \geq 0$, nota a meno di una costante moltiplicativa, per cui $p(x) \propto f(x)$. Se (v, u) è un campione estratto dalla distribuzione uniforme bivariata $(V, U) \sim \mathcal{U}(\mathcal{A})$, dove*

$$\mathcal{A} = \{(v, u) : 0 \leq u \leq \sqrt{2p(v/u)}\}, \tag{2.13}$$

allora si ha che $x = \frac{v}{u}$ è un campione da $f(x)$.
Dimostrazione. Data la trasformazione $(v, u) \to (x, y)$

$$\begin{cases} x = \frac{v}{u} \\ y = u \end{cases} \tag{2.14}$$

e la variabile casuale bivariata (V, U) *distribuita uniformemente su* $\mathcal{A}$*, si può scrivere la funzione di densità congiunta* $q(x, y)$ *della trasformazione della variabile casuale* (X, Y) *come*

$$q(x,y) = \frac{1}{|\mathcal{A}|}\left|\mathbf{J}^{-1}\right| \quad \forall y: \quad 0 \leq y \leq \sqrt{2p(x)}, \tag{2.15}$$

dove $|\mathcal{A}|$ *è l'area di* $\mathcal{A}$*, e* $\mathbf{J}^{-1}$ *è il Jacobiano della trasformazione inversa*

$$\mathbf{J}^{-1} = \det \begin{bmatrix} y & x \\ 0 & 1 \end{bmatrix} = y. \tag{2.16}$$

Sostituendo si ottiene

$$q(x,y) = \begin{cases} \frac{1}{|\mathcal{A}|}y & se \quad 0 \leq y \leq \sqrt{2p(x)} \\ 0 & altrimenti. \end{cases} \tag{2.17}$$

Allora la funzione di densità della distribuzione marginale X*, coincide con* $f(x)$*:*

$$\begin{aligned} \int_{-\infty}^{+\infty} q(x,y)dy &= \int_0^{\sqrt{2p(x)}} \frac{y}{|\mathcal{A}|}dy = \\ &= \frac{1}{|\mathcal{A}|}\left[\frac{y^2}{2}\right]_0^{\sqrt{2p(x)}} = \frac{1}{|\mathcal{A}|}p(x). \end{aligned} \tag{2.18}$$

La prima uguaglianza discende da 2.17. Valendo che $f(x) \propto p(x)$ *e* $\int_{-\infty}^{+\infty} q(x,y)dy$ *è una funzione di densità propria, si ha necessariamente che* $f(x) = \frac{1}{|\mathcal{A}|}p(x)$*, per cui* $|\mathcal{A}| = c_v = \int_{\mathcal{D}} p(x)dx$*.*

Questo teorema afferma quindi che se risulta possibile generare valori (v', u') campionati uniformemente da $\mathcal{A}$, il campione $x' = \frac{v'}{u'}$ proviene dalla distribuzione $f(x)$.
La regione $\mathcal{A}$ può inoltre essere ridefinita come

$$\mathcal{A} = \left\{(v,u) : 0 \leq u \leq \sqrt{2c_A p(v/u)}\right\}, \tag{2.19}$$

dove c_A è una costante positiva. L'area di $\mathcal{A}$ in questo modo varia, ossia vale che $|\mathcal{A}| = c_A c_v$, tuttavia il metodo rimane valido. Normalmente per semplicità si pone $c_A = \frac{1}{2}$, ottenendo

$$\mathcal{A} = \{(v,u) : 0 \leq u \leq \sqrt{p(v/u)}\}. \tag{2.20}$$

Dalla definizione di $\mathcal{A}$ e dalla trasformazione 2.14, discende che $u_i \leq p(x)$ e $v_i = xu_i \leq x\sqrt{p(x)}$. Dunque il confine di $\mathcal{A}$ viene definito dai punti (v_b, u_b) che

soddisfano il sistema di equazioni

$$\begin{cases} u_b = \sqrt{p(x)} \\ v_b = x\sqrt{p(x)} \end{cases} \tag{2.21}$$

La regione $\mathcal{A}$ è quindi limitata se e solo se le funzioni $\sqrt{p(x)}$ e $x\sqrt{p(x)}$ sono limitate. In questi casi di interesse, chiaramente l'efficacia del metodo dipende dalla facilità con cui si riesce a campionare da $\mathcal{A}$. Per poterlo fare si parte dal fatto che, stando alla definizione della regione $\mathcal{A}$, essa può essere racchiusa in una regione rettangolare $\mathcal{R}$, definita da

$$\mathcal{R} = \left\{ (v,u) : 0 \leq u \leq \sup_x \sqrt{p(x)}, \inf_x x\sqrt{p(x)} \leq v \leq \sup_x x\sqrt{p(x)} \right\}. \tag{2.22}$$

Successivamente alla costruzione di $\mathcal{R}$, risulta semplice campionare uniformemente da questa regione, utilizzando poi un algoritmo accettazione-rifiuto che verifichi che i campioni ottenuti da $\mathcal{R}$ siano appartenenti anche ad $\mathcal{A}$, accettando quindi i valori che soddifino la definizione 2.19, i quali costituiscono un campione dalla distribuzione target per il risultato fornito dal Teorema 2.3.
La seguente rappresentazione grafica permette di ottenere un'intuizione del funzionamento del metodo.

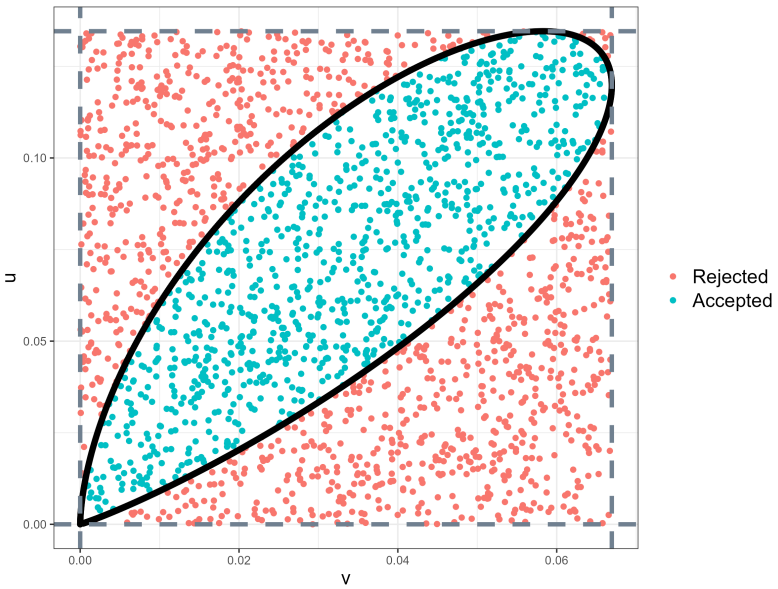

Figura 2.5: Metodo del rapporto di uniformi sulla distribuzione $\sqrt{Gamma(1.5,2)}$, N=1000.

La tecnica descritta può quindi essere impiegata per la generazione di un campione di numerosità N da una distribuzione continua implementando il seguente algoritmo.

Algoritmo 3: Standard ratio of uniforms

Input : funzione di densità (anche non normalizzata) $p(x)$,
numerosità del campione N.

Output : campione $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ da $f(x) \propto p(x)$

$i \leftarrow 1$
while $i \leq N$ **do**
  Costruire il rettangolo $\mathcal{R} \supseteq \mathcal{A}$
  Campionare (v', u') uniformemente dalla regione rettangolare $\mathcal{R}$
  if $u' \leq \sqrt{p\left(\frac{v'}{u'}\right)}$ **then**
    $x^{(i)} = x' = \frac{v'}{u'}$
    $i \leftarrow i + 1$
  end
  else
    Rifiutare $x' = \frac{v'}{u'}$
  end
end

Capitolo 3

Caso applicativo

Come illustrato, i metodi descritti trovano la loro utilità nei contesti in cui si vuole campionare da una distribuzione sulla quale non sono impiegabili metodi diretti. La distribuzione che viene presentata in questo capitolo ricade in questi casi ed è stata quindi scelta per poter effettuare un confronto empirico tra gli algoritmi presentati.

3.1 La distribuzione Modified-Half-Normal

La distribuzione *Modified-Half-Normal*, indicata con $\text{MHN}(\alpha, \beta, \gamma)$, è una distribuzione continua appartenente alla famiglia esponenziale. È caratterizzata da tre parametri e possiede come supporto l'insieme dei numeri reali positivi $\mathbb{R}^+$. Per le sue caratteristiche distintive, illustrate di seguito, essa può essere vista come una generalizzazione di alcune distribuzioni continue più comuni, aspetto che la rende uno strumento adattabile a diversi utilizzi.
Chiaramente in questo contesto l'interesse verso questa distribuzione è dovuto al fatto che essa si presti all'applicazione dei metodi di campionamento indiretti discussi. Prima di poter procedere con questa analisi, risulta utile fornire alcune delle informazioni caratterizzanti questa distribuzione, le quali sono tratti da Jingchao Sun & Pal (2023), dove essa viene presentata in dettaglio.

Definizione 3.1 (Funzione di densità e funzione di ripartizione della distribuzione *Modified-Half-Normal*)**.** Data la distribuzione $\text{MHN}(\alpha, \beta, \gamma)$, con parametri $\alpha > 0$, $\beta > 0$ e $\gamma \in \mathbb{R}$, se ne definiscono la funzione di densità e la funzione di ripartizione rispettivamente come

$$f_{\text{MHN}}(x \mid \alpha, \beta, \gamma) = \frac{2\beta^{\frac{\alpha}{2}} x^{\alpha-1} \exp\left(-\beta x^2 + \gamma x\right)}{\Psi\left[\frac{\alpha}{2}, \frac{\gamma}{\sqrt{\beta}}\right]} \mathbb{I}(x > 0), \tag{3.1}$$

e

$$F_{\text{MHN}}(x \mid \alpha, \beta, \gamma) = \frac{2\beta^{\frac{\alpha}{2}}}{\Psi\left[\frac{\alpha}{2}, \frac{\gamma}{\sqrt{\beta}}\right]} \sum_{i=0}^{\infty} \frac{\gamma^i}{2i!} \beta^{-\frac{\alpha+i}{2}} \gamma\left(\frac{\alpha+i}{2}, \beta x^2\right). \tag{3.2}$$

Dove $\gamma(s, y) = \int_0^y t^{s-1} e^{-t} dt$ indica la funzione gamma incompleta e $\frac{1}{2\beta^{\frac{\alpha}{2}}} \Psi\left[\frac{\alpha}{2}, \frac{\gamma}{\sqrt{\beta}}\right]$ è la costante di normalizzazione.
L'espressione $\Psi\left[\frac{\alpha}{2}, x\right]$ rappresenta un'abbreviazione notazionale con cui si identifica il seguente caso particolare della funzione Fox-Wright:

$${}_1\Psi_1\left[\begin{matrix}\left(\frac{\alpha}{2}, \frac{1}{2}\right) \\ (1, 0)\end{matrix}; x\right] = \sum_{n=0}^{\infty} \frac{\Gamma\left(\frac{\alpha}{2} + \frac{1}{2}n\right)}{n!} x^n.$$

Considerando la difficoltà del calcolo della funzione Fox-Wright ed in generale data la complessità della forma analitica della funzione di ripartizione, è evidente che la distribuzione *Modified-Half-Normal* non si presti all'utilizzo del metodo dell'inversione, motivo per il quale per campionare da essa si necessita di metodi indiretti.
I grafici riportati di seguito rappresentano alcuni casi delle funzioni di densità della distribuzione $\text{MHN}(\alpha, \beta, \gamma)$ per differenti valori dei tre parametri.

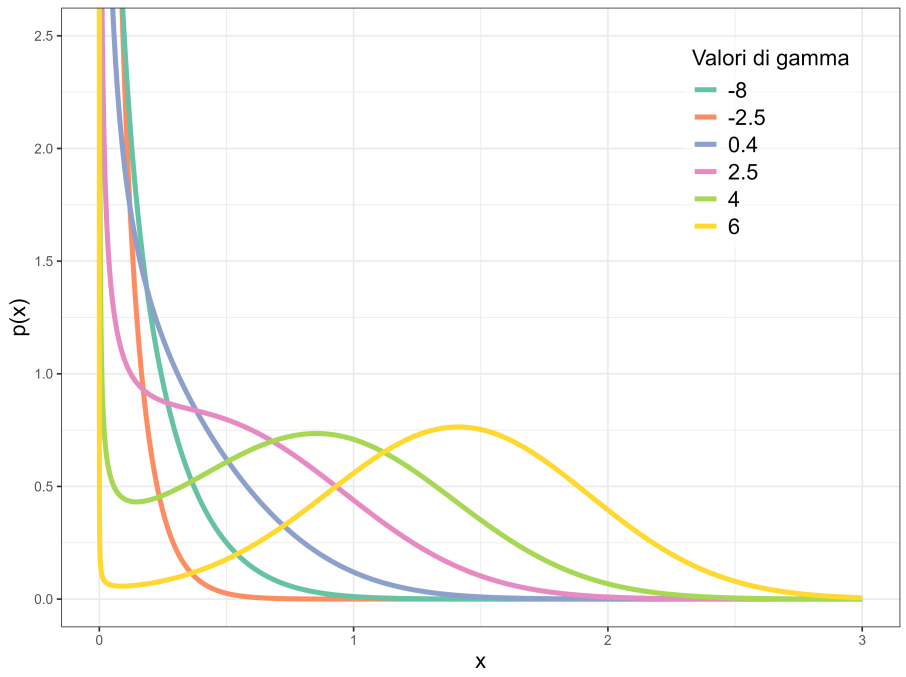

Figura 3.1: Funzioni di densità di $\text{MHN}(0.5, 2, \gamma)$ per alcuni valori di γ ed $\alpha < 1$.

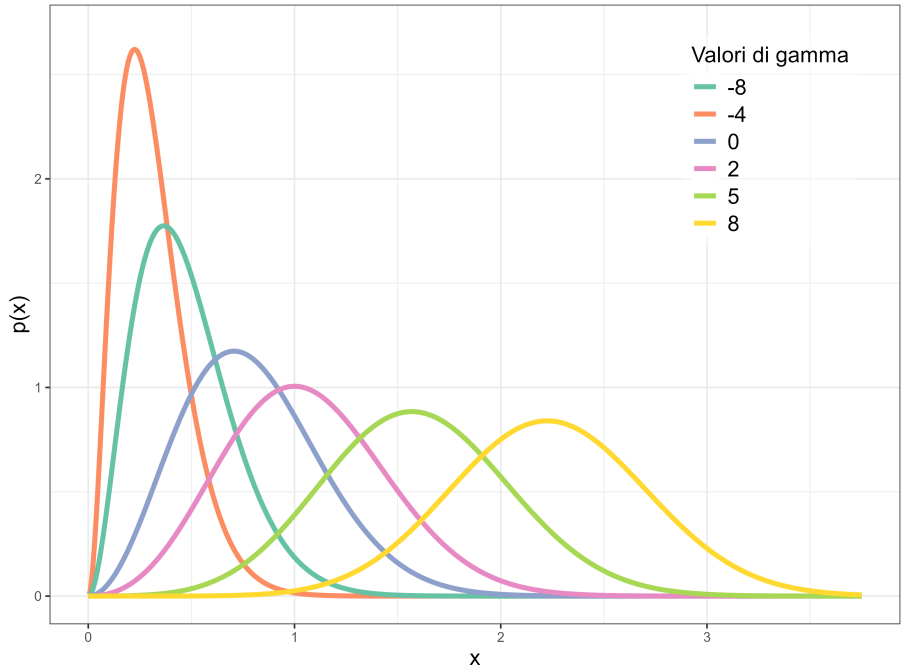

Figura 3.2: Funzioni di densità di $\text{MHN}(3, 2, \gamma)$ per alcuni valori di γ ed $\alpha > 1$.

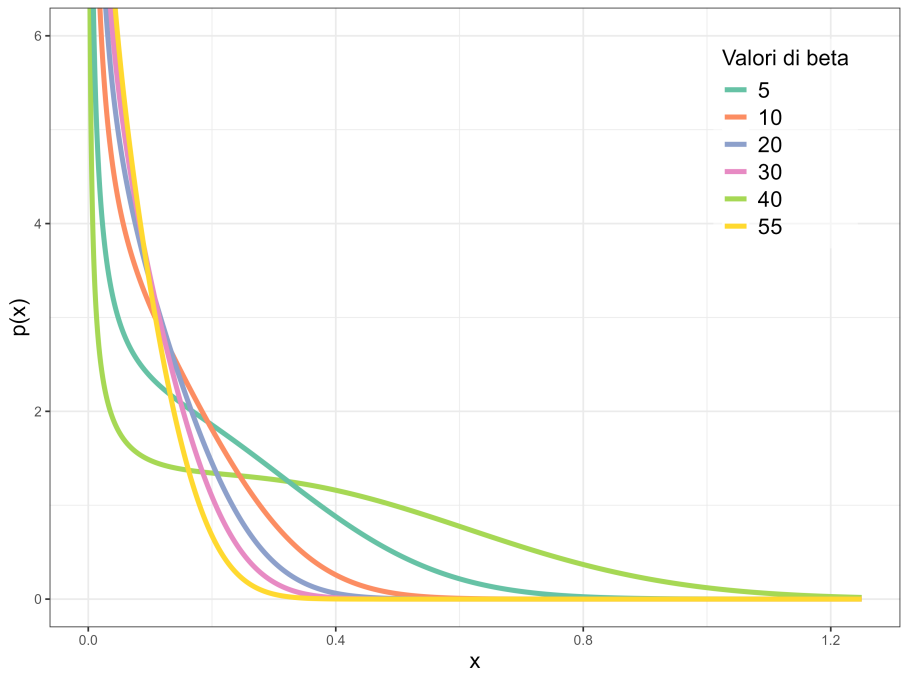

Figura 3.3: Funzioni di densità di $\text{MHN}(0.5, \beta, 4)$ per alcuni valori di β ed $\alpha < 1$.

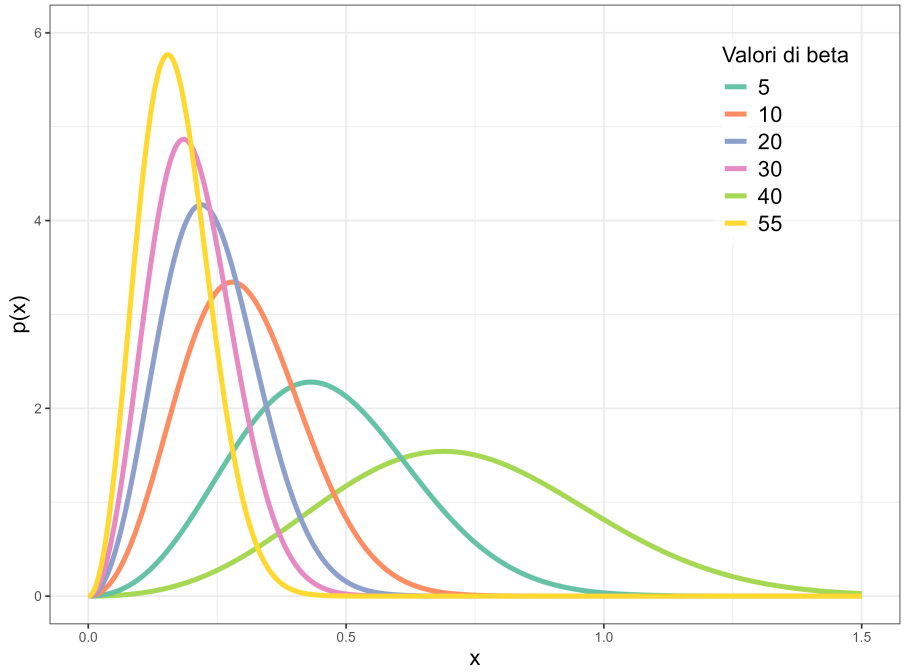

Figura 3.4: Funzioni di densità di MHN$(3, \beta, 4)$ per alcuni valori di β ed $\alpha > 1$.

In questa fase di presentazione della distribuzione, risulta inoltre utile fornire alcune delle sue principali proprietà, descritte in Jingchao Sun & Pal (2023) e riportate nel seguente teorema.

Teorema 3.1. *Data* $X \sim \text{MHN}(\alpha, \beta, \gamma)$*, valgono i seguenti risultati:*

3.1.1 Assumendo $(\alpha + k) \in \mathbb{R}^+$*, il momento k-esimo centrato in* 0 *della distribuzione è dato da*

$$\mathbb{E}\left(X^k\right) = \frac{\Psi\left[\frac{\alpha+k}{2}, \frac{\gamma}{\sqrt{\beta}}\right]}{\beta^{\frac{k}{2}}\Psi\left[\frac{\alpha}{2}, \frac{\gamma}{\sqrt{\beta}}\right]}. \tag{3.3}$$

3.1.2 La varianza della distribuzione è data da

$$\text{Var}(X) = \frac{\alpha}{2\beta} + \mathbb{E}(X)\left[\frac{\gamma}{2\beta} - \mathbb{E}(X)\right]. \tag{3.4}$$

3.1.3 Se $\alpha > 1$*, la moda della distribuzione è data da*

$$\frac{\gamma + \sqrt{\gamma^2 + 8\beta(\alpha - 1)}}{4\beta}. \tag{3.5}$$

3.1.4 Se $\alpha \geq 1$*, la funzione di densità della distribuzione è log-concava.*

Considerando gli obiettivi prefissati, l'ultimo punto del teorema risulta particolarmente utile in quanto identifica tutti i casi in cui la condizione necessaria per l'applicazione del metodo ARS viene soddisfatta. Per questa ragione, nelle fase di confronto tra differenti metodi di campionamento saranno considerati soltanto i casi in cui la distribuzione presenta $\alpha \geq 1$.
Come ultimo risultato della sezione viene riportato un teorema, tratto da Jingchao Sun & Pal (2023), in cui vengono identificati alcuni casi per cui per particolari valori dei parametri di $\text{MHN}(\alpha, \beta, \gamma)$, essa coincide con altre distribuzioni più note.

Teorema 3.2 (Casi particolari della distribuzione *Modified-Half-Normal*)**.**
Data $X \sim \text{MHN}(\alpha, \beta, \gamma)$*, vale che :*

3.2.1 *Se* $\gamma = 0$*, allora* $X^2 \sim \mathit{Gamma}\left(\frac{\alpha}{2}, \beta\right)$.

3.2.2 *Se* $\alpha = 1$ *e* $\gamma = 0$*, allora* $X \sim \mathit{Half-Normal}\left(\frac{1}{\sqrt{2\beta}}\right)$.

3.2.3 *Se* $\alpha = 1$*, allora* $X \sim \mathit{Truncated-Normal}\left(\frac{\gamma}{2\beta}, \frac{1}{\sqrt{2\beta}}, 0, \infty\right)$ *sul supporto* $(0, \infty)$.

3.2.4 *Se* $\beta = 0$ *e* $\gamma < 0$*, allora* $X \sim \mathit{Gamma}(\alpha, -\gamma)$.

Questo tipo di risultato può ritornare utile per valutare la validità dei metodi di campionamento elaborati per il caso generale, confrontando i risultati ottenuti da questi con i casi particolari, per i quali esiste una maggiore disponibilità di tecniche.

3.2 Metodo accettazione-rifiuto ad hoc per la distribuzione Modified-Half-Normal nel caso $\gamma \leq 0$

La distribuzione $\text{MHN}(\alpha, \beta, \gamma)$ è stata descritta nelle sue principali caratteristiche in quanto, come accennato, non risulta possibile campionare direttamente da essa, per cui si necessita dell'utilizzo delle tecniche presentate in precedenza. Di queste, gli algoritmi ARS e del rapporto di uniformi richiedono come funzione di input soltanto la funzione di densità obiettivo, mentre questo non vale nel caso degli algoritmi accettazione-rifiuto standard, i quali necessitano anche di una funzione strumentale che soddisfi la condizione 2.1. I metodi descritti in Jingchao Sun & Pal (2023), sviluppati per campionare specificatamente da questa distribuzione in maniera efficiente, risultano essere di questo ultimo tipo. Essendo quindi questa classe di algoritmi costruiti a partire dalla forma assunta dalla funzione di densità obiettivo e dal momento che nel caso di interesse $\text{MHN}(\alpha, \beta, \gamma)$ questa può variare

in modo considerevole, i metodi accettazione rifiuto ad hoc per questa distribuzione sono differenti in base ai valori assunti dai suoi tre parametri.
In particolare è stato scelto di illustrare il metodo ideato da Jingchao Sun & Pal (2023) per i casi in cui si ha $\gamma \leq 0$, in quanto la funzione strumentale individuata in queste situazioni risulta essere una trasformazione della funzione di densità di una distribuzione nota.

Funzionamento dell'algoritmo

Data la distribuzione $\text{MHN}(\alpha, \beta, \gamma)$, nel caso $\gamma \leq 0$ si ha che

$$f_{\text{MHN}}(x \mid \alpha, \beta, \gamma) \propto x^{\alpha-1} \exp\left(-\beta x^2 - |\gamma| x\right) \mathbb{I}(x > 0). \tag{3.6}$$

Facendo uso della generalizzazione della disuguaglianza tra media aritmetica e media geometrica (*generalized AM-GM inequality*) si può ottenere una funzione strumentale che soddisfa la seguente disuguaglianza:

$$x^{\alpha-1} \exp\left(-\beta x^2 - |\gamma| x\right) \leq x^{\alpha-1} \exp\left(-(\beta + |\gamma|) x^{\frac{2\beta+|\gamma|}{\beta+|\gamma|}}\right). \tag{3.7}$$

Indipendentemente dal valore dei parametri α, β, γ, la diseguaglianza 3.7 si riduce ad un'uguaglianza per $x = 1$: in un intorno di questo punto dunque la funzione strumentale offre una migliore approssimazione della funzione obiettivo rispetto che negli altri punti del supporto. Facendo in modo che il punto di tangenza tra le due funzioni possa variare pur mantenendo valida la relazione 3.7, si può ottenere uno strumento maggiormente flessibile, utile per l'elaborazione di un metodo di campionamento. Introducendo un ulteriore parametro nella funzione strumentale, rappresentante il punto di tangenza, si ottiene una generalizzazione del risultato precedente, la quale viene formalizzata nel seguente teorema.

Teorema 3.3. *Sia* $f_{\text{MHN}}(x \mid \alpha, \beta, \gamma)$ *la funzione di densità della distribuzione* $\text{MHN}(\alpha, \beta, \gamma)$*, con* $\gamma \leq 0$*. Dati* $m \in \mathbb{R}^+$ *e* $K_0(m, \alpha, \beta, \gamma) = \frac{2\beta^{\frac{\alpha}{2}}}{\Psi\left[\frac{\alpha}{2}, \frac{-|\gamma|}{\sqrt{\beta}}\right]}$ *vale che*

$$f_{\text{MHN}}(x \mid \alpha, \beta, \gamma) \leq K_0(m, \alpha, \beta, \gamma) x^{\alpha-1} \exp\left(-m(\beta m + |\gamma|)\left(\frac{x}{m}\right)^{\frac{2\beta m+|\gamma|}{\beta m+|\gamma|}}\right). \tag{3.8}$$

L'importanza di questo teorema è dovuta al fatto che la funzione strumentale proposta può essere normalizzata, andando ad ottenere la funzione di densità della potenza

$\left(\frac{\beta m+|\gamma|}{2\beta m+|\gamma|}\right)$-esima della seguente distribuzione Gamma:

$$T \sim \text{Gamma}\left(\frac{\alpha(\beta m+|\gamma|)}{2\beta m+|\gamma|}, m(\beta m+|\gamma|)\right).$$

Allora, ponendo $Y = T^{\frac{\beta m+|\gamma|}{2\beta m+|\gamma|}}$ si ottiene che

$$f_Y(y) \propto y^{\alpha-1} \exp\left(-m(\beta m+|\gamma|) y^{\frac{2\beta m+|\gamma|}{\beta m+|\gamma|}}\right), \tag{3.9}$$

la quale corrisponde alla funzione strumentale proposta nel Teorema 3.3, a meno di una costante moltiplicativa. Da questo risultato discende che per campionare da una MHN(α, β, γ) con $\gamma \leq 0$, è sufficiente effettuare una trasformazione di un campione ottenuto da una appropriata distribuzione Gamma, per poi valutare se questo debba essere accettato o meno. Questo rappresenta essere un vantaggio notevole, dal momento che essendo la Gamma una distribuzione comune, per campionare da essa sono già stati ideati algoritmi di elevata efficienza, per cui sfruttandoli si ottiene un metodo poco dispendioso a livello computazionale.
Quanto appena descritto discorsivamente può quindi essere implementato nel seguente algoritmo accettazione-rifiuto.

Algoritmo 4: Rejection sampling per MHN(α, β, γ) con $\gamma \leq 0$

Input: $\alpha > 0$, $\beta > 0$, $\gamma \leq 0$, $m > 0$, numerosità N del campione.
Output: Campione $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ da MHN(α, β, γ).

```
i = 0
while i ≤ N do
    Generare x' ~ Gamma((α(βm+|γ|))/(2βm+|γ|), m(βm+|γ|))
    Generare u' ~ U(0,1)
    if u' < exp(m(βm+|γ|)(T/m)^((2βm+|γ|)/(βm+|γ|)) − βx'^2 − |γ|x') then
        x^(i) = x'
        i = i + 1
    end
    else
        Rifiutare x'
    end
end
```

L'algoritmo descritto richiede in input un qualunque valore $m > 0$, tuttavia non tutti i valori di m conducono alle stesse performance. Infatti, in base al punto in cui la

funzione obiettivo e la funzione strumentale sono tangenti, la differenza tra le due aree sottostanti alle due curve risulta essere differente. L'obiettivo di conseguenza è riuscire ad individuare il valore di m che minimizzi l'area compresa tra le due curve, in modo da ottenere una elevata probabilità di accettazione, la quale, come stabilito nella definizione 2.1, risulta essere funzione della distanza tra la funzione obiettivo e quella strumentale. In questo specifico caso quindi l'area corrispondente alla probabilità di accettazione possiederà anch'essa parametri α, β, γ e sarà funzione di m. Questi aspetti sono stati formulati in dettaglio nel seguente teorema.

Teorema 3.4. *Dati $\alpha > 1, \beta > 0$ $\gamma \leq 0$, sia $\hat{a}_{\text{MHN}}(m, \alpha, \beta, \gamma)$ la probabilità di accettazione nell'algoritmo 4. Valgono allora i risultati:*

1. $\hat{a}_{\text{MHN}}(m, \alpha, \beta, \gamma) = \dfrac{(2\beta m+|\gamma|)(\beta m+|\gamma|)^{\frac{\alpha(\beta m+|\gamma|)}{2\beta m+|\gamma|}-1}\Psi\left[\frac{\alpha}{2}, \frac{-|\gamma|}{\sqrt{\beta}}\right]}{2\beta^{\frac{\alpha}{2}} m^{\frac{\alpha\beta m}{2\beta m+|\gamma|}}\Gamma\left(\frac{\alpha(\beta m+|\gamma|)}{2\beta m+|\gamma|}\right)}$.
2. $\forall \alpha > 1, \beta > 0$, $\gamma \leq 0$, *la funzione* $m \mapsto \hat{a}_{\text{MHN}}(m, \alpha, \beta, \gamma)$ *possiede come unico punto di massimo* m_{opt}, *dove* $m_{\text{opt}} > X_{\text{mode}}$, *con* X_{mode} *moda della distribuzione.*

Quanto affermato significa che l'area $\hat{a}_{\text{MHN}}(m, \alpha, \beta, \gamma)$ possiede un unico punto di massimo, per la cui individuazione ci si riconduce quindi ad un problema di ottimizzazione. La strategia che viene proposta è l'applicazione dell'algoritmo Newton-Raphson sulla trasformata logaritmica dell'area di accettazione, definita da

$$\begin{aligned} l(m, \alpha, \beta, \gamma) &= \log\left[\hat{a}_{\text{MHN}}\left(m, \alpha, \beta, \gamma\right)\right] \\ &= \log\left(\Psi\left[\frac{\alpha}{2}, \frac{|\gamma|}{\sqrt{\beta}}\right]\right) + \frac{\alpha(\beta m + |\gamma|)}{2\beta m + |\gamma|}\log\left(\beta m^2 + m|\gamma|\right) \\ &+ \log(2\beta m + |\gamma|) - \log 2 - \frac{\alpha}{2}\log\left(\beta m^2\right) \\ &- \log\left[\Gamma\left(\frac{\alpha(\beta m + |\gamma|)}{2\beta m + |\gamma|}\right)\right] - \log(\beta m + |\gamma|). \end{aligned} \tag{3.10}$$

Essendo un algoritmo iterativo, il suo impiego rende la procedura dispendiosa in termini computazionali, per cui utilizzandolo va considerato che il miglioramento che si ottiene in termini di probabilità di accettazione va a discapito dei tempi di esecuzione dell'algoritmo. Una valida alternativa a questa procedura è data dall'utilizzo del valore m_{init} , il quale risulta essere sufficientemente vicino al valore m_{opt} , pur non richiedendo per la sua individuazione una procedura numerica. Esso inoltre può essere impiegato come punto di inizializzazione nel caso si voglia utilizzare l'algoritmo

Newton-Raphson per l'individuazione di m_{opt} , in quanto ridurrebbe il numero di iterazioni necessarie rispetto alla scelta di un altro valore m per l'inizializzazione. Per il suo calcolo ci si riconduce alla seguente definizione.

Definizione 3.2 (m_{init})**.** Data MHN(α, β, γ), con $\gamma \leq 0$ e $\alpha > 1$, data $\hat{a}_{\text{MHN}}(m, \alpha, \beta, \gamma)$ probabilità di accettazione nell'algoritmo 4, il punto m_{init} viene definito come

$$m_{\text{init}} = \begin{cases} \frac{\alpha^2}{1+\alpha} & \text{se } \alpha \leq 1.1, \\ \frac{3\lambda}{2} X_{\text{mode}} + \left(1 - \frac{3\lambda}{2}\right) X_{\text{inflex}} & \text{se } \alpha > 1.1, \end{cases} \tag{3.11}$$

dove $\lambda = \frac{f_{\text{MHN}}(2X_{\text{mode}} - X_{\text{inflex}})}{f_{\text{MHN}}(2X_{\text{mode}} - X_{\text{inflex}}) + f_{\text{MHN}}(X_{\text{inflex}})}$.
Il valore X_{mode} indica la moda della distribuzione, mentre X_{inflex} indica il punto di flesso della funzione di densità di MHN(α, β, γ).
Il valore di X_{inflex} viene individuato dalla maggior radice reale dall'equazione $\frac{d^2}{dx^2}[x^{\alpha-1} \exp(-\beta x^2 - |\gamma| x)] = 0$, ossia risolvendo

$$\begin{aligned} & x^{\alpha-3} \left(4\beta^2 x^4 + 4\beta |\gamma| x^3 + \left(\gamma^2 + (2 - 4\alpha)\beta\right) x^2 \right. \\ & \left. + (2 - 2\alpha) |\gamma| x + \alpha^2 - 3\alpha + 2\right) \mathrm{e}^{-\beta x^2 - |\gamma| x} = 0 \end{aligned}$$

Con questo si conclude la discussione sulla scelta di m, la quale risulta essere in effetti cruciale per quanto riguarda l'efficienza del metodo di campionamento descritto.
I grafici riportati di seguito permettono di cogliere intuitivamente l'importanza di questo aspetto.

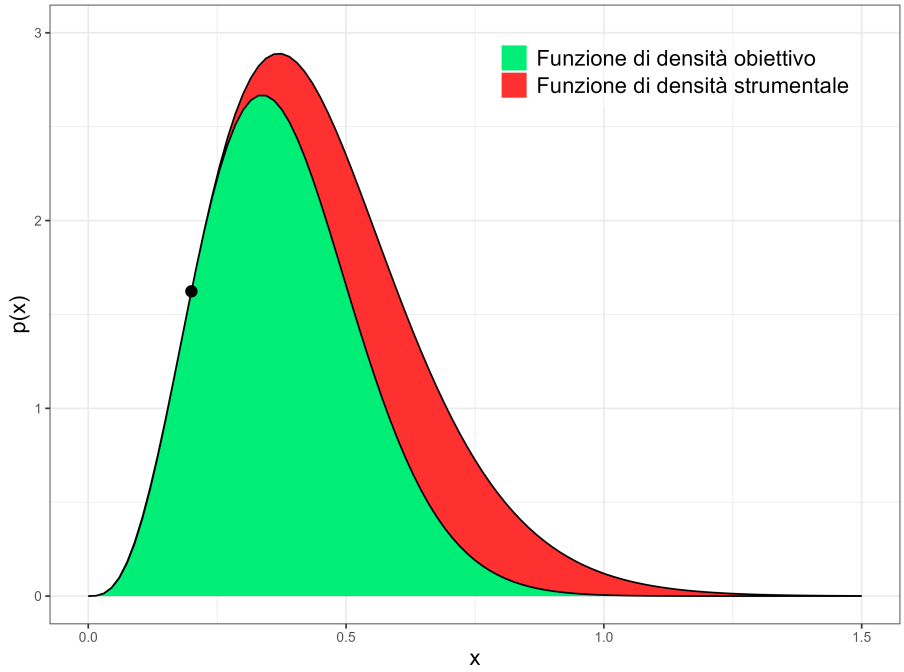

Figura 3.5: Rejection sampling su MHN$(4, 8, -3.5)$, con $m < X_{\text{mode}}$ generico.

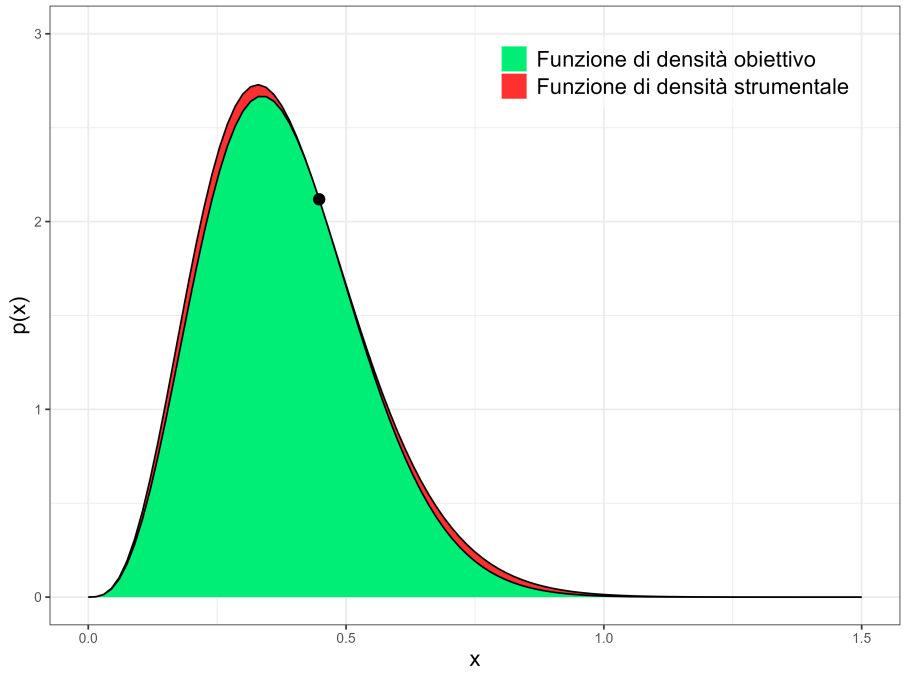

Figura 3.6: Rejection sampling su MHN$(4, 8, -3.5)$, con m_{opt} .

Capitolo 4

Studio empirico

Nei due capitoli precedenti sono state illustrate rispettivamente la teoria che sta dietro i metodi di campionamento indiretti descritti e la distribuzione sulla quale questi verranno applicati. Per poter sfruttare a livello pratico questi risultati teorici, è stato necessario implementarli sotto forma di algoritmi, utilizzando il software R. In questo modo è stato possibile effettuare in primo luogo una verifica per quanto riguarda la loro corretta esecuzione, ed in secondo luogo un confronto tra di essi, in modo da evidenziare i vantaggi ed i limiti presentati dall'utilizzo di ognuno rispetto agli altri.

4.1 Efficacia degli algoritmi

L'aspetto di primaria importanza da verificare prima dell'utilizzo intensivo di un metodo di campionamento indiretto, è chiaramente che questo produca dei valori casuali distribuiti effettivamente in base alla densità da cui si vuole campionare. Dal momento che la distribuzione di interesse MHN(α, β, γ) prevede dei casi in cui essa rappresenta una trasformazione di distribuzioni note, per svolgere il compito descritto si è prima prima affrontato il caso generale e successivamente sono stati sfruttati dei risultati validi per i casi particolari, in modo da ottenere conferme maggiormente affidabili sulla qualità del lavoro svolto.

4.1.1 Caso generale

Il metodo più naturale per accertare l'efficacia degli algoritmi prevede la rappresentazione dei campioni che questi generano sotto forma di istogramma, in modo da ottenere una conferma visiva di quanto la densità di interesse $f(x)$ sia correttamente stimata, la cui curva viene sovraimposta al grafico ottenuto.

Di seguito sono quindi stati riportati gli istogrammi di un campione casuale di numerosità $N = 1000000$ ottenuto campionando con ognuno dei tre algoritmi descritti dalla distribuzione $MHN(4, 3, -2)$.

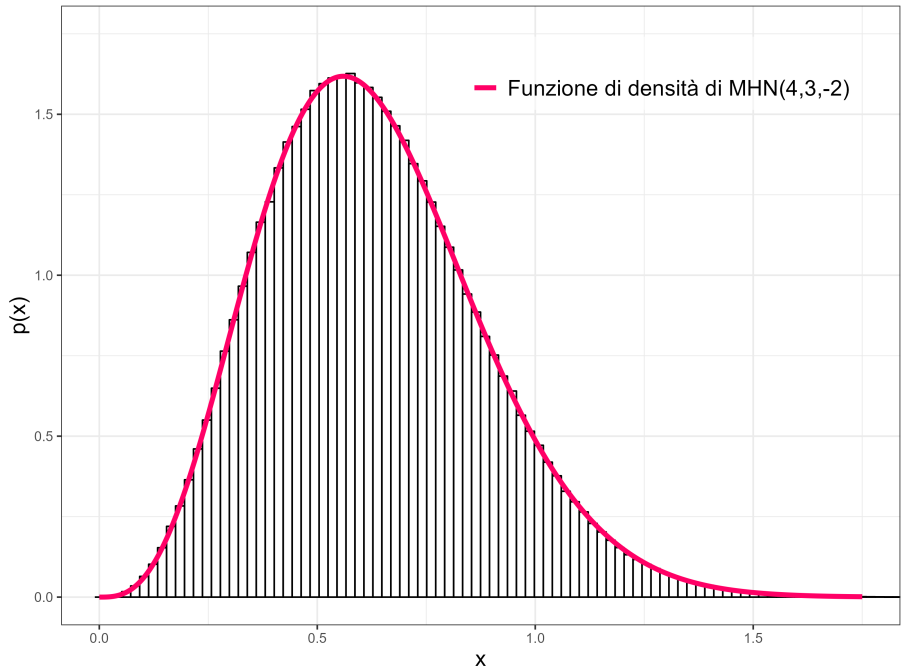

Figura 4.1: Istogramma di un campione casuale di numerosità $N = 1000000$ ottenuto da MHN$(4, 3, -2)$, utilizzando l'algoritmo ARS.

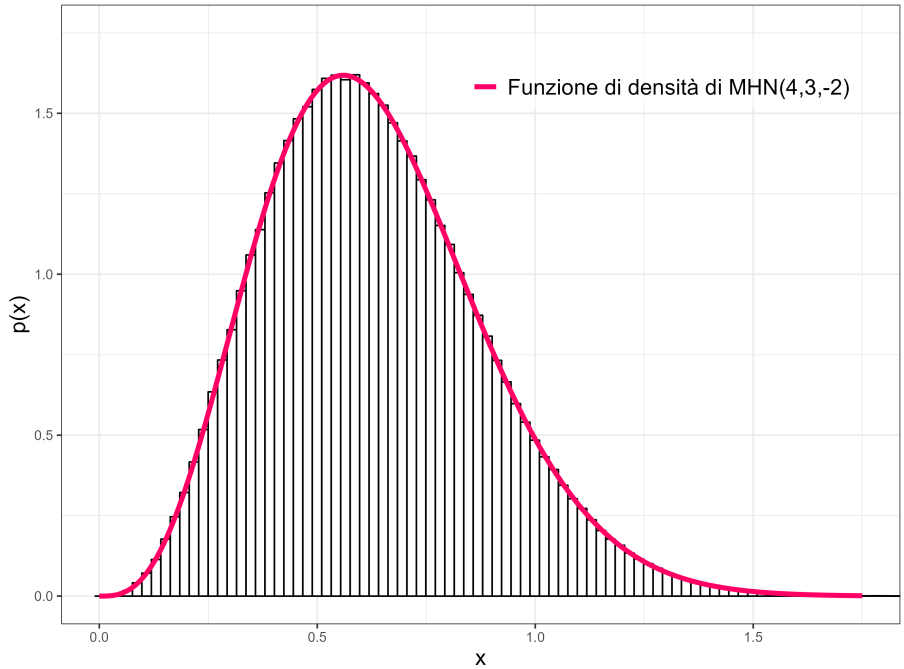

Figura 4.2: Istogramma di un campione casuale di numerosità $N = 1000000$ ottenuto da MHN$(4, 3, -2)$, utilizzando l'algoritmo RS 4.

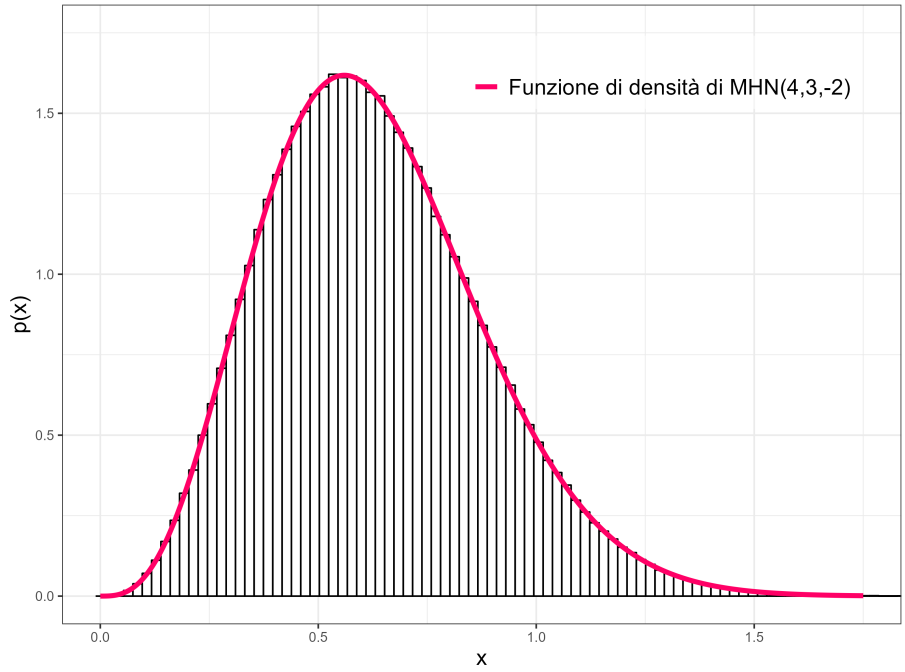

Figura 4.3: Istogramma di un campione casuale di numerosità $N = 1000000$ ottenuto da $\text{MHN}(4, 3, -2)$, utilizzando l'algoritmo ROU.

Si nota che l'utilizzo di una qualsiasi dei tre metodi proposti produce il medesimo risultato: l'istogramma ottenuto dal campione casuale generato si adatta in modo sostanzialmente perfetto alla curva rappresentate la funzione di densità di $MHN(4, 3, -2)$, indice del fatto che gli algoritmi di campionamento siano stati implementati correttamente.

4.1.2 Caso particolare

Nel terzo capitolo il teorema 3.2 fornisce un elenco dei casi in cui, per determinati valori dei suoi parametri, la distribuzione *Modified-Half-Normal* coincide con distribuzioni più comuni o con loro trasformazioni. Come era stato accennato, questo tipo di risultati si dimostra di grande utilità, dal momento che permettono di sfruttare tecniche elaborate per altre distribuzioni per validare i risultati ottenuti nei casi generali. In particolare, di seguito viene utilizzato il risultato fornito in 3.2.1, essendo la distribuzione Gamma tra le più note. Per farlo, i valori dei campioni casuali generati con i tre algoritmi oggetto di analisi sono stati elevati al quadrato. Nel caso questi risultino come provenienti dall'appropriata distribuzione Gamma come stabilito nel teorema citato, si otterrebbe un'ulteriore conferma della correttezza del lavoro svolto. Per verificare questo aspetto ci si basa sul confronto tra le funzione di ripartizione empirica (ECDF) ottenuta dai campioni casuali e la funzione di

ripartizione teorica (CDF) della distribuzione Gamma, attraverso l'utilizzo di un test statistico non parametrico, noto come test di Kolmogorov-Smirnov. Il test nella sua forma "a due code" sottopone a verifica il sistema di ipotesi:

$$\begin{cases} \mathcal{H}_0 : & F_n(x) = F(x) \quad \forall x \\ \mathcal{H}_1 : & F_n(x) \neq F(x) \quad \text{per qualche } x, \end{cases}$$

impiegando la statistica test $D_n = \sup_x |F_n(x) - F(x)|$.
$F(x)$ corrisponde alla CDF della distribuzione teorica ed $F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{(-\infty,x]}(X_i)$ corrisponde alla ECDF del campione. Esaminando la differenza massima che intercorre tra le due funzioni si possono ottenere quindi informazioni utili sulla distribuzione dei valori campionati.
Il test in questo caso è stato utilizzato per osservare se i valori al quadrato di un campione casuale da MHN$(3, 2, 0)$, generati con i metodi indiretti descritti, si distribuissero in base a Gamma$(1.5, 2)$. Il test è stato ripetuto 500 volte per ogni algoritmo su campioni di numerosità $N = 100000$, per poi calcolare per ognuno di questi la percentuale di volte in cui il p-value del test è risultato essere maggiore della soglia 0.1, per le quali l'ipotesi nulla viene accettata.

Algoritmo	**Percentuale di p-values $>$ 0.1**
ARS	83.6 %
RS	88.6 %
ROU	92.2 %

Tabella 4.1: Percentuale di p-values nel test Kolmogorov-Smirnov superiori alla soglia 0.1 su 500 campioni di numerosità $N = 100000$ ottenuti con ciascun algoritmo.

I risultati ottenuti sono decisamente positivi: la percentuale minore è superiore ad 83%, fino ad arrivare al 92% nel caso dell'algoritmo ROU. Quanto ottenuto dunque attesta come nella maggioranza dei casi non esista una differenza statisticamente significativa tra la distribuzione dei campioni generati e quella teorica. Si può quindi affermare come l'implementazione degli algoritmi sia stata eseguita correttamente, in quanto ognuno di essi riesce a produrre con precisione valori dalla distribuzione di interesse. Questo aspetto tuttavia non costituisce l'unico metro di valutazione, per cui nelle fasi seguenti ne sono stati confrontati di ulteriori.

4.2 Percentuali di campioni rifiutati

Gli algoritmi di campionamento utilizzati prevedono ognuno una componente di verifica per l'eventuale accettazione dei campioni generati, facendo parte della categoria delle tecniche di *rejection sampling*. Come reso noto in precedenza, una qualità che ne contraddistingue un'elevata efficienza è il numero di valori che devono essere rifiutati per ottenere un campione della numerosità N desiderata, il quale chiaramente si desidera essere il minore possibile. Risulta quindi naturale effettuare un confronto rispetto a questo aspetto tra gli algoritmi, in modo da evidenziare quale di questi conduca alla maggiore probabilità di accettazione. In questo contesto va inoltre ricordata la particolare natura dell'algoritmo ARS, il quale presenta il vantaggio di utilizzare l'informazione fornita dai campioni rifiutati per incrementare gradualmente la probabilità di accettazione. Lo stesso chiaramente non accade nel caso degli altri due metodi, i quali presenteranno necessariamente una proporzione di valori accettati sostanzialmente costante all'aumentare di N.
Di seguito è stato riportato un grafico raffigurante le percentuali dei campioni rifiutati sul totale dei valori generati, a fronte di alcune diverse numerosità campionarie. Per ogni valore di N è stata riportata la mediana su 100 campioni ottenuti dalla distribuzione MHN$(4, 8, -2.5)$, la quale rappresenta una stima della probabilità di rifiuto.

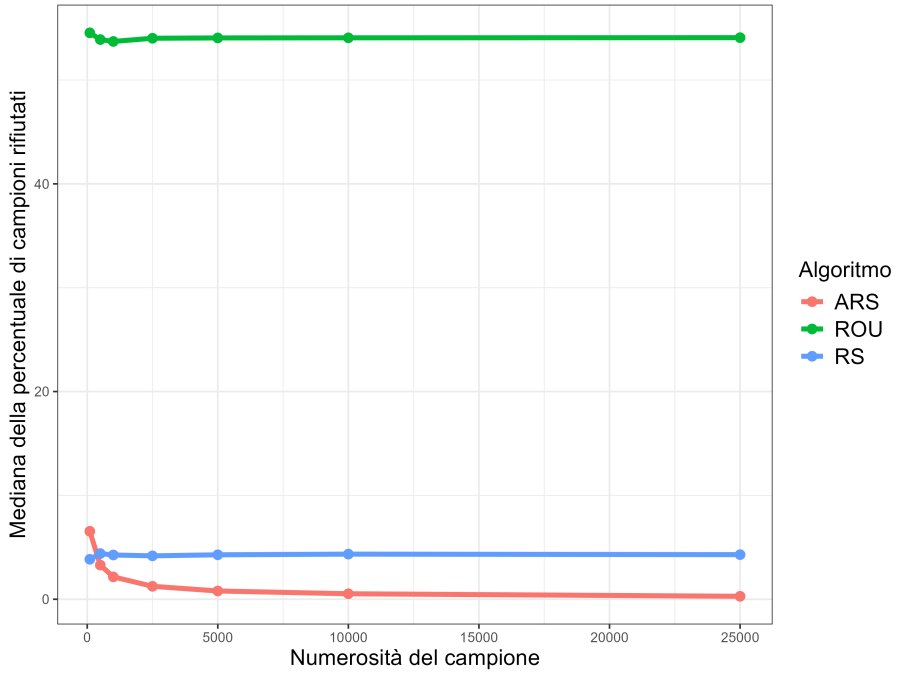

Figura 4.4: Mediana della percentuale di campioni rifiutati per ognuno dei tre algoritmi, considerando campioni da MHN$(4, 8, -2.5)$.

Come ci si aspettava, nei casi degli algoritmi ROU ed RS, la percentuale di rifiuti rimane stazionaria all'aumentare del valore di N, tuttavia ciò che è interessante notare è la differenza notevole che intercorre tra le soglie dei due valori. L'algoritmo ROU infatti conduce ad una percentuale di campioni rifiutati visibilmente superiore rispetto agli altri, con valori che risultano essere persino di poco maggiori al 50%. Questa situazione deve essere dovuta al fatto che l'area compresa nella curva di densità della distribuzione nel nuovo sistema di coordinate rappresenta una percentuale poco elevata sullo spazio totale compreso nella regione rettangolare da cui si campiona in questo metodo. Al contrario, essendo l'algoritmo RS utilizzato ideato ad hoc per il tipo di distribuzione da cui si sta campionando, esso possiede una percentuale di rifiuti molto contenuta. Vale invece un discorso a parte per l'algoritmo ARS, il quale essendo adattivo come illustrato in precedenza, presenta una evidente decrescita della percentuale di rifiuti sul totale dei campioni generati, la quale già per $N = 25000$ si nota essere vicina allo 0%. Questi aspetti possono essere compresi con un maggiore livello di dettaglio osservando la tabella seguente, la quale riporta la mediana delle percentuali di campioni accettati su 100 generazioni di campioni di diverse numerosità dalla distribuzione sopracitata. Va considerato anche in questo caso che i valori riportati costituiscono delle stime e non rappresentano il valore effettivo della probabilità di accettazione, la quale è un valore fissato e strutturale per gli algoritmi, calcolabile secondo quanto fornito nella definizione 2.1.

N:	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**	**25000**
RS	95.24%	95.79%	95.97%	95.79%	95.75%	95.69%	95.75%	95.75%
ARS	92.59%	95.42%	96.90%	97.80%	98.80%	99.19%	99.47%	99.71%
ROU	45.66%	46.00%	46.17%	46.06%	45.90%	45.99%	46.07%	45.89%

Tabella 4.2: Percentuale mediana dei valori accettati su 100 campioni per ognuno degli algoritmi, considerando campioni da $\text{MHN}(4, 8, -2.5)$.

Osservando questi valori, emerge ancora più chiaramente il limite del metodo ROU, il quale conduce ad una accettazione mediana intorno al 46%, significando quindi che per ogni campione generato ne viene rifiutato più di uno: questo aspetto rappresenta un chiaro svantaggio in termini di efficienza. Per quanto riguarda gli altri due metodi, i risultati ottenuti sono invece ottimi. L'algoritmo RS si attesta infatti su percentuali di accettazioni sul totale dei valori generati decisamente elevate, intorno al 95.7%. Le performance migliori sono tuttavia state ottenute dall'algoritmo ARS, come era facile prevedere, avendo esso la peculiarità di poter migliorare la propria probabilità di accettazione. Esso infatti, sfruttando i campioni via via rifiutati, passa da una

percentuale di 92.59% per $N = 100$, inferiore a quella ottenuta dall'algoritmo RS, fino ad una percentuale pari a 99.7% per $N = 25000$. In questo caso quindi il numero di rifiuti diventa sostanzialmente irrisorio all'aumentare della numerosità campionaria, in quanto con il suo incremento sempre meno valori non vengono accettati. Questo tipo di vantaggio tuttavia presenta inevitabilmente un costo: dover aggiornare la funzione strumentale con ogni valore rifiutato comporta sicuramente un dispendio in termini di risorse computazionali. L'analisi appena effettuata dunque non esaurisce gli aspetti attraverso cui questi algoritmi possono essere confrontati e analizzati.

4.3 Tempi di esecuzione

Un ulteriore aspetto da considerare nella valutazione di un algoritmo di campionamento è sicuramente il tempo che questo impiega per la generazione del campione voluto. Soprattutto nei casi in cui si necessiti di campioni a numerosità elevata, disporre di metodi quanto più possibilmente veloci risulta infatti cruciale. Per questa ragione l'ultimo confronto effettuato risuta essere in termini di tempo di esecuzione degli algoritmi. Questo tipo di analisi è stata eseguita facendo uso della funzione `microbenchmark` di `R`, la quale permette di ottenere il tempo mediano di esecuzione delle porzioni di codice che le vengono forniti in input, considerando un numero di iterazioni arbitrario, in questo caso impostato pari a 100. Di seguito sono riportati i confronti grafici dei risultati così ottenuti, sempre considerando diverse numerosità N di campioni dalla distribuzione $\text{MHN}(4, 8, -2.5)$.

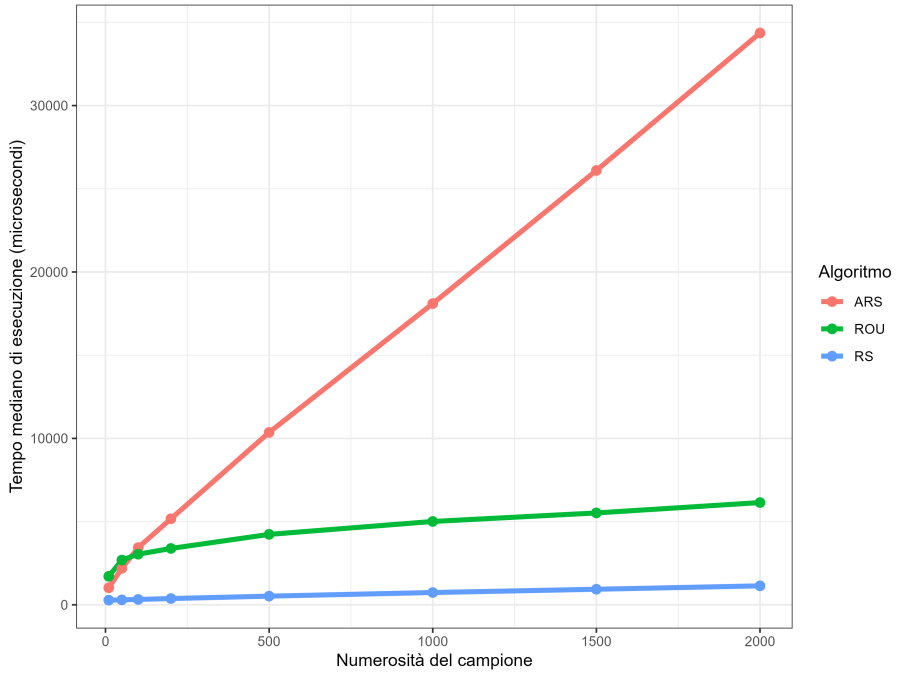

Figura 4.5: Mediana dei tempi per ognuno dei tre algoritmi, considerando campioni da MHN$(4, 8, -2.5)$, da $N = 10$ fino ad $N = 2000$.

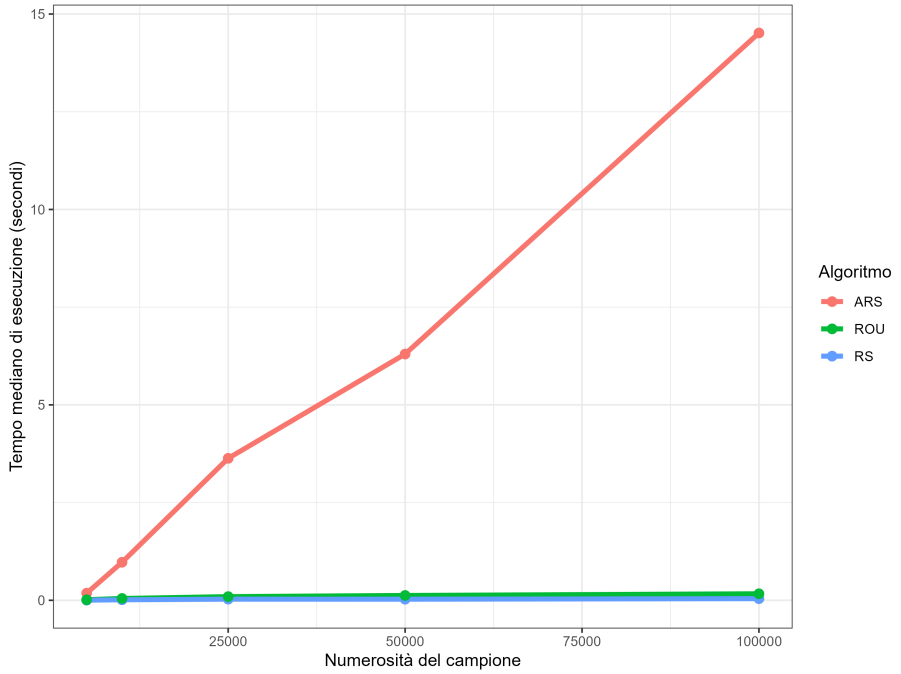

Figura 4.6: Mediana dei tempi per ognuno dei tre algoritmi, considerando campioni da MHN$(4, 8, -2.5)$, da $N = 5000$ fino ad $N = 100000$.

Si osserva una differenza sostanziale in termini di tempistiche per quanto riguarda i tre algoritmi. L'algoritmo ARS ha ottenuto le performance peggiori con un evidente

distacco, presentando i tempi di gran lunga più elevati, i quali risultano inoltre crescere ad un ritmo estremamente rapido all'aumentare della numerosità campionaria. Questa comportamento era già stato supposto nella fase precedente: per poter ottenere un probabilità di accettazione sempre più elevata deve essere eseguito un processo di aggiornamento della funzione strumentale, il quale sicuramente influisce in maniera pesante sui tempi in cui l'algoritmo viene eseguito, essendo computazionalmente esigente. Per permettere il funzionamento dell'algoritmo ARS, deve inoltre essere estratto un intervallo per decidere da quale componente della funzione strumentale si debba campionare, i quali possiedono pesi molto variabili: nel momento in cui questi diventano un numero elevato, selezionarne casualmente uno richiede più tempo di quello che servirebbe per campionare da una distribuzione comune, come avviene negli altri due casi. Questo algoritmo fa poi utilizzo del metodo dell'inversione, per cui richiede la valutazione di una funzione ad ogni iterazione, aspetto che ne limita ulteriormente la rapidità. Gli altri due algoritmi infatti, appoggiandosi ad algoritmi di campionamento per distribuzioni note, presentano tempi di esecuzione decisamente inferiori. L'algoritmo ROU conduce al secondo miglior risultato, con un tempo di molto inferiore rispetto al peggiore appena descritto, compensando quindi almeno in parte l'elevata proporzione di rifiuti che effettua. Il risultato migliore invece, come era ragionevole aspettarsi, lo ottiene l'algoritmo RS ideato specificatamente per la distribuzione in esame, il quale presenta i tempi di esecuzione nettamente minori e con la crescita meno marcata. In ogni caso si nota però come il ritmo di crescita dei tempi rispetto all'algoritmo ROU sia sostanzialmente lo stesso, a fronte di una differenza in termini di tempi di inizializzazione che diventa praticamente irrilevante nella generazione di grossi campioni. Il quadro che emerge in questa analisi depone quindi a favore dell'utilizzo delle tecniche non adattive, le quali conducono ad un evidente risparmio, aspetto che diventa di fondamentale importanza soprattutto nel caso vogliano essere impiegati campioni a numerosità particolarmente elevate.

4.4 Conclusioni

Come discusso, gli algoritmi di campionamento presentati sfruttano ognuno la tecnica dell'accettazione-rifiuto per ottenere il medesimo risultato, il quale è stato dimostrato essere stato raggiunto. I differenti risultati teorici su cui questi metodi sono stati ideati e le rispettive implementazioni hanno tuttavia condotto a delle evidenti differenze in termini di performance. Nonostante l'obiettivo fosse infatti comune, le caratteristiche strutturali degli algoritmi erano del tutto peculiari. In particolar modo, gli aspetti di

maggior rilevanza in questo contesto risultano essere la quantità di input necessari e l'universalità di questi algoritmi. Considerando per primo l'algoritmo ARS si intuisce il ruolo queste caratteristiche. Esso necessita del maggior numero di input e risulta applicabile nei casi, comunque piuttosto numerosi, di log-concavità della funzione di densità di interesse, richieste date dal fatto che esso può in questo modo sfruttare un maggior numero di informazioni, andando così ad ottenere un miglioramento notevole in termini di probabilità di accettazione. In questo processo la questione computazionale diventa però importante: in modo da migliorare una singola caratteristica, i tempi di esecuzione si prolungano notevolmente. Si comprende allora come i singoli punti di forza di un algoritmo di campionamento vadano tendenzialmente a discapito di altri aspetti. Considerando invece l'algoritmo RS si evidenzia infatti la situazione opposta: esso richiede input molto limitati e presenta sia un'alta probabilità di accettazione che tempi di esecuzione molto rapidi, tuttavia non è in alcun modo universale, dal momento che non copre neppure tutti i casi della particolare distribuzione per cui è stato ideato. Esso risulta quindi di notevole efficienza, ma essendo applicabile in contesti estremamente specifici. Il metodo ROU presenta invece un'elevata universalità e tempi di esecuzione decisamente buoni, pur richiedendo in input la sola funzione di densità non normalizzata. Queste caratteristiche del tutto positive hanno però come controparte uno svantaggio evidente: la probabilità di accettazione è infatti stata stimata come inferiore al 50%, significando quindi un dispendio di risorse di calcolo rilevante, in questo modo infatti più di un campione viene rifiutato per ognuno accettato. Emerge quindi come non sia possibile decretare in maniera netta il metodo di campionamento da preferire, non eccellendo nessuno in ogni aspetto, motivo appunto per il quale nel tempo ne sono stati elaborati diversi. Nell'effettuare una scelta tra essi ciò che può essere quindi consigliabile fare è comprendere a fondo l'obiettivo che si vuole raggiungere e le informazioni di cui si dispone, in modo tale da selezionare il compromesso disponibile più conveniente, dal momento che, come accertato, non esiste una soluzione necessariamente migliore altre.

Capitolo 5

Codice R

```
####################
#### CAPITOLO 2 ####
####################

library(ggplot2)

#Campionamento da Half-Normal(0,1)
rs_halfnormal_std=function(N){
  fun=function(x) {sqrt(2 / pi) * exp(- x^2 / 2)}

  sample=c()
  rejected=c()

  samples_left=N-length(sample)

  #aggiunti per rappresentazione grafica
  dd=as.data.frame(matrix(ncol = 3, nrow = 0))
  colnames(dd)=c('x','y','accepted')

  while (samples_left!=0) {

    #con l'esponenziale
    u=runif(samples_left,0,1)
    x=-log(u)
    y=runif(samples_left, 0, prop_f(x))
```

```
   #con l'uniforme
   #x=runif(samples_left,0,4.5)
   #y=runif(samples_left,0,fun(0))

   condition=y<=fun(x)

   sample=append(sample,x[condition])
   rejected=append(rejected,x[!condition])

   dd=rbind(dd, data.frame(x,y, accepted = condition))

   samples_left=N-length(sample)
  }

  return(list(sample=sample,rejected=rejected,data=dd))
}
samp=ar_halfnormal_std(10000)

#grafico campionando da Exp(1)
f1 = function(x) {sqrt(2 / pi) * exp(- x^2 / 2)}
f2 = function(x) {sqrt(2 * exp(1) / pi) * exp(-x)}
ggplot(samp$data, aes(x=x,y=y,color = accepted))+
  geom_point()+xlim(0,5.5)+
  stat_function(fun = f1, color = "blue", size = 1.5)+
  stat_function(fun = f2, color = "red3", size = 1.5 )+
  theme_bw()+
  scale_color_discrete(name = "", labels = c("Rejected", "Accepted"))+
  theme(legend.text = element_text(size = 14),
        axis.title = element_text(size = 14))+
  labs(x='x',y='p(x)')

#grafico campionando da U(0,4.5)
ggplot(samp$data, aes(x=x,y=y,color = accepted))+
  geom_point()+xlim(0,4.5)+
  stat_function(fun = f1, color = "blue", size = 1.5)+
  geom_segment(aes(x = 0, y = f1(0), xend = 4.5, yend = f1(0)), size=1.5,
      color='red3')+
  theme_bw()+
```

```
  scale_color_discrete(name = "", labels = c("Rejected", "Accepted"))+
  theme(legend.text = element_text(size = 14),
       axis.title = element_text(size = 14))+
  labs(x='x',y='p(x)')


#Funzione di densita' di MHN non normalizzata, nel caso gamma<=0
mhn.nonorm=function(a,b,c){
  #closure
  function(x){
    r=x^(a-1) * exp(-(b*x^2) - (abs(c)*x))
    return(r)
  }
}


#ALGORITMO ARS

#Funzioni ausiliarie

#Funzione che restituisce il logaritmo della funzione in input
logfun=function(fun){
  return(function(x) log(fun(x)))
}

#Funzione per l'individuazione dei punti di supporto iniziali (J. James)
find_start_points=function(fun, high_d_point){
  x0=high_d_point
  eps=.Machine$double.eps^(1/3)
  x1=x0
  x2=x1+eps
  h=logfun(fun)
  h1=h(x1)
  v1=numDeriv::grad(h,x1)
  h2=h(x2)
  v2=numDeriv::grad(h,x2)

  while (v1*v2>=0){
```

```
   theta=(v2-v1)/(x2-x1)
   x2=x1
   h2=h1
   v2=v1
   mu=x2-(v2/theta)
   sigma=1/sqrt(-theta)
   x1=mu+sign(v2)*sigma
   h1=h(x1)
   v1=numDeriv::grad(h,x1)
  }

  if(x2<x1){
   temp=x1
   x1=x2
   x2=temp
  }
  start.points=c(x1,x2)

  theta=(v2-v1)/(x2-x1)
  mu=x2-(v2/theta)
  sigma=1/sqrt(abs(theta))

  #aggiunge nuovi punti di supporto, se necessario
  if((x1-mu)/sigma>-0.16){
   x3=x1-sigma
   start.points=append(start.points,x3)
  }

  if((x2-mu)/sigma<0.16){
   x4=x2+sigma
   start.points=append(start.points,x4)
  }

  return(c(sort(start.points,decreasing=F)))
}

#Funzione che calcola i punti di intersezione tra due rette
calc_int_point=function(intercepts,slopes){
```

```
  n=length(intercepts)
  x=(intercepts[2:n]-intercepts[1:(n-1)])/(slopes[1:(n-1)]-slopes[2:n])
  y=slopes[1:(n-1)]*x+intercepts[1:(n-1)]
  return(coord.int.point=list(x=x,y=y))
}

#Funzione che calcola tutti i punti di intersezione tra rette consecutive,
#prende in input la funzione ed i punti di tangenza tra funzione e rette.
#Restituisce le coordinate dei punti ed intercetta
#e coefficiente angolare delle rette ordinate.
calc_intersection_points=function(fun,xx){

  #coefficienti angolari delle rette
  m=numDeriv::grad(fun,xx)

  #intercette delle rette
  q=fun(xx)-m*xx

  n=length(xx)
  int.points=calc_int_point(q[1:n],m[1:n])
  lines.data=append(int.points,list(m,q))
  names(lines.data)=c('x','y','m','q')

  return(lines.data)
}

#Funzione che calcola i pesi di ognuna delle aree sottostanti
#alle differenti componenti della funzione strumentale
calc_weights=function(xx,mm,qq){

  n=length(xx)

  #nel caso uno dei coefficienti angolari delle rette sia esattamente 0:
  if(0 %in% mm) {
    mm[which(mm==0)]=.Machine$double.xmin
  }

  #calcolo delle aree (integrale svolto)
```

```
  areas=(1/mm[1:(n-1)]) * (exp(mm[1:(n-1)]*xx[2:n] +qq[1:(n-1)]) -
                       exp(mm[1:(n-1)]*xx[1:(n-1)] +qq[1:(n-1)]))

  #calcolo dei pesi
  weights=areas/sum(areas)

  return(weights)
}

#Funzione principale
#Restituisce il campione, i valori rifiutati, i punti di tangenza e
#di intersezione delle rette.
ars_alg=function(N,fun,bounds=c(-Inf,Inf),high_dens_point,start_points=NULL)
    {

  if(N<=0 | N%%1 != 0){stop("Error: N must be a positive integer.")}
  if(!is.numeric(bounds)){stop("Error: bound values are not numeric.")}
  if(length(bounds)!=2){stop("Error: bounds argument must be of length 2.")}
  if(!is.function(fun)){stop("Error: argoment fun must be a function.")}

  sample=c()
  rejected=c()

  f=logfun(fun)

  #Inizializzazione:
  #starting points
  #(o inseriti manualmente o individuati con un algoritmo apposito)
  if (is.null(start_points)) {
    tang.points = find_start_points(fun, high_dens_point)
  } else {
    if (!is.numeric(start_points)) {
      stop("Error: start_points must be numeric.")
    }
    tang.points = sort(start_points)
  }

  #punti di intersezione tra le rette tangenti
```

```
  int.lines=calc_intersection_points(f,tang.points)
  m=int.lines$m
  q=int.lines$q
  x.pieces=sort(c(bounds[1],int.lines$x,bounds[2]))

  #pesi
  w=calc_weights(x.pieces,m,q)

  #estrazione
  while (length(sample)<N) {

    i=sample(length(w),1,replace=T,prob=w)
    u=runif(1)
    s=(1/m[i])*log((u*exp(m[i]*(x.pieces[i+1]-x.pieces[i]))) -u +1) + x.
        pieces[i]

    if(u<=(fun(s)/exp(int.lines$m[i]*s +int.lines$q[i]))){
      sample=append(sample,s)
    }
    #eventuale aggiornamento
    else{
      rejected=sort(append(rejected,s))
      tang.points=sort(append(tang.points,s))
      int.lines=calc_intersection_points(f,tang.points)
      m=int.lines$m
      q=int.lines$q
      x.pieces=sort(c(bounds[1],int.lines$x,bounds[2]))
      w=calc_weights(x.pieces,m,q)
    }

  }

  x.int=sort(x.pieces)[2:(length(x.pieces)-1)]

  return(res=list(sample=sample,rejected=rejected,int.points.x=x.int, tang.
      points=tang.points))
}
```

```
#Funzione per il grafico dell'algoritmo ARS
plot_ars=function(N,funz,start_p,seme){
 #per rendere il grafico riproducibile
 set.seed(seme)

 f1=function(x,m,q){
  return(exp(m*x+q))}

 h=logfun(funz)

 sam=ars_alg(N,funz,c(0,Inf),start_p)

 #coordinate dei punti di intersezione tra le rette
 zz=calc_intersection_points(h,sam$tang.points)

 #coordinate dei punti di di tangenza delle rette ad h
 punti=data.frame(t(rbind(sam$tang.points,funz(sam$tang.points))))
 colnames(punti)=c('x.tang.points','y.tang.points')

 p=ggplot(data=punti, aes(x=x.tang.points, y=y.tang.points))+
  geom_point()+xlim(0,1.7)+ylim(0, 0.025)+
  stat_function(fun=funz, geom = "area", color = "black",
   fill = "turquoise2", alpha=1)+
  #vengono aggiunti manualmente solo la prima e l'ultima componente della
  #funzione strumentale
  stat_function(fun = function(x) f1(x, zz$m[1], zz$q[1]), xlim=c(0,zz$x
      [1]),
  geom = "area", color = "black", fill = "gold", alpha=0.3)+
  stat_function(fun = function(x) f1(x, zz$m[length(zz$m)], zz$q[length(zz$
      m)]), xlim=c(zz$x[length(zz$x)],1.7), geom = "area", color = "black",
       fill = "gold", alpha=0.3)+
  theme_bw()+labs(x='x',y='p(x)')

 #altre componenti
 exp_plots = function(m, q, xx) {
  stat_function(fun = function(x) f1(x, m, q) , xlim=c(xx[1],xx[2]),
  geom = "area", color = "black", fill = "gold", alpha=0.3)}
```

```
  xx_pairs = map2(zz$x[-length(zz$x)], zz$x[-1], c)
  #Necessaria la funzione pmap per aggiungere diverse funzioni ad un ggplot,
  #senza aggiungerle una ad una manualmente
  #un ciclo for avrebbe aggiunto soltanto l'ultima del ciclo
  p <- p + unlist(pmap(list(zz$m[2:(length(zz$m)-1)],
  zz$q[2:(length(zz$m)-1)], xx_pairs), exp_plots))

  #dataset fittizio, utile solo per scopi grafici (la legenda)
  #questione cromatica: l'area della funzione strumentale colorata
  #con gold con alpha=0.3, si sovrappone all'area della funzione obiettivo
  #colorata con turquoise2, generando un nuovo colore, sostanzialmente
  #uguale a springgreen2, colore selezionato per la legenda
  legend_data = data.frame(
    x = c(Inf, Inf),
    y = c(Inf, Inf),
    fill = c("springgreen2", "gold"),
    label = c("Funzione␣di␣densit␣obiettivo", "Funzione␣di␣densit␣
        strumentale"))

  p = p +
    geom_point(data = legend_data, aes(x = x, y = y, color = label),
    size = 0, show.legend = TRUE) +
    scale_color_manual(name = "",
      values = c("Funzione␣di␣densit␣obiettivo" = "springgreen2",
               "Funzione␣di␣densit␣strumentale" = "gold"))+
    guides(color = guide_legend(override.aes = list(size = 5)))+
    theme(
      legend.position = c(0.95, 0.95),
      legend.justification = c("right", "top"),
      legend.background = element_rect(fill = alpha('white', 0.5)),
      legend.text = element_text(size = 14),
      axis.title = element_text(size = 14))

  return(p)
}
```

```
#ALGORITMO ROU

#Funzione principale
rou_sampler=function(N,fun){

 sample=c()
 rejected=c()

 sqrt_ff=function(x){return(sqrt(fun(x)))}
 u_sup=optimize(sqrt_ff, c(0,1), maximum = T)$objective

 x_sqrt_ff=function(x){return(x*sqrt(fun(x)))}
 v_sup=optimize(x_sqrt_ff, c(0,1), maximum = T)$objective
 v_inf=optimize(x_sqrt_ff, c(0,0.001), maximum = F)$objective

 samples_left=N-length(sample)

 #Informazioni salvate per il grafico
 dd=as.data.frame(matrix(ncol = 3, nrow = 0))
 colnames(dd)=c('u','v','rej')

 #estrazione
 while (samples_left!=0) {
  u=runif(samples_left,0,u_sup)
  v=runif(samples_left,v_inf,v_sup)
  x=v/u

  condition=u<=sqrt_ff(v/u)

  sample=append(sample,x[condition])
  rejected=append(rejected,x[!condition])

  dd=rbind(dd, data.frame(u = u, v = v, accepted = condition))

  samples_left=N-length(sample)
 }

 return(list(sample=sample,rejected=rejected,data=dd))
```

```
}

#Rappresentazione grafica della funzione di densita' non normalizzata
#nel nuovo sistema di coordinate (esempio riportato)
ff=mhn.nonorm(4,8,-2)

sqrt_f=function(x){return(sqrt(ff(x)))}
x_sqrt_f=function(x){return(x*sqrt(ff(x)))}

xx = seq(0,1.5, by = 0.001)
df = data.frame(x = x_sqrt_f(xx), y = sqrt_f(xx))

ggplot(df, aes(x = x, y = y)) + geom_point() +theme_bw()

#Rappresentazione grafica precedente, ma con campioni accettati e non
#e regione rettangolare di campionamento
u_sup=optimize(sqrt_f, c(0,1), maximum = T)$objective
v_sup=optimize(x_sqrt_f, c(0,1), maximum = T)$objective
v_inf=optimize(x_sqrt_f, c(0,0.001), maximum = F)$objective

ggplot() +
 geom_point(data=sam$data, aes(x = v, y = u, color = accepted))+
 geom_point(data=df, aes(x = x, y = y))+
 scale_color_discrete(name = "", labels = c("Rejected", "Accepted"))+
 geom_hline(yintercept = u_sup, linetype = "dashed", color = "slategray",
     size = 1.5) +
 geom_hline(yintercept = 0, linetype = "dashed", color = "slategray",size =
      1.5)+
 geom_vline(xintercept = v_sup , linetype = "dashed", color = "slategray",
     size = 1.5) +
 geom_vline(xintercept = v_inf , linetype = "dashed", color = "slategray",
     size = 1.5)+
 theme_bw()+
 theme(legend.text = element_text(size = 14),
      axis.title = element_text(size = 14))
```

```
####################
#### CAPITOLO 3 ####
####################

#Approssimazione funzione Fox-Wright nel caso di interesse
fw_approx=function(a,b,c,n,z){
 #n massimo pari a 170 altrimenti il fattoriale diventa Inf
 i=c(0:n)
 res=sum(gamma((a/2)+(i/2))/(factorial(i)) * z^i)
 return(res)
}

#Funzione di densita' di MHN normalizzata
mhn_density=function(a,b,c){
 function(x){
  cost=(2*(b^(a/2)))*(1/fw_approx(a,b,c,170,c/sqrt(b)))
  f=cost * (x^(a-1)) * exp(c*x - b*x^2)
  return(f)
 }
}

#Derivata prima e seconda del logaritmo dell'area di accettazione
#prima
der1_logA=function(a,b,c){
 function(m){
  aa=((a*b*abs(c))/(2*b*m + abs(c))^2)
  bb=(digamma(a*(b*m + abs(c))/(2*b*m + abs(c)))-log(b*m^2 + m*abs(c)))
  cc=2*b/(2*b*m + abs(c)) - b/(b*m + abs(c))
  res=aa*bb + cc
  return(res)
 }
}

#seconda
der2_logA=function(a,b,c){
 function(m){
  aa=((4*a*abs(c)*b^2)/(2*b*m +abs(c))^3)
  bb=(digamma(a*(b*m + abs(c))/(2*b*m + abs(c)))-log(b*m^2 + m*abs(c)))
```

```
   cc=((a*b*abs(c))/(2*b*m + abs(c))^2)
   dd=((-a*b*abs(c))/(2*b*m + abs(c))^2) * trigamma(a*(b*m + abs(c))/(2*b*m
       + abs(c)))
   ee=(-2*b*m+abs(c))/(m*abs(c)+b*m^2)
   ff=(-1/(m + abs(c)/2*b)^2) + (-1/(m + abs(c)/b)^2)
   res=aa*bb + cc*(dd+ee) + ff
   return(res)
  }
}

#calcolo x_flex
library(polynom)

calc_x_flex=function(a,b,c){
  aa=4*b^2
  bb=4*b*abs(c)
  cc=b*(2-4*a) + c^2
  dd=(2-2*a)*abs(c)
  ee=-3*a +2 +a^2
  polyn=polynomial(c(dd,cc,bb,aa))
  roots=solve(a=polyn,b=ee)
  real_roots=Re(roots)[abs(Im(roots)) < .Machine$double.eps]
  #print(real_roots)
  x_flex=max(real_roots)
  return(x_flex)
}

#calcolo m_init
calc_m_init=function(a,b,c){
  f=mhn.norm(a,b,c)
  x_flex=calc_x_flex(a,b,c)
  x_mode=(c + sqrt(8*b*(c-1) +c^2))/(4*b)
  if(a<1.1){
    m_init=(a^2)/(1+a)
  }else{
    sigma=f(2*x_mode-x_flex)/(f(2*x_mode-x_flex)-f(x_flex))
    m_init=(3*sigma/2)*x_mode + (1 - 3*sigma/2)*x_flex
  }
```

```
  return(m_init)
}

#calcolo m_opt
calc_m_opt=function(tol,maxiters,a,b,c){
  m=calc_m_init(a,b,c)
  l1=der1_logA(a,b,c)
  l2=der2_logA(a,b,c)
  tolerance=10
  iters_left=maxiters
  while (maxiters!=0 & tolerance>tol) {
    m_next=m-l1(m)/l2(m)
    tolerance=abs(m_next-m)
    m=m_next
    iters_left=iters_left-1
  }
  return(m)
}

#Funzione principale
rMHN_negGamma=function(N,alpha,beta,gamma){

  if(gamma>0){stop("Error: gamma must be less than or equal to 0.")}

  sample=c()
  rejected=c()

  m=calc_m_opt(0.1,100,alpha,beta,gamma)

  shape=(alpha*(beta*m + abs(gamma)))/(2*beta*m + abs(gamma))
  rate=m*(beta*m + abs(gamma))

  samples_left=N-length(sample)
  while (samples_left!=0) {
    t=rgamma(samples_left,shape=shape,rate=rate)
    x=m*(t^((beta*m + abs(gamma))/(2*beta*m + abs(gamma))))
    u=runif(samples_left)
```

```
   condition=u<exp(m*(beta*m + abs(gamma))*((x/m)^((2*beta*m + abs(gamma))/(
       beta*m + abs(gamma)))) -beta*x^2 - abs(gamma)*x)

   sample=append(sample,x[condition])
   rejected=append(rejected,x[!condition])

   samples_left=N-length(sample)
  }

  return(list(sample=sample,rejected=rejected))

}

#Funzione strumentale
proposal_density=function(a,b,c,m){
  function(x){
   k=(2*b^(a/2))/fw_approx(a,b,c,170,-abs(c)/sqrt(b))
   r=k*(x^(a-1))*exp(-m*(b*m + abs(c)) * (x/m)^((2*b*m + abs(c))/(b*m + abs(
       c))))
   return(r)
  }
}

#Rappresentazione grafica delle due funzioni
rMHN_plot = function(a, b, c, m, xlim_vals, ylim_vals) {
  pd=proposal_density(a,b,c,m)
  td=mhn_density(a,b,c)
  p = ggplot() +
   xlim(xlim_vals[1], xlim_vals[2]) +
   ylim(ylim_vals[1], ylim_vals[2]) +
   stat_function(aes(fill = "Funzione di densit strumentale"), fun = pd,
       geom = 'area')+
   stat_function(aes(fill = "Funzione di densit obiettivo"), fun = td, geom
       = 'area') +
   stat_function(fun = pd, geom = "line", color = "black", size = 0.5) +
   stat_function(fun = td, geom = "line", color = "black", size = 0.5) +
   geom_point(aes(x = m, y = td(m)), color = "black", size = 3) +
   labs(x = 'x', y = 'p(x)') + theme_bw() +
```

```
   scale_fill_manual(values = c("Funzione di densit strumentale" = "
       firebrick1",
                        "Funzione di densit obiettivo" = "springgreen2"),
                name = "Function Type") +
   theme(legend.title = element_blank(),
        legend.position = c(0.95, 0.95),
        legend.justification = c("right", "top"),
        legend.background = element_rect(fill = alpha('white', 0.5)),
        legend.text = element_text(size = 14),
        axis.title = element_text(size = 14))
  return(p)
}


####################
#### CAPITOLO 4 ####
####################

#Istogrammi con funzione di densita' (esempio riportato)
funz=mhn.nonorm(4,3,-2)
sam=rou_sampler(1000000,funz)
#sam=rMHN_negGamma(1000000,4,3,-2)
#sam=ars_alg(1000000,funz,c(0,Inf),0.5)

x_vals = seq(0, 1.75, length.out = 1000000)
dens = data.frame(
  x = x_vals,
  y = mhn_density(4,3,-2)(x_vals),
  col="Funzione di densit di MHN(4,3,-2)")

ggplot(as.data.frame(sam$sample), aes(x = sam$sample)) +
  geom_histogram(aes(x=sam$sample,y=..density..),bins=100, col='black',fill='
      white')+
  geom_line(aes(x = dens$x, y = dens$y, linetype = dens$col), color = '#
      FF0066', size = 1.5) +
  coord_cartesian(xlim = c(0,1.75), ylim = c(0, 1.75))+theme_bw()+
  labs(x = 'x', y = 'p(x)', linetype = "")+
```

```
  scale_linetype_discrete(name = "", labels = c("Funzione␣di␣densit␣di␣MHN
      (4,3,-2)" = "Funzione␣di␣densit␣di␣MHN(4,3,-2)"))+
  theme(legend.position = c(0.95, 0.95),
       legend.justification = c("right", "top"),
       legend.background = element_rect(fill = alpha('white', 0.5)),
       legend.text = element_text(size = 14),
       legend.title = element_text(size = 14),
       axis.title = element_text(size = 14))

#P-values del test Kolmogorov-Smirnov (esempio riportato)
set.seed(456)
#Funzione per il calcolo dei p-value in base al metodo utilizzato
ks_test_pvalue = function(shape, rate, method = c("ars", "rou", "rs")) {

  funz = mhn.nonorm(shape * 2, rate, 0)

  sample_data = switch(method,
                  ars = (ars_alg(10000, funz, bound = c(0, Inf),0.5)$sample)
                      ^2,
                  rou = (rou_sampler(10000, funz)$sample)^2,
                  rs = (rMHN_negGamma(10000, shape * 2, rate, 0)$sample)^2)

  ks_test = ks.test(sample_data, "pgamma", shape, rate)

  return(ks_test$p.value)
}

pvalues_ks = matrix(NA, ncol = 3, nrow = 500)
pvalues_ks[, 1] = replicate(500, ks_test_pvalue(1.5, 2, method = "ars"))
pvalues_ks[, 2] = replicate(500, ks_test_pvalue(1.5, 2, method = "rou"))
pvalues_ks[, 3] = replicate(500, ks_test_pvalue(1.5, 2, method = "rs"))
pvalues_ks = as.data.frame(pvalues_ks)
colnames(pvalues_ks) = c("ks_pv_ARS", "ks_pv_ROU", "ks_pv_RS")

mean(pvalues_ks[, 1] > 0.1)
mean(pvalues_ks[, 2] > 0.1)
mean(pvalues_ks[, 3] > 0.1)
```

```
#Percentuale campioni rifiutati (esempio riportato)
sample_sizes = c(100, 500, 1000, 2500, 5000, 10000, 25000)

p_rejected = sapply(sample_sizes, function(size) {
 # Perform the sampling and rejection counting 100 times
 rejections = replicate(100, {
  r_ars=ars_alg(size, mhn.nonorm(4, 8, -2.5), c(0, Inf), 0.3)
  r_rs=rMHN_negGamma(size, 4, 8, -2.5)
  r_rou=rou_sampler(size, mhn.nonorm(4, 8, -2.5))
  p_rej = c(length(r_ars$rejected)/(length(r_ars$sample)+length(r_ars$
      rejected))*100,
          length(r_rs$rejected)/(length(r_rs$sample)+length(r_rs$rejected))
              *100,
          length(r_rou$rejected)/(length(r_rou$sample)+length(r_rou$
              rejected))*100)
  return(p_rej)
 })

 median_p_rej = apply(rejections, 1, median)

 return(median_p_rej)
})

results = data.frame(
 Size = rep(sample_sizes, each = 3),
 Algoritmo = rep(c("ARS", "RS", "ROU"), times = length(sample_sizes)),
 n_rejected = as.vector(rbind(p_rejected[1,], p_rejected[2,], p_rejected
     [3,]))
)

ggplot(data = results, aes(x = Size, y = p_rejected, color = Algoritmo)) +
 geom_line(size=1.5) +
 geom_point(size=2.5) +
 scale_y_continuous(name = "Mediana␣della␣percentuale␣di␣campioni␣rifiutati
     ") +
 scale_x_continuous(name = "Numerosit␣del␣campione") +
 theme_bw()+
 theme(legend.text = element_text(size = 14),
```

```
      legend.title = element_text(size = 14),
      axis.title = element_text(size = 14))

#Percentuali campioni accettati (esempio riportato)
sample_sizes = c(100, 250, 500, 1000, 2500, 5000, 10000, 25000)
prob_accept = sapply(sample_sizes, function(size) {
  p_acc = replicate(100, {
    r_ars=ars_alg(size, mhn.nonorm(4, 8, -2.5), c(0, Inf), 0.3)
    r_rs=rMHN_negGamma(size, 4, 8, -2.5)
    r_rou=rou_sampler(size, mhn.nonorm(4, 8, -2.5))
    p_acc = c(length(r_ars$sample)/(length(r_ars$sample)+length(r_ars$
        rejected)),
             length(r_rs$sample)/(length(r_rs$sample)+length(r_rs$rejected)),
             length(r_rou$sample)/(length(r_rou$sample)+length(r_rou$rejected)
                 ))
    return(p_acc)
  })

  median_p_acc = apply(p_acc, 1, median)

  return(median_p_acc)
})

res_p_a = data.frame(
  Size = rep(sample_sizes, each = 3),
  Algoritmo = rep(c("ARS", "RS", "ROU"), times = length(sample_sizes)),
  Prob_accept = round(as.vector(rbind(prob_accept[1,], prob_accept[2,], prob
      _accept[3,])),3)
)

#Tempi di esecuzione (esempio riportato)
library(microbenchmark)

funz=mhn.nonorm(4,8,-2.5)
sample_sizes = c(10, 50, 100, 200, 500, 1000, 1500, 2000)
#sample_sizes = c(5000, 10000, 25000, 50000, 100000)
medianExTime_algs = function(size) {
  bm = microbenchmark(
```

```
   ars = ars_alg(size, funz, c(0, Inf), 0.3),
   rs = rMHN_negGamma(size, 4, 8, -2.5),
   rou = rou_sampler(size, funz),
   times = 100
  )
  bm_summary = summary(bm, unit='us')
  #unit impostata su secondi con campioni grandi
  #bm_summary = summary(bm, unit='s')
  median_times = bm_summary$median
  return(median_times)
}

medians = sapply(sample_sizes, medianExTime_algs)
results = data.frame(
  Size = rep(sample_sizes, each = 3),
  Algoritmo = rep(c("ARS", "RS", "ROU"), times = length(sample_sizes)),
  MedianTime = as.vector(medians)
)

ggplot(data = results, aes(x = Size, y = MedianTime, color = Algoritmo)) +
  geom_line(size = 1.5) +
  geom_point(size = 2.5) +
  scale_y_continuous(name = "Tempo␣mediano␣di␣esecuzione␣(microsecondi)") +
  scale_x_continuous(name = "Numerosit␣del␣campione") +
  theme_bw()
```

Bibliografia

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. New York, NY, USA: Springer-Verlag.

Gilks, W. R. & Wild, P. (1992). Adaptive Rejection Sampling for Gibbs Sampling. *Journal of the Royal Statistical Society Series C: Applied Statistics* **41**, 337–348.

James, J. (2024). Automated generation of initial points for adaptive rejection sampling of log-concave distributions. *Statistics and Computing* **34**.

Jingchao Sun, M. K. & Pal, S. (2023). The modified-half-normal distribution: Properties and an efficient sampling scheme. *Communications in Statistics - Theory and Methods* **52**, 1591–1613.

Kinderman, A. J. & Monahan, J. F. (1977). Computer generation of random variables using the ratio of uniform deviates. *ACM Trans. Math. Softw.* **3**, 257–260.

L'Ecuyer, P. (1994). Uniform random number generation. *Annals of Operations Research* **53**, 77–120.

Martino, L., Luengo, D. & Mguez, J. (2018). *Independent Random Sampling Methods*. Springer Publishing Company, Incorporated, 1st ed.

Robert, C. & Casella, G. (2004). *Monte Carlo statistical methods*. Springer Verlag.