Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in
Scienze Statistiche

Approximate Bayesian computation: sviluppo e valutazione di un nuovo algoritmo

Relatore: Prof. Guido Masarotto
Dipartimento di Scienze Statistiche

Laureando: Matteo Benetti
Matricola n. 2058520

Anno Accademico 2023/2024

Indice

Elenco delle tabelle

Elenco delle figure

Introduzione

Lo sviluppo e i progressi negli algoritmi di campionamento da distribuzioni multivariate hanno permesso di trovare soluzioni a problemi inferenziali che altrimenti non sarebbero mai stati presi in considerazione nelle scienze applicate. Tali algoritmi trovano grande impiego nell'ambito della statistica Bayesiana, il cui obbiettivo principale è quello di trovare la distribuzione a posteriori dei parametri $\pi(\theta|y_{obs})$.

Questo elaborato si struttura in tre capitoli, con la finalità di introdurre e spiegare il funzionamento di un nuovo algoritmo per la simulazione nell'ambito della statistica Bayesiana.

Nel primo capitolo, dopo aver fatto una panoramica generale sull'inferenza statistica Bayesiana, vengono presentate le caratteristiche di alcuni metodi classici per quanto riguarda la simulazione: l'algoritmo accetto-rifiuto, il campionamento per importanza e i metodi di Markov chain Monte Carlo (MCMC). Seppur utilizzati in molte circostanze, questi metodi hanno il limite di non poter essere impiegati quando non si è in grado di valutare la funzione di verosimiglianza perché difficile da trattare analiticamente o numericamente, oppure perché addirittura non disponibile. Gli algoritmi di *approximate Bayesian computation*, che fanno parte degli algoritmi senza verosimiglianza, sono l'oggetto principale di discussione all'interno di questo lavoro; questi metodi hanno la finalità di rendere possibile l'inferenza via si-

mulazione quando la funzione di verosimiglianza diventa intrattabile, proprio come accade in un numero sempre crescente di problemi reali.

Nel secondo capitolo si vuole portare l'attenzione sugli algoritmi di *approximate Bayesian computation* facendo un particolare riferimento al funzionamento dell'algoritmo di Lenormand. Questo viene considerato lo standard di riferimento in termini di efficienza e consistenza delle stime per il nuovo algoritmo presentato per la prima volta in questo lavoro.

Il terzo capitolo presenta i risultati di tre analisi condotte: la prima è relativa ad un esempio giocattolo, in cui si conosce il vero modello generatore dei dati ed il parametro su cui viene condotta l'inferenza è scalare; la seconda analisi è invece eseguita su un insieme di dati reali che riguardano un'indagine effettuata da una banca tedesca sul rischio di insolvenza dei clienti. La performance dell'algoritmo di *approximate Bayesian computation* proposto viene confrontata con quella dell'algoritmo di Lenormand e comunque per entrambi viene valutata la correttezza delle stime dei parametri del modello di regressione rispetto a quelli del modello logistico di riferimento, che in questo caso viene considerato come il vero modello generatore dei dati. Per questa situazione lo standard di riferimento è l'algoritmo JAGS, un tipo di MCMC. La terza analisi invece è relativa a un processo Markoviano a tempo continuo in cui non è possibile calcolare le stime mediante i tradizionali metodi di massima verosimiglianza o MCMC.

Al termine del lavoro si riportano le conclusioni relative ai vantaggi e agli aspetti critici del nuovo algoritmo di *approximate Bayesian computation* proposto.

Capitolo 1

Statistica Bayesiana e algoritmi di simulazione

In questo capitolo si presentano sommariamente alcuni dei concetti chiave della statistica Bayesiana e di alcuni metodi computazionali associati, facendo riferimento in particolare all'algoritmo di accetto-rifiuto, al campionamento per importanza, agli algoritmi di Markov chain Monte Carlo e ai metodi senza verosimiglianza.

Si introducono i concetti fondamentali della statistica Bayesiana, delineando i principi chiave e illustrando il processo di aggiornamento attraverso l'uso delle distribuzioni a priori e a posteriori (Albert 2009, Sisson, Fan e Beaumont 2018, Wilkinson 2019, Salvan, Sartori e Pace 2022).

Si espone il funzionamento dell'algoritmo accetto–rifiuto, una tecnica basilare ma potente per generare campioni da generiche distribuzioni nell'ambito della statistica computazionale (Robert e Casella 2010, Sartori e Guolo 2022, Sisson, Fan e Beaumont 2018, Albert 2009).

Successivamente si presenta il funzionamento del campionamento per importanza, un metodo utile sia per il calcolo approssimato del valore di integra-

li via Monte Carlo sia per simulare estrazioni da una distribuzione obbiettivo (Sartori e Guolo 2022, Sisson, Fan e Beaumont 2018, Albert 2009).

Si presentano poi i concetti alla base degli algoritmi di Markov chain Monte Carlo spiegando il funzionamento dell'algoritmo Metropolis-Hastings che rappresenta un semplice metodo per introdursi a questa classe di metodi nella pratica (Robert e Casella 2010, Wilkinson 2019, Liseo 2010 Sartori e Guolo 2022).

Si conclude il capitolo riportando i principi fondamentali dei metodi senza verosimiglianza concentrando l'attenzione sugli algoritmi di *approximate Bayesian computation* (Sisson, Fan e Beaumont 2018, Drovandi e Pettitt n.d., Turner e Van Zandt 2012, Drovandi e Pettitt 2011).

1.1 Introduzione alla Statistica Bayesiana

Nell'ambito dell'inferenza statistica Bayesiana la probabilità è intesa come incertezza rispetto a quantità osservabili (il campione y_{obs} o osservazioni future y) e quantità non osservabili (i parametri $\theta \in \Theta \subseteq \mathbb{R}^p$); ogni quantità considerata nell'ambito dell'inferenza Bayesiana è trattata come una variabile casuale. Quindi, oltre ad assumere un determinato modello statistico per y_{obs}, si assume che $\theta \in \Theta$ sia realizzazione di una variabile casuale; la sua densità, $\pi(\theta)$, prende il nome di distribuzione a priori perché contiene le informazioni preliminari che si hanno a disposizione per θ. La definizione del modello statistico $\mathcal{M}$ permette di calcolare la funzione di verosimiglianza $p(y_{obs}|\theta)$ necessaria per aggiornare l'informazione contenuta nella distribuzione a priori a partire da quella contenuta nei dati.

Attraverso il teorema di Bayes, da qui il nome "Bayesiana", è possibile calcolare la distribuzione a posteriori nel seguente modo:

$$\pi(\theta|y_{obs}) = \frac{\pi(\theta)p(y_{obs}|\theta)}{\int_\Theta \pi(\theta)p(y_{obs}|\theta)d\theta}.$$

A denominatore compare una quantità, la costante di normalizzazione, che non dipende da θ, quindi $\pi(\theta|y_{obs}) \propto \pi(\theta)p(y_{obs}|\theta)$.

Ottenere la distribuzione a posteriori $\pi(\theta|y_{obs})$ è il principale obbiettivo dell'inferenza Bayesiana: a partire da essa, infatti, si riescono ad ottenere tutte le informazioni richieste per l'analisi del modello, tra cui la sua validazione e l'inferenza previsiva.

La definizione della distribuzione a priori è un aspetto delicato di questo approccio inferenziale perché, a differenza di ciò che avviene per il modello statistico, non esiste un metodo per valutarne la sua bontà.

La specificazione della distribuzione a priori può derivare da diversi aspetti:

- studi precedenti: se in precedenza sono stati osservati dati simili, questi possono essere usati per estrarre qualche informazione utile all'analisi in questione;

- convenienza matematica: alcune distribuzioni a priori possono portare semplificazioni nei calcoli della distribuzione a posteriori. In tal caso si parla di distribuzioni a priori coniugate;

- a priori soggettive: la definizione della distribuzione a priori dipende da un'opinione soggettiva da parte di qualche esperto del fenomeno di interesse;

- a priori non informative: queste distribuzioni a priori non contengono nessuna informazione preliminare sul parametro; vengono utilizzate,

per esempio, quando non si è in grado di specificare la distribuzione a priori in un altro modo.

Nel caso in cui non si scelga di utilizzare distribuzioni a priori coniugate, solitamente la complessità del modello e della distribuzione a priori fanno sì che la distribuzione a posteriori non sia disponibile in forma chiusa, quindi si rendono necessari metodi numerici per svolgere le varie procedure inferenziali.

Alcuni algoritmi popolari per simulare dalla distribuzione a posteriori sono per esempio il campionamento per importanza, Markov chain Monte Carlo (MCMC) e sequential Monte Carlo (SMC). Nel caso dell'algoritmo Metropolis–Hastings, un esempio di MCMC, l'utilizzo di metodi numerici si presenta nel calcolo della probabilità che la catena di Markov accetti la proposta di muoversi da un valore corrente θ ad un nuovo valore θ'. Allo stesso modo, nel caso dell'algoritmo SMC, i metodi numerici intervengono per il calcolo del peso da attribuire al parametro generato. In entrambi i casi comunque è richiesto di calcolare il valore della funzione di verosimiglianza.

Tuttavia, in un numero sempre maggiore di problemi reali, valutare la funzione di verosimiglianza $p(y_{obs}|\theta)$ è proibitivo a livello computazionale, se non impossibile. In questi scenari quindi il modello selezionato non è trattabile con le classiche procedure inferenziali; la necessità di valutare ripetutamente la distribuzione a posteriori per estrarre un campione da questa distribuzione rende impraticabili l'utilizzo e l'implementazione delle tecniche standard di simulazione Bayesiana.

Per risolvere questo problema si potrebbe ricorrere all'utilizzo di un modello differente, più adatto ai calcoli statistici, che presenta però uno svantaggio dovuto al fatto che il modello scelto può non essere così realistico per il fenomeno di interesse e quindi non permette di condurre le adeguate analisi. Un'alternativa più attraente può essere quella di considerare un'approssima-

zione del modello in grado di garantirne la veridicità a spese della presenza di un errore di approssimazione.

Nonostante siano disponibili vari metodi per approssimare la distribuzione a posteriori, i metodi Bayesiani senza verosimiglianza sono risultati molto efficaci e intuitivi per condurre un'analisi Bayesiana approssimata; tra questi vi è il metodo di *approximate Bayesian computation*.

1.2 Algoritmi di simulazione

Nell'ambito della statistica computazionale si è spesso interessati a simulare osservazioni a partire da distribuzioni arbitrarie che possono avere forme più o meno familiari. L'algoritmo accetto-rifiuto, il metodo del campionamento per importanza e gli algoritmi Markov chain Monte Carlo sono dei metodi generali che possono trovare utilità in qualsiasi ambito e non solo nella statistica Bayesiana.

1.2.1 Algoritmo accetto-rifiuto

Nel caso in cui la distribuzione a posteriori di un parametro sia nota, risulta semplice simulare direttamente da essa. Tuttavia, in molte situazioni, la distribuzione a posteriori non ha una forma familiare e bisogna avere a disposizione strumenti alternativi per poter generare campioni da essa.

L'algoritmo accetto-rifiuto (*rejection sampling algorithm*) è un metodo generale per poter simulare numeri casuali da una densità d'interesse $f(\theta)$, anche se questa è nota a meno di costanti moltiplicative. L'algoritmo presuppone di avere a disposizione una densità $g(\theta)$, tale che:

- permetta di simulare osservazioni facilmente;

- sia simile alla densità d'interesse in termini di supporto; in particolare il supporto di g deve contenere quello di f;

- si individua il $\sup_\theta \dfrac{f(\theta)}{g(\theta)} = K < \infty$ tale che $\forall\theta$ e $\forall K$, $f(\theta) \leq Kg(\theta)$.

Una volta trovata la densità g, da cui si è in grado di simulare osservazioni, l'algoritmo accetto-rifiuto consiste nei seguenti passaggi:

1. si genera indipendentemente un valore di θ da g e un valore $U \sim U(0,1)$;

2. se $UKg(\theta) \leq f(\theta)$ allora si accetta il valore di θ come realizzazione di una simulazione da f, altrimenti lo si rifiuta. In altre parole il valore di θ proposto viene accettato con probabilità $\dfrac{f(\theta)}{Kg(\theta)}$;

3. si ripetono gli step 1 e 2 fino a che non si raggiunge un numero sufficiente di valori di θ accettati.

Il metodo di campionamento tramite l'algoritmo accetto-rifiuto è uno strumento particolarmente efficace per emulare l'estrazione da diverse distribuzioni. Infatti, gli approcci standard per simulare da distribuzioni di probabilità comuni, come la normale, la gamma e la beta, spesso utilizzano metodi basati proprio su questo algoritmo.

I suoi aspetti critici sono la scelta di un'opportuna densità g da cui simulare e la scelta della costante b. L'algoritmo accetto-rifiuto è tanto più efficiente quanto più alta è la proporzione di valori di θ accettati come estrazioni della distribuzione d'interesse f.

1.2.2 Campionamento per importanza

Questo metodo nasce con l'obbiettivo di calcolare integrali via Monte Carlo.

In generale la stima Monte Carlo classica può risultare pessima, in particolare nel caso in cui si ha a che fare con eventi rari. L'idea è che, invece di generare da una densità f, si generi da una densità g che simuli valori nella parte più importante del supporto, così da non sprecare valori di θ che non verrebbero considerati nel caso della classica stima Monte Carlo; da qui il nome campionamento per importanza (*importance sampling*) e distribuzione d'importanza per g.

Di seguito si illustra il funzionamento dell'algoritmo considerando che l'obbiettivo è generare dalla distribuzione a posteriori, quindi $f(\theta) = p(\theta|y_{obs})$.

In molti casi la costante di normalizzazione della densità a posteriori è ignota, quindi il valore atteso a posteriori di una certa funzione $m(\theta)$ è dato dal seguente rapporto di integrali:

$$\psi = E[m(\theta)|y_{obs}] = \frac{\int_\Theta m(\theta)\pi(\theta)p(y_{obs}|\theta)d\theta}{\int_\Theta \pi(\theta)p(y_{obs}|\theta)d\theta}, \tag{1.1}$$

con $\pi(\theta)$ densità a priori e $p(y_{obs}|\theta)$ funzione di verosimiglianza. Se si riuscisse a generare un campione casuale dalla distribuzione a posteriori $\pi(\theta|y_{obs})$, si potrebbe calcolare in modo approssimato il valore dell'integrale via Monte Carlo. In questo caso si suppone che non si sia in grado di generare dalla distribuzione a priori, ma che si possa costruire una funzione di densità h da cui è possibile simulare. La media a posteriori calcolata in (1.1) diventa:

$$\begin{aligned}\psi = E[m(\theta)|y_{obs}] &= \frac{\int_\Theta m(\theta)\dfrac{\pi(\theta)p(y_{obs}|\theta)}{h(\theta)}h(\theta)d\theta}{\int_\Theta \dfrac{\pi(\theta)p(y_{obs}|\theta)}{h(\theta)}h(\theta)d\theta} \\ &= \frac{\int_\Theta m(\theta)w(\theta)h(\theta)d\theta}{\int_\Theta w(\theta)h(\theta)d\theta}.\end{aligned}$$

dove $w(\theta) = \pi(\theta)p(y_{obs}|\theta)/h(\theta)$ è la funzione peso (*importance weigth function*).

Siano quindi $\theta_1, ..., \theta_n \sim h$, allora

$$\hat{\psi} = \frac{\sum_{i=1}^{n} m(\theta_i)w(\theta_i)}{\sum_{i=1}^{n} w(\theta_i)}.$$

Come nel caso dell'algoritmo accetto-rifiuto, la difficoltà principale è trovare una densità di campionamento h adatta. Innanzitutto deve essere una densità da cui è possibile estrarre simulazioni. Una scelta sbagliata può portare alla restituzione di risultati errati; per questo motivo, la scelta deve essere fatta in modo che le code siano più pesanti della distribuzione a posteriori per assicurarsi che la funzione peso sia limitata e che non sia la causa della divergenza della varianza dello stimatore.

L'algoritmo di campionamento per importanza è utile perché, oltre al calcolo di stime Monte Carlo più accurate, può essere usato anche per l'estrazione di campioni distribuiti secondo la densità a posteriori.

Come nel caso dell'algoritmo accetto-rifiuto, si simulano n estrazioni $\theta_1, ..., \theta_n$ a partire dalla densità delle proposte h ma si calcolano anche i pesi associati $\{w_i = \pi(\theta_i|y_{obs})/h(\theta_i)\}$. Si procede quindi con il calcolo dei pesi normalizzati in modo da intenderli in termini di probabilità

$$\bar{w}_j = \frac{w(\theta_i)}{\sum_{i=1}^{n} w(\theta_i)}, \quad \text{con} \quad j = 1, ..., n.$$

È quindi possibile estrarre un nuovo campione $\theta_1^*, ..., \theta_n^*$ a partire dalla distribuzione discreta descritta dalle n coppie $(\theta_j, \bar{w}_j)$; questo campione sarà approssimativamente distribuito secondo la densità a posteriori.

Questo metodo è chiamato *sampling importance resampling* (SIR).

1.2.3 Markov chain Monte Carlo

Il campionamento mediante l'algoritmo accetto-rifiuto è un metodo generale per simulare da una distribuzione a posteriori arbitraria, ma può essere difficile da impostare poiché richiede la scelta di un'adeguata densità per le proposte g. Allo stesso modo, anche il campionamento per importanza e gli algoritmi SIR, che sono di uso comune, possono essere problematici se si sceglie di utilizzarli per problemi ad elevata dimensionalità.

Gli algoritmi di Markov chain Monte Carlo (MCMC) sono utilizzati per campionare da distribuzioni di probabilità complesse che spesso si incontrano nell'ambito della statistica Bayesiana. Questi algoritmi sono particolarmente utili quando è difficile campionare direttamente dalla distribuzione a posteriori o quando si ha una conoscenza solo parziale della sua forma funzionale. Questi metodi permettono la costruzione di successioni di valori pseudocasuali che approssimativamente si possono considerare come realizzazioni indipendenti dalla densità d'interesse $f(\theta)$.

Il nome "Markov chain Monte Carlo" riflette due aspetti fondamentali dell'algoritmo. In primo luogo, la descrizione e la comprensione dei metodi MCMC necessitano di alcune conoscenze sulle catene di Markov. In secondo luogo, se si fosse in grado di generare una simulazione $(\theta^{(1)}, \theta^{(2)}, ...)$ di una catena di Markov, la cui distribuzione limite è proprio la distribuzione obbiettivo, per il teorema di ergodicità è possibile approssimare via Monte Carlo il valore di integrali in modo consistente.

L'idea è quella di generare una catena di Markov che presenti come distribuzione stazionaria la distribuzione di probabilità di interesse. Per ottenere un campione da essa è quindi sufficiente simulare un numero adeguato di stati successivi della catena.

Il problema dei metodi MCMC è quello di riuscire a costruire una cate-

na di Markov $(\theta_1, \theta_2, ..., \theta_n, ...)$ la cui distribuzione limite sia proprio $f(\theta)$ e che allo stesso tempo la converga verso tale distribuzione il più velocemente possibile.

Si presenta ora l'algoritmo Metropolis-Hastings, un metodo che può essere visto come il più generale degli algoritmi di Markov chain Monte Carlo e che, grazie alla sua semplicità, permette di fare chiarezza sul funzionamento di questi tipi di metodi.

L'algoritmo Metropolis-Hastings

L'idea che si utilizza in questo algoritmo per generare una catena con distribuzione limite uguale alla distribuzione di interesse $f(\theta)$ è quella di creare una successione di valori candidati generati da una densità di proposta $q(\theta^{(t+1)}|\theta^{(t)})$ che rappresenta la probabilità che l'algoritmo proponga uno spostamento della catena dal valore attuale $\theta^{(t)}$ ad un nuovo valore $\theta^{(t+1)}$. I requisiti per tale funzione è che sia semplice in modo da consentire la possibilità di generare valori da essa e che sia sufficientemente sparsa da permettere l'esplorazione dell'intero supporto di $f(\theta)$.

I passi dell'algoritmo sono riassunti nei seguenti punti:

1. Al tempo 0, la catena si trova in $\theta^{(0)}$;

2. Al tempo t si genera un valore candidato $\theta^* \sim q(\cdot|\theta^{(t-1)})$;

3. Si calcola il valore di $\alpha = \alpha(\theta^*, \theta^{(t-1)})$, dove

$$\alpha(\theta^*, \theta^{(t-1)}) = \min\left(\frac{q\left(x_{t-1} \mid \theta^*\right) f(\theta^*)}{q\left(\theta^* \mid x_{t-1}\right) f\left(x_{t-1}\right)}, 1\right);$$

4. Si definisce

$$\theta^{(t+1)} = \begin{cases} \theta^* & \text{con probabilità } \alpha \\ \theta^{(t-1)} & \text{con probabilità } 1-\alpha \end{cases}$$

5. Si pone $t = t + 1$ e si torna al punto 2.

Sotto la condizione che la catena dei valori candidati con densità $q(\theta^{(t+1)}|\theta^{(t)})$ sia ergodica, allora la successione di valori accettati $(\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n)}, ...)$ compone un campione pseudo-casuale dalla distribuzione obbiettivo f.

1.3 Metodi senza verosimiglianza

Per capire l'intuizione alla base dei metodi senza verosimiglianza è opportuno fare riferimento, in prima battuta, all'algoritmo accetto-rifiuto.

Se si specifica come densità obbiettivo la densità a posteriori e come densità delle proposte la densità a priori, si ottiene che la probabilità di accettare è proporzionale alla funzione di verosimiglianza. Considerando rispettivamente $f(\theta) = \pi(\theta|y_{obs})$ e $g(\theta) = \pi(\theta)$ è noto che la probabilità di accettare un valore di θ proposto è $\dfrac{f(\theta)}{Kg(\theta)} \propto p(y_{obs}|\theta)$.

La valutazione diretta di questa probabilità può non essere disponibile se la funzione di verosimiglianza non è trattabile, ma è possibile determinare stocasticamente se accettare o meno un valore proposto senza utilizzare procedure numeriche per calcolare la probabilità di accettazione.

Ciò può essere compreso considerando che la probabilità di accettazione è direttamente proporzionale alla probabilità di generare i dati osservati, y_{obs}, secondo il modello $p(y|\theta)$ con θ fissato. In altre parole, se la funzione di verosimiglianza $p(y|\theta)$ fosse adeguatamente normalizzata, essa potrebbe essere

interpretata come una probabilità di generare i dati. Quindi la probabilità di ottenere esattamente il campione osservato è data da $p(y_{obs}|\theta)$.

Utilizzando questa idea, è possibile semplificare il processo di accettazione o rifiuto di un'estrazione dalla distribuzione, basandosi su una variabile aleatoria di Bernoulli che indica se si genera $y = y_{obs}$ o meno, anziché valutare direttamente la probabilità $p(y_{obs}|\theta)$.

Questa intuizione, tenendo conto che per semplicità si ipotizza che i dati y siano discreti (assunzione che sarà rilassata in seguito), permette di riscrivere l'algoritmo accetto-rifiuto senza verosimiglianza come illustrato di seguito:

1. si genera un valore di $\theta \sim \pi(\theta)$ campionando dalla densità a priori;

2. si genera un campione $y \sim p(y|\theta)$ a partire dal modello;

3. se $y = y_{obs}$ allora si accetta il valore di θ generato, altrimenti si torna al punto 1.

Si ripetono questi passi finché non si ottiene un campione $\theta_1, ..., \theta_n$ di numerosità n desiderata.

Nonostante la correttezza dell'algoritmo dal punto di vista teorico, evitare di calcolare la verosimiglianza può creare problemi. Infatti, come è probabile che avvenga in generale, se le analisi diventano più complesse, la probabilità che $y = y_{obs}$ può diventare molto piccola o esattamente nulla nel caso in cui i dati non siano discreti ma continui.

Per ovviare a questo problema si può richiedere che l'algoritmo accetti un valore di θ non se $y = y_{obs}$ in modo esatto, ma che i dati generati siano semplicemente "vicini" ai dati osservati. Si accettano i valori di θ tali per cui $\|y - y_{obs}\| \leq h$, per qualche valore $h \geq 0$ e per qualche misura di distanza $\|\cdot\|$, come per esempio la distanza Euclidea.

A seguito di queste considerazioni è possibile quindi rilassare l'ipotesi che i dati siano discreti. Di conseguenza, l'algoritmo di campionamento si modifica solo nel punto (3) in questo modo:

3. se $\|y - y_{obs}\| \leq h$ allora si accetta il valore di θ generato, altrimenti si torna al punto 1.

Come risultato si ottiene che il campione estratto $\theta_1, ..., \theta_n$ non sarà più esattamente estratto dalla distribuzione a posteriori $\pi(\theta|y_{obs})$, ma solo in modo approssimato, ad eccezione del caso in cui si decida di porre $h = 0$. La logica alla base di questo metodo è che all'aumentare di h il tasso di accettazione dell'algoritmo cresce, ma di conseguenza ci si allontana dalla vera forma della distribuzione a posteriori, con la variabilità del parametro che cresce.

Un'ulteriore complicazione può essere dettata dalla numerosità del campione y_{obs}. Infatti, anche se ci si condiziona al vero valore del parametro, le probabilità di generare un campione y vicino a y_{obs} quando la numerosità campionaria cresce sono estremamente basse, in termini di misura della distanza considerata. Per provare ad aumentare le probabilità che y e y_{obs} siano vicini si potrebbero ridefinire entrambi secondo le loro statistiche d'ordine. Tuttavia è importante notare che le probabilità rimarrebbero comunque molto basse (si veda l'esempio 3 della sezione 1.7.1 di Sisson, Fan e Beaumont 2018 per un'illustrazione).

Il valore di h dovrebbe essere perciò grande; in questo modo si accetterebbero però valori di θ in corrispondenza di campioni y distanti da y_{obs}; ne segue che la distribuzione a posteriori prodotta fornisce un'approssimazione scarsa della vera distribuzione a posteriori.

Una strada che risulta essere ragionevole per ovviare a questo problema è quella di utilizzare delle statistiche di sintesi per diminuire la dimensione del confronto $\|y - y_{obs}\|$.

Si suppone che sia disponibile una statistica del campione $s = S(y)$ tale che questa sia sufficiente, così da non perdere troppa informazione, ma con il requisito che $\dim(S(y)) \ll \dim(y)$. Il vantaggio che comporta questa scelta è che il confronto $\|y - y_{obs}\|$ viene ora sostituito dal confronto $\|s - s_{obs}\|$, dove $s_{obs} = S(y_{obs})$, che ha una dimensione molto minore rispetto a quella dell'intero campione.

Con questa nuova specifica, l'algoritmo di campionamento si modifica solo nel punto (3) in questo modo:

3. Si calcola $s = S(y)$.

   Se $\|s - s_{obs}\| \leq h$ allora si accetta il valore di θ generato, altrimenti si torna al punto 1.

È possibile mostrare (paragrafo 1.7 di Sisson, Fan e Beaumont 2018) che l'utilizzo di statistiche di sintesi al posto dell'intero campione non peggiora la qualità della stima della distribuzione a posteriori sebbene questo comporti una perdita di informazione. In generale utilizzare le statistiche di sintesi come confronto con il campione osservato si rivela essere molto utile nel caso dell'inferenza Bayesiana approssimata.

I termini relativi ai metodi senza verosimiglianza e di *approximate Bayesian computation* fanno riferimento a metodi computazionali nell'ambito della statistica Bayesiana quando la funzione di verosimiglianza è computazionalmente intrattabile o addirittura non disponibile.

L'espressione "senza verosimiglianza" tuttavia è impropria in quanto sembra sottintendere il fatto che la verosimiglianza non entri in gioco nell'analisi

in alcun modo. Questo fatto però non è vero in quanto tale funzione deve esistere essendo quella che determina il modello generatore dei dati $y \sim p(y|\theta)$, indipendentemente dal fatto che sia possibile o meno valutarla o scriverla. In questo contesto l'espressione "senza verosimiglianza" indica che le classiche analisi basate sulla verosimiglianza vengono processate senza la valutazione numerica o analitica della funzione di verosimiglianza.

Approximate Bayesian computation, comunemente abbreviata con la sigla ABC, è un esempio di metodo senza verosimiglianza. Il motivo per cui ci si trova davanti ad un'approssimazione della distribuzione a posteriori è dato dal fatto che il parametro proposto θ, viene accettato solo se $\|y - y_{obs}\| \leq h$, con $h > 0$ oppure, analogamente, se si considerano le statistiche di sintesi del campione, viene accettato solo se se $\|s - s_{obs}\| \leq h$ con $h \geq 0$ (incluso $h = 0$).

Nel caso in cui si misura la distanza di y da y_{obs} e si pone $h = 0$, l'algoritmo non rientra nell'insieme dei metodi di ABC perché non ci si trova di fronte a un'approssimazione dal momento che il metodo produce campioni provenienti esattamente dalla distribuzione a posteriori. Allo stesso modo, se si misura la distanza tra le statistiche di sintesi dei campioni e tali statistiche sono sufficienti, con $h = 0$ non ci si trova di fronte ad un metodo di ABC perché ancora una volta i campioni estratti provengono dalla distribuzione a posteriori in modo esatto e non approssimato.

1.3.1 Approssimazione della distribuzione a posteriori

Come già evidenziato nel paragrafo precedente, i metodi di *approximate Bayesian computation* producono un'approssimazione della distribuzione obbiettivo da cui si vuole simulare, che in questo lavoro corrisponde alla distribuzione a posteriori.

I passi dell'algoritmo sono sempre tre, come nel caso dell'algoritmo accetto-rifiuto (sezione 1.2.1) e dell'algoritmo di campionamento di importanza (sezione 1.2.2): (i) si genera un valore di θ a partire dalla distribuzione a priori $\pi(y\theta)$, (ii) si generano i dati simulati y a partire dal modello considerato $y \sim p(|\theta)$, (iii) si accetta θ se $\|y - y_{obs}\| \leq h$, il che è equivalente ad estrarre un campione (θ, y) dalla distribuzione congiunta che è proporzionale a:

$$\mathbb{1}(\|y - y_{obs}\| \leq h)p(y|\theta)\pi(\theta), \tag{1.2}$$

dove $\mathbb{1}$ è la funzione indicatrice tale che $\mathbb{1}(x) = 1$ se la condizione x è vera, $\mathbb{1}(x) = 0$ altrimenti.

Si nota che per $h \to 0$ la distribuzione marginale per θ, a partire da (1.2), è:

$$\begin{aligned}\lim_{h\to 0} \int \mathbb{1}(\|y - y_{obs}\| \leq h)p(y|\theta)\pi(\theta)dy &= \int \delta_{y_{obs}}(y)p(y|\theta)\pi(\theta)dy \\ &= p(y_{obs}|\theta)\pi(\theta),\end{aligned}$$

dove $\delta_{y_{obs}}$ è la funzione delta di Dirac che assume valore 1 se $y = y_{obs}$ e 0 altrimenti.

Quindi per $h = 0$ la distribuzione marginale di θ è proprio la distribuzione a posteriori $\pi(\theta|y_{obs})$.

È già stato presentato il motivo per cui la scelta di $h = 0$ non è adeguata nella pratica perché, nel caso di dati continui, la probabilità di accettazione dell'algoritmo è nulla. Le medesime conclusioni si ottengono utilizzando una statistica di sintesi per i dati $s = S(y)$ che risulta essere più sensata per ridurre la dimensione dei confronti dal momento che è raro ottenere $y = y_{obs}$ con numerosità campionarie elevate.

1.3.2 Campionamento per importanza via ABC

In questa sezione si riportano i passi di un algoritmo di campionamento per importanza via ABC. Dal momento che finora si è sempre parlato dell'utlizzo di una misura per il calcolo della distanza tra la statistica del campione simulato (s) e la statistica del campione osservato (s_{obs}) senza però fare chiarezza sul suo effettivo utilizzo, è importante fare una specificazione: le statistiche di sintesi dei dati sono per natura più variabili e avranno un impatto maggiore nel calcolo della distanza con le statistiche osservate se non vengono riscalate. Diventa quindi più opportuno l'utilizzo di una misura di distanza che tenga conto della scala delle statistiche e di eventuali correlazioni tra esse. A questo proposito una scelta adeguata per $\|\cdot\|$ è la distanza di Mahalanobis. Nel caso in cui $q = \dim(S(y)) = 1$ allora un'altra scelta possibile è quella della distanza Euclidea.

L'algoritmo di campionamento per importanza via ABC richiede che siano definite le seguenti quantità:

- la distribuzione a posteriori $\pi(\theta|y_{obs}) \propto p(y_{obs}|\theta)\pi(\theta)$ obbiettivo delle simulazioni;
- la distribuzione $g(\theta)$ da cui sia possibile campionare e tale per cui se $g(\theta) > 0$ allora $\pi(\theta|y_{obs}) > 0$;
- lo scalare $h > 0$ che regola la probabilità di accettazione di un parametro θ;
- la funzione $s = S(y)$ che calcola le statistiche di sintesi a partire dal campione y.

I passi dell'algoritmo sono i seguenti:

1. si genera $\theta_i \sim g(\theta)$ dalla distribuzione di importanza;

2. si genera un campione $y_i \sim p(y|\theta_i)$ dal modello;

3. si calcola la statistica di sintesi del campione generato: $s = S(y_i)$;

4. si calcolano i pesi $w_i = \mathbb{1}(\|s - s_{obs}\| \leq h)\dfrac{\pi(\theta_i)}{g(\theta_i)}$;

5. si continua ripetendo gli step elencati in precedenza fino a che non si ottiene un campione $(\theta_1, w_1), ..., (\theta_n, w_n)$.

Capitolo 2

Algoritmi di ABC

In questo capitolo si vuole portare l'attenzione sull'argomento principale di questo lavoro: gli algoritmi di *approximate Bayesian computation*. Dopo una panoramica generale (Turner e Van Zandt 2012 e Sisson, Fan e Beaumont 2018), si descrive il funzionamento della classe degli algoritmi sequenziali (Drovandi e Pettitt 2011) all'interno della quale si trova l'algoritmo di Lenormand (Lenormand, Jabot e Deffuant 2013). Oltre a questo algoritmo, si presenta in modo dettagliato il funzionamento del nuovo algoritmo proposto, knnABC, elencando le differenze principali con l'algoritmo di Lenormand considerato come riferimento in questo lavoro perché è uno dei più utilizzati e perché è l'unico completamente automatico in questa classe di algoritmi.

2.1 Approximate Bayesian computation

Gli algoritmi di *approximate Bayesian computation* sono strutturati come presentato in sezione 1.3.2 ma è possibile evidenziare due principali specificazioni a seconda dell'approccio che si decide di implementare. Il primo approccio prevede che si generino i parametri (spesso chiamati anche con il

nome di particelle) da una distribuzione di importanza $g(\theta)$ e che si calcolino i pesi. Nel caso in cui $g(\theta) = \pi(\theta)$ si ha l'algoritmo accetto-rifiuto. In entrambi i casi si considerano le osservazioni in corrispondenza dei campioni più vicini come provenienti dalla distribuzione a posteriori. Il secondo è un approccio sequenziale: si presuppone che non sia immediato definire la distribuzione di importanza da usare e quindi si sceglie di costruirne una sequenza $g^{(0)}(\theta), ..., g^{(T)}(\theta)$, con $g^{(0)}(\theta) = \pi(\theta)$. Iterazione dopo iterazione la distribuzione di importanza $g^{(i)}(\theta)$ fornisce un'approssimazione della distribuzione a posteriori sempre migliore perché la soglia h, che determina il livello di accettazione, diventa man mano più piccola.

A prescindere dalla scelta dell'approccio, tutti gli algoritmi di *approximate Bayesian computation* si basano su un campionamento per importanza, i cui ingredienti principali da definire sono:

- la distribuzione di importanza $g(\theta)$;
- il parametro h che definisce la soglia;
- la scelta della metrica per misurare la distanza tra le statistiche e la scelta di come stimare la matrice di varianza e covarianza di queste distanze.

Nei prossimi paragrafi si presenta il funzionamento della classe degli algoritmi sequenziali facendo un focus sull'algoritmo di Lenormand, chiamato per brevità con la sigla APCM, e su un nuovo algoritmo pensato con alcune diverse specifiche rispetto allo standard della maggior parte degli algoritmi di *approximate Bayesian computation.*

2.2 Algoritmi sequenziali

Entrambi gli algoritmi citati alla fine della sezione 2.1 appartengono all'insieme degli algoritmi sequenziali in quanto ad ogni iterazione vengono estratti, a partire da una distribuzione di importanza $g^{(i)}(\theta)$, un numero fisso di parametri simulati e poi secondo qualche criterio vengono tenuti e considerati come simulazioni della distribuzione a posteriori o scartati ed eventualmente sostituiti con altre generazioni di simulazioni. La classe degli algoritmi sequenziali quindi approssima progressivamente la distribuzione a posteriori usando sequenze di campioni $\theta^{(t)} = \{\theta_i^{(t)}\}$ con $i = 1, ..., n$, derivati dal campione al passo precedente $\theta^{(t-1)}$ assieme all'utilizzo di un insieme di livelli di tolleranza $\{\varepsilon_1, ..., \varepsilon_T\}$ che identificano il livello della soglia per la distanza tra le statistiche osservate e quelle calcolate a partire dal campione simulato. Questa tecnica porta a focalizzare l'attenzione del campionamento sulle regioni dello spazio dei parametri che presentano un'elevata verosimiglianza, e quindi le regioni caratterizzate da distanze tra le statistiche basse, riducendo così il tempo impiegato nel campionamento dell'intero spazio parametrico.

2.2.1 Algoritmo di Lenormand - APCM

Questo algoritmo segue i principi dell'ABC sequenziale, illustrati al paragrafo precedente, e definisce ad ogni iterazione i livelli di tolleranza ad ogni step (come in Drovandi e Pettitt 2011, Del Moral, Doucet e Jasra 2012, Wegmann et al. 2010). Per ogni livello di tolleranza ε_t viene generato un campione $\theta^{(t)}$ di parametri e vengono calcolati i pesi di importanza associati.

L'algoritmo APMC richiede che siano definite le seguenti quantità:

- la distribuzione a priori $\pi(\theta)$;
- il modello per i dati $y \sim p(y|\theta)$;

- la statistica di sintesi $s = S(y)$;
- la misura di distanza $d_i^{(t)} = \|s_i - s_{obs}\|$;
- un parametro $\alpha \in (0, 1)$.

Al primo passo dell'algoritmo, $t = 0$, viene estratto un campione di n particelle $\theta_i^{(0)} \sim g^{(0)}(\theta) = \pi(\theta)$ con $i = 1, ..., n$ e si simulano dei dati y a partire dal modello, per ogni parametro fissato $\theta_i^{(0)}$. Si calcolano quindi le statistiche di sintesi dei dati simulati e per ciascuna si ottengono la distanza dalla statistica del campione osservato: $d_i^{(0)} = \|s_i - s_{obs}\|$. Successivamente si calcolano i pesi che al passo $t = 0$ sono tutti pari a $w_i^{(0)} = 1 \ \forall i$.

Il funzionamento dell'algoritmo alla generica iterazione $t \in \{1, ..., T\}$, avendo a disposizione il campione $\mathcal{S}^{(t-1)} = (\theta_i^{(t-1)}, w_i^{(t-1)})$ con $i = 1, ..., n_\alpha$ di numerosità $n_\alpha = \lfloor \alpha n \rfloor$, si riassume nelle seguenti fasi:

1. si generano $n - n_\alpha$ nuovi parametri $\{\theta_j^{(t-1)}\}$ con $j = n_\alpha + 1, ..., n$ dove $\theta_j^{(t-1)} \sim \mathcal{N}(\theta_j^*, \sigma_{t-1}^2)$. Le medie θ_j^* sono scelte in modo casuale dall'insieme $(\theta_i^{(t-1)})$ con $i = 1, ..., n_\alpha$ pesato secondo $w_i^{(t-1)}$ con $i = 1, ..., n_\alpha$, mentre la varianza σ_{t-1}^2 è calcolata come la varianza campionaria dello stesso insieme. Quindi, la densità utilizzata per generare una nuova particella $\theta_i^{(t)}$ al tempo t è quella di una mistura di normali:

$$f^{(t)}(\theta_i^{(t)}) = \sum_{j=1}^{n_\alpha} \frac{w_j^{(t-1)}}{\sum_{k=1}^{n_\alpha} w_k^{(t-1)}} \sigma_{t-1}^{-1} \varphi \left(\sigma_{t-1}^{-1} \left(\theta_i^{(t)} - \theta_j^{(t-1)} \right) \right),$$

   dove $\varphi(x)$ è la funzione di densità della normale standard;

2. si calcolano i pesi $w_j^{(t-1)}$ con $j = n_\alpha + 1, ..., n$ delle nuove particelle

$\{\theta_j^{(t-1)}\}$ con $j = n_\alpha + 1, ..., n$ secondo la seguente formula:

$$w_i^{(t)} = \frac{\pi(\theta_i^{(t)})}{f^{(t)}(\theta_i^{(t)})};$$

3. si calcolano le distanze $d_j^{(t-1)}$ con $j = n_\alpha + 1, ..., n$ delle nuove particelle $\{\theta_j^{(t-1)}\}$ con $j = n_\alpha + 1, ..., n$;

4. si concatenano le n_α particelle che sono state generate al primo passo e si definisce il campione temporaneo $\mathcal{S}_{tmp}^{(t)} = (\theta_i^{(t)}, w_i^{(t)}, d_i^{(t)})$ con $i = 1, ..., n$;

5. il successivo livello di tolleranza ε_t è pari all'α-esimo quantile di $\{d_i^{(t)})\}$ con $i = 1, ..., n$;

6. il nuovo campione $\mathcal{S}^{(t)} = (\theta_i^{(t)}, w_i^{(t)})$ con $i = 1, ..., n_\alpha$ ed è costituito dalle n_α particelle provenienti da $\mathcal{S}_{tmp}^{(t)}$ che soddisfano $d_i^{(t)} \leq \varepsilon_t$.

L'algoritmo illustrato ha un criterio di arresto: se la porzione di particelle p_{acc} che soddisfano il livello di tolleranza ε_{t-1} tra le nuove $n - n_\alpha$ è più piccola di un valore scelto $p_{acc_{min}}$, l'algoritmo si ferma e restituisce come risultato il campione $(\theta_i^{(t)}, w_i^{(t)})$ con $i = 1, ..., n_\alpha$. La porzione di particelle p_{acc} è calcolata nel seguente modo:

$$p_{acc}(t) = \frac{1}{n - n_\alpha} \sum_{k=n_\alpha+1}^{n} \mathbb{1}_{d_k^{(t-1)} < \varepsilon_{t-1}}.$$

2.2.2 Nuovo algoritmo - knnABC

Il principio di funzionamento è simile a livello di strutturale a quello dell'algoritmo APCM di Lenormand ad eccezione di alcune differenze che si presentano di seguito.

La prima è relativa al fatto che la sequenza di distribuzioni $g^{(i)}(\theta)$ usata per proporre nuovi punti non fornisce una approssimazione della distribuzione a posteriori sempre più precisa (con il decrescere di h) come nel caso dell'algoritmo di Lenormand. Questo aspetto è una conseguenza del criterio di scelta di un parametro θ in corrispondenza del quale si ha un campione simulato "vicino" a quello osservato. Come introdotto in sezione 1.3.2 nei metodi senza verosimiglianza, anche l'algoritmo APCM utilizza una metrica, come la distanza di Mahalanobis, per misurare la distanza tra le statistiche di sintesi del campione simulato e del campione osservato. Questo approccio però verifica la vicinanza delle statistiche in modo puntuale nel senso che viene valutata la distanza della singola statistica di sintesi simulata sotto un certo θ_i che, per effetto del caso, può risultare vicina nonostante il relativo parametro sia in un punto dello spazio dove la curva di verosimiglianza non ha valori così elevati. L'intuizione alla base di questo nuovo algoritmo è ragionare in termini di media della distanza tra le statistiche in una determinata regione dello spazio. L'idea è quella che una statistica relativa al campione simulato in corrispondenza del parametro θ_i viene considerata come vicina alla statistica del campione osservato se la sua media con le distanze delle statistiche di sintesi simulate, relative ai k parametri più vicine a θ_i, è vicina. Per poter sviluppare questo passaggio si ricorre all'utilizzo della regressione locale non parametrica KNN (*k-nearest neighbors*, Azzalini, Scarpa e Walton 2012). Il modello ha come variabile risposta la distanza $\delta_i = s_i - s_{obs}$ mentre la variabile esplicativa è il parametro (che può avere naturalmente anche una dimensione $p > 1$). La stessa procedura di regressione locale si potrebbe fare utilizzando un kernel per ottenere lo stesso grado di lisciamento della curva. Si è scelto questo metodo non parametrico per la sua semplicità a livello concettuale dato che l'ampiezza di banda che regola il grado di lisciamento

della curva è solo legata al numero di vicini che si vogliono considerare.

La regressione lineare locale stima quindi il valore della distanza media $\hat{\delta}_i$ di k distanze individuate nella regione definita dai parametri più vicini nello spazio a θ_i. Tale distanza lisciata è quella utilizzata all'interno della formula per il calcolo della distanza di Mahalanobis per la selezione dei parametri in corrispondenza dei quali si hanno campioni con statistiche di sintesi (in media) vicine a quelle osservate. La formula è la seguente:

$$d(\hat{s}_i) = \sqrt{\hat{\delta}_i^{\,T} \mathbf{W}^{-1} \hat{\delta}_i} = \sqrt{(\hat{s}_i - s_{obs})^T \mathbf{W}^{-1} (\hat{s}_i - s_{obs})}. \tag{2.1}$$

La seconda differenza è legata alla stima della matrice $\mathbf{W}$ che compare in equazione (2.1). Nell'algoritmo di Lenormand la matrice è calcolata all'inizio a partire dalle prime estrazioni delle particelle effettuate dalla distribuzione a priori e mantenuta invariata per tutte le iterazioni dell'algoritmo. Nell'algoritmo proposto la matrice di varianza e covarianza viene stimata in modo robusto (Raymaekers e Rousseeuw 2021) e aggiornata sulla base di tutte le informazioni che si hanno a disposizione dall'inizio fino alla simulazione corrente. La scelta di stimare tale matrice in modo robusto è data dal fatto che le stime iniziali possono portare a campioni con statistiche di sintesi distanti da quella osservata. Tali informazioni non vengono comunque scartate, ma questa scelta fa in modo che siano meno influenti.

Il terzo aspetto differente rispetto all'algoritmo APCM riguarda il numero di punti più vicini (che compongono il "campione di elite") da conservare all'interno del campione. Nel caso di questo nuovo algoritmo, il numero di punti più vicini è una funzione decrescente con il passare delle iterazioni; il motivo di questa scelta è dettato dal fatto che all'inizio si parte con un numero elevato per esplorare una grande porzione dello spazio parametrico, mentre con il passare delle iterazioni servirà simulare un numero *add* di nuove

particelle a partire da sempre meno punti. È questo il senso dell'algoritmo sequenziale che si concentra con il passare delle iterazioni nella regione dello spazio parametrico con un valore della verosimiglianza più elevata. La funzione deve essere decrescente e limitata inferiormente da un asintoto orizzontale di modo che da una certa iterazione in avanti il numero di punti migliori rimanga costante in un numero diverso da 0. La velocità della funzione nel tendere al valore minimo è invece sintomo di come si vuole esplorare la regione: se tale funzione va verso il minimo velocemente significa che si è fiduciosi del fatto che si sta esplorando bene lo spazio parametrico già dall'inizio (ci si può trovare in questo caso nel momento in cui si ha una distribuzione priori informativa e si conosce già dove il fenomeno di interesse possa concentrarsi); nel caso in cui questa funzione tenda al valore minimo più lentamente vuol dire che si vuole esplorare la regione più lentamente a partire da punti più sparsi.

Per questo algoritmo è stata scelta come funzione la seguente

$$elite = n_{elite} + \frac{1}{2} n_{start} exp\left(-a_{elite}\left(1 - \frac{n}{{n_{elite}}^2}\right)\right). \tag{2.2}$$

In formula (2.2) *elite* è il numero di punti migliori che si considerano in quell'iterazione. Le costanti n_{elite}, n_{start} e a_{elite} sono dei parametri che regolano l'andamento della curva. L'altezza dell'asintoto orizzontale è identificata da n_{elite}, n_{start} è la costante che indica il numero di punti che sono stati simulati nella prima iterazione, mentre a_{elite} è un parametro che regola la velocità con cui la curva raggiunge l'asintoto orizzontale.

Infine, l'ultima differenza riguarda la regola d'arresto. Nel paragrafo precedente è stata presentata la regola d'arresto per l'algoritmo APCM che fa riferimento alla porzione di nuove particelle accettate. Il ragionamento alla base della regola d'arresto proposta mira a confrontare il volume della di-

stribuzione di importanza, che viene aggiornata ad ogni iterazione, con il volume della distribuzione a posteriori. L'idea è quella di arrestare l'algoritmo quando il volume della distribuzione di importanza sia nello stesso ordine di grandezza di quella della distribuzione a posteriori e che allo stesso tempo abbia ridotto la variabilità della distribuzione a priori.

Nel caso di una distribuzione p-variata, calcolare il suo volume è complicato, ma è possibile ottenerne un'approssimazione calcolando il volume dell'ellissoide che fornisce una misura della variabilità (e quindi di quanto è "ampia") della distribuzione di riferimento. Per il calcolo bisogna fare riferimento agli autovalori della matrice di varianza e covarianza della distribuzione di importanza. Sia x un nuovo valore generato a partire dalla distribuzione di importanza, allora la variabile casuale $X = E + Z$ è così definita, dove E è la variabile per un punto estratto in modo casuale a partire dal campione dei punti di elite e Z è una variabile normale p-variata di media nulla e varianza pari a una matrice $\mathbf{S}$ indipendente da E. Per come è stato scritto l'algoritmo, la matrice $\mathbf{S}$ è proprio la matrice di varianza e covarianza campionaria dei punti appartenenti al campione di punti migliori $E_1, ..., E_{elite}$ calcolata in modo robusto e aggiornata ad ogni iterazione. Quindi:

$$\mathrm{Var}(X) = \mathrm{Var}(E + Z) = \mathrm{Var}(E) + \mathrm{Var}(Z) = \mathbf{S} + \mathbf{S} = 2\mathbf{S}.$$

In appendice A sono riassunti alcuni risultati legati al calcolo del volume dell'ellissoide che si dimostra essere proporzionale alla radice quadrata del determinante della matrice di varianza e covarianze della distribuzione di riferimento. Sia $\mathcal{V}$ il volume dell'ellissoide della distribuzione e $\lambda_1, ..., \lambda_p$ gli autovalori della matrice di varianza e covarianza $\mathbf{S}$, si ottiene:

$$\mathcal{V} \propto \sqrt{\det(\mathbf{2S})} = \sqrt{\prod_{i=1}^{p} 2\lambda_i} = 2^{p/2} \prod_{i=1}^{p} \sqrt{\lambda_i}. \tag{2.3}$$

Per quanto riguarda il calcolo del volume dell'ellissoide della distribuzione a posteriori, il ragionamento da fare è analogo. Il problema è che non si ha a disposizione la vera matrice di varianza e covarianza di tale distribuzione. Per far fronte a questo problema si calcola la matrice di varianza e covarianza del campione di punti migliori selezionato questa volta a partire dalle distanze puntuali e non dalle distanze lisciate tramite regressione lineare locale. Successivamente, una volta ottenuta la stima della matrice di varianza e covarianza dei punti appartenenti al campione di elite si procede allo stesso modo calcolando un'approssimazione del volume dell'ellissoide attraverso la radice quadrata del determinante di questa matrice.

L'utilizzo di questo metodo per calcolare il volume della distribuzione a posteriori fornisce però una sovra-stima del vero volume a posteriori per due principali motivi:

- non essendo a conoscenza della vera forma della distribuzione a posteriori è possibile solo calcolarne un'approssimazione empirica sulla baste delle distanze puntuali. Queste, per effetto del caso, possono essere molto basse anche in regioni dello spazio parametrico dove la vera distribuzione a posteriori non ha molta massa di probabilità, ma il criterio scelto, $\|s_i - s_{obs}\| \leq h$, dato che guarda solo alle differenze puntuali, considererà tale estrazione come appartenente al campione di elite e quindi verrà utilizzato per il calcolo della matrice di varianza e covarianza empirica anche se si può trattare di un valore anomalo. Si valuta la matrice di varianza e covarianza empirica sulla base degli *elite* punti non tenendo conto però dei loro pesi di importanza. L'utilizzo

dei pesi di importanza farebbe in modo che la distribuzione risulti più concentrata perché questi punti anomali hanno un peso di importanza basso.

- il valore della costante h che regola la soglia per cui un campione viene accettato o meno non è ancora fissa perché si sta usando il valore dell'iterazione corrente.

Funzionamento dell'algoritmo

Prima di riassumere i passaggi di funzionamento dell'algoritmo, si deve tenere presente che il seguente metodo è costituito da due sotto-algoritmi:

- il primo permette di trovare la distribuzione di importanza, in un modo differente rispetto a Lenormand, ma sempre con una logica sequenziale;
- il secondo serve ad estrarre osservazioni provenienti dalla distribuzione a posteriori mediante un algoritmo di campionamento per importanza via ABC.

L'algoritmo knnABC funziona in modo analogo, nella struttura, all'algoritmo APCM con le differenze specificate nella sezione 2.2.2 che in realtà sono tutte proprie della prima parte dell'algoritmo, quella che permette di trovare una distribuzione di importanza appropriata per il campionamento.

L'algoritmo, come qualsiasi altro nell'ambito dell'*approximate Bayesian computation*, richiede che si sia in grado di simulare un campione y a partire dal modello $p(y|\theta)$, generare particelle da una certa distribuzione a priori e calcolare la densità in corrispondenza di tali punti. Quest'ultimo aspetto sarà essenziale per il calcolo dei pesi di importanza.

In prima battuta vengono simulati un numero pari a n_{start} parametri dalla distribuzione a priori $\pi(\theta)$ e a partire da questi si simulano osservazioni

y per le quali poi si calcolano le statistiche di sintesi $s = s(Y)$ e le relative distanze dalle statistiche osservate nei dati a disposizione $d_i = s_i - s_{obs}$ con $i = 1, ..., n_{start}$. A partire dalle osservazioni $\theta_1, ..., \theta_{n_{start}}$ si calcola un'approssimazione del volume dell'ellissoide della distribuzione a priori definita dal prodotto delle radici degli autovalori della matrice di varianza e covarianza campionaria come in equazione (2.3).

Si presenta il funzionamento dell'algoritmo alla t-esima generica iterazione, tenendo conto delle premesse iniziali e che dall'inizio fino all'iterazione precedente si è arrivati a collezionare un numero di parametri pari a n.

1. Si calcolano le distanze lisciate mediante regressione lineare locale, tenendo conto che i parametri sono standardizzati, altrimenti la scelta dei k vicini può essere influenzata dalle diverse scale che i parametri possono avere;

2. si calcola un'approssimazione robusta della matrice di varianza e covarianza di tali distanze lisciate;

3. a partire dal numero n di parametri calcolati fino all'iterazione corrente si calcola il valore di *elite*, ovvero quanti punti verranno selezionati come migliori, mediante la formula (2.2). Vengono scelti come appartenenti al campione di elite i parametri che hanno il valore della distanza di Mahalanobis minore;

4. siano $\mathcal{V}_{prior}$, $\mathcal{V}_{post}$ e $\mathcal{V}_{imp}$ rispettivamente l'approssimazione del volume dell'ellissoide della distribuzione a priori, a posteriori e di importanza. Si calcolano:

$$\frac{\mathcal{V}_{imp}}{\mathcal{V}_{post}} \quad \text{e} \quad \frac{\mathcal{V}_{imp}}{\mathcal{V}_{prior}}.$$

Questi rapporti forniscono una misura di quanto è grande il volume della distribuzione di importanza rispetto alla distribuzione a priori e a posteriori;

5. si verifica se la regola d'arresto viene rispettata;

6. si generano nuove osservazioni a partire da una mistura di *elite* normali tutte con peso pari a $\frac{1}{elite}$ ognuna centrata in un punto θ_i con $i = 1, ..., elite$ e con varianza pari alla varianza campionaria dei punti migliori.

La regola d'arresto a cui si fa riferimento nel passo (5) dell'algoritmo confronta il rapporto $\frac{\mathcal{V}_{imp}}{\mathcal{V}_{post}}$ con una certo valore di tolleranza, tol, fissato.

Per come è stato pensato l'algoritmo il valore di default impostato è $tol = 1.1$, ad indicare che l'algoritmo si arresta quando la distribuzione di importanza è circa il 10% più 'grande" della distribuzione a posteriori. La logica è che si vuole fare in modo che la distribuzione di importanza sia comparabile a quella a posteriori in termini del volume dell'ellissoide, ma allo stesso tempo che l'algoritmo si arresti quando questa rimane un po' più variabile per poter essere ragionevolmente sicuri che la maggior parte del supporto della distribuzione a posteriori sia incluso in quello della distribuzione di importanza.

Il rapporto $\frac{\mathcal{V}_{imp}}{\mathcal{V}_{prior}}$ invece non viene usato come controllo ai fini della regola d'arresto ma fornisce l'informazione di quanto grande sia il volume della distribuzione di importanza nei confronti della distribuzione a priori. È quindi auspicabile che $\frac{\mathcal{V}_{imp}}{\mathcal{V}_{prior}} \ll 1$, ad indicare che la distribuzione di importanza abbia un volume sostanzialmente minore della distribuzione a priori sintomo del fatto che si estraggono delle proposte per θ nella regione dello spazio

parametrico più interessante, ovvero quella occupata dalla distribuzione a posteriori.

Risultati dell'algoritmo

Al termine di questa prima parte dell'algoritmo si ottiene una distribuzione di importanza da cui campionare in modo efficace.

Il secondo passo dell'algoritmo serve ad ottenere un campione proveniente dalla distribuzione a posteriori con i relativi pesi di importanza (θ_i, w_i) con $i = 1, ..., n$. Per farlo si utilizza un l'algoritmo di campionamento per importanza via ABC presentato in sezione 1.3.2 con $g(\theta)$ che è la mistura di normali ciascuna centrata in uno dei punti migliori θ_i con $i = 1, ..., elite$ e tutte di egual peso.

Si deve quindi fissare un valore della soglia h con cui si decide se accettare o meno un valore proposto di θ e, una volta ottenuti le n estrazioni si calcolano i pesi di importanza nel seguente modo:

$$w_i = \frac{\pi(\theta_i)}{g(\theta_i)} \quad \text{con} \quad i = 1, ..., n,$$

e poi opportunamente riscalati per intenderli in termini di probabilità.

A partire dal primo passo dell'algoritmo è possibile scegliere il valore della soglia h in modo adeguato. Al campione di elite sono associate le distanze delle statistiche simulate dalla statistica osservata che, per costruzione del campione, sono le più basse. Il valore di default per questa soglia è il quantile di livello $\alpha = 0.1$ di queste distanze osservate che rimane invariato per tutta la parte di campionamento successiva. La scelta di questo quantile è un compromesso tra il fatto che questo valore non può essere troppo grande, in tal caso l'approssimazione non risulterebbe soddisfacente, e nemmeno troppo

piccola, altrimenti la probabilità di accettare un valore θ proposto sarebbe molto bassa. Tuttavia per come è costruito l'algoritmo, la distribuzione di importanza si trova già nella regione della distribuzione a posteriori e quindi non serve prendere un quantile troppo piccolo. Inoltre per la scelta di h i pesi di importanza non entrano in gioco, si considera ogni elemento del campione di elite con la stessa probabilità pari a $\frac{1}{elite}$.

Al termine delle simulazioni le stime dei parametri vengono aggiustate sulla base di modelli di regressione per tenere conto della discrepanza che c'è tra le statistiche di sintesi osservate e quelle del campione simulato, come illustrato in appendice B.

Capitolo 3

Esempi e simulazioni

In questo capitolo si riportano i risultati di analisi condotte per valutare la performance di knnABC nei confronti dell'algoritmo di Lenormand. I risultati riportati sono relativi a tre differenti situazioni:

- un esempio con dati binomiali. Questa è una situazione in cui si conosce il vero modello generatore dei dati e anche la vera forma della distribuzione a posteriori;

- un esempio con dati reali in cui si adatta un modello logistico con 6 variabili esplicative e quindi la dimensione del parametro θ è $p = 7$. In questa situazione si è a conoscenza delle vere stime di massima verosimiglianza ed è possibile ottenere anche una approssimazione della distribuzione a posteriori con gli algoritmi MCMC;

- un esempio con dati simulati in cui il modello di riferimento è un processo Markoviano a tempo continuo con uno stato a 4 dimensioni. In questo caso non è possibile calcolare gli stimatori di massima verosimiglianza oppure usare i tradizionali algoritmi MCMC poiché la verosimiglianza è intrattabile.

3.1 Esempio binomiale

Per capire a livello pratico il comportamento del nuovo algoritmo knnABC si presenta un esempio giocattolo con i dati y_{obs} che sono una realizzazione di una variabile casuale $Y \sim Bin(n, \theta)$ con $n = 50$ e $\theta = 0.2$. La conoscenza del vero modello generatore dei dati è essenziale per poter misurare la qualità dello stimatore $\hat{\theta}$ restituito dall'algoritmo. Si ipotizza di non avere informazioni a priori e quindi $\theta \sim U(0, 1)$. La scelta di questa distribuzione a priori permette di conoscere la vera forma della distribuzione a posteriori. La distribuzione di una variabile $U(0, 1)$ infatti coincide con la distribuzione $Beta(\alpha, \beta)$, con $\alpha = \beta = 1$, che è la distribuzione a priori coniugata quando i dati y_{obs} provengono dalla distribuzione binomiale. Seguendo la formula per il calcolo della distribuzione a posteriori è possibile definire che $\theta|y_{obs} \sim Beta(y_{obs} + \alpha, n - y_{obs} + \beta)$.

Per verificare la ripetibilità del metodo, dato che si stanno simulando le osservazioni relative a y_{obs}, la procedura della stima della distribuzione a posteriori viene replicata per 5 volte.

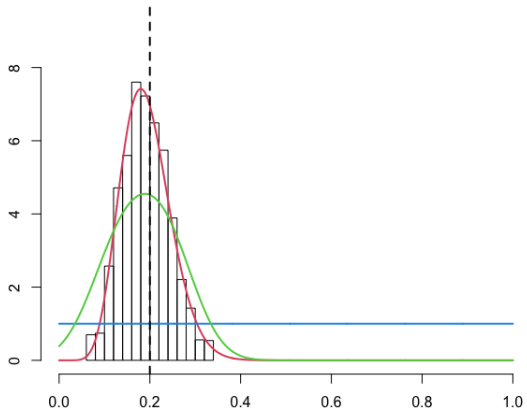

Figura 3.1: Nel grafico sono riportate: la densità a priori (linea blu), la vera densità a posteriori (linea rossa), la distribuzione di importanza dell'algoritmo knnABC (linea verde) e la distribuzione a posteriori stimata con knnABC (istogramma) nella prima iterazione. La linea tratteggiata corrisponde al vero valore del parametro $\theta = 0.2$.

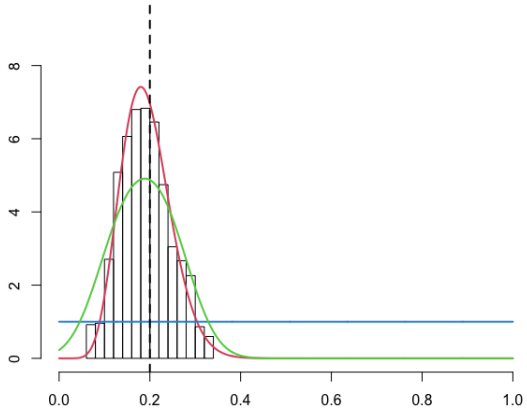

Figura 3.2: Nel grafico sono riportate: la densità a priori (linea blu), la vera densità a posteriori (linea rossa), la distribuzione di importanza dell'algoritmo knnABC (linea verde) e la distribuzione a posteriori stimata con knnABC (istogramma) nella seconda iterazione. La linea tratteggiata corrisponde al vero valore del parametro $\theta = 0.2$.

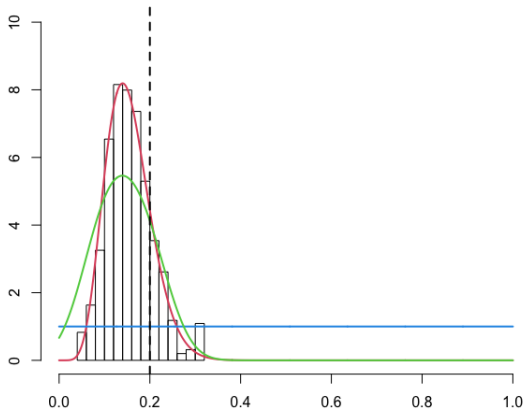

Figura 3.3: Nel grafico sono riportate: la densità a priori (linea blu), la vera densità a posteriori (linea rossa), la distribuzione di importanza dell'algoritmo knnABC (linea verde) e la distribuzione a posteriori stimata con knnABC (istogramma) nella terza iterazione. La linea tratteggiata corrisponde al vero valore del parametro $\theta = 0.2$.

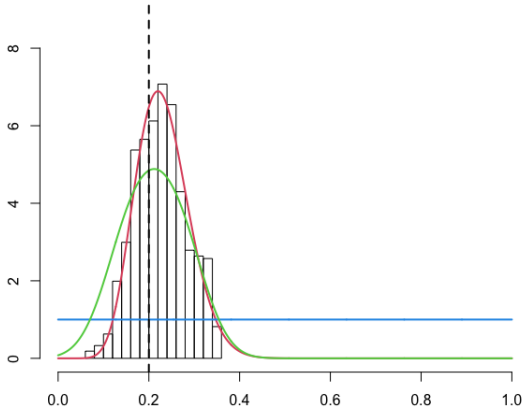

Figura 3.4: Nel grafico sono riportate: la densità a priori (linea blu), la vera densità a posteriori (linea rossa), la distribuzione di importanza dell'algoritmo knnABC (linea verde) e la distribuzione a posteriori stimata con knnABC (istogramma) nella quarta iterazione. La linea tratteggiata corrisponde al vero valore del parametro $\theta = 0.2$.

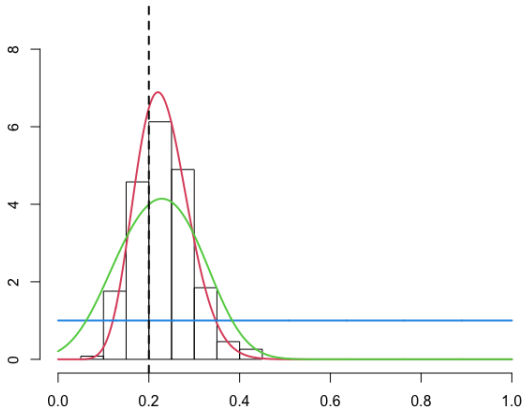

Figura 3.5: Nel grafico sono riportate: la densità a priori (linea blu), la vera densità a posteriori (linea rossa), la distribuzione di importanza dell'algoritmo knnABC (linea verde) e la distribuzione a posteriori stimata con knnABC (istogramma) nella quinta iterazione. La linea tratteggiata corrisponde al vero valore del parametro $\theta = 0.2$.

I grafici nelle figure 3.1, 3.2, 3.3, 3.4 e 3.5 illustrano come tra le 5 diverse replicazioni l'algoritmo fornisce qualitativamente gli stessi risultati. Si può notare come la distribuzione di importanza comporti un miglioramento rispetto alla distribuzione a priori uniforme nell'intervallo unitario in termini

di proposte per il valore del parametro. Per quanto riguarda invece la distribuzione a posteriori stimata dall'algoritmo si nota che mima bene la vera distribuzione a posteriori. Inoltre la densità a posteriori stimata tramite l'algoritmo knnABC ha la moda della distribuzione nei pressi del valore $\theta \approx 0.2$ che, come riportato in fase di definizione del modello, coincide con il vero valore del parametro.

Replicazione	Simulazioni
1	6091
2	13544
3	13171
4	5875
5	6260

Tabella 3.1: Numero di simulazioni dell'algoritmo knnABC nelle 5 replicazioni per la stima di un campione di $n = 1000$ valori di θ.

In tabella 3.1 vengono riportate il numero di simulazioni necessarie per stimare $n = 1000$ valori di θ dalla distribuzione a posteriori per ogni replicazione dell'algoritmo.

Il funzionamento dell'algoritmo è quello spiegato in sezione 2.2.2 con la distribuzione a priori evidenziata all'inizio di questa sezione. Un aspetto degno di nota è relativo al calcolo dei volumi degli ellissoidi che, avendo a che fare con un parametro scalare, coincidono con la deviazione standard della distribuzione che è proprio la radice della varianza, ovvero dell'autovalore.

Si può concludere che l'algoritmo proposto funziona per questo esempio in quanto restituisce risultati che sono in linea con i risultati analitici che per questo caso è possibile calcolare.

3.2 Esempio con dati reali: modello logistico

In questa sezione si presentano i risultati delle simulazioni per un'analisi di un'insieme di dati reali. Il dataset `Credit` (Salvan, Sartori e Pace 2020) contiene dati relativi a $n = 1000$ clienti di una banca della Germania meridionale. La variabile risposta y è una variabile dicotomica cosi definita:

$$y_i = \begin{cases} \texttt{buen} & \text{se l'individuo è un } \textit{good payer} \\ \texttt{mal} & \text{se l'individuo è un } \textit{bad payer}. \end{cases}$$

Sono stati osservati 700 *good payers* e 300 *bad payers* e, oltre a questa caratteristica, sono disponibili altre 6 variabili relative al profilo di ciascun individuo o al tipo di prestito richiesto che sono di seguito elencate:

$$X_1 = \begin{cases} 1 & \text{se l'individuo ha una buona gestione del conto corrente} \\ 0 & \text{altrimenti;} \end{cases}$$

$$X_2 = \begin{cases} 1 & \text{se l'individuo ha una cattiva gestione del conto corrente} \\ 0 & \text{altrimenti;} \end{cases}$$

$X_3 =$ durate del prestito in mesi;

$$X_4 = \begin{cases} 1 & \text{se l'individuo è stato in precedenza un } \textit{bad payer} \\ 0 & \text{se l'individuo è stato in precedenza un } \textit{good payer}; \end{cases}$$

$$X_5 = \begin{cases} 1 & \text{se il prestito è per uso professionale} \\ 0 & \text{se il prestito è per uso privato;} \end{cases}$$

$$X_6 = \begin{cases} 1 & \text{se l'individuo vive da solo} \\ 0 & \text{altrimenti.} \end{cases}$$

Il modello di riferimento per la variabile risposta è un modello logistico con queste 6 variabili esplicative:

$$\text{logit}(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6.$$

Si suppone che questo modello lineare generalizzato sia il vero modello generatore dei dati e che sia quindi possibile calcolare analiticamente il vero valore dei parametri sui quali è d'interesse condurre le procedure inferenziali.

Di seguito si riporta il vettore dei parametri e le relative stime di massima verosimiglianza:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix} = \begin{bmatrix} -0.53401 \\ -1.93772 \\ -0.61739 \\ 0.46854 \\ 0.98772 \\ 0.46940 \\ 0.53273 \end{bmatrix} \tag{3.1}$$

Oltre alla stima dei parametri classica, si è condotta l'analisi via MCMC; si utilizza anche questo metodo poiché è considerato lo strumento di riferimento per condurre simulazioni in situazioni più complesse come questa.

Per quanto riguarda invece l'approccio via ABC, per rendere le performance dell' algoritmi di Lenormand e dell'algoritmo knnABC confrontabili si sono scelte le stesse condizioni iniziali:

- si simulano $n = 1000$ estrazioni per il vettore $\boldsymbol{\beta}$ dalla distribuzione a posteriori;

- la distribuzione a priori è $\beta_i \sim U(-3, 3)$ con $i = 1, ..., 7$;

- la statistica di sintesi utilizzata è il prodotto tra la matrice del modello $\mathbf{X}$ e il vettore risposta $\mathbf{y}$: $s(\mathbf{y}) = \mathbf{X}^T\mathbf{y}$.

Di seguito si riportano i grafici e le tabelle per valutare i risultati prodotti dagli algoritmi, tenendo conto che per l'algoritmo di Lenormand si è utilizzata l'implementazione ufficiale contenuta nella funzione della libreria `EasyABC`.

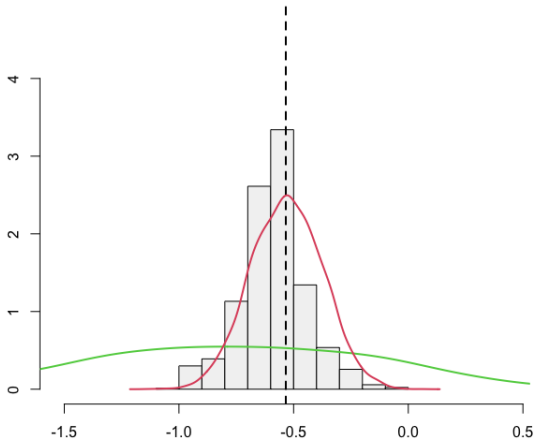

Figura 3.6: Distribuzioni stimate per β_1 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

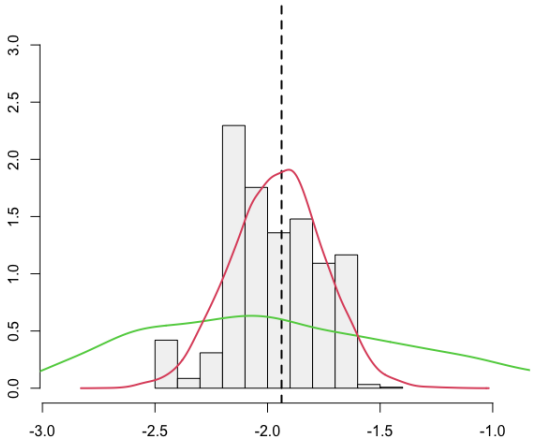

Figura 3.7: Distribuzioni stimate per β_2 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

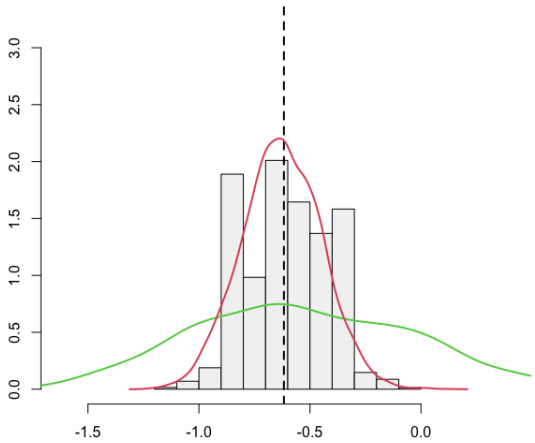

Figura 3.8: Distribuzioni stimate per β_3 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

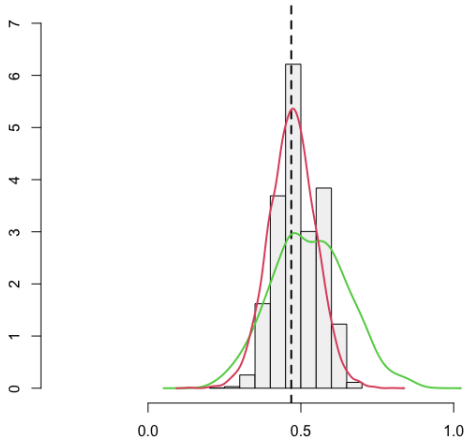

Figura 3.9: Distribuzioni stimate per β_4 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

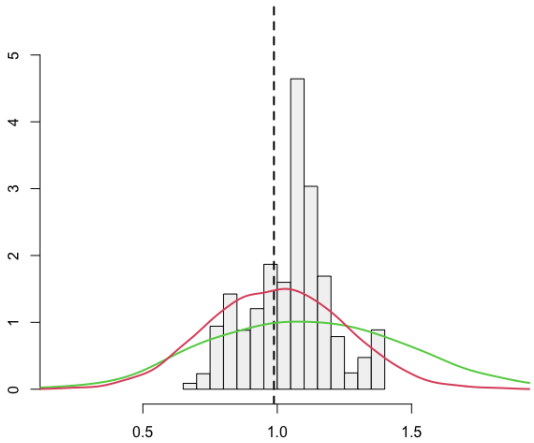

Figura 3.10: Distribuzioni stimate per β_5 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

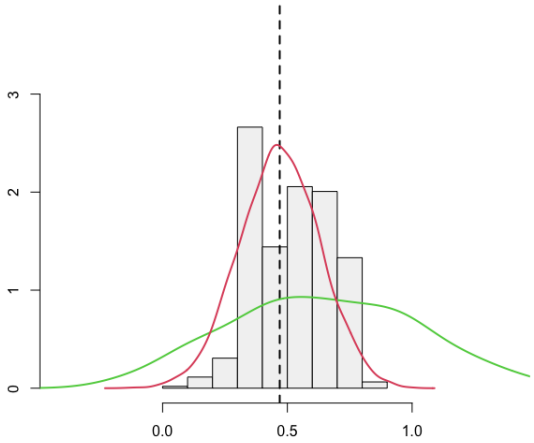

Figura 3.11: Distribuzioni stimate per β_6 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

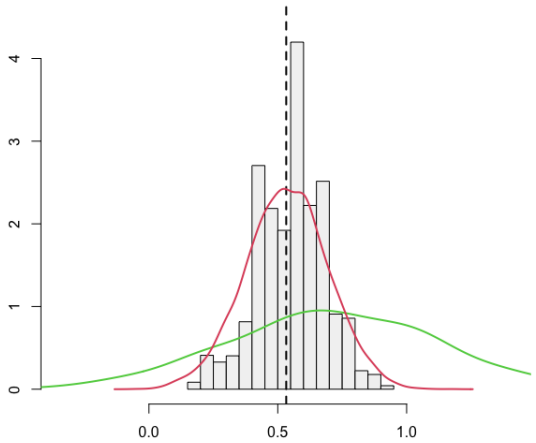

Figura 3.12: Distribuzioni stimate per β_7 dall'algoritmo knnABC (istogramma), dall'algoritmo di Lenormand (linea verde) e dall'algoritmo MCMC (linea rossa). La linea tratteggiata corrisponde alla stima di massima verosimiglianza (3.1).

	IMP	knnABC	LEN	JAGS	ML
β_1	-0.6085193	-0.5842396	-0.8110827	0.5326357	-0.5340083
β_2	-1.9220924	-1.9701124	-1.9400759	1.9598623	-1.9377175
β_3	-0.6075951	-0.6073577	-0.5197396	0.6258544	-0.6173896
β_4	0.4680179	0.4876876	0.5285338	0.4741448	0.4685445
β_5	1.0477010	1.0498990	1.1121127	0.9961111	0.9877161
β_6	0.5107908	0.5195157	0.6428257	0.4710259	0.4694008
β_7	0.5707101	0.5549740	0.7147565	0.5360106	0.5327310

Tabella 3.2: Stime puntuali dei coefficienti restituite dalla distribuzione di importanza stimata con l'algoritmo knnABC (IMP), dalla distribuzione a posteriori stimata con l'algoritmo knnABC (knnABC), dalla distribuzione a posteriori stimata con l'algoritmo di Lenormand (LEN), dalla distribuzione a posteriori stimata con l'algoritmo di MCMC (JAGS) e dal metodo della massima verosimiglianza (ML).

	IMP	knnABC	LEN	JAGS	ML
β_1	0.4966098	0.14578827	0.6746910	0.15630631	0.15531454
β_2	0.3694424	0.20381644	0.5980747	0.20560184	0.20546039
β_3	0.3494695	0.18037880	0.5097490	0.17424122	0.17573947
β_4	0.1450795	0.07435062	0.1231170	0.07666977	0.07589366
β_5	0.1979610	0.14989982	0.3610048	0.25111436	0.25266989
β_6	0.2401304	0.15871884	0.3776263	0.16048654	0.15968199
β_7	0.2455984	0.13261930	0.4074833	0.15905850	0.15912180

Tabella 3.3: Stime degli standard error dei coefficienti restituite dalla distribuzione di importanza stimata con l'algoritmo knnABC (IMP), dalla distribuzione a posteriori stimata con l'algoritmo knnABC (knnABC), dalla distribuzione a posteriori stimata con l'algoritmo di Lenormand (LEN), dalla distribuzione a posteriori stimata con l'algoritmo di MCMC (JAGS) e dal metodo della massima verosimiglianza (ML).

I grafici in figure 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 e 3.12 riportano gli istogrammi relativi alle distribuzioni a posteriori dei coefficienti $\boldsymbol{\beta}$ stimate mediante l'algoritmo knnABC e l'algoritmo di Lenormand.

In tabella 3.2 sono riportate le stime prodotte dai vari metodi utilizzati. Si può notare come le stime con knnABC siano a livello puntuale molto vicine a quelle reali, riportate nella colonna ML, e comunque sono comparabili alle stime prodotte dall'algoritmo JAGS che è lo standard di riferimento in questa situazione. Inoltre l'algoritmo knnABC ha nettamente una performance migliore rispetto all'algoritmo di Lenormand poiché è evidente che le stime di quest'ultimo hanno una distorsione nettamente maggiore.

La tabella 3.3 evidenzia un buon comportamento dell'algoritmo knnABC: le stime degli standard error sono nello stesso ordine di grandezza degli standard error stimati via massima verosimiglianza e tramite l'algoritmo JAGS. Anche se per alcuni parametri la distorsione di tale stima è maggiore, non è mai paragonabile alle distorsioni delle stime dell'algoritmo di Lenormand che

sistematicamente stima standard error almeno due volte più grandi. L'unico problema si può riscontrare nella stima dello standard error del coefficiente β_5 che da parte dell'algoritmo knnABC viene sottostimato. Non si considera così rilevante questo aspetto da poter ritenere la performance dell'algoritmo knnABC negativa poiché comunque si si tratta di un problema con una dimensione del parametro pari a $p = 7$.

knnABC	LEN
32194	67000

Tabella 3.4: Numero di simulazioni per stimare il vettore $\boldsymbol{\beta}$ con l'algoritmo (knnABC) e con l'algoritmo di Lenormand (LEN).

Si ritiene buono il comportamento dell'algoritmo knnABC che, in questo caso è migliore dell'algoritmo concorrente, poiché permette, in generale, con un minor numero di simulazioni (meno della metà) di ottenere stime più precise e con uno standard error minore.

3.3 Esempio con dati simulati: processo Markoviano

In questa sezione si riporta l'analisi di dati simulati nel caso del modello per la dinamica enzimatica di Michaelis e Menten (paragrafo 7.3 di Wilkinson 2019). In questa situazione un substrato S viene convertito in un prodotto P solo in presenza di un catalizzatore E (per l'enzima).

Un modello plausibile per illustrare questa dinamica è:

$$S + E \xrightarrow{\theta_1} SE \tag{3.2}$$

$$SE \xrightarrow{\theta_2} S + E \tag{3.3}$$

$$SE \xrightarrow{\theta_3} P + E \tag{3.4}$$

Dove θ_1, θ_2 e θ_3 sono costanti deterministiche di velocità cinetica dell'azione di massa che governano la dinamica della reazione.

Il modello è un processo Markoviano a tempo continuo con uno stato formato da 4 dimensioni $S(t), E(t), SE(t)$ e $P(t)$ dove le possibili reazioni sono governate dalle seguenti leggi di probabilità:

- Prima reazione (3.2) (S e E si incontrano e formano SE)

$$\begin{aligned} Pr\left[S(t+\delta)\right] &= S(t) - 1, \\ E(t+\delta) &= E(t) - 1, \\ SE(t+\delta) &= SE(t) + 1) = \theta_1 S(t) E(t) + o(\delta). \end{aligned}$$

- Seconda reazione (3.3) (SE si spezza e riforma S e E)

$$\begin{aligned} Pr\left[S(t+\delta)\right] &= S(t) + 1, \\ E(t+\delta) &= E(t) + 1, \\ SE(t+\delta) &= SE(t) - 1) = \theta_2 SE(t) + o(\delta). \end{aligned}$$

- Terza reazione (3.4) (SE si spezza e forma una molecola P e una E)

$$\begin{aligned} Pr\left[SE(t+\delta)\right] &= SE(t) - 1, \\ P(t+\delta) &= P(t) + 1, \\ E(t+\delta) &= E(t) + 1) = \theta_3 SE(t) + o(t). \end{aligned}$$

In questo caso l'inferenza sui parametri può essere condotta solo via simulazione. È noto che il vero valore dei parametri è pari a:

$$\theta = (\theta_1, \theta_2, \theta_3) = (0.01, 0.05, 0.02). \tag{3.5}$$

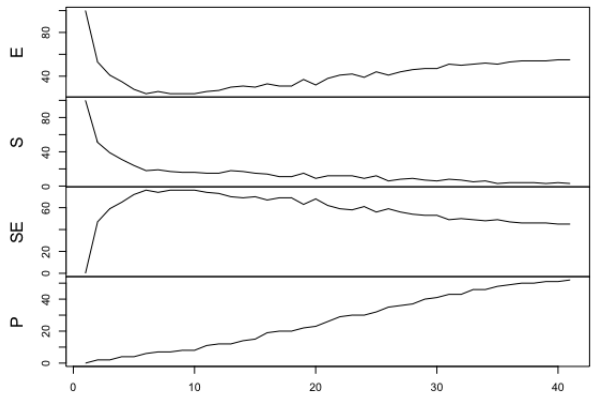

Figura 3.13: Grafico di un esempio di traiettorie simulate delle 4 dimensioni dello stato del processo utilizzando come parametri $\theta = (\theta_1, \theta_2, \theta_3) = (0.01, 0.05, 0.02)$ in funzione del tempo.

Per queste simulazioni la statistica sufficiente utilizzata è il prodotto tra la base $\mathbf{B}$ di una spline cubica con $gdl = 4$ e la matrice del campione di riferimento $\mathbf{Y}$. La scelta di utilizzare una base della spline cubica è dovuta al fatto che si vuole cogliere i trend delle traiettorie, che come si vede in figura 3.13, sono regolari, sintetizzando l'informazione. La statistica $s(\mathbf{Y}) = \mathbf{B}^T\mathbf{Y}$

coincide con la statistica sufficiente che si utilizzerebbe in un modello lineare basato sulla spline.

I risultati di questa analisi riportano il confronto della performance dell'algoritmo knnABC con l'algoritmo di Lenormand per 200 simulazioni. Si considera la trasformazione dei parametri $\eta_i = -\log_{10}(\theta_i)$ assumendo come distribuzioni a priori $\eta_i \sim U(-1, 5)$.

	knnABC	LEN1	LEN2
θ_1	0.0188	0.0935	0.0208
θ_2	0.0896	0.6904	0.1346
θ_3	0.0204	0.0203	0.0204

Tabella 3.5: Stime puntuali per i parametri θ_1, θ_2 e θ_3 del modello per la dinamica enzimatica.

	knnABC	LEN1	LEN2
θ_1	0.0155	0.0917	0.0294
θ_2	0.0802	0.7007	0.2258
θ_3	0.0031	0.0031	0.0031

Tabella 3.6: Radice dell'errore quadratico medio (RMSE) per i parametri θ_1, θ_2 e θ_3 del modello per la dinamica enzimatica.

	knnABC	LEN1	LEN2
Minimo	21673	28000	105000
I Quartile	31168	33000	178000
Mediana	37155	41000	198000
Media	38228	41715	194495
III Quartile	44429	50000	215000
Massimo	64297	57000	280000

Tabella 3.7: Statistiche di sintesi del numero di simulazioni effettuate nelle 200 replicazioni dell'algoritmo.

Le tabelle 3.5, 3.6 e 3.7 riportano i risultati delle simulazioni dell'algoritmo knnABC e dell'algoritmo di Lenormand. In particolare per il secondo si illustrano i risultati per l'algoritmo di Lenormand con l'accuratezza standard pari a $p_{acc} = 0.05$ (LEN1) e con un'accuratezza elevata $p_{acc} = 0.005$ (LEN2).

I risultati delle stime di queste simulazioni a livello puntuale e di precisione evidenziano come l'algoritmo knnABC fornisca stime più precise e con un RMSE nettamente minore (per θ_3 è uguale) dell'algoritmo di Lenormand con accuratezza standard, nonostante il numero di simulazioni effettuate, elencate in tabella 3.7, sia grosso modo lo stesso. I risultati invece diventano quasi equivalenti, con gli RMSE che comunque sono migliori, se si confronta l'algoritmo knnABC con l'algoritmo di Lenormand con precisione maggiore nonostante che per quest'ultimo confronto emerge in tabella 3.7 che il numero di simulazioni per ottenere un campione per θ di numerosità $n = 1000$ è circa 5 volte maggiore.

Capitolo 4

Conclusioni

L'obbiettivo di questo lavoro è stato quello di presentare un nuovo algoritmo con l'obbiettivo di migliorare le prestazioni degli algoritmi di *approximate Bayesian computation* sequenziali come, per esempio, l'algoritmo di Lenormand. La scelta di questo algoritmo è dovuta al fatto che questo è uno dei più usati in questo ambito e inoltre perché è l'unico algoritmo completamente automatico, dunque non richiede alcuna conoscenza preliminare del problema. Il paragone con questo nuovo algoritmo quindi viene fatto a parità di informazione. Nella sezione 2.2.2 sono state presentate le principali differenze e queste dimostrano che le scelte fatte conducono a simulazioni più soddisfacenti andando a valutare i risultati relativi all'esempio della stima dei parametri di un modello logistico con dei dati reali (sezione 3.2) e all'esempio di un processo Markoviano a tempo continuo (sezione 3.3).

A questo proposito è quindi possibile affermare che ci sono presupposti per poter utilizzare questo algoritmo anche per altri problemi visti i risultati convincenti.

È opportuno far presente che per capire quanto questo nuovo algoritmo sia performante in un senso più ampio rispetto agli esempi presentati in

questo lavoro, andrebbero esplorate situazioni più complesse in termini di dimensione del problema; la dimensione del problema può fare riferimento sia al numero di parametri sia al numero di statistiche di sintesi coinvolte. Per esempio ci possono essere casi in cui si vuole condurre inferenza su una funzione del tempo per la quale si hanno un numero limitato di osservazioni in un intervallo e quindi è ragionevole scegliere di non sintetizzare le osservazioni con qualche statistica ma di usare l'intero campione.

Nell'ottica di possibili ricerche future l'idea potrebbe essere quella di combinare la logica degli algoritmi sequenziali con la seconda parte del nuovo algoritmo proposto. In fase di presentazione del funzionamento dell'algoritmo, sezione 2.2.2, si è evidenziato come la parte di aggiornamento sequenziale dell'algoritmo sia relativa solo alla ricerca della distribuzione per importanza che rimane fissa nella fase di campionamento. Anche la soglia che definisce la distanza massima e che determina quali valori di θ accettare come provenienti dalla distribuzione a posteriori rimane invariata. L'interesse a voler combinare gli algoritmi sequenziali con questa seconda parte dell'algoritmo ha come obbiettivo quello di voler aggiornare in modo adattivo sia la distribuzione di importanza che il valore della soglia con l'idea che le cose possano migliorare in termini di numero di iterazioni dell'algoritmo.

Appendice A

Volume di un ellissoide

In questa sezione l'obbiettivo è quello di fornire una definizione di ellissoide e di alcune utili proprietà ad esso legate (Gupta e O'Donnell 2013).

Si definisce una sfera $B(c, r)$ (in $\mathbb{R}^n$) di centro $c \in \mathbb{R}^n$ e di raggio r l'insieme:

$$B(c, r) := \{x \in \mathbb{R}^n : x^T x \leq r^2\},$$

con $B(0, 1)$ che prende il nome di *sfera unitaria*.

A partire da questa definizione, un ellissoide non è altro che una trasformazione affine di tale sfera.

Un ellissoide E centrato nell'origine è il risultato di una trasformazione lineare invertibile $L : \mathbb{R}^n \to \mathbb{R}^n$. L'insieme di punti che appartengono ad un ellissoide è definito nel seguente modo:

$$\begin{aligned} E = L(B(0,1)) &= \{Lx : x \in B(0,1)\} \\ &= \left\{y : L^{-1}y \in B(0,1)\right\} \\ &= \left\{y : \left(L^{-1}y\right)^T L^{-1}y \leq 1\right\} \\ &= \left\{y : y^T \left(LL^T\right)^{-1} y \leq 1\right\} \\ &= \left\{y : y^T Q^{-1} y \leq 1\right\}. \end{aligned}$$

La traslazione $c+E$, con $c \in \mathbb{R}^n$, di un ellissoide centrato nell'origine (E) definisce un ellissoide di generico centro c. Nella definizione dell'ellissoide E come trasformazione lineare L di una sfera ci si torva di fronte alla matrice $Q \in \mathbb{R}^{n\times n}$, che gode delle seguenti proprietà:

1. è una matrice simmetrica;
2. $Q = LL^T$ per qualche matrice non singolare L;
3. tutti gli n autovalori di Q sono strettamente positivi.

Un ellissoide può essere quindi definito in modo equivalente a partire da una matrice Q definita positiva e da un centro $c \in \mathbb{R}^n$:

$$E(c,Q) := \left\{c + y : y^T Q^{-1} y \leq 1\right\} = \left\{y : (y-c)^T Q^{-1}(y-c) \leq 1\right\}.$$

Come ultima cosa si mostra il calcolo del volume di un ellissoide. Si indica con $\mathrm{vol}(A)$ il volume dell'insieme $A \subseteq \mathbb{R}^n$ e L una trasformazione lineare, allora:

$$\mathrm{vol}(L(A)) = \det(L) \cdot \mathrm{vol}(A).$$

In particolare, il volume di un ellissoide è:

$$\mathrm{vol}(E(c,Q)) = \det(L) \cdot \mathrm{vol}(B(0,1)) = \sqrt{\det(Q)} \cdot \mathrm{vol}(B(0,1)).$$

Quest'ultima formula mette in relazione il volume di un'ellissoide con il volume di una sfera unitaria; il volume di E è da considerarsi proporzionale alla radice quadrata del determinante della matrice Q, a meno del valore del volume della sfera unitaria che è constante e irrilevante ai fini pratici.

Appendice B

Aggiustamento dei parametri basato sulla regressione

In questa sezione si discutono gli approcci di regressione per i metodi di *approximate Bayesian computation* (Sisson, Fan e Beaumont 2018).

Quando si esegue la seconda fase dell'algoritmo, quella del campionamento per importanza, si accettano i valori di θ_i tali per cui i relativi campioni simulati tramite il modello di riferimento hanno statistiche di sintesi s_i vicine alla statistica di sintesi del campione osservato s_{obs}. Questi approcci mirano ad aggiustare le stime dei parametri per tenere conto della discrepanza che intercorre tra s_i e s_{obs}; per effettuare questa correzione si utilizzano dei modelli di regressione, da qui infatti deriva il nome di questi approcci.

Si ha a disposizione il campione (θ_i, d_i) con $i = 1, ..., n$ dove $d_i = s_i - s_{obs}$ sono appunto le discrepanze dalla statistica di sintesi del campione osservato. Si costruisce un modello di regressione lineare

$$\theta_i = a + \beta d_i + \epsilon_i,$$

e si stimano ai minimi quadrati i coefficienti così da ottenere $(\hat{a}, \hat{\beta})$.

Se il campione simulato $(\theta_1, ..., \theta_n)$ fosse tale che le statistiche simulate siano $s_i = 0 \ \ \forall i$ allora risulterebbe che il vero valore del parametro sia:

$$\theta_i = \hat{a} + \epsilon_i,$$

con $\hat{a}$ il valore stimato dall'algoritmo di ABC e ϵ_i l'errore di stima.

Naturalmente questa situazione non è verosimile perché le distanze d_i possono essere non nulle. L'aggiustamento delle stime si effettua correggendole per l'effetto delle d_i in modo da ottenere

$$\theta_i^* = \theta_i - \hat{\beta} d_i.$$

Essendo che tale correzione viene fatta alla fine dell'algoritmo, solitamente le d_i sono molto piccole e allo stesso modo anche il valore di $\hat{\beta}$; come risultato quindi le stime aggiustate con questo metodo non sono così diverse dalle stime di partenza, ma si dimostra con un risultato teorico che questo approccio fornisce un miglioramento in termini di convergenza dell'algoritmo (Li e Fearnhead 2018).

Appendice C

Codice R

In questa sezione si riporta il codice `R` per implementare l'algoritmo knnABC.

La funzione `findImportanceSampler` permette di trovare la distribuzione di importanza.

```
findImportanceSampler <- function(tobs, tsim, rprior, dprior,
                                  n_start = 1000, add = 50, n_max = 10000,
                                  n_elite = 200, a_elite = 1,
                                  tol = 1.1, trace = 0) {
    p <- length(rprior())
    q <- length(tobs)
    r_sim <- function(theta) tsim(theta) - tobs
    par <- matrix(NA, p, n_max)
    stat <- matrix(NA, q, n_max)
    w <- rep(1, n_max)
    idx <- seq.int(n_start)
    par[, idx] <- replicate(n_start, rprior())
    stat[, idx] <- apply(par[, idx, drop = FALSE], 2, r_sim)
    a_prior <- sqrt(eigen(
        tcrossprod((par[, idx, drop = FALSE] -
                        rowMeans(par[, idx, drop = FALSE]))
                        / sqrt(n_start - 1)),
        symmetric = TRUE, only.values = TRUE)$values)
    c_vol <- 2^(p / 2)
    n <- n_start
    nextreport <- trace
    while (1) {
        idx <- seq.int(n)
```

```
        ## find the actual importance sampler
        par_scale <- par[, idx, drop = FALSE]
        par_scale <- (par_scale - rowMeans(par_scale))
                        / apply(par_scale, 1, sd)
        stat_n <- stat[, idx, drop = FALSE]
        shat <- knnreg(par_scale, stat_n, par_scale)
        sc <- isqm(wrap(stat_n - shat))
        elite <- floor(n_elite + 0.5 * n_start *
                         exp(-a_elite * (1 - n / n_start)^2))
        mix <- select_elite(par, sc %*% shat, elite)
        ## compute an estimate of the "volume" of the
        ## importance sampler relative to the prior "volume"
        ## and to an (overestimate) of the posterior "volume"
        a_post <- select_elite(par, sc %*% stat_n, elite)$l
        v_prior <- c_vol * prod(mix$l / a_prior)
        v_post <- c_vol * prod(mix$l / a_post)
        ## report and check convergence
        done <- (n >= n_max) || (v_post < tol)
        if ((trace > 0) && (done || (n >= nextreport))) {
            cat(
                "After", n, "simulations \n",
                " number of components:", elite, "\n",
                " mixture mean:   ", mix$em, "\n",
                " mixture volume (relative to prior):", v_prior, "\n",
                " mixture volume (relative to posterior):", v_post, "\n"
            )
            nextreport <- n + trace
        }
        if (done) break

        ## sample new points
        mn <- min(n_max - n, add)
        idx <- seq.int(n + 1, n + mn)
        par[, idx] <- replicate(mn, rmix(mix, dprior))
        stat[, idx] <- apply(par[, idx, drop = FALSE], 2, r_sim)
        n <- n + mn
    }
    stat <- stat[, mix$best, drop = FALSE]
    list(
        p = p, q = q, n = n, mix = mix,
        sc = sc, stat = stat, d = colSums((sc %*% stat)^2),
        dprior = dprior, r_sim = r_sim
    )
}
```

La funzione `importanceABC` permette di trovare un campione di una certa numerosità a partire dalla distribuzione di importanza trovata tramite la

funzione `findImportanceSampler`.

```
importanceABC <- function(sampler, n = 1000,
                          dmax = quantile(sampler$d, 0.1),
                          adjust = TRUE, ...) {
    mix <- sampler$mix
    dprior <- sampler$dprior
    rsim <- sampler$r_sim
    sc <- sampler$sc
    par <- matrix(NA, sampler$p, n)
    stat <- matrix(NA, sampler$q, n)
    d <- numeric(n)
    nreplace <- n
    ireplace <- seq.int(n)
    nsim <- 0
    while (nreplace > 0) {
        par[, ireplace] <- replicate(nreplace, rmix(mix, dprior))
        stat[, ireplace] <- apply(par[, ireplace, drop = FALSE], 2, rsim)
        nsim <- nsim + nreplace
        d[ireplace] <- colSums((sc %*% stat[, ireplace, drop = FALSE])^2)
        ireplace <- which(d > dmax)
        nreplace <- length(ireplace)
    }
    w <- apply(par, 2, function(x) dprior(x) / dmix(x, mix))
    w <- w / sum(w)
    if (adjust) par <- adjustPar(par, stat, ...)
    list(param = par, stats = stat, weights = w,
         ess = 1 / sum(w * w), n = nsim, d = d, dmax = max(d))
}
```

La funzione `select_elite` permette di trovare gli *elite* punti migliori.

```
select_elite <- function(par, stat, elite) {
    best <- knnsearch0(stat, elite)
    e <- par[, best, drop = FALSE]
    em <- rowMeans(e)
    a <- eigen(tcrossprod((e - em) / sqrt(elite)), symmetric = TRUE)
    list(
        best = best, em = em,
        w = rep(1 / elite, elite), e = e,
        V = a$vectors,
        l = sqrt(pmax(.Machine$double.eps, a$values))
    )
}
```

La funzione `adjustPar` aggiusta linearmente i parametri (Li e Fearnhead 2018).

```
adjustPar <- function(par, stat, transform = "none",
                      lower = 0, upper = 1) {
    p <- NROW(par)
    if (length(transform) == 1) transform <- rep(transform, p)
    if (length(lower) == 1) lower <- rep(lower, p)
    if (length(upper) == 1) upper <- rep(upper, p)
    ilog <- which(transform == "log")
    ilogit <- which(transform == "logit")
    par[ilog, ] <- log(par[ilog, ] - lower[ilog])
    par[ilogit, ] <- log((par[ilogit, ] - lower[ilogit])
                         / (upper[ilogit] - par[ilogit, ]))
    m <- lm(t(par) ~ t(stat))
    par <- par - t(predict(m)) + (if (NROW(par) == 1) coef(m)[1]
                                  else coef(m)[1, ])
    par[ilog, ] <- lower[ilog] + exp(par[ilog, ])
    par[ilogit, ] <- lower[ilogit] + (upper[ilogit] - lower[ilogit])
                         / (1 + exp(-par[ilogit, ]))
    par
}
```

Le funzioni `rmix` e `dmix` permettono rispettivamente di generare nuovi punti da una mistura di normali troncata e di calcolare il valore della densità della mistura di normali in un punto.

```
rmix <- function(mix, accept) {
    mu <- mix$e
    p <- NROW(mix$e)
    m <- NCOL(mix$e)
    w <- mix$w
    V <- mix$V
    l <- mix$l
    while (1) {
        x <- as.numeric(mu[, sample.int(m, 1, prob = w)]
                + V %*% rnorm(p, sd = l))
        if (accept(x) > 0) break
    }
    x
}
```

```
dmix <- function(x, mix) {
    a <- sum(log(sqrt(2 * pi) * mix$l))
    sum(mix$w * exp(-0.5 * colSums((crossprod(mix$V, mix$e - x)
                                    / mix$l)^2) - a))
}
```

La funzione `wrap` permette di calcolare la matrice di varianza e covarianza in modo robusto (Raymaekers e Rousseeuw 2021).

```
wrap <- function(x) {
    B <- 1.5
    C <- 4
    Q1 <- 1.5407929
    Q2 <- 0.86227309
    a <- abs(x)
    s <- apply(a, 1, median)
    a <- a / pmax(.Machine$double.eps, s)
    x[a > C] <- 0
    i <- which((B < a) & (a < C))
    x[i] <- Q1 * sign(x[i]) * tanh(Q2 * (C - a[i]))
    outer(s, s) * tcrossprod(x / sqrt(NCOL(x)))
}
```

La funzione `isqm` permette di calcolare la matrice inversa della radice quadrata di una matrice.

```
isqm <- function(A, eps = sqrt(.Machine$double.eps)) {
    A <- eigen(A, symmetric = TRUE)
    n <- sum(A$values > eps)
    d <- c(1 / sqrt(A$values[seq_len(n)]), rep(0, length(A$values) - n))
    A$vectors %*% (d * t(A$vectors))
}
```

Bibliografia

[1] J. Albert. *Bayesian Computation with R*. en. New York, NY: Springer New York, 2009.

[2] A. Azzalini, B. Scarpa e G. Walton. *Data analysis and data mining: an Introduction*. en. Oxford ; New York: Oxford University Press, 2012.

[3] P. Del Moral, A. Doucet e A. Jasra. "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Statistics and computing* 22 (2012), pp. 1009–1020.

[4] C. C. Drovandi e A. N. Pettitt. "Bayesian Algorithms with Applications". en. Tesi di dott.

[5] C. C. Drovandi e A. N. Pettitt. "Estimation of Parameters for Macroparasite Population Evolution Using Approximate Bayesian Computation". en. In: *Biometrics* 67.1 (2011), pp. 225–233.

[6] A. Gupta e R. O'Donnell. "Lecture notes for CMU's course on Linear Programming & Semidefinite Programming". en. 2013.

[7] M. Lenormand, F. Jabot e G. Deffuant. "Adaptive approximate Bayesian computation for complex models". en. In: *Computational Statistics* 28.6 (2013), pp. 2777–2796.

[8] W. Li e P. Fearnhead. “Convergence of regression-adjusted approximate Bayesian computation”. en. In: *Biometrika* 105.2 (2018), pp. 301–318.

[9] B. Liseo. “Introduzione alla statistica bayesiana”. 2010.

[10] J. Raymaekers e P. J. Rousseeuw. In: *Technometrics* 63.2 (2021), pp. 184–198.

[11] C. Robert e G. Casella. *Introducing Monte Carlo Methods with R*. en. New York, NY: Springer New York, 2010.

[12] A. Salvan, N. Sartori e L. Pace. *Modelli lineari generalizzati*. Springer, 2020.

[13] A. Salvan, N. Sartori e L. Pace. “Statistical Inference: Theory and Methods”. 2022.

[14] N. Sartori e A. Guolo. “Statistica Computazionale (progredito)”. 2022.

[15] S. A. Sisson, Y. Fan e M. Beaumont. *Handbook of Approximate Bayesian Computation*. en. First edition. New York: Taylor & Francis Ltd, 2018.

[16] B. M. Turner e T. Van Zandt. “A tutorial on approximate Bayesian computation”. en. In: *Journal of Mathematical Psychology* 56.2 (2012), pp. 69–85.

[17] D. Wegmann et al. “ABCtoolbox: a versatile toolkit for approximate Bayesian computations”. In: *BMC bioinformatics* 11.1 (2010), pp. 1–7.

[18] D. J. Wilkinson. *Stochastic modelling for systems biology*. en. Third edition. Chapman & Hall/CRC mathematical and computational biology. Boca Raton: CRC Press, Taylor e Francis Group, 2019.