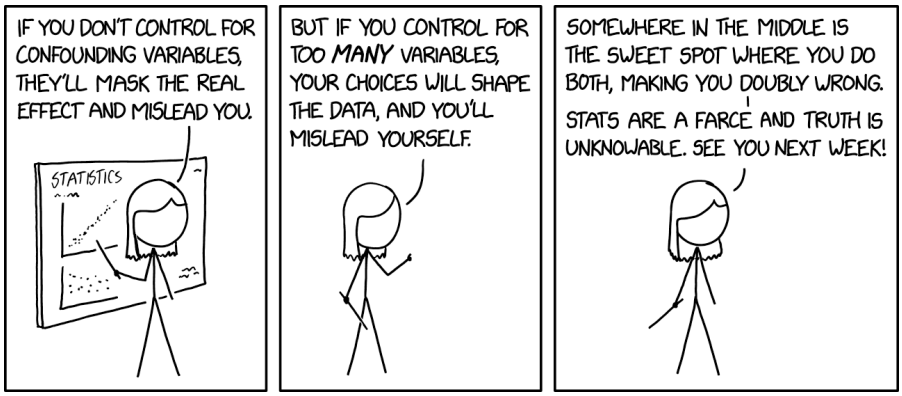
Corrado Caudek

Data Science per psicologi

Psicometria – AA 2021/2022



Indice

Elenco delle figure

Elenco delle tabelle

Prefazione

Data Science per psicologi contiene il materiale delle lezioni dell'insegnamento di *Psicometria B000286* (A.A. 2021/2022) rivolto agli studenti del primo anno del Corso di Laurea in Scienze e Tecniche Psicologiche dell'Università degli Studi di Firenze. *Psicometria* si propone di fornire agli studenti un'introduzione all'analisi dei dati in psicologia. Le conoscenze/competenze che verranno sviluppate in questo insegnamento sono quelle della Data science, ovvero un insieme di conoscenze/competenze che si pongono all'intersezione tra statistica (ovvero, richiedono la capacità di comprendere teoremi statistici) e informatica (ovvero, richiedono la capacità di sapere utilizzare un software).

La psicologia e la Data science

Sembra sensato spendere due parole su un tema che è importante per gli studenti: quello indicato dal titolo di questo Capitolo. È ovvio che agli studenti di psicologia la statistica non piace. Se piacesse, forse studierebbero Data science e non psicologia; ma non lo fanno. Di conseguenza, gli studenti di psicologia si chiedono: "perché dobbiamo perdere tanto tempo a studiare queste cose quando in realtà quello che ci interessa è tutt'altro?" Questa è una bella domanda.

C'è una ragione molto semplice che dovrebbe farci capire perché la Data science è così importante per la psicologia. Infatti, a ben pensarci, la psicologia è una disciplina intrinsecamente statistica, se per statistica intendiamo quella disciplina che studia la variazione delle caratteristiche degli individui nella popolazione. La psicologia studia *gli individui* ed è proprio la variabilità inter- e intra-individuale ciò che vogliamo descrivere e, in certi casi, predire. In questo senso, la psicologia è molto diversa dall'ingegneria, per esempio. Le proprietà di un determinato ponte sotto certe condizioni, ad esempio, sono molto simili a quelle di un altro pon-

te, sotto le medesime condizioni. Quindi, per un ingegnere la statistica è poco importante: le proprietà dei materiali sono unicamente dipendenti dalla loro composizione e restano costanti. Ma lo stesso non può dirsi degli individui: ogni individuo è unico e cambia nel tempo. E le variazioni tra gli individui, e di un individuo nel tempo, sono l'oggetto di studio proprio della psicologia: è dunque chiaro che i problemi che la psicologia si pone sono molto diversi da quelli affrontati, per esempio, dagli ingegneri. Questa è la ragione per cui abbiamo tanto bisogno della Data science in psicologia: perché la Data science ci consente di descrivere la variazione e il cambiamento. E queste sono appunto le caratteristiche di base dei fenomeni psicologici.

Sono sicuro che, leggendo queste righe, a molti studenti sarà venuta in mente la seguente domanda: perché non chiediamo a qualche esperto di fare il "lavoro sporco" (ovvero le analisi statistiche) per noi, mentre noi (gli psicologi) ci occupiamo solo di ciò che ci interessa, ovvero dei problemi psicologici slegati dai dettagli "tecnici" della Data science? La risposta a questa domanda è che non è possibile progettare uno studio psicologico sensato senza avere almeno una comprensione rudimentale della Data science. Le tematiche della Data science non possono essere ignorate né dai ricercatori in psicologia né da coloro che svolgono la professione di psicologo al di fuori dell'Università. Infatti, anche i professionisti al di fuori dall'università non possono fare a meno di leggere la letteratura psicologica più recente: il continuo aggiornamento delle conoscenze è infatti richiesto dalla deontologia della professione. Ma per potere fare questo è necessario conoscere un bel po' di Data science! Basta aprire a caso una rivista specialistica di psicologia per rendersi conto di quanto ciò sia vero: gli articoli che riportano i risultati delle ricerche psicologiche sono zeppi di analisi statistiche e di modelli formali. E la comprensione della letteratura psicologica rappresenta un requisito minimo nel bagaglio professionale dello psicologo.

Le considerazioni precedenti cercano di chiarire il seguente punto: la Data science non è qualcosa da studiare a malincuore, in un singolo insegnamento universitario, per poi poterla tranquillamente dimenticare. Nel bene e nel male, gli psicologi usano gli strumenti della Data science in tantissimi ambiti della loro attività professionale: in particolare quando costruiscono, somministrano e interpretano i test psicometrici. È dunque chiaro che possedere delle solide basi di Data science è un tassello imprescindibile del bagaglio professionale dello psicologo. In questo insegnamento verranno trattati i temi base della Data science e verrà

adottato un punto di vista bayesiano, che corrisponde all'approccio più recente e sempre più diffuso in psicologia.

Come studiare

Il giusto metodo di studio per prepararsi all'esame di Psicometria è quello di seguire attivamente le lezioni, assimilare i concetti via via che essi vengono presentati e verificare in autonomia le procedure presentate a lezione. Incoraggio gli studenti a farmi domande per chiarire ciò che non è stato capito appieno. Incoraggio gli studenti a utilizzare i forum attivi su Moodle e, soprattutto, a svolgere gli esercizi proposti su Moodle. I problemi forniti su Moodle rappresentano il livello di difficoltà richiesto per superare l'esame e consentono allo studente di comprendere se le competenze sviluppate fino a quel punto sono sufficienti rispetto alle richieste dell'esame.

La prima fase dello studio, che è sicuramente individuale, è quella in cui è necessario acquisire le conoscenze teoriche relative ai problemi che saranno presentati all'esame. La seconda fase di studio, che può essere facilitata da scambi con altri e da incontri di gruppo, porta ad acquisire la capacità di applicare le conoscenze: è necessario capire come usare un software (R) per applicare i concetti statistici alla specifica situazione del problema che si vuole risolvere. Le due fasi non sono però separate: il saper fare molto spesso ci aiuta a capire meglio.

Sviluppare un metodo di studio efficace

Avendo insegnato molte volte in passato un corso introduttivo di analisi dei dati ho notato nel corso degli anni che gli studenti con l'atteggiamento mentale che descriverò qui sotto generalmente ottengono ottimi risultati. Alcuni studenti sviluppano naturalmente questo approccio allo studio, ma altri hanno bisogno di fare uno sforzo per maturarlo. Fornisco qui sotto una breve descrizione del "metodo di studio" che, nella mia esperienza, è il più efficace per affrontare le richieste di questo insegnamento.

- Dedicate un tempo sufficiente al materiale di base, apparentemente facile; assicuratevi di averlo capito bene. Cercate le lacune nella vostra comprensione. Leggere presentazioni diverse dello stesso materiale (in libri o articoli diversi) può fornire nuove intuizioni.
- Gli errori che facciamo sono i nostri migliori maestri. Istintivamente cerchiamo di dimenticare subito i nostri errori. Ma il miglior modo di imparare è apprendere dagli errori che commettiamo. In questo senso, una soluzione corretta è meno utile di una soluzione sbagliata. Quando commettiamo un errore questo ci fornisce un'informazione importante: ci fa capire qual è il materiale di studio sul quale dobbiamo ritornare e che dobbiamo capire meglio.
- C'è ovviamente un aspetto "psicologico" nello studio. Quando un esercizio o problema ci sembra incomprensibile, la cosa migliore da fare è dire: "mi arrendo", "non ho idea di cosa fare!". Questo ci rilassa: ci siamo già arresi, quindi non abbiamo niente da perdere, non dobbiamo più preoccuparci. Ma non dobbiamo fermarci qui. Le cose "migliori" che faccio (se ci sono) le faccio quando non ho voglia di lavorare. Alle volte, quando c'è qualcosa che non so fare e non ho idea di come affontare, mi dico: "oggi non ho proprio voglia di fare fatica", non ho voglia di mettermi nello stato mentale per cui "in 10 minuti devo risolvere il problema perché dopo devo fare altre cose". Però ho voglia di *divertirmi* con quel problema e allora mi dedico a qualche aspetto "marginale" del problema, che so come affrontare, oppure considero l'aspetto più difficile del problema, quello che non so come risolvere, ma invece di cercare di risolverlo, guardo come altre persone hanno affrontato problemi simili, opppure lo stesso problema in un altro contesto. Non mi pongo l'obiettivo "risolvi il problema in 10 minuti", ma invece quello di farmi un'idea "generale" del problema, o quello di capire un caso più specifico e più semplice del problema. Senza nessuna pressione. Infatti, in quel momento ho deciso di non lavorare (ovvero, di non fare fatica). Va benissimo se "parto per la tangente", ovvero se mi metto a leggere del materiale che sembra avere poco a che fare con il problema centrale (le nostre intuizioni e la nostra curiosità solitamente ci indirizzano sulla strada giusta). Quando faccio così, molto spesso trovo la soluzione del problema che mi ero posto e, paradossalmente, la trovo in un tempo minore di quello che, in precedenza, avevo dedicato a "lavorare" al problema. Allora perché non faccio sempre così? C'è ovviamente l'aspetto dei "10 minuti" che non è sempre facile da dimenticare. Sotto pressione, possiamo solo agire in maniera automatica, ovvero possia-

mo solo applicare qualcosa che già sappiamo fare. Ma se dobbiamo imparare qualcosa di nuovo, la pressione è un impedimento.

- È utile farsi da soli delle domande sugli argomenti trattati, senza limitarsi a cercare di risolvere gli esercizi che vengono assegnati. Quando studio qualcosa mi viene in mente: "se questo è vero, allora deve succedere quest'altra cosa". Allora verifico se questo è vero, di solito con una simulazione. Se i risultati della simulazione sono quelli che mi aspetto, allora vuol dire che ho capito. Se i risultati sono diversi da quelli che mi aspettavo, allora mi rendo conto di non avere capito e ritorno indietro a studiare con più attenzione la teoria che pensavo di avere capito – e ovviamente mi rendo conto che c'era un aspetto che avevo frainteso. Questo tipo di verifica è qualcosa che dobbiamo fare da soli, in prima persona: nessun altro può fare questo al posto nostro.
- Non aspettatevi di capire tutto la prima volta che incontrate un argomento nuovo.[1] È utile farsi una nota mentalmente delle lacune nella vostra comprensione e tornare su di esse in seguito per carcare di colmarle. L'atteggiamento naturale, quando non capiamo i dettagli di qualcosa, è quello di pensare: "non importa, ho capito in maniera approssimativa questo punto, non devo preoccuparmi del resto". Ma in realtà non è vero: se la nostra comprensione è superficiale, quando il problema verrà presentato in una nuova forma, non riusciremo a risolverlo. Per cui i dubbi che ci vengono quando studiamo qualcosa sono il nostro alleato più prezioso: ci dicono esattamente quali sono gli aspetti che dobbiamo approfondire per potere migliorare la nostra preparazione.
- È utile sviluppare una visione d'insieme degli argomenti trattati, capire l'obiettivo generale che si vuole raggiungere e avere chiaro il contributo che i vari pezzi di informazione forniscono al raggiungimento di tale obiettivo. Questa organizzazione mentale del materiale di studio facilita la comprensione. È estremamente utile creare degli schemi di ciò che si sta studiando. Non aspettate che sia io a fornirvi un riepilogo di ciò che dovete imparare: sviluppate da soli tali schemi e tali riassunti.
- Tutti noi dobbiamo imparare l'arte di trovare le informazioni, non solo nel caso di questo insegnamento. Quando vi trovate di fronte a qualcosa che non capite, o ottenete un oscuro messaggio di errore da

[1] Ricordatevi inoltre che gli individui tendono a sottostimare la propria capacità di apprendere (Horn and Loewenstein, 2021).

un software, ricordatevi: “Google is your friend”!

Corrado Caudek
Marzo 2022

Parte I

Inferenza bayesiana

1

Flusso di lavoro bayesiano

La moderna statistica bayesiana viene per lo più eseguita utilizzando un linguaggio di programmazione probabilistico implementato su computer. Ciò ha cambiato radicalmente il modo in cui venivano eseguite le statistiche bayesiane anche fin pochi decenni fa. La complessità dei modelli che possiamo costruire è aumentata e la barriera delle competenze matematiche e computazionali che sono richieste è diminuita. Inoltre, il processo di modellazione iterativa è diventato, sotto molti aspetti, molto più facile da eseguire. Anche se formulare modelli statistici complessi è diventato più facile che mai, la statistica è un campo pieno di sottigliezze che non scompaiono magicamente utilizzando potenti metodi computazionali. Pertanto, avere una buona preparazione sugli aspetti teorici, specialmente quelli rilevanti nella pratica, è estremamente utile per applicare efficacemente i metodi statistici.

1.1 Modellizzazione bayesiana

L'analisi bayesiana corrisponde alla costruzione di un modello statistico che si può rappresentare con una quaterna

$$(\mathcal{Y}, p(y \mid \theta), p(\theta), \theta \in \Theta), \tag{1.1}$$

dove $\mathcal{Y}$ è l'insieme di tutti i possibili risultati ottenuti dall'esperimento casuale e $p(y \mid \theta)$ è una famiglia di leggi di probabilità, indicizzata dal parametro $\theta \in \Theta$, che descrive l'incertezza sull'esito dell'esperimento. Secondo l'approccio bayesiano, il parametro incognito θ è considerato una variabile casuale che segue la legge di probabilità $p(\theta)$. L'incertezza su θ è la sintesi delle opinioni e delle informazioni che si hanno sul parametro prima di avere osservato il risultato dell'esperimento e prende il nome di *distribuzione a priori*. La costruzione del modello statistico

passa attraverso la scelta di una densità $p(y \mid \theta)$ che rappresenta, in senso probabilistico, il fenomeno d'interesse, e attraverso la scelta di una distribuzione a priori $p(\theta)$. Le informazioni che si hanno a priori sul parametro di interesse θ, contenute in $p(\theta)$, vengono aggiornate attraverso quelle provenienti dal campione osservato $y = (y_1, \dots, y_n)$ contenute nella funzione $p(y \mid \theta)$, che, osservata come funzione di θ per y, prende il nome di *funzione di verosimiglianza*. L'aggiornamento delle informazioni avviene attraverso la formula di Bayes

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int_\Theta p(y \mid \theta)p(\theta)\, \mathrm{d}\theta} \quad \theta \in \Theta, \tag{1.2}$$

in cui $p(\theta \mid y)$ prende il nome di *distribuzione a posteriori*.

Il denominatore del Teorema di Bayes (1.2), che costituisce la costante di normalizzazione, è la densità marginale dei dati (o verosimiglianza marginale). In ambito bayesiano la distribuzione a posteriori viene utilizzata per calcolare le principali quantità di interesse dell'inferenza, ad esempio la media a posteriori di θ.

Possiamo descrivere la modellazione bayesiana distinguendo tre passaggi (Martin et al., 2022).

1. Dati alcuni dati e alcune ipotesi su come questi dati potrebbero essere stati generati, progettiamo un modello combinando e trasformando variabili casuali.
2. Usiamo il teorema di Bayes per condizionare i nostri modelli ai dati disponibili. Chiamiamo questo processo "inferenza" e come risultato otteniamo una distribuzione a posteriori.
3. Critichiamo il modello verificando se il modello abbia senso utilizzando criteri diversi, inclusi i dati e la nostra conoscenza del dominio. Poiché generalmente siamo incerti sui modelli, a volte confrontiamo modelli diversi.

Questi tre passaggi vengono eseguiti in modo iterativo e danno luogo a quello che è chiamato "flusso di lavoro bayesiano" (*bayesian workflow*).

1.1.1 Notazione

Per fissare la notazione, nel seguito y rappresenterà i dati e θ rappresenterà i parametri incogniti di un modello statistico. Sia y che θ ven-

gono concepiti come variabili casuali. Con x vengono invece denotate le quantità note, come ad esempio i predittori del modello lineare. Per rappresentare in un modo conciso i modelli probabilistici viene usata una notazione particolare. Ad esempio, invece di scrivere $p(\theta) = \text{Beta}(1, 1)$ scriviamo $\theta \sim \text{Beta}(1, 1)$. Il simbolo "$\sim$" viene spesso letto "è distribuito come". Possiamo anche pensare che significhi che θ costituisce un campione casuale estratto dalla distribuzione Beta(1, 1). Allo stesso modo, ad esempio, la verosimiglianza del modello binomiale può essere scritta come $y \sim \text{Bin}(n, \theta)$.

1.2 Distribuzioni a priori

Quando adottiamo un approccio bayesiano, i parametri della distribuzione di riferimento non venono considerati come delle costanti incognite ma bensì vengono trattati come variabili casuali; di conseguenza, i parametri assumono una particolare distribuzione che nelle statistica bayesiana viene definita "a priori". I parametri (o il parametro), che possiamo indicare con θ, possono assumere delle distribuzioni a priori differenti: a seconda delle informazioni disponibili bisogna selezionare una distribuzione di θ in modo tale che venga assegnata una probabilità maggiore a quei valori che si ritengono più plausibili per θ. Idealmente, le credenze a priori che portano alla specificazione di una distribuzione a priori dovrebbero essere supportate da una qualche motivazione, come ad esempio i risultati di ricerche precedenti.

1.2.1 Tipologie di distribuzioni a priori

Possiamo distinguere tra diverse distribuzioni a priori in base a quanto fortemente impegnano il ricercatore a ritenere come plausibile un particolare intervallo di valori dei parametri. Il caso più estremo è quello che rivela una totale assenza di conoscenze a priori, il che conduce alle *distribuzioni a priori non informative*, ovvero quelle che assegnano lo stesso livello di credibilità a tutti i valori dei parametri. Le distribuzioni a priori informative, d'altra parte, possono essere *debolmente informative* o *fortemente informative*, a seconda della forza della credenza che esprimono. Il caso più estremo di credenza a priori è quello che riassume il punto di vista del ricercatore nei termini di un *unico valore* del parametro, il che assegna tutta la probabilità (massa o densità) ad di un

singolo valore del parametro. Poiché questa non è più una distribuzione di probabilità, sebbene ne soddisfi la definizione, in questo caso si parla di una *distribuzione a priori degenerata*. La figura seguente motra alcuni esempi di distribuzioni a priori per il modello Binomiale:

- distribuzione *non informativa*: $\theta_c \sim \text{Beta}(1, 1)$;
- distribuzione *debolmente informativa*: $\theta_c \sim \text{Beta}(5, 2)$;
- distribuzione *fortemente informativa*: $\theta_c \sim \text{Beta}(50, 20)$;
- *valore puntuale*: $\theta_c \sim \text{Beta}(\alpha, \beta)$ con $\alpha, \beta \to \infty$ e $\frac{\alpha}{\beta} = \frac{5}{2}$.

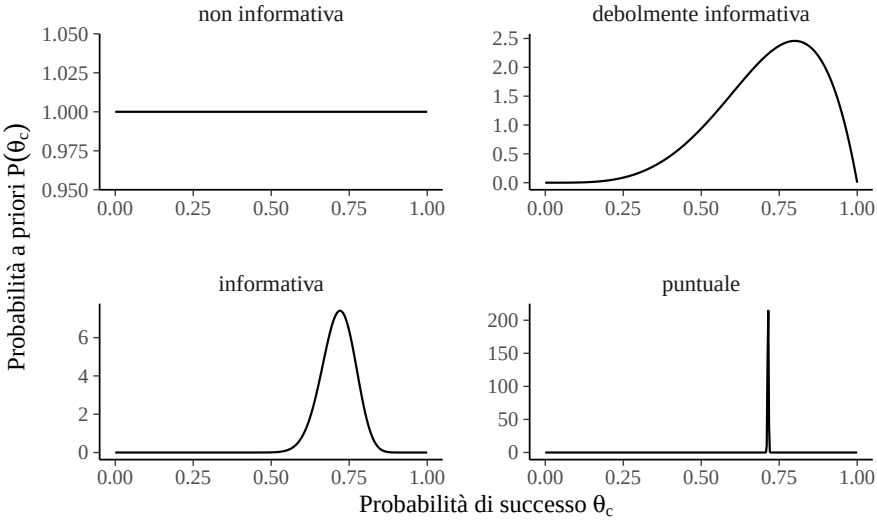


Figura 1.1: *Esempi di distribuzioni a priori per il parametro θ_c nel Modello Binomiale.*

1.2.2 Selezione della distribuzione a priori

La selezione delle distribuzioni a priori è stata spesso vista come una delle scelte più importanti che un ricercatore fa quando implementa un modello bayesiano in quanto può avere un impatto sostanziale sui risultati finali. La soggettività delle distribuzioni a priori è evidenziata dai critici come un potenziale svantaggio dei metodi bayesiani. A questa critica, van de Schoot et al. (2021) rispondono dicendo che, al di là della scelta delle distribuzioni a priori, ci sono molti elementi del processo di inferenza statistica che sono soggettivi, ovvero la scelta del modello statistico e le ipotesi sulla distribuzione degli errori. In secondo luogo, van de Schoot et al. (2021) notano come le distribuzioni a priori svol-

gono due importanti ruoli statistici: quello della "regolarizzazione della stima", ovvero, il processo che porta ad indebolire l'influenza indebita di osservazioni estreme, e quello del miglioramento dell'efficienza della stima, ovvero, la facilitazione dei processi di calcolo numerico di stima della distribuzione a posteriori. L'effetto della distribuzione a priori sulla distribuzione a posteriori verrà discusso in dettaglio nel Capitolo **??**.

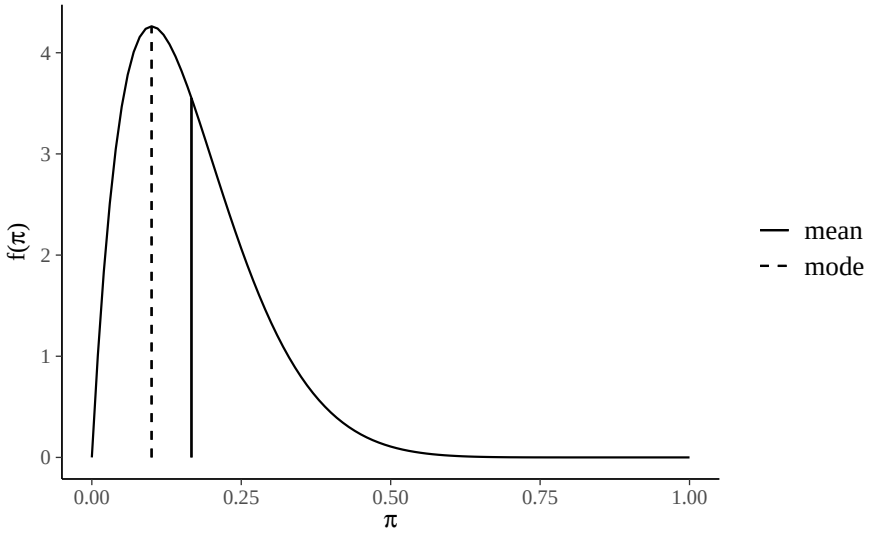
1.2.3 Un'applicazione empirica

Per introdurre la modelizzazione bayesiana useremo qui i dati riportati da Zetsche et al. (2019) (si veda l'appendice **??**). Tali dati corrispondono a 23 "successi" in 30 prove e possono dunque essere considerati la manifestazione di una variabile casuale Bernoulliana.

Se non abbiamo alcuna informazione a priori su θ (ovvero, la probabilità che l'aspettativa dell'umore futuro del partecipante sia distorta negativamente), potremmo pensare di usare una distribuzione a priori uniforme, ovvero una Beta di parametri $\alpha = 1$ e $\beta = 1$. Una tale scelta, tuttavia, è sconsigliata in quanto è più vantaggioso usare una distribuzione debolmente informativa, come ad esempio Beta$(2, 2)$, che ha come scopo la regolarizzazione, cioè quello di mantenere le inferenze in un intervallo ragionevole. Qui useremo una Beta$(2, 10)$.

$$p(\theta) = \frac{\Gamma(12)}{\Gamma(2)\Gamma(10)}\theta^{2-1}(1-\theta)^{10-1}.$$

```
bayesrules::plot_beta(alpha = 2, beta = 10, mean = TRUE, mode = TRUE)
```



La Beta$(2, 10)$ esprime la credenza che θ assume valori < 0.5, con il valore più plausibile pari a circa 0.1. Questo è assolutamente implausibile per il caso dell'esempio in discussione: la Beta$(2, 10)$ verrà usata solo per scopi didattici, ovvero, per esplorare le conseguenze di tale scelta sulla distribuzione a posteriori.

1.3 La funzione di verosimiglianza

Iniziamo con una definizione.

Definizione 1.1. La *funzione di verosimiglianza* $\mathcal{L}(\theta \mid y) = f(y \mid \theta), \theta \in \Theta$, è la funzione di massa o di densità di probabilità dei dati y vista come una funzione del parametro sconosciuto (o dei parametri sconosciuti) θ.

Detto in altre parole, le funzioni di verosimiglianza e di (massa o densità di) probabilità sono formalmente identiche, ma è completamente diversa la loro interpretazione. Nel caso della funzione di massa o di densità di probabilità la distribuzione del vettore casuale delle osservazioni campionarie y dipende dai valori assunti dal parametro (o dai parametri) θ; nel caso della la funzione di verosimiglianza la credibilità assegnata a ciascun possibile valore θ viene determinata avendo acquisita l'informazione campionaria y che rappresenta l'elemento condizionante. In altri termini, la

funzione di verosimiglianza descrive in termini relativi il sostegno empirico che $\theta \in \Theta$ riceve da y. Infatti, la funzione di verosimiglianza assume forme diverse al variare di y. Possiamo dunque pensare alla funzione di verosimiglianza come alla risposta alla seguente domanda: avendo osservato i dati y, quanto risultano (relativamente) credibili i diversi valori del parametro θ? In termini più formali possiamo dire: sulla base dei dati, $\theta_1 \in \Theta$ risulta più credibile di $\theta_2 \in \Theta$ quale indice del modello probabilistico generatore dei dati se $\mathcal{L}(\theta_1) > \mathcal{L}(\theta_1)$.

Notiamo un punto importante: la funzione $\mathcal{L}(\theta \mid y)$ non è una funzione di densità. Infatti, essa non racchiude un'area unitaria.

1.3.1 Notazione

Seguendo una pratica comune, in questa dispensa spesso useremo la notazione $p(\cdot)$ per rappresentare due quantità differenti, ovvero la funzione di verosimiglianza e la distribuzione a priori. Questo piccolo abuso di notazione riflette il seguente punto di vista: anche se la verosimiglianza non è una funzione di densità di probabilità, noi non vogliamo stressare questo aspetto, ma vogliamo piuttosto pensare alla verosimiglianza e alla distribuzione a priori come a due elementi che sono egualmente necessari per calcolare la distribuzione a posteriori. In altri termini, per così dire, questa notazione assegna lo stesso status epistemologico alle due diverse quantità che si trovano al numeratore della regola di Bayes.

1.3.2 La log-verosimiglianza

Dal punto di vista pratico risulta più conveniente utilizzare, al posto della funzione di verosimiglianza, il suo logaritmo naturale, ovvero la funzione di log-verosimiglianza:

$$\ell(\theta) = \log \mathcal{L}(\theta). \tag{1.3}$$

Poiché il logaritmo è una funzione strettamente crescente (usualmente si considera il logaritmo naturale), allora $\mathcal{L}(\theta)$ e $\ell(\theta)$ assumono il massimo (o i punti di massimo) in corrispondenza degli stessi valori di θ (per un approfondimento, si veda l'Appendice **??**):

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta) = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta).$$

Per le proprietà del logaritmo, si ha

$$\ell(\theta) = \log \left(\prod_{i=1}^{n} f(y \mid \theta) \right) = \sum_{i=1}^{n} \log f(y \mid \theta). \tag{1.4}$$

Si noti che non è necessario lavorare con i logaritmi, ma è fortemente consigliato. Il motivo è che i valori della verosimiglianza, in cui si moltiplicano valori di probabilità molto piccoli, possono diventare estremamente piccoli – qualcosa come 10^{-34}. In tali circostanze, non è sorprendente che i programmi dei computer mostrino problemi di arrotondamento numerico. Le trasformazioni logaritmiche risolvono questo problema.

1.3.3 Un'applicazione empirica

Se i dati di Zetsche et al. (2019) possono essere riassunti da una proporzione allora è sensato adottare un modello probabilistico binomiale quale meccanismo generatore dei dati:

$$y \sim \text{Bin}(n, \theta), \tag{1.5}$$

laddove θ è la probabiltà che una prova Bernoulliana assuma il valore 1 e n corrisponde al numero di prove Bernoulliane. Questo modello assume che le prove Bernoulliane y_i che costituiscono il campione y siano tra loro indipendenti e che ciascuna abbia la stessa probabilità $\theta \in [0, 1]$ di essere un "successo" (valore 1). In altre parole, il modello generatore dei dati avrà una funzione di massa di probabilità

$$p(y \mid \theta) \; = \; \text{Bin}(y \mid n, \theta).$$

Nei capitoli precedenti è stato mostrato come, sulla base del modello binomiale, sia possibile assegnare una probabilità a ciascun possibile valore $y \in \{0, 1, \dots, n\}$ *assumendo noto il valore del parametro* θ. Ma ora abbiamo il problema inverso, ovvero quello di fare inferenza su θ alla luce dei dati campionari y. In altre parole, riteniamo di conoscere il modello probabilistico che ha generato i dati, ma di tale modello non conosciamo i parametri: vogliamo dunque ottenere informazioni su θ avendo osservato i dati y.

Per i dati di Zetsche et al. (2019) la funzione di verosimiglianza corrisponde alla funzione binomiale di parametro $\theta \in [0, 1]$ sconosciuto. Abbiamo osservato un "successo" 23 volte in 30 "prove", dunque, $y = 23$ e $n = 30$. La funzione di verosimiglianza diventa

$$\mathcal{L}(\theta \mid y) = \frac{(23+7)!}{23!7!}\theta^{23} + (1-\theta)^7. \tag{1.6}$$

Per costruire la funzione di verosimiglianza dobbiamo applicare la (1.6) tante volte, cambiando ogni volta il valore θ ma *tenendo sempre costante il valore dei dati*. Per esempio, se poniamo $\theta = 0.1$

$$\mathcal{L}(\theta \mid y) = \frac{(23+7)!}{23!7!}0.1^{23} + (1-0.1)^7$$

otteniamo

```
dbinom(23, 30, 0.1)
#> [1] 9.737e-18
```

Se poniamo $\theta = 0.2$

$$\mathcal{L}(\theta \mid y) = \frac{(23+7)!}{23!7!}0.2^{23} + (1-0.2)^7$$

otteniamo

```
dbinom(23, 30, 0.2)
#> [1] 3.581e-11
```

e così via. La figura 1.2 — costruita utilizzando 100 valori equispaziati $\theta \in [0, 1]$ — fornisce una rappresentazione grafica della funzione di verosimiglianza.

```
n <- 30
y <- 23
theta <- seq(0, 1, length.out = 100)
like <- choose(n, y) * theta^y * (1 - theta)^(n - y)
tibble(theta, like) %>%
  ggplot(aes(x = theta, y = like)) +
  geom_line() +
  labs(
    y = expression(L(theta)),
    x = expression("Valori possibili di" ~ theta)
  )
```

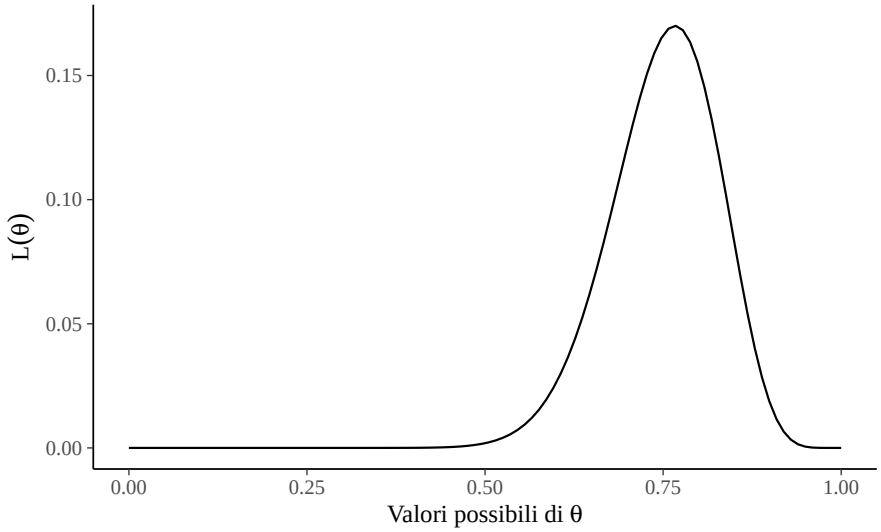


Figura 1.2: *Funzione di verosimiglianza nel caso di 23 successi in 30 prove.*

Come possiamo interpretare la curva che abbiamo ottenuto? Per alcuni valori θ la funzione di verosimiglianza assume valori piccoli; per altri valori θ la funzione di verosimiglianza assume valori più grandi. Questi ultimi sono i valori di θ più credibili e il valore 23/30 (la moda della funzione di verosimiglianza) è il valore più credibile di tutti.

1.4 La verosimiglianza marginale

Per il calcolo di $p(\theta \mid y)$ è necessario dividere il prodotto tra la distribuzione a priori e la verosimiglianza per una costante di normalizzazione. Tale costante di normalizzazione, detta *verosimiglianza marginale*, ha lo scopo di fare in modo che $p(\theta \mid y)$ abbia area unitaria.

Si noti che, nel caso di variabili continue, la verosimiglianza marginale è espressa nei termini di un integrale. Tranne in pochi casi particolari, tale integrale non ha una soluzione analitica. Per questa ragione, l'inferenza bayesiana procede calcolando una approssimazione della distribuzione a posteriori mediante metodi numerici.

1.4.1 Un'applicazione empirica

Consideriamo nuovamente i dati di Zetsche et al. (2019). Supponiamo che nel numeratore bayesiano la verosimiglianza sia moltiplicata per una distribuzione uniforme, ovvero Beta$(1, 1)$. In tali circostanze, il prodotto si riduce alla funzione di verosimiglianza. Per i dati di Zetsche et al. (2019), dunque, la costante di normalizzazione si ottiene marginalizzando la funzione di verosimiglianza $p(y = 23, n = 30 \mid \theta)$ sopra θ, ovvero risolvendo l'integrale:

$$p(y = 23, n = 30) = \int_0^1 \binom{30}{23} \theta^{23}(1 - \theta)^7 \, \mathrm{d}\theta. \tag{1.7}$$

Una soluzione numerica si trova facilmente usando R:

```
like_bin <- function(theta) {
  choose(30, 23) * theta^23 * (1 - theta)^7
}
integrate(like_bin, lower = 0, upper = 1)$value
#> [1] 0.03226
```

La derivazione analitica è fornita nell'Appendice **??**.

1.5 Distribuzione a posteriori

La distribuzione a postreriori si trova applicando il teorema di Bayes:

$$\text{probabilità a posteriori} = \frac{\text{probabilità a priori} \cdot \text{verosimiglianza}}{\text{costante di normalizzazione}}$$

Una volta trovata la distribuzione a posteriori, possiamo usarla per derivare altre quantità di interesse. Questo viene generalmente ottenuto calcolando il valore atteso:

$$J = \int f(\theta) p(\theta \mid y) \, \mathrm{d}y$$

Se $f(\cdot)$ è la funzione identità, ad esempio, J risulta essere la media di θ:

$$\bar{\theta} = \int_{\Theta} \theta p(\theta \mid y) \, \mathrm{d}\theta.$$

Ripeto qui quanto detto sopra: le quantità di interesse della statistica bayesiana(costante di normalizzazione, valore atteso della distribuzione a posteriori, ecc.) contengono integrali che risultano, nella maggior parte dei casi, impossibili da risolvere analiticamente. Per questo motivo, si ricorre a metodi di stima numerici, in particolare a quei metodi Monte Carlo basati sulle proprietà delle catene di Markov (MCMC). Questo argomento verrà discusso nel Capitolo 2.

1.6 Distribuzione predittiva a priori

La distribuzione a posteriori è l'oggetto centrale nella statistica bayesiana, ma non è l'unico. Oltre a fare inferenze sui valori dei parametri, potremmo voler fare inferenze sui dati. Questo può essere fatto calcolando la *distribuzione predittiva a priori*:

$$p(y^*) = \int_{\Theta} p(y^* \mid \theta) p(\theta) \, \mathrm{d}\theta. \tag{1.8}$$

La (1.8) descrive la distribuzione prevista dei dati in base al modello (che include la distribuzione a priori e la verosimiglianza), ovvero descrive i dati y^* che ci aspettiamo di osservare, dato il modello, prima di avere osservato i dati del campione.

È possibile utilizzare campioni dalla distribuzione predittiva a priori per valutare e calibrare i modelli utilizzando le nostre conoscenze dominio-specifiche. Ad esempio, ci possiamo chiedere: "È sensato che un modello dell'altezza umana preveda che un essere umano sia alto -1.5 metri?". Già prima di misurare una singola persona, possiamo renderci conto dell'assurdità di questa domanda. Se la distribuzione prevista dei dati consente domande di questo tipo (ovvero, prevede di osservare dati che risultano insensati alla luce delle nostre conoscenze dominio-specifiche), è chiaro che il modello deve essere riformulato.

1.7 Distribuzione predittiva a posteriori

Un'altra quantità utile da calcolare è la distribuzione predittiva a posteriori:

$$p(\tilde{y} \mid y) = \int_{\Theta} p(\tilde{y} \mid \theta) p(\theta \mid y) \, \mathrm{d}\theta. \tag{1.9}$$

Questa è la distribuzione dei dati attesi futuri $\tilde{y}$ alla luce della distribuzione a posteriori $p(\theta \mid y)$, che a sua volta è una conseguenza del modello adottato (distribuzione a priori e verosimiglianza) e dei dati osservati. In altre parole, questi sono i dati che il modello si aspetta dopo aver osservato i dati de campione. Dalla (1.9) possiamo vedere che le previsioni sui dati attesi futuri sono calcolate integrando (o marginalizzando) sulla distribuzione a posteriori dei parametri. Di conseguenza, le previsioni calcolate in questo modo incorporano l'incertezza relativa alla stima dei parametri del modello.

Commenti e considerazioni finali

Questo Capitolo ha brevemente passato in rassegna i concetti di base dell'inferenza statistica bayesiana. In base all'approccio bayesiano, invece di dire che il parametro di interesse di un modello statistico ha un valore vero ma sconosciuto, diciamo che, prima di eseguire l'esperimento, è possibile assegnare una distribuzione di probabilità, che chiamano stato di credenza, a quello che è il vero valore del parametro. Questa distribuzione a priori può essere nota (per esempio, sappiamo che la distribuzione dei punteggi del QI è normale con media 100 e deviazione standard 15) o può essere del tutto arbitraria. L'inferenza bayesiana procede poi nel modo seguente: si raccolgono alcuni dati e si calcola la probabilità dei possibili valori del parametro alla luce dei dati osservati e delle credenze a priori. Questa nuova distribuzione di probabilità è chiamata "distribuzione a posteriori" e riassume l'incertezza dell'inferenza.

2

Approssimazione della distribuzione a posteriori

In generale, in un problema bayesiano i dati y provengono da una densità $p(y \mid \theta)$ e al parametro θ viene assegnata una densità a priori $p(\theta)$. Dopo avere osservato i dati $Y = y$, la funzione di verosimiglianza è uguale a $\mathcal{L}(\theta) = p(y \mid \theta)$ e la densità a posteriori diventa

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int p(y \mid \theta)p(\theta) \, \mathrm{d}\theta}.$$

Se vogliamo trovare la distribuzione a posteriori con metodi analitici è necessario ricorrere all'impiego di distribuzioni a priori coniugate, come nello schema beta-binomiale. Per quanto "semplice" in termini formali, la scelta di distribuzioni a priori coniugate limita di molto le possibili scelte del ricercatore. Inoltre, non è sempre sensato, dal punto di vista teorico, utilizzare tali distribuzioni per la stima dei parametri di interesse. Il mancato ricorso all'impiego delle distribuzioni a priori coniugate richiede necessariamente il computo dell'espressione a denominatore della formula di Bayes che solo in rare occasioni può essere ottenuta per via analitica. In altre parole, è possibile ottenere analiticamenre la distribuzione a posteriori solo per alcune specifiche combinazioni di distribuzioni a priori e verosimiglianza, il che limita considerevolmente la flessibilità della modellizzazione. Inoltre, i sommari della distribuzione a posteriori sono espressi come rapporto di integrali. Ad esempio, la media a posteriori di θ è data da

$$\mathbb{E}(\theta \mid y) = \frac{\int \theta p(y \mid \theta)p(\theta) \, \mathrm{d}\theta}{\int p(y \mid \theta)p(\theta) \, \mathrm{d}\theta}.$$

Il calcolo del valore atteso a posteriori richiede dunque il computo di due integrali, quello a denominatore e quello a numeratore dell'espressione, ciascuno dei quali non esprimibile in forma chiusa. Per questa ragione, la

strada principale che viene seguita nella modellistica bayesiana è quella che porta a determinare la distribuzione a posteriori non per via analitica, ma bensì mediante metodi numerici. La simulazione fornisce dunque la strategia generale del calcolo bayesiano. A questo fine vengono principalmente usati i metodi di campionamento Monte Carlo basati su Catena di Markov (MCMC). Tali metodi costituiscono una potente e praticabile alternativa per la costruzione della distribuzione a posteriori per modelli complessi e consentono di decidere quali distribuzioni a priori e quali distribuzioni di verosimiglianza usare sulla base di considerazioni teoriche soltanto, senza dovere preoccuparsi di altri vincoli.

Dato che è basata su metodi computazionalmente intensivi, la stima numerica della funzione a posteriori può essere svolta soltanto mediante software. In anni recenti i metodi Bayesiani di analisi dei dati sono diventati sempre più popolari proprio perché la potenza di calcolo necessaria per svolgere tali calcoli è ora alla portata di tutti. Questo non era vero solo pochi decenni fa.

In questo Capitolo verranno presentati due metodi di simulazione iterativa[1] che consentono di generare dalle distribuzioni a posteriori campioni dei parametri del modello:

- **metodi basati su griglia:** dove, sebbene non sia disponibile alcuna formula algebrica in forma chiusa, le proprietà della distribuzione a posteriori possono essere calcolate con una precisione arbitraria;
- **metodi Monte Carlo:** dove, utilizzando appropriate funzioni di numeri casuali, viene generato un ampio campione di casi della variabile casuale per poi stimare empiricamente la proprietà di interesse in base al campione così otttenuto.

2.1 Metodo basato su griglia

Il metodo basato su griglia (*grid-based*) è un metodo numerico esatto basato su una griglia di punti uniformemente spaziati. Anche se la maggior parte dei parametri è continua (ovvero, in linea di principio ciascun parametro può assumere un numero infinito di valori), possiamo ottenere un'eccellente approssimazione della distribuzione a posteriori conside-

[1] Si veda anche l'Appendice **??**.

rando solo una griglia finita di valori dei parametri. In un tale metodo, la densità di probabilità a posteriori può dunque essere approssimata tramite le densità di probabilità calcolate in ciascuna cella della griglia.

Il metodo basato su griglia si sviluppa in quattro fasi:

- fissare una griglia discreta di possibili valori θ;
- valutare la distribuzione a priori $p(\theta)$ e la funzione di verosimiglianza $p(y \mid \theta)$ in corrispondenza di ciascun valore θ della griglia;
- ottenere un'approssimazione discreta della densità a posteriori:
  - per ciascun valore θ della griglia, calcolare il prodotto $p(\theta)p(y \mid \theta)$;
  - normalizzare i prodotti così ottenuti in modo tale che la loro somma sia 1;
- selezionare N valori casuali della griglia in modo tale da ottenere un campione casuale delle densità a posteriori normalizzate.

Possiamo migliorare l'approssimazione aumentando il numero di punti della griglia. Infatti utilizzando un numero infinito di punti si otterrebbe la descrizione esatta della distribuzione a posteriori, dovendo però pagare il costo dell'utilizzo di infinite risorse di calcolo. Il limite maggiore dell'approccio basato su griglia è che, al crescere della dimensionalità N dello spazio dei parametri, i punti della griglia necessari per avere una buona stima crescerebbero esponenzialmente con N, rendendo questo metodo inattuabile.

2.1.1 Modello Beta-Binomiale

Per fare un esempio, consideriamo lo schema beta-binomiale di cui conosciamo la soluzione esatta. Utilizziamo nuovamente i dati di Zetsche et al. (2019): 23 "successi" in 30 prove Bernoulliane indipendenti.[2] Imponiamo alla distribuzione a priori su θ (probabilità di successo in una singola prova, laddove per "successo" si intende una aspettativa distorta negativamente dell'umore futuro) una $\text{Beta}(2, 10)$ per descrivere la nostra incertezza sul parametro prima di avere osservato i dati. Dunque, il modello diventa:

$$
\begin{aligned}
Y \mid \theta &\sim \text{Bin}(n = 30, \theta), \\
\theta &\sim \text{Beta}(2, 10).
\end{aligned}
$$

[2]Nel presente esempio useremo lo stesso codice R utilizzato da Johnson et al. (2022).

In queste circostanze, l'aggiornamento bayesiano produce una distribuzione a posteriori Beta di parametri 25 ($y + \alpha = 23 + 2$) e 17 ($n - y + \beta = 30$ - 23 + 10):

$$\theta \mid (y = 23) \sim \text{Beta}(25, 17).$$

Per approssimare la distribuzione a posteriori, fissiamo una griglia di $n = 11$ valori equispaziati: $\theta \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$:

```
grid_data <- tibble(
  theta_grid = seq(from = 0, to = 1, length.out = 11)
)
grid_data
#> # A tibble: 11 x 1
#>   theta_grid
#>        <dbl>
#> 1        0
#> 2        0.1
#> 3        0.2
#> 4        0.3
#> 5        0.4
#> 6        0.5
#> 7        0.6
#> 8        0.7
#> # ... with 3 more rows
```

In corrispondenza di ciascun valore della griglia, valutiamo la distribuzione a priori Beta$(2, 10)$ e la verosimiglianza Bin$(y = 23, n = 30)$.

```
grid_data <- grid_data %>%
  mutate(
    prior = dbeta(theta_grid, 2, 10),
    likelihood = dbinom(23, 30, theta_grid)
  )
```

In ciascuna cella della griglia calcoliamo poi il prodotto della verosimiglianza e della distribuzione a priori. Troviamo così un'approssimazione discreta e non normalizzata della distribuzione a posterio-

ri (`unnormalized`). Normalizziamo questa approssimazione dividendo ciascun valore `unnormalized` per la somma di tutti i valori del vettore:

```
grid_data <- grid_data %>%
  mutate(
    unnormalized = likelihood * prior,
    posterior = unnormalized / sum(unnormalized)
  )
```

Verifichiamo:

```
grid_data %>%
  summarize(
    sum(unnormalized),
    sum(posterior)
  )
#> # A tibble: 1 x 2
#>   `sum(unnormalized)` `sum(posterior)`
#>                 <dbl>            <dbl>
#> 1            0.000869                1
```

Abbiamo dunque ottenuto la seguente distribuzione a posteriori discretizzata $p(\theta \mid y)$:

```
round(grid_data, 2)
#> # A tibble: 11 x 5
#>   theta_grid prior likelihood unnormalized posterior
#>        <dbl> <dbl>      <dbl>        <dbl>     <dbl>
#> 1        0    0          0               0      0
#> 2        0.1  4.26       0               0      0
#> 3        0.2  2.95       0               0      0
#> 4        0.3  1.33       0               0      0
#> 5        0.4  0.44       0               0      0.02
#> 6        0.5  0.11       0               0      0.23
#> 7        0.6  0.02       0.03            0      0.52
#> 8        0.7  0          0.12            0      0.21
#> # ... with 3 more rows
```

La figura 2.1 mostra un grafico della distribuzione a posteriori discretizzata così ottenuta:

```
grid_data %>%
  ggplot(
    aes(x = theta_grid, y = posterior)
  ) +
  geom_point() +
  geom_segment(
    aes(
      x = theta_grid,
      xend = theta_grid,
      y = 0,
      yend = posterior
    )
  )
```

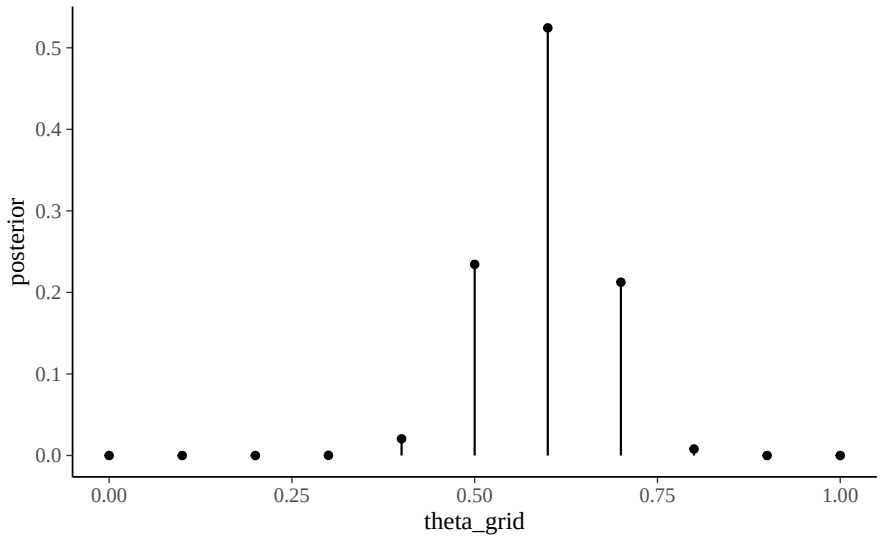


Figura 2.1: *Distribuzione a posteriori discretizzata ottenuta con il metodo grid-based per $y = 23$ successi in 30 prove Bernoulliane, con distribuzione a priori $Beta(2, 10)$. È stata utilizzata una griglia di solo $n = 11$ punti.*

L'ultimo passo della simulazione è il campionamento dalla distribuzione a posteriori discretizzata:

```
set.seed(84735)
post_sample <- sample_n(
  grid_data,
  size = 1e5,
  weight = posterior,
  replace = TRUE
)
```

La figura 2.2 mostra che, con una griglia così sparsa abbiamo ottenuto una versione approssimata della vera distribuzione a posteriori (all'istogramma è stata sovrapposta l'esatta distribuzione a posteriori Beta$(25, 17)$).

```
ggplot(post_sample, aes(x = theta_grid)) +
  geom_histogram(aes(y = ..density..), color = "white") +
  stat_function(fun = dbeta, args = list(25, 17)) +
  lims(x = c(0, 1))
```

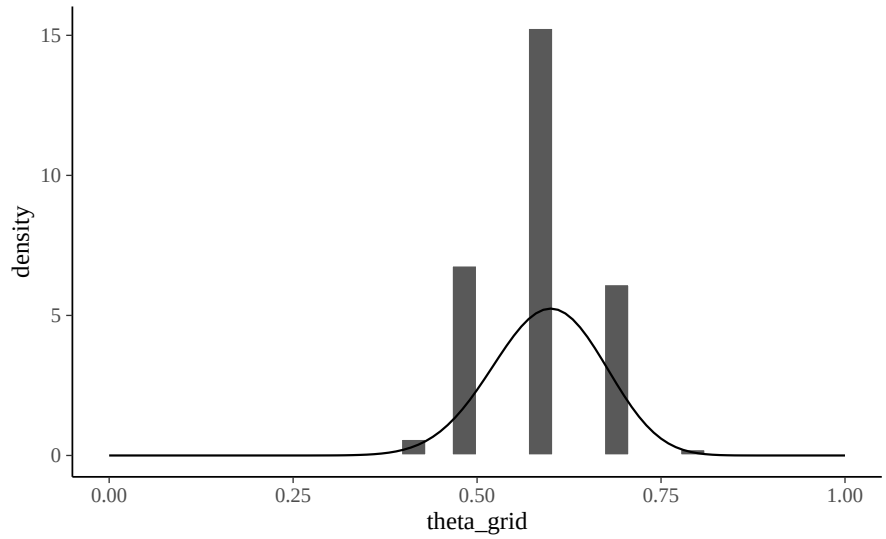


Figura 2.2: *Campionamento dalla distribuzione a posteriori discretizzata ottenuta con il metodo grid-based per $y = 23$ successi in 30 prove Bernoulliane, con distribuzione a priori Beta$(2, 10)$. È stata utilizzata una griglia di solo $n = 11$ punti.*

Possiamo ottenere un risultato migliore con una griglia più fine, come indicato nella figura 2.3:

```
grid_data <- tibble(
  theta_grid = seq(from = 0, to = 1, length.out = 100)
)
grid_data <- grid_data %>%
  mutate(
    prior = dbeta(theta_grid, 2, 10),
    likelihood = dbinom(23, 30, theta_grid)
  )
grid_data <- grid_data %>%
  mutate(
    unnormalized = likelihood * prior,
    posterior = unnormalized / sum(unnormalized)
  )
grid_data %>%
  ggplot(
    aes(x = theta_grid, y = posterior)
  ) +
  geom_point() +
  geom_segment(
    aes(
      x = theta_grid,
      xend = theta_grid,
      y = 0,
      yend = posterior
    )
  )
```

Campioniamo ora 10000 punti:

```
# Set the seed
set.seed(84735)
post_sample <- sample_n(
  grid_data,
  size = 1e4,
  weight = posterior,
```

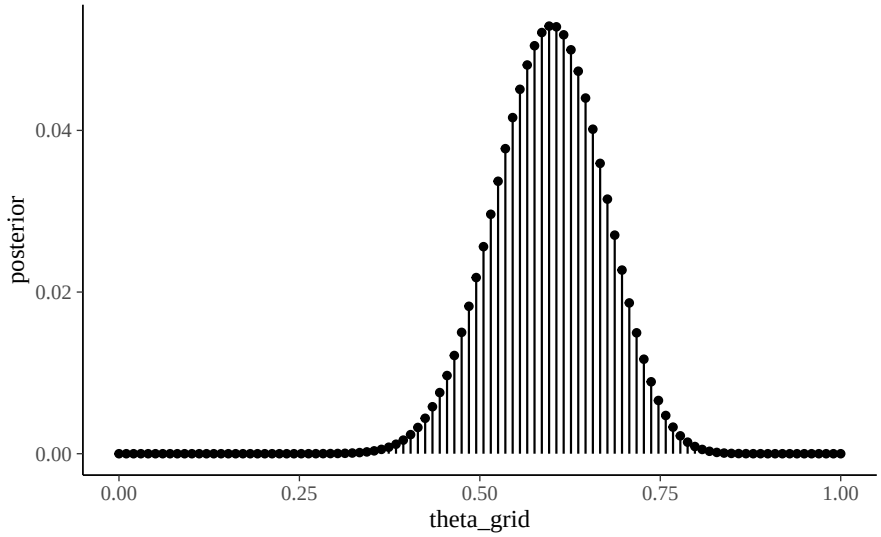


Figura 2.3: *Distribuzione a posteriori discretizzata ottenuta con il metodo grid-based per $y = 23$ successi in 30 prove Bernoulliane, con distribuzione a priori Beta$(2, 10)$. È stata utilizzata una griglia di $n = 100$ punti.*

```
  replace = TRUE
)
```

Con il campionamento dalla distribuzione a posteriori discretizzata costruita mediante una griglia più densa ($n = 100$) otteniamo un risultato soddisfacente (figura 2.4): ora la distribuzione dei valori prodotti dalla simulazione approssima molto bene la corretta distribuzione a posteriori $p(\theta \mid y) = \text{Beta}(25, 17)$.

```
post_sample %>%
  ggplot(aes(x = theta_grid)) +
  geom_histogram(
    aes(y = ..density..),
    color = "white",
    bins = 50
  ) +
  stat_function(fun = dbeta, args = list(25, 17)) +
  lims(x = c(0, 1))
```

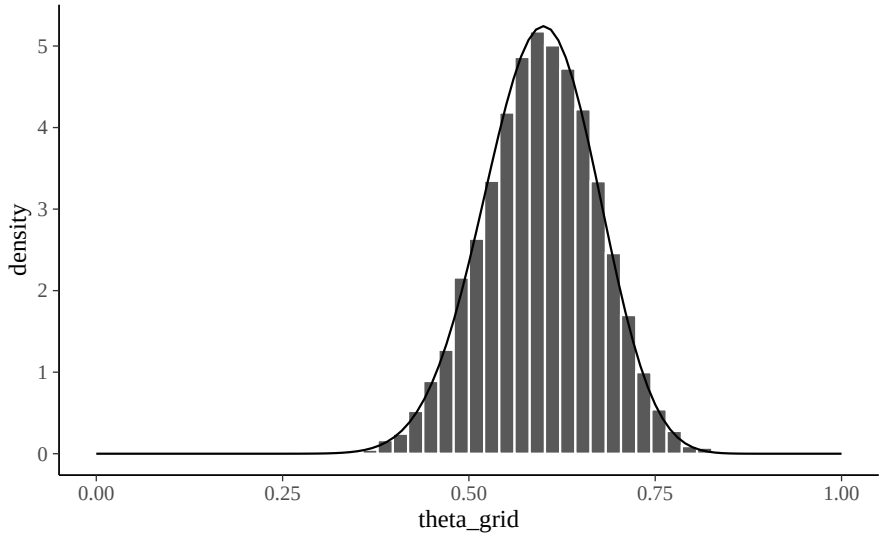


Figura 2.4: *Campionamento dalla distribuzione a posteriori discretizzata ottenuta con il metodo grid-based per $y = 23$ successi in 30 prove Bernoulliane, con distribuzione a priori $Beta(2, 10)$. È stata utilizzata una griglia di $n = 100$ punti. All'istogramma è stata sovrapposta la corretta distribuzione a posteriori, ovvero la densità $Beta(25, 17)$.*

In conclusione, il metodo basato su griglia è molto intuitivo e non richiede particolari competenze di programmazione per essere implementato. Inoltre, fornisce un risultato che, per tutti gli scopi pratici, può essere considerato come un campione casuale estratto da $p(\theta \mid y)$. Tuttavia, anche se tale metodo fornisce risultati accuratissimi, esso ha un uso limitato. A causa della *maledizione della dimensionalità*[3], tale metodo può solo essere solo nel caso di semplici modelli statistici, con non più di due parametri. Nella pratica concreta tale metodo viene dunque sostituito da altre tecniche più efficienti in quanto, anche nei più comuni modelli utilizzati in psicologia, vengono solitamente stimati centinaia se non migliaia di parametri.

[3]Per capire cosa sia la maledizione della dimensionalità, supponiamo di utilizzare una griglia di 100 punti equispaziati. Nel caso di un solo parametro, è necessario calcolare 100 valori. Per due parametri devono essere calcolari 100^2 valori. Ma già per 10 parametri è necessario calcolare 10^{10} valori – è facile capire che una tale quantità di calcoli è troppo grande anche per un computer molto potente. Per modelli che richiedono la stima di un numero non piccolo di parametri è dunque necessario procedere in un altro modo.

2.2 Metodo Monte Carlo

I metodi più ampiamente adottati nell'analisi bayesiana per la costruzione della distribuzione a posteriori per modelli complessi sono i metodi di campionamento MCMC. Tali metodi consentono al ricercatore di decidere quali distribuzioni a priori e quali distribuzioni di verosimiglianza usare sulla base di considerazioni teoriche soltanto, senza doversi preoccupare di altri vincoli. Dato che è basata su metodi computazionalmente intensivi, la stima numerica MCMC della funzione a posteriori può essere svolta soltanto mediante software. In anni recenti i metodi Bayesiani di analisi dei dati sono diventati sempre più popolari proprio perché la potenza di calcolo necessaria per svolgere tali calcoli è ora alla portata di tutti. Questo non era vero solo pochi decenni fa.

2.2.1 Integrazione di Monte Carlo

Il termine Monte Carlo si riferisce al fatto che la computazione fa ricorso ad un ripetuto campionamento casuale attraverso la generazione di sequenze di numeri casuali. Una delle sue applicazioni più potenti è il calcolo degli integrali mediante simulazione numerica. Supponiamo di essere in grado di estrarre campioni casuali dalla distribuzione continua $p(\theta \mid y)$ di media μ. Se possiamo ottenere una sequenza di realizzazioni indipendenti

$$\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(T)} \overset{\text{iid}}{\sim} p(\theta \mid y)$$

allora diventa possibile calcolare

$$\mathbb{E}(\theta \mid y) = \int \theta p(\theta \mid y) \, \mathrm{d}\theta \approx \frac{1}{T} \sum_{i=1}^{T} \theta^{(t)}.$$

In altre parole, l'aspettazione teorica di θ può essere approssimata dalla media campionaria di un insieme di realizzazioni indipendenti ricavate da $p(\theta \mid y)$. Per la Legge Forte dei Grandi Numeri, l'approssimazione diventa arbitrariamente esatta per $T \to \infty$.[4]

[4] L'integrazione Monte Carlo può essere utilizzata anche per la valutazione di integrali più complessi.

Quello che è stato detto sopra non è altro che un modo sofisticato per dire che, se vogliamo calcolare un'approssimazione del valore atteso di una variabile casuale, non dobbiamo fare altro che la media aritmetica di un grande numero di realizzazioni indipendenti della variabile casuale. Come è facile intuire, l'approssimazione migliora al crescere del numero di dati che abbiamo a disposizione.

Un'altra importante funzione di θ è la funzione indicatore, $I(l < \theta < u)$, che assume valore 1 se θ giace nell'intervallo (l, u) e 0 altrimenti. Il valore di aspettazione di $I(l < \theta < u)$ rispetto a $p(\theta)$ dà la probabilità che θ rientri nell'intervallo specificato, $Pr(l < \theta < u)$, e può essere approssimato usando l'integrazione Monte Carlo, ovvero prendendo la media campionaria del valore della funzione indicatore per ogni realizzazione $\theta^{(t)}$. È semplice vedere come

$$Pr(l < \theta < u) \approx \frac{\text{numero di realizzazioni } \theta^{(t)} \in (l, u)}{T}.$$

Presentiamo qui l'integrazione di Monte Carlo perché, nell'analisi bayesiana, il metodo Monte Carlo viene usato per ottenere un'approssimazione della distribuzione a posteriori, quando tale distribuzione non può essere calcolata con metodi analitici. In altre parole, il metodo Monte Carlo consente di ottenere un gran numero di valori θ che, nelle circostanze ideali, avrà una distribuzione identica alla distribuzione a posteriori $p(\theta \mid y)$.

2.2.2 Descrizione intuitiva

Se la funzione di densità $p(\theta \mid y)$ è conosciuta, è facile ottenere una sequenza di realizzazioni *iid* della variabile casuale, per esempio, usando R. Ma ora supponiamo di non conoscere $p(\theta \mid y)$. Quello che vogliamo fare è ottenere comunque una sequenza di valori θ. Anche se tali valori non saranno *iid*, per qualunque coppia di valori θ_a e θ_b nella sequenza vogliamo che sia soddisfatto il seguente vincolo:

$$\frac{\#\theta' \text{ nella sequenza} = \theta_a}{\#\theta' \text{ nella sequenza} = \theta_b} \approx \frac{p(\theta_a \mid y)}{p(\theta_b \mid y)}.$$

L'algoritmo di Metropolis ci consente di ottenere una tale sequenza di valori, la cui distribuzione sarà dunque uguale a $p(\theta \mid y)$. In forma in-

tuitiva, l'algoritmo di Metropolis può essere descritto come indicato di seguito.

- Data una sequenza di valori $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(t)}\}$, ci poniamo il problema di aggiungere un nuovo valore θ^{t+1} alla sequenza.
- Consideriamo un valore θ^* simile a $\theta^{(t)}$; ci chiediamo se dobbiamo inserire un tale valore nella sequenza oppure no.
- Se $p(\theta^* \mid y) > p(\theta^{(t)} \mid y)$, allora sicuramente lo dobbiamo aggiungere alla sequenza perché, nella sequenza, il numero di valori θ^* deve essere maggiore del numero dei valori $\theta^{(t)}$.
- Se invece $p(\theta^* \mid y) < p(\theta^{(t)} \mid y)$, allora non dobbiamo necessariamente aggiungere θ^* alla sequenza.
- La decisione di aggiungere o no θ^* alla sequenza dipenderà dal confronto tra $p(\theta^* \mid y)$ e $p(\theta^{(t)} \mid y)$.
  - Calcoliamo il rapporto

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{(t)} \mid y)} = \frac{p(y \mid \theta^*)p(\theta^*)}{p(y \mid p(\theta^{(t)})p(p(\theta^{(t)})}.$$

- Se $r > 1$, accettiamo θ^* e lo aggiungiamo alla sequenza: $\theta^{(t+1)} = \theta^*$, in quanto $\theta^{(t)}$ è già presente nella sequenza e θ^* ha una probabilità maggiore di $\theta^{(t)}$.
- Se $r < 1$, per ciasuna istanza di $\theta^{(t)}$, accettiamo θ^* solo una frazione di volte uguale a

$$\frac{p(\theta^* \mid y)}{p(\theta^{(t)} \mid y)}$$

in quanto la frequenza relativa dei valori $\theta^{(t)}$ e θ^* nella sequenza deve essere uguale al rapporto precedente. Per ottenere questo risultato, poniamo $\theta^{(t+1)}$ uguale a θ^* o $\theta^{(t)}$ con probabilità rispettivamente uguali a r o $1 - r$.

Questa è l'intuizione che sta alla base dell'algoritmo di Metropolis et al. (1953).

2.2.3 Un'applicazione empirica

Poniamoci ora il problema di usare l'algoritmo di Metropolis per calcolare la distribuzione a posteriori di una proporzione θ. Usiamo nuovamente i dati di Zetsche et al. (2019) (ovvero, 23 "successi" in 30 prove Bernoulliane) e, per rendere il problema più interessante, assumiamo per θ una distribuzione a priori Beta$(2, 10)$. Sappiamo che, in tali circostanze, la distribuzione a posteriori può essere ottenuta analiticamente tramite lo schema beta-binomiale ed è una Beta$(25, 17)$. Se vogliamo il valore della media a posteriori di θ, il risultato esatto è dunque

$$\bar{\theta}_{post} = \frac{\alpha}{\alpha + \beta} = \frac{25}{25 + 17} \approx 0.5952.$$

È anche possibile ottenere il valore della media a posteriori per via numerica. Sapendo che la distribuzione a posteriori è una Beta$(25, 17)$, possiamo estrarre un campione di osservazioni da una tale distribuzione e calcolare la media. Con poche osservazioni (diciamo 10) otteniamo un risultato molto approssimato

```
set.seed(84735)
print(mean(rbeta(1e2, shape1 = 25, shape2 = 17)), 6)
#> [1] 0.584251
```

ma, per la legge dei grandi numeri, l'approssimazione migliora all'aumentare del numero di osservazioni:

```
print(mean(rbeta(1e4, shape1 = 25, shape2 = 17)), 6)
#> [1] 0.595492
print(mean(rbeta(1e6, shape1 = 25, shape2 = 17)), 6)
#> [1] 0.595192
```

Lo stesso si può dire delle altre statistiche descrittive: moda, varianza, eccetera.

Nel presente esempio, la simulazione Monte Carlo produce il risultato desiderato perché

- sappiamo che la distribuzione a posteriori è una Beta$(25, 17)$,
- è possibile usare le funzioni R per estrarre campioni casuali da una tale distribuzione.

Tuttavia, capita raramente di usare una distribuzione a priori coniugata alla verosimiglianza. Quindi, in generale, le due condizioni descritte sopra non si applicano. Ad esempio, nel caso di una verosimiglianza binomiale e di una distribuzione a priori gaussiana, la distribuzione a posteriori di θ è

$$p(\theta \mid y) = \frac{\mathrm{e}^{-(\theta-1/2)^2}\theta^y(1-\theta)^{n-y}}{\int_0^1 \mathrm{e}^{-(t-1/2)^2}t^y(1-t)^{n-y}dt}.$$

Una tale distribuzione non è implementata in R e dunque non possiamo ottenere dei campioni casuali da una tale distribuzione.

In tali circostanze, però, è ancora possibile ottenere ottenere un campione causale dalla distribuzione a posteriori in un altro modo. Questo risultato si ottiene utilizzando i metodi Monte Carlo basati su Catena di Markov (MCMC). I metodi MCMC, di cui l'algoritmo di Metropolis è un caso particolare e ne rappresenta il primo esempio, sono una classe di algoritmi che consentono di ottenere campioni casuali da una distribuzione a posteriori *senza dovere conoscere la rappresentazione analitica di una tale distribuzione*.[5] Le tecniche MCMC sono il metodo computazionale maggiormente usato per risolvere i problemi dell'inferenza bayesiana.

2.2.4 Una passeggiata casuale sui numeri naturali

Prima di applicare l'algoritmo di Metropolis ai dati di Zetsche et al. (2019), consideriamo un caso più semplice. In questo esempio preliminare useremo l'algoritmo di Metropolis per ottenere un campione casuale da una distribuzione di massa di probabilità; esamineremo il caso continuo in seguito.[6]

Definiamo la distribuzione di probabilità discreta della variabile casuale X che assume valori nell'insieme dei numeri naturali $1, 2, \ldots, K$. Scri-

[5] In termini più formali, si può dire che i metodi MCMC consentono di costruire sequenze di punti (detti catene di Markov) nello spazio dei parametri le cui densità sono proporzionali alla distribuzione a posteriori. In altre parole, dopo aver simulato un grande numero di passi della catena si possono usare i valori così generati come se fossero un campione casuale della distribuzione a posteriori. Un'introduzione alle catene di Markov è fornita nell'Appendice **??**.

[6] Seguo qui la trattazione di Albert and Hu (2019). Per una presentazione dell'idea che sta alla base dell'algoritmo di Metropolis, si possono anche consultare Kruschke (2014) e McElreath (2020).

viamo in R la funzione `pd()` che assegna a $X = \{1, 2, \dots, 8\}$ valori di probabilità proporzionali agli interi 5, 10, 4, 4, 20, 20, 12 e 5:

```
pd <- function(x) {
  values <- c(5, 10, 4, 4, 20, 20, 12, 5)
  ifelse(
    x %in% 1:length(values),
    values[x] / sum(values),
    0
  )
}
prob_dist <- tibble(
  x = 1:8,
  prob = pd(1:8)
)
```

La figura 2.5 illustra la distribuzione di massa di probabilità che è stata generata in questo modo.

```
x <- 1:8
prob_dist %>%
  ggplot(aes(x = x, y = prob)) +
  geom_bar(stat = "identity", width = 0.06) +
  scale_x_continuous("x", labels = as.character(x), breaks = x) +
  labs(
    y = "Probabilità",
    x = "X"
  )
```

Per i dati di questo esempio, l'algoritmo di Metropolis corrisponde alla seguente passeggiata aleatoria.[7]

[7] Per passeggiata aleatoria si intende un processo stocastico a parametro discreto, nel quale si ipotizza che una variabile casuale X_t descriva la posizione assunta al tempo t da un punto in movimento. Inizialmente, il punto si trova nella posizione 0, cioè $X_0 = 0$. Al tempo $t = 1$ il punto compie un salto o in avanti, e raggiunge la posizione 1 con probabilità 1/2, o all'indietro, e raggiunge la posizione -1 con probabilità 1/2. Al tempo $t = 2$ fa ancora un salto in avanti con probabilità 1/2, oppure all'indietro con probabilità 1/2. La probabilità di andare avanti o indietro ai diversi tempi rimane costante e non dipende dai risultati precedenti (Treccani: *Enciclopedia della Matematica* online, https://www.treccani.it/enciclopedia/).

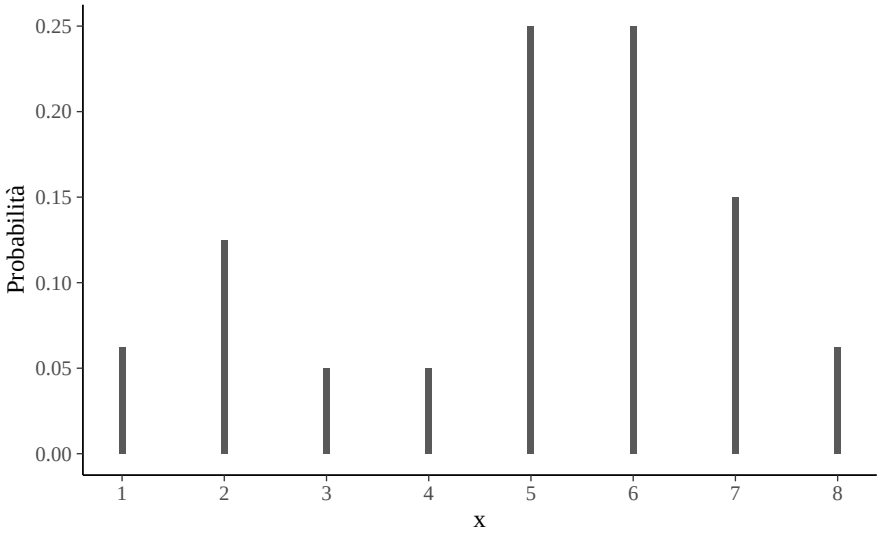


Figura 2.5: *Distribuzione di massa di probabilità della variabile casuale discreta X avente supporto $\{1, 2, ..., 8\}$.*

1. L'algoritmo inizia con un valore iniziale qualsiasi da 1 a $K = 8$ della variabile casuale.
2. Per simulare il valore successivo della sequenza, lanciamo una moneta equilibrata. Se esce testa, consideriamo come valore candidato il valore immediatamente precedente al valore corrente nella sequenza; se esce croce, il candidato è il valore nella sequenza immediatamente successivo a quello corrente.
3. Si calcola il rapporto r tra la probabilità del valore candidato e la probabilità del valore corrente:

$$r = \frac{pd(\text{valore candidato})}{pd(\text{valore corrente})}.$$

4. Si estrae un numero a caso $\in [0, 1]$. Se tale valore è minore di r si accetta il valore candidato come valore successivo della catena markoviana; altrimenti il valore successivo della catena rimane il valore corrente.

In termini tecnici (si veda l'Appendice **??**), i passi da 1 a 4 definiscono una catena di Markov irriducibile e aperiodica sui valori di stato

$\{1, 2, \ldots, 8\}$, dove il passo 1 fornisce il valore iniziale della catena e i passi da 2 a 4 definiscono la matrice di transizione P. Il campionamento dalla distribuzione di massa `pd` corrisponde ad una passeggiata aleatoria che inizia da una posizione qualsiasi e che ripete le fasi 2, 3 e 4 dell'algoritmo di Metropolis. Dopo un gran numero di passi, la distribuzione dei valori della catena markoviana approssimerà la distribuzione di probabilità `pd`.

La funzione `random_walk()` implementa l'algoritmo di Metropolis. Tale funzione prende in input la distribuzione di probabilità `pd`, la posizione di partenza `start` e il numero di passi dell'algoritmo `num_steps`.

```
random_walk <- function(pd, start, num_steps) {
  y <- rep(0, num_steps)
  current <- start
  for (j in 1:num_steps) {
    candidate <- current + sample(c(-1, 1), 1)
    prob <- pd(candidate) / pd(current)
    if (runif(1) < prob) {
      current <- candidate
    }
    y[j] <- current
  }
  return(y)
}
```

Implementiamo ora l'algoritmo di Metropolis utilizzando, quale valore iniziale, $X = 4$. Ripetiamo la simulazione 10,000 volte.

```
out <- random_walk(pd, 4, 1e4)

S <- tibble(out) %>%
  group_by(out) %>%
  summarize(
    N = n(),
    Prob = N / 10000
  )

prob_dist2 <- rbind(
  prob_dist,
```

```
  tibble(
    x = S$out,
    prob = S$Prob
  )
)
prob_dist2$Type <- rep(
  c("Prob. corrette", "Prob. simulate"),
  each = 8
)
```

```
x <- 1:8
prob_dist2 %>%
  ggplot(aes(x = x, y = prob, fill = Type)) +
  geom_bar(
    stat = "identity",
    width = 0.1,
    position = position_dodge(0.3)
  ) +
  scale_x_continuous(
    "x",
    labels = as.character(x),
    breaks = x
  ) +
  scale_fill_manual(values = c("black", "gray80")) +
  theme(legend.title = element_blank()) +
  labs(
    y = "Probabilità",
    x = "X"
  )
```

La figura 2.6 confronta l'istogramma dei valori simulati dalla passeggiata aleatoria con l'effettiva distribuzione di probabilità `pd`. Si noti che le due distribuzioni sono molto simili.

2.2.5 L'algoritmo di Metropolis

Dopo avere introdotto l'algoritmo di Metropolis con l'esempio proposto da Albert and Hu (2019), consideriamo ora l'algoritmo nella sua for-

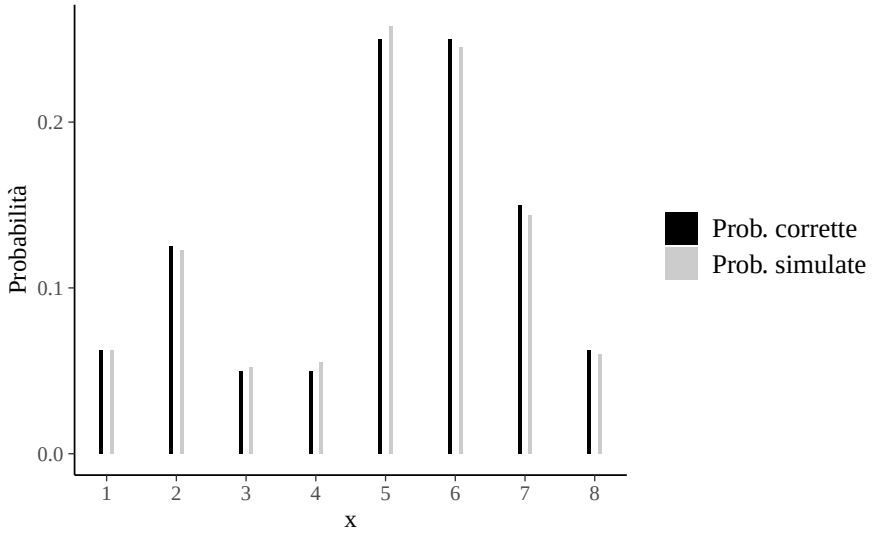


Figura 2.6: *L'istogramma confronta i valori prodotti dall'algoritmo di Metropolis con i valori corretti della distribuzione di massa di probabilità.*

ma più generale.[8] Nelle iterazioni dell'algoritmo di Metropolis possiamo distinguere le seguenti fasi.

1. Si inizia con un punto arbitrario $\theta^{(1)}$; quindi il primo valore della catena di Markov $\theta^{(1)}$ può corrispondere semplicemente ad un valore a caso tra i valori possibili del parametro.

(b) Per ogni passo successivo della catena, $m + 1$, si estrae un valore candidato θ' da una distribuzione proposta: $\theta' \sim \Pi(\theta)$. La distribuzione proposta può essere qualunque distribuzione, anche se, idealmente, è meglio che sia simile alla distribuzione a posteriori. In pratica, però, la distribuzione a posteriori è sconosciuta e quindi il valore θ' viene estratto a caso da una qualche distribuzione simmetrica centrata sul valore corrente $\theta^{(m)}$ del parametro. Nell'esempio presente useremo la gaussiana quale distribuzione proposta. La distribuzione proposta (gaussiana) sarà centrata sul valore corrente della catena e avrà una

[8] Un'illustrazione visiva di come si svolge il processo di "esplorazione" dell'algoritmo di Metropolis è fornita in questo post: https://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/.

deviazione standard appropriata: $\theta' \sim \mathcal{N}(\theta^{(m)}, \sigma)$. In pratica, questo significa che, se σ è piccola, il valore candidato θ' sarà simile al valore corrente $\theta^{(m)}$.

(c) Si calcola il rapporto r tra la densità della distribuzione a posteriori non normalizzata calcolata nel punto θ' e la densità nel punto $\theta^{(m)}$:

$$r = \frac{p(y \mid \theta')p(\theta')}{p(y \mid \theta^{(m)})p(\theta^{(m)})}. \tag{2.1}$$

Il numeratore della (2.1) contiene il prodotto tra la verosimiglianza $p(y \mid \theta')$ e la densità a priori di θ, entrambe calcolate nel punto θ'. Il denominatore contiene il prodotto tra la verosimiglianza $p(y \mid \theta^{(m)})$ e la densità a priori di θ, entrambe calcolate nel punto $\theta^{(m)}$. Si noti che, essendo un rapporto, la (2.1) cancella la costante di normalizzazione.

(d) Si decide se accettare il candidato θ' oppure se rigettarlo e estrarre un nuovo valore dalla distribuzione proposta. Possiamo pensare al rapporto r come alla risposta alla seguente domanda: alla luce dei dati, quale stima di θ è più plausibile il valore candidato o il valore corrente? Se r è maggiore di 1, ciò significa che il candidato è più plausibile del valore corrente; dunque il candidato viene sempre accettato. Altrimenti, si decide di accettare il candidato con una probabilità minore di 1, ovvero non sempre, ma soltanto con una probabilità uguale ad r. Se r è uguale a 0.10, ad esempio, questo significa che la credibilità a posteriori del valore candidato è 10 volte più piccola della credibilità a posteriori del valore corrente. Dunque, il valore candidato verrà accettato solo nel 10% dei casi. Come conseguenza di questa strategia di scelta, l'algoritmo di Metropolis ottiene un campione casuale dalla distribuzione a posteriori, dato che la probabilità di accettare il valore candidato è proporzionale alla densità del candidato nella distribuzione a posteriori. Dal punto di vista algoritmico, la procedura descritta sopra viene implementata confrontando il rapporto r con un valore estratto a caso da una distribuzione uniforme Unif$(0, 1)$. Se $r > u \sim$ Unif$(0, 1)$, allora il candidato θ' viene accettato e la catena si muove in quella nuova posizione, ovvero $\theta^{(m+1)} = \theta'$.

Altrimenti $\theta^{(m+1)} = \theta^{(m)}$ e si estrae un nuovo candidato dalla distribuzione proposta.

(e) Il passaggio finale dell'algoritmo calcola l'*accettanza* in una specifica esecuzione dell'algoritmo, ovvero la proporzione di candidati θ' che sono stati accettati quali valori successivi della catena.

L'algoritmo di Metropolis prende come input il numero T di passi da simulare, la deviazione standard σ della distribuzione proposta e la densità a priori, e ritorna come output la sequenza $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(T)}$. La chiave del successo dell'algoritmo di Metropolis è il numero di passi fino a che la catena approssima la stazionarietà. Tipicamente i primi da 1000 a 5000 elementi sono scartati. Dopo un certo periodo k (detto di *burn-in*), la catena di Markov converge ad una variabile casuale che è distribuita secondo la distribuzione a posteriori. In altre parole, i campioni del vettore $\left(\theta^{(k+1)}, \theta^{(k+2)}, \dots, \theta^{(T)}\right)$ diventano campioni di $p(\theta \mid y)$.

2.2.6 Un'applicazione empirica

Possiamo ora utilizzare l'algoritmo di Metropolis per trovare, nel caso dei pazienti clinici depressi di Zetsche et al. (2019), la distribuzione a posteriori di θ, ovvero la probabilità che l'umore futuro atteso sia negativo. I dati di Zetsche et al. (2019) ci dicono che, nel caso dei 30 pazienti che sono stati esaminati, 23 hanno manifestato aspettative distorte negativamente circa il loro stato d'animo futuro. A priori, abbiamo deciso di imporre su θ una Beta$(2, 10)$.[9]

2.2.6.1 Funzioni

Definiamo la funzione `likelihood()`, considerati come fissi i dati di Zetsche et al. (2019), ritorna l'ordinata della verosimiglianza binomiale per ciascun valore `param` in input:

```
likelihood <- function(param, x = 23, N = 30) {
  dbinom(x, N, param)
}
```

[9] L'Appendice **??** mostra come usare il metodo basato su griglia per trovare $p(\theta \mid y)$ con questi dati.

La funzione `prior()` ritorna l'ordinata della distribuzione a priori Beta$(2, 10)$ per ciascun valore `param` in input:

```
prior <- function(param, alpha = 2, beta = 10) {
  dbeta(param, alpha, beta)
}
```

La funzione `posterior()` ritorna, per ciascun valore `param` in input, il prodotto della densità a priori e della verosimiglianza:

```
posterior <- function(param) {
  likelihood(param) * prior(param)
}
```

2.2.6.2 Implementazione

Per implementare l'algoritmo di Metropolis utilizzeremo una distribuzione proposta gaussiana. Il valore candidato sarà dunque un valore selezionato a caso da una gaussiana di parametri μ uguale al valore corrente nella catena e $\sigma = 0.9$. In questo esempio, la deviazione standard σ è stata scelta empiricamente in modo tale da ottenere una accettanza adeguata. L'accettanza ottimale èpari a circa 0.20/0.30 — se l'accettanza è troppo grande, l'algoritmo esplora uno spazio troppo ristretto della distribuzione a posteriori.[10]

```
proposal_distribution <- function(param) {
  while (1) {
    res <- rnorm(1, mean = param, sd = 0.9)
    if (res > 0 & res < 1) {
      break
    }
  }
  res
}
```

[10] L'accettanza dipende dalla distribuzione proposta: in generale, tanto più la distribuzione proposta è simile alla distribuzione target, tanto più alta diventa l'accettanza.

Ho inserito un controllo che impone al valore candidato di essere incluso nell'intervallo [0, 1], com'è necessario per il valore di una proporzione.[11]

L'algoritmo di Metropolis viene implementato nella seguente funzione:

```
metropolis <- function(startvalue, iterations) {
  chain <- vector(length = iterations + 1)
  chain[1] <- startvalue
  for (i in 1:iterations) {
    proposal <- proposal_distribution(chain[i])
    r <- posterior(proposal) / posterior(chain[i])
    if (runif(1) < r) {
      chain[i + 1] <- proposal
    } else {
      chain[i + 1] <- chain[i]
    }
  }
  chain
}
```

Mediante la funzione precedente, generiamo una catena di valori θ:

```
set.seed(84735)
startvalue <- runif(1, 0, 1)
niter <- 1e4
chain <- metropolis(startvalue, niter)
```

In questo modo, abbiamo ottenuto una catena di Markov costituita da 10,001 valori. Escludiamo i primi 5,000 valori considerati come burn-in. Consideriamo i restanti 5,001 valori come un campione casuale estratto dalla distribuzione a posteriori $p(\theta \mid y)$.

L'accettanza è pari a

[11] Si possono trovare implementazioni più eleganti di quella presentata qui. Il presente esercizio ha solo lo scopo di illustrare la logica soggiacente all'algoritmo di Metropolis; non ci preoccupiamo di trovare un'implementazione efficente dell'algoritmo.

```
burnin <- niter / 2
acceptance <- 1 - mean(duplicated(chain[-(1:burnin)]))
acceptance
#> [1] 0.2585
```

il che conferma la bontà della deviazione standard ($\sigma = 0.9$) scelta per la distribuzione proposta.

Mediante i valori della catena così ottenuta è facile trovare una stima a posteriori del parametro θ. Per esempio, la stima della media a posteriori è:

```
mean(chain[-(1:burnin)])
#> [1] 0.5957
```

Una figura che mostra l'approssimazione di $p(\theta \mid y)$ ottenuta con l'algoritmo di Metropolis, insieme ad un *trace plot* dei valori della catena di Markov, è prodotta nel modo seguente:

```
p1 <- tibble(
  x = chain[-(1:burnin)]
) %>%
  ggplot(aes(x)) +
  geom_histogram() +
  labs(
    x = expression(theta),
    y = "Frequenza",
    title = "Distribuzione a posteriori"
  ) +
  geom_vline(
    xintercept = mean(chain[-(1:burnin)])
  ) +
  xlim(c(0.3, 0.85)) +
  coord_flip()

p2 <- tibble(
  x = 1:length(chain[-(1:burnin)]),
  y = chain[-(1:burnin)]
```

```
) %>%
  ggplot(aes(x, y)) +
  geom_line() +
  labs(
    x = "Numero di passi",
    y = expression(theta),
    title = "Valori della catena"
  ) +
  geom_hline(
    yintercept = mean(chain[-(1:burnin)]),
    colour = "gray"
  ) +
  ylim(c(0.3, 0.85))

p1 + p2
```

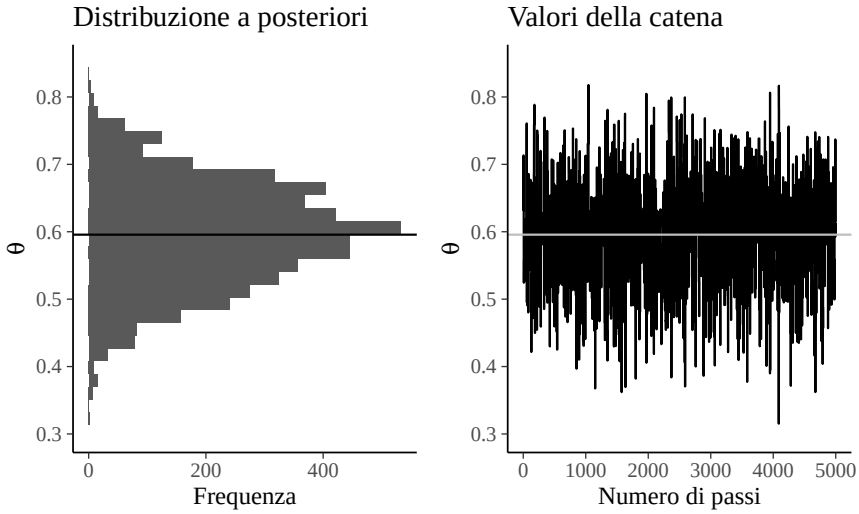


Figura 2.7: *Sinistra. Stima della distribuzione a posteriori della probabilità di una aspettativa futura distorta negativamente per i dati di Zetsche et al. (2019). Destra. Trace plot dei valori della catena di Markov escludendo il periodo di burn-in.*

2.2.6.3 Animazione

È facile creare un'animazione che illustra il progresso dell'algoritmo di Metropolis. In R, sono necessari i seguenti pacchetti:

```
library("gganimate")
library("magick")
```

Creiamo un DataFrame che contiene i primi 1000 valori della catena che è stata generata in precedenza:

```
d <- tibble(
  iter = 1:1000,
  mcmc = chain[1:1000]
)
```

L'animazione del trace plot si ottiene mediante la funzione `transition_reveal()`:

```
ggplot(d, aes(x = iter, y = mcmc)) +
  geom_line() +
  transition_reveal(iter)
```

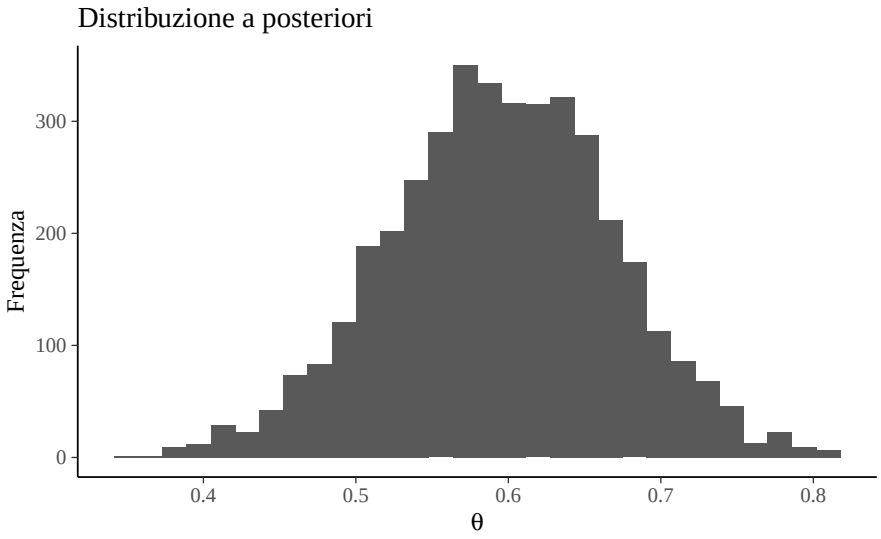
Ovviamente, è necessario fare tutto questo in RStudio: non ho inserito un GIF animato nel presente file pdf.

2.2.6.4 Funzione `metropolis()`

I calcoli precedenti possono anche essere svolti usando la funzione `metropolis()` del pacchetto `ProbBayes`. Per fare un esempio, usiamo i dati corrispondenti a 23 successi in 30 prove, con una distribuzione a priori Beta$(2, 10)$. Le istruzioni sono le seguenti:

```
set.seed(123)
lpost <- function(theta, s) {
  dbeta(theta, 2, 10, log = TRUE) +
    dbinom(23, 30, theta, log = TRUE)
}
post <- ProbBayes::metropolis(lpost, 0.5, 0.2, 8000)
```

```
tibble(
  x = post$S[4001:8000]
) %>%
  ggplot(aes(x)) +
  geom_histogram() +
  labs(
    x = expression(theta),
    y = "Frequenza",
    title = "Distribuzione a posteriori"
  )
```



Utilizzando un burn-in piuttosto lungo, la stima a posteriori di θ diventa:

```
mean(post$S[4001:8000])
#> [1] 0.5948
```

L'accettanza però non è ottimale:

```
post$accept_rate
#> [1] 0.5464
```

2.2.7 Input

Negli esempi discussi in questo Capitolo abbiamo illustrato l'esecuzione di una singola catena in cui si parte un unico valore iniziale e si raccolgono i valori simulati da molte iterazioni. È possibile però che i valori di una catena siano influenzati dalla scelta del valore iniziale. Quindi una raccomandazione generale è di eseguire l'algoritmo di Metropolis più volte utilizzando diversi valori di partenza. In questo caso, si avranno più catene di Markov. Confrontando le proprietà delle diverse catene si esplora la sensibilità dell'inferenza alla scelta del valore di partenza. I software MCMC consentono sempre all'utente di specificare diversi valori di partenza e di generare molteplici catene di Markov.

2.2.8 Stazionarietà

Un punto importante da verificare è se il campionatore ha raggiunto la sua distribuzione stazionaria. La convergenza di una catena di Markov alla distribuzione stazionaria viene detta "mixing".

2.2.8.1 Autocorrelazione

Informazioni sul "mixing" della catena di Markov sono fornite dall'autocorrelazione. L'autocorrelazione misura la correlazione tra i valori successivi di una catena di Markov. Il valore m-esimo della serie ordinata viene confrontato con un altro valore ritardato di una quantità k (dove k è l'entità del ritardo) per verificare quanto si correli al variare di k. L'autocorrelazione di ordine 1 (*lag 1*) misura la correlazione tra valori successivi della catena di Markow (cioè, la correlazione tra $\theta^{(m)}$ e $\theta^{(m-1)}$); l'autocorrelazione di ordine 2 (*lag 2*) misura la correlazione tra valori della catena di Markow separati da due "passi" (cioè, la correlazione tra $\theta^{(m)}$ e $\theta^{(m-2)}$); e così via.

L'autocorrelazione di ordine k è data da ρ_k e può essere stimata come:

$$\begin{aligned}\rho_k &= \frac{\text{Cov}(\theta_m, \theta_{m+k})}{\mathbb{V}(\theta_m)} \\ &= \frac{\sum_{m=1}^{n-k}(\theta_m - \bar{\theta})(\theta_{m-k} - \bar{\theta})}{\sum_{m=1}^{n-k}(\theta_m - \bar{\theta})^2} \qquad \text{con} \quad \bar{\theta} = \frac{1}{n}\sum_{m=1}^{n}\theta_m. \end{aligned} \tag{2.2}$$

Per fare un esempio pratico, simuliamo dei dati autocorrelati con la funzione R `colorednoise::colored_noise()`:

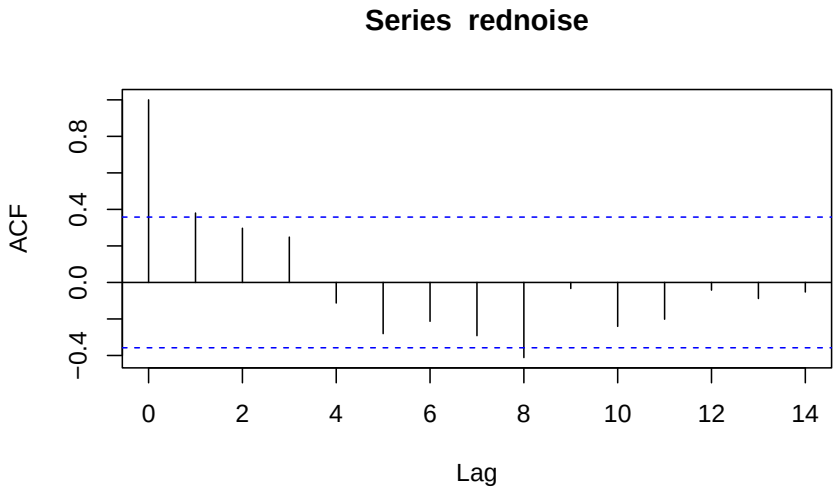
```
suppressPackageStartupMessages(library("colorednoise"))
set.seed(34783859)
rednoise <- colored_noise(
  timesteps = 30, mean = 0.5, sd = 0.05, phi = 0.3
)
```

L'autocorrelazione di ordine 1 è semplicemente la correlazione tra ciascun elemento e quello successivo nella sequenza. Nell'esempio, il vettore `rednoise` è una sequenza temporale di 30 elementi. Il vettore `rednoise[-length(rednoise)]` include gli elementi con gli indici da 1 a 29 nella sequenza originaria, mentre il vettore `rednoise[-1]` include gli elementi 2:30. Gli elementi delle coppie ordinate dei due vettori avranno dunque gli indici $(1, 2), (2, 3), ... (29, 30)$ degli elementi della sequenza originaria. La correlazione di Pearson tra i vettori `rednoise[-length(rednoise)]` e `rednoise[-1]` corrisponde all'autocorrelazione di ordine 1 della serie temporale.

```
cor(rednoise[-length(rednoise)], rednoise[-1])
#> [1] 0.3967
```

Il Correlogramma è uno strumento grafico usato per la valutazione della tendenza di una catena di Markov nel tempo. Il correlogramma si costruisce a partire dall'autocorrelazione ρ_k di una catena di Markov in funzione del ritardo (*lag*) k con cui l'autocorrelazione è calcolata: nel grafico ogni barretta verticale riporta il valore dell'autocorrelazione (sull'asse delle ordinate) in funzione del ritardo (sull'asse delle ascisse). In R, il correlogramma può essere prodotto con una chiamata a `acf()`:

```
acf(rednoise)
```



Nel correlogramma precedente vediamo che l'autocorrelazione di ordine 1 è circa pari a 0.4 e diminuisce per lag maggiori; per lag di 4, l'autocorrelazione diventa negativa e aumenta progressivamente fino ad un lag di 8; eccetera.

In situazioni ottimali l'autocorrelazione diminuisce rapidamente ed è effettivamente pari a 0 per piccoli lag. Ciò indica che i valori della catena di Markov che si trovano a più di soli pochi passi di distanza gli uni dagli altri non risultano associati tra loro, il che fornisce una conferma del "mixing" della catena di Markov, ossia della convergenza alla distribuzione stazionaria. Nelle analisi bayesiane, una delle strategie che consentono di ridurre l'autocorrelazione è quella di assottigliare l'output immagazzinando solo ogni m-esimo punto dopo il periodo di burn-in. Una tale strategia va sotto il nome di *thinning*.

2.2.9 Test di convergenza

Un test di convergenza può essere svolto in maniera grafica mediante le tracce delle serie temporali (*trace plot*), cioè il grafico dei valori simulati rispetto al numero di iterazioni. Se la catena è in uno stato stazionario le tracce mostrano assenza di periodicità nel tempo e ampiezza costante, senza tendenze visibili o andamenti degni di nota. Un esempio di *trace plot* è fornito nella figura 2.7 (destra).

Ci sono inoltre alcuni test che permettono di verificare la stazionarie-

tà del campionatore dopo un dato punto. Uno è il test di Geweke che suddivide il campione, dopo aver rimosso un periodo di burn in, in due parti. Se la catena è in uno stato stazionario, le medie dei due campioni dovrebbero essere uguali. Un test modificato, chiamato Geweke z-score, utilizza un test z per confrontare i due subcampioni ed il risultante test statistico, se ad esempio è più alto di 2, indica che la media della serie sta ancora muovendosi da un punto ad un altro e quindi è necessario un periodo di burn-in più lungo.

Commenti e considerazioni finali

In generale, la distribuzione a posteriori dei parametri di un modello statistico non può essere determinata per via analitica. Tale problema viene invece affrontato facendo ricorso ad una classe di algoritmi per il campionamento da distribuzioni di probabilità che sono estremamente onerosi dal punto di vista computazionale e che possono essere utilizzati nelle applicazioni pratiche solo grazie alla potenza di calcolo dei moderni computer. Lo sviluppo di software che rendono sempre più semplice l'uso dei metodi MCMC, insieme all'incremento della potenza di calcolo dei computer, ha contribuito a rendere sempre più popolare il metodo dell'inferenza bayesiana che, in questo modo, può essere estesa a problemi di qualunque grado di complessità.

Bibliografia

Albert, J. and Hu, J. (2019). *Probability and Bayesian Modeling*. Chapman and Hall/CRC.

Horn, S. and Loewenstein, G. (2021). Underestimating learning by doing. *Available at SSRN 3941441*.

Johnson, A. A., Ott, M., and Dogucu, M. (2022). *Bayes Rules! An Introduction to Bayesian Modeling with R*. CRC Press.

Kruschke, J. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.

Martin, O. A., Kumar, R., and Lao, J. (2022). *Bayesian Modeling and Computation in Python*. CRC Press.

McElreath, R. (2020). *Statistical rethinking: A Bayesian course with examples in R and Stan*. CRC Press, Boca Raton, Florida, 2nd edition edition.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

van de Schoot, R., Depaoli, S., King, R., Kramer, B., Märtens, K., Tadesse, M. G., Vannucci, M., Gelman, A., Veen, D., Willemsen, J., and Yau, C. (2021). Bayesian statistics and modelling. *Nature Reviews Methods Primer*, 1(1):1–26.

Zetsche, U., Bürkner, P.-C., and Renneberg, B. (2019). Future expectations in clinical depression: Biased or realistic? *Journal of Abnormal Psychology*, 128(7):678–688.