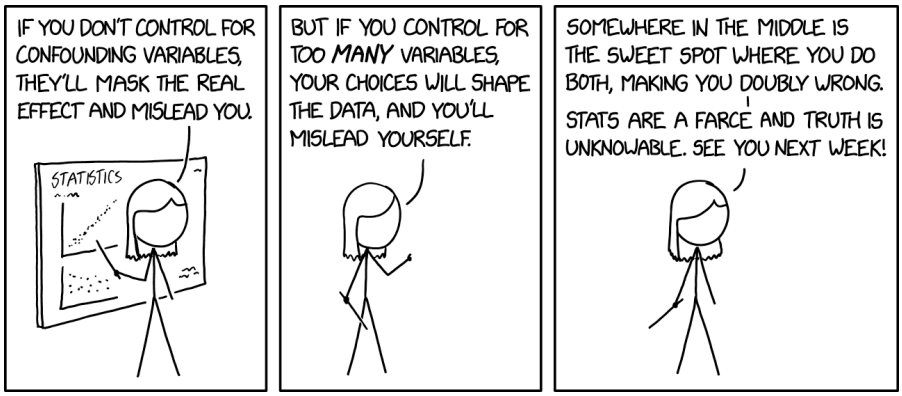
*Corrado Caudek*

# *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 $\theta$ è considerato una variabile casuale che segue la legge di probabilità $p(\theta)$. L'incertezza su $\theta$ 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 $\theta$, 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 $\theta$ 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 $\theta$.

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 $\theta$ rappresenterà i parametri incogniti di un modello statistico. Sia $y$ che $\theta$ 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 $\theta$ 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 $\theta$, possono assumere delle distribuzioni a priori differenti: a seconda delle informazioni disponibili bisogna selezionare una distribuzione di $\theta$ in modo tale che venga assegnata una probabilità maggiore a quei valori che si ritengono più plausibili per $\theta$. 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 $\frac{\alpha}{\beta} = \frac{5}{2}$.

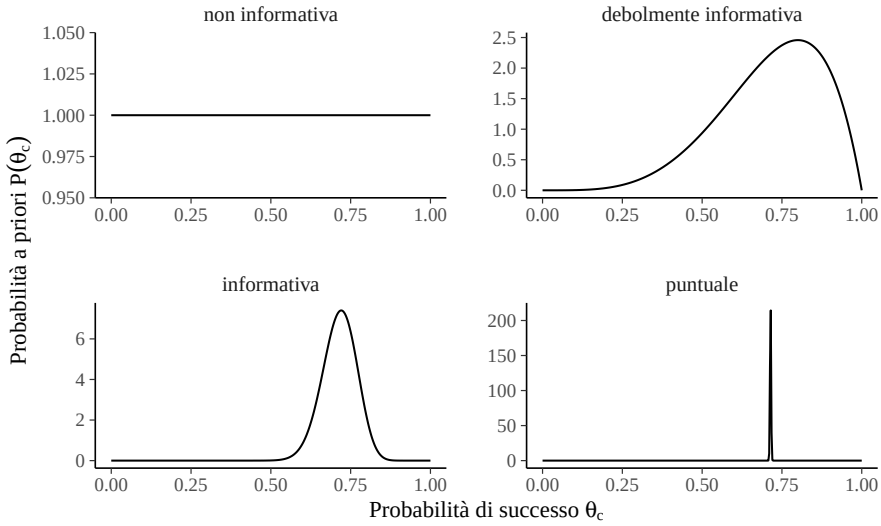



***Figura 1.1:*** *Esempi di distribuzioni a priori per il parametro $\theta_c$ nel Modello Binomiale.*

### 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 **??**.

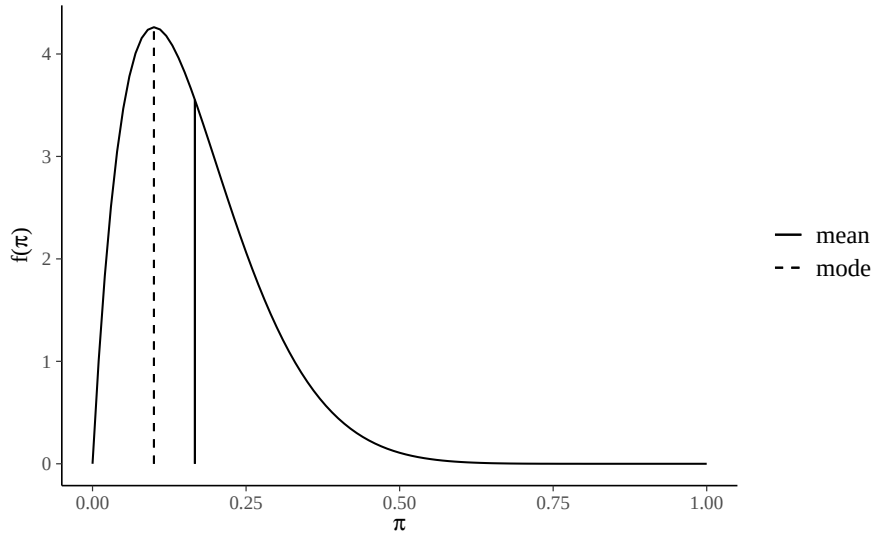
### 1.2.3 Un esempio concreto

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 $\theta$ (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 $\theta$ 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) $\theta$.

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) $\theta$; nel caso della la funzione di verosimiglianza la credibilità assegnata a ciascun possibile valore $\theta$ 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 $\theta$? 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 $\theta$ (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 esempio concreto

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 $\theta$ 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* $\theta$. Ma ora abbiamo il problema inverso, ovvero quello di fare inferenza su $\theta$ 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 $\theta$ 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 $\theta$ 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)
  )
```

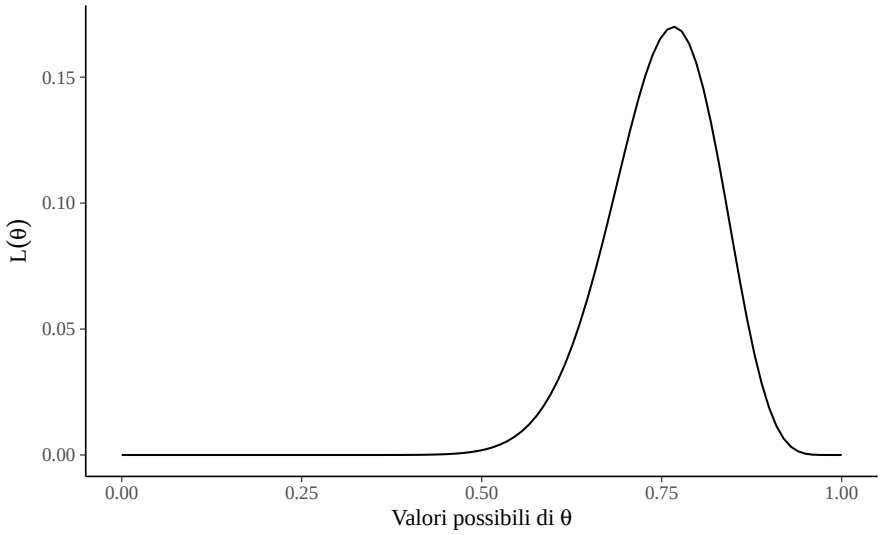



***Figura 1.2:*** *Funzione di verosimiglianza nel caso di 23 successi in 30 prove.*

Come possiamo interpretare la curva che abbiamo ottenuto? Per alcuni valori $\theta$ la funzione di verosimiglianza assume valori piccoli; per altri valori $\theta$ la funzione di verosimiglianza assume valori più grandi. Questi ultimi sono i valori di $\theta$ 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 esempio concreto

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 $\theta$, 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 $\theta$:

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

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 **??**.

## 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

# *Il modello beta-binomiale in linguaggio Stan*

In questo Capitolo introdurremo un linguaggio di programmazione probabilistica chiamato Stan. Stan[1] è un linguaggio di programmazione probabilistica usato per l'inferenza bayesiana (Carpenter et al., 2017). Prende il nome da uno dei creatori del metodo Monte Carlo, Stanislaw Ulam (Eckhardt, 1987). Stan consente di generare campioni da distribuzioni di probabilità basati sulla costruzione di una catena di Markov avente come distribuzione di equilibrio (o stazionaria) la distribuzione desiderata. Un'introduzione al linguaggio Stan è fornita nell'Appendice A. In questo Capitolo useremo Stan per fare inferenza su una proporzione.

## 2.1 Il presidente Trump e l'idrossiclorochina

Per fare un esempio concreto, consideriamo un set di dati reali. Cito dal *Washington Post* del 7 aprile 2020: *"One of the most bizarre and disturbing aspects of President Trump's nightly press briefings on the coronavirus pandemic is when he turns into a drug salesman. Like a cable TV pitchman hawking 'male enhancement' pills, Trump regularly extols the virtues of taking hydroxychloroquine, a drug used to treat malaria and lupus, as a potential 'game changer' that just might cure Covid-19."* Tralasciamo qui il fatto che il Donald Trump non sia un esperto in questo campo. Esaminiamo invece le evidenze iniziali a supporto dell'ipotesi che l'idrossiclorochina possa essere utile per la cura del Covid-19, ovvero le evidenze che erano disponibili nel momento in cui il Donald Trump ha fatto le affermazioni riportate sopra (in seguito, quest'idea è stata screditata). Tali evidenze sono state fornite da uno studio di Gautret et al. (2020). Il disegno sperimentale di Gautret et al. (2020) comprende, tra le altre cose, il confronto tra una condizione sperimentale e una

[1] http://mc-stan.org/

condizione di controllo. Il confronto importante è tra la proporzione di paziente positivi al virus SARS-CoV-2 nel gruppo sperimentale (a cui è stata somministrata l'idrossiclorochina; 6 su 14) e la proporzione di paziente positivi nel gruppo di controllo (a cui non è stata somministrata l'idrossiclorochina; ovvero 14 su 16). Obiettivo di questo Capitolo è mostrare come si possa fare inferenza sui dati di Gautret et al. (2020) usando il linguaggio Stan. Per semplicità, iniziamo considerando solo il gruppo di controllo.

## 2.2 Una proporzione

Sulla base di ciò che è stato detto nel Capitolo **??**, sappiamo che, quando i dati sono rappresentati da una proporzione $\theta$, e quando utilizziamo una distribuzione a priori Beta per $\theta$, la distribuzione a posteriori di $\theta$ è specificata dallo schema beta-binomiale. Se scegliamo, ad esempio, una Beta$(2, 2)$ quale distribuzione a priori per $\theta$, il modello diventa:

$$\begin{aligned} y &\sim \text{Bin}(n, \theta) \\ \theta &\sim \text{Beta}(2, 2), \end{aligned} \tag{2.1}$$

dove la prima riga definisce la funzione di verosimiglianza e la seconda riga definisce la distribuzione a priori. Vediamo ora come specificare tale modello beta-binomiale in linguaggio Stan.

## 2.3 Interfaccia `cmdstanr`

Nella seguente discussione useremo l'interfaccia `cmdstanr` di CmdStan.[2]. Carichiamo i pacchetti necessari:

```
library("cmdstanr")
library("posterior")
```

[2]I modelli discussi in questo capitolo sono discussi da Gelman et al. (1995) mentre il codice è stato ricavato dalla seguente pagina web[3].

```
rstan_options(auto_write = TRUE) # avoid recompilation of models
# parallelize across all CPUs
options(mc.cores = parallel::detectCores())
# improve execution time
Sys.setenv(LOCAL_CPPFLAGS = '-march=native')
SEED <- 374237 # set random seed for reproducibility
```

Per svolgere l'analisi mediante `cmdstanr` è necessario prima specificare la struttura del modello bayesiano nella notazione Stan e, poi, eseguire il campionamento dalla distribuzione a posteriori. Esaminiamo questi due passaggi per il caso dell'esempio presente.

### 2.3.1 Fase 1

Nella prima fase dell'analisi dobbiamo definire i dati, i parametri e il modello. I *dati* devono essere contenuti in un oggetto di classe `list`. Per l'esempio presente abbiamo:

```
data1_list <- list(
  N = 16,
  y = c(rep(1, 14), rep(0, 2))
)
```

Il *modello* è Binomial$(n, \theta)$ e, in linguaggio Stan, può essere specificato come

```
y ~ bernoulli(theta);
```

Il modello dipende dal *parametro* `theta`. In Stan, dobbiamo specificare che `theta` può essere un qualsiasi numero reale compreso tra 0 e 1. Inoltre, dobbiamo specificare la distribuzione a priori di $\theta$. Per questo parametro abbiamo scelto una Beta$(2, 2)$ e, in linguaggio Stan, scriviamo

```
theta ~ beta(2, 2);
```

Memorizziamo ora il modello beta-binomiale specificato in linguaggio Stan come stringa di caratteri nel file `oneprop.stan`:

```
modelString = "
data {
  int<lower=0> N;
  int<lower=0, upper=1> y[N];
}
parameters {
  real<lower=0, upper=1> theta;
}
model {
  theta ~ beta(2, 2);
  y ~ bernoulli(theta);
}
"
writeLines(modelString, con = "code/oneprop.stan")
```

### 2.3.2 Fase 2

Per utilizzare il modello che abbiamo specificato, leggiamo l'indirizzo del file che contiene il codice Stan:

```
file <- file.path("code", "oneprop.stan")
```

Compiliamo il codice

```
mod <- cmdstan_model(file)
```

ed eseguiamo il campionamento MCMC con la chiamata

```
fit1 <- mod$sample(
  data = data1_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = 84735,
  chains = 4L,
  parallel_chains = 4L,
  refresh = 0,
```

```
  thin = 1
)
```

Avendo assunto una distribuzione a priori per il parametro $\theta$, l'algoritmo procede in maniera ciclica, correggendo la distribuzione a priori di $\theta$ condizionandola ai valori già generati. Dopo un certo numero di cicli, necessari per portare l'algoritmo a convergenza, i valori estratti possono essere assunti come campionati dalla distribuzione a posteriori di $\theta$.

Si noti che `$sample()` richiede due tipi di informazioni. Innanzitutto, dobbiamo specificare le informazioni sul modello in base a:

- `mod` = la stringa di caratteri che definisce il modello (qui `oneprop.stan`),
- `data` = i dati in formato lista (`data1_list`).

Dobbiamo inoltre specificare le informazioni sul campionamento MCMC utilizzando tre argomenti aggiuntivi:

- L'argomento `chains` specifica quante catene di Markov parallele eseguire. Eseguiamo qui quattro catene, quindi otteniamo quattro campioni distinti di valori $\pi$.
- L'argomento `iter` specifica il numero desiderato di iterazioni o la lunghezza di ciascuna catena di Markov. Per impostazione predefinita, la prima metà di queste iterazioni è costituita da campioni "burn-in" o "warm-up" che verranno ignorati. La seconda metà è conservata e costituisce un campione della distribuzione a posteriori.
- L'argomento `seed` per impostare il numero casuale che genera il seme per una simulazione `cmdstanr`.

### 2.3.3 Burn-in

Al crescere del numero di passi della catena, la distribuzione di target viene sempre meglio approssimata. All'inizio del campionamento, però, la distribuzione può essere significativamente lontana da quella stazionaria, e ci vuole un certo tempo prima di raggiungere la distribuzione stazionaria di equilibrio, detto, appunto, periodo di *burn-in*. I campioni provenienti da tale parte iniziale della catena vanno tipicamente scartati perché possono non rappresentare accuratamente la distribuzione a posteriori

### 2.3.4 Inferenza

Un sommario della distribuzione a posteriori si ottiene con:

```
fit1$summary(c("theta"))
#> # A tibble: 1 x 10
#>   variable  mean median     sd    mad    q5   q95  rhat
#>   <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>
#> 1 theta    0.798  0.808 0.0876 0.0872 0.638 0.924  1.00
#> # ... with 2 more variables: ess_bulk <dbl>,
#> #   ess_tail <dbl>
```

Creiamo un oggetto di classe `stanfit`

```
stanfit1 <- rstan::read_stan_csv(fit1$output_files())
```

di dimensioni

```
dim(as.matrix(stanfit1, pars = "theta"))
#> [1] 16000     1
```

I primi 10 valori sono presentati qui di seguito

```
as.matrix(stanfit1, pars = "theta") %>%
  head(10)
#>           parameters
#> iterations  theta
#>       [1,] 0.8521
#>       [2,] 0.7845
#>       [3,] 0.7845
#>       [4,] 0.7551
#>       [5,] 0.7256
#>       [6,] 0.7744
#>       [7,] 0.7744
#>       [8,] 0.8062
#>       [9,] 0.8266
#>      [10,] 0.8499
```

La matrice precedente include i valori assunti dalla catena di Markov, ovvero un insieme di valori plausibili $\theta$ estratti dalla distribuzione a posteriori. Un tracciato della catena di Markov illustra questa esplorazione rappresentando il valore $\theta$ sulle ordinate e l'indice progressivo di in ogni iterazione sull'ascissa. Usiamo la funzione `mcmc_trace()` del pacchetto `bayesplot` (Gabry et al. 2019) per costruire il grafico che include tutte e quattro le catene di Markov:

```
stanfit1 %>%
  mcmc_trace(pars = c("theta"), size = 0.1)
```

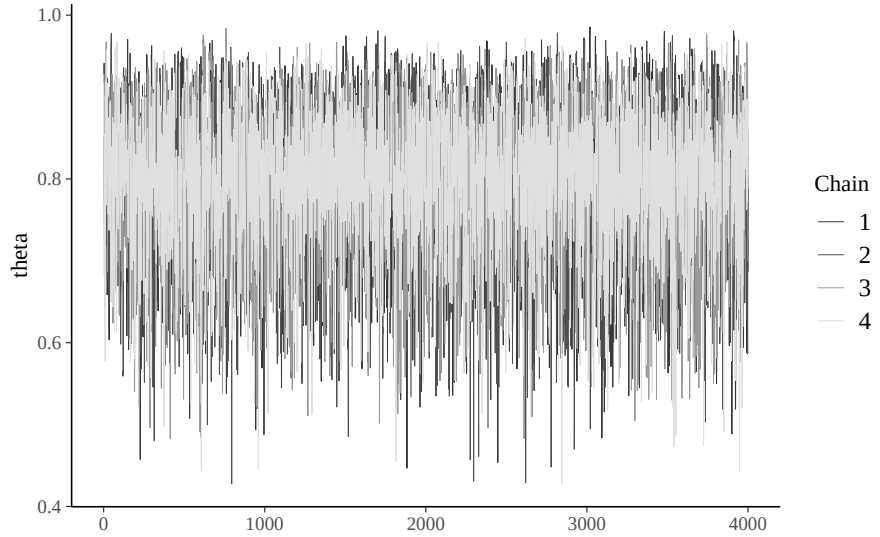



***Figura 2.1:*** *Trace-plot per il parametro $\theta$ nel modello Beta-Binomiale.*

La figura 2.1 mostra che le catene esplorano uno spazio compreso approssimativamenre tra 0.7 e 0.9; tale figura descrive il comportamento *longitudinale* delle catene di Markov.

Possiamo anche esaminare la distribuzione degli stati della catena di Markov, ovvero, dei valori che queste catene visitano lungo il loro percorso, ignorando l'ordine di queste visite. L'istogramma della figura 2.2 fornisce una rappresentazione grafica di questa distribuzione per i 16000 valori complessivi delle quattro catene, ovvero per 4000 valori provienienti da ciascuna catena.

```
mcmc_hist(stanfit1, pars = "theta") +
  yaxis_text(TRUE) +
  ylab("count")
```

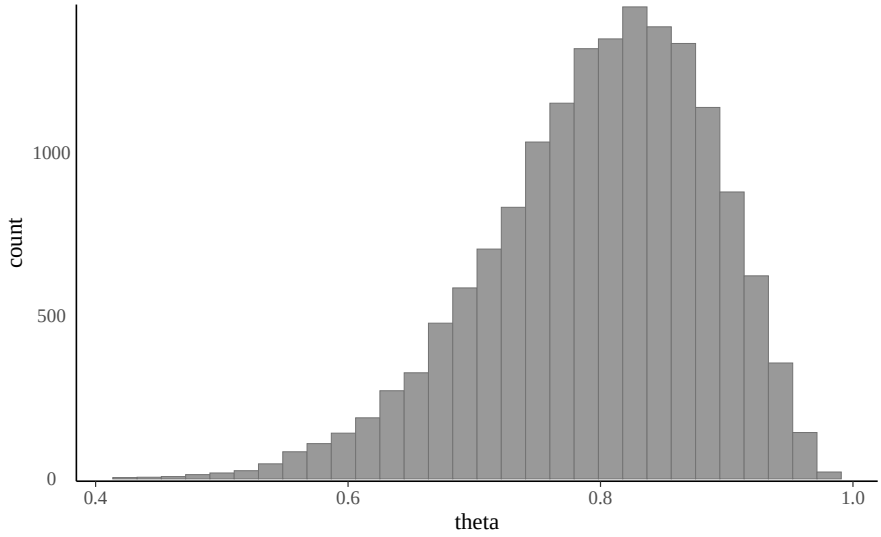



***Figura 2.2:*** *Istogramma che illustra l'approssimazione della distribuzione a posteriori per il parametro $\theta$ nel modello Beta-Binomiale.*

Nel modello Beta-Binomiale in cui la verosimiglianza è binomiale con 14 successi su 16 prove e in cui assumiamo una distribuzione a priori di tipo Beta$(2, 2)$ sul parametro $\theta$, la distribuzione a posteriori è ancora una distribuzione Beta di parametri $\alpha = 2 + 14$ e $\beta = 2 + 16$ - 14. La figura 2.3 riporta un kernel density plot per i valori delle quattro catene di Markov con sovrapposta in nero la densità Beta$(16, 4)$. Il punto importante è che la distribuzione dei valori delle catene di Markov produce un'eccellente approssimazione alla distribuzione bersaglio.[4]

[4]Nel caso presente, il risultato è poco utile dato che è disponibile una soluzione analitica. Tuttavia, questo esercizio mette in evidenza il fatto cruciale che, nei casi in cui possiamo verificarne la soluzione, il campionamento Monte Carlo a catena di Markov è in grado di trovare la risposta corretta. Di conseguenza, possiamo anche essere sicuri che fornirà un'approssimazione alla distribuzione a posteriori anche in quei casi in cui una soluzione analitica non è disponibile.

```
mcmc_dens(stanfit1, pars = "theta") +
  yaxis_text(TRUE) +
  ylab("density") +
  stat_function(fun = dbeta, args = list(shape1 = 16, shape2=4))
```

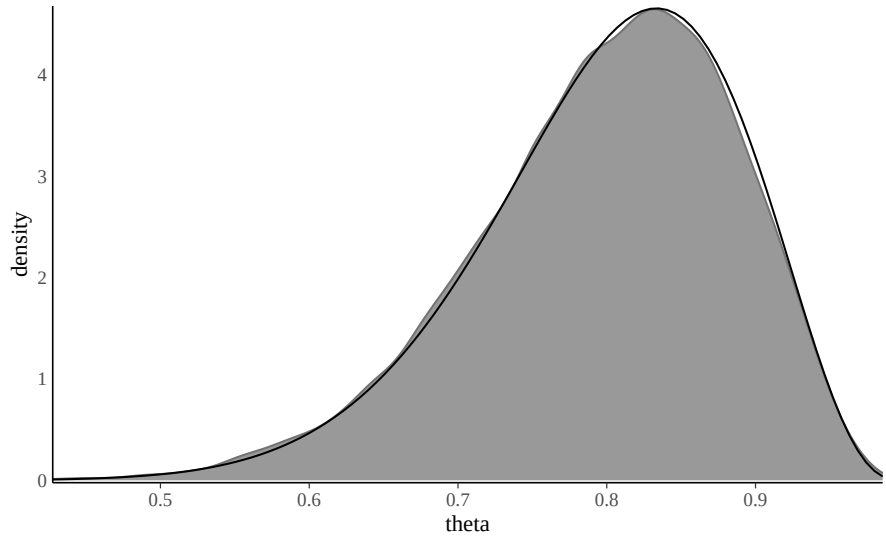



***Figura 2.3:*** *Istogramma che illustra l'approssimazione della distribuzione a posteriori per il parametro $\theta$ nel modello Beta-Binomiale. La curva nera rappresenta la corretta distribuzione a posteriori Beta(16, 4).*

Un intervallo di credibilità al 95% per $\theta$ si ottiene con la seguente chiamata:

```
posterior1 <- extract(stanfit1)
rstantools::posterior_interval(as.matrix(stanfit1), prob = 0.95)
#>          2.5%    97.5%
#> theta   0.599   0.9376
#> lp__  -12.582 -10.0086
```

Svolgendo un'analisi bayesiana simile a questa, Gautret et al. (2020) hanno trovato che gli intervalli di credibilità del gruppo di controllo e del gruppo sperimentale non si sovrappongono. Questo fatto viene interpretato dicendo che il parametro $\theta$ è diverso nei due gruppi. Sulla base di queste evidenza, Gautret et al. (2020) hanno concluso, con un

grado di certezza soggettiva del 95%, che nel gruppo sperimentale vi è una probabilità più bassa di risultare positivi al SARS-CoV-2 rispetto al gruppo di controllo. In altri termini, questa analisi dei dati suggerisce che l'idrossiclorochina sia efficace come terapia per il Covid-19.

## 2.4 La critica di Hulme et al. (2020)

Un articolo pubblicato da Hulme et al. (2020) si è posto il problema di rianalizzare i dati di Gautret et al. (2020).[5] Tra gli autori di questo articolo figura anche Eric-Jan Wagenmakers, uno psicologo molto conosciuto per i suoi contributi metodologici. Hulme et al. (2020) hanno osservato che, nelle analisi statistiche riportate, Gautret et al. (2020) hanno escluso alcuni dati. Nel gruppo sperimentale, infatti, vi erano alcuni pazienti i quali, anziché migliorare, sono in realtà peggiorati. L'analisi statistica di Gautret et al. (2020) ha escluso i dati di questi pazienti. Se consideriamo tutti i pazienti — non solo quelli selezionati da Gautret et al. (2020) — la situazione diventa la seguente:

- gruppo sperimentale: 10 positivi su 18;
- gruppo di controllo: 14 positivi su 16.

L'analisi dei dati proposta da Hulme et al. (2020) richiede l'uso di alcuni strumenti statistici che, in queste dispense, non verranno discussi. Ma possiamo giungere alle stesse conclusioni raggiunte da questi ricercatori anche usando le procedure statistiche descritte nel Paragrafo successivo.

## 2.5 Due proporzioni

Svolgiamo ora l'analisi considerando tutti i dati, come suggerito da Hulme et al. (2020). Per fare questo verrà creato un modello bayesiano per fare inferenza sulla differenza tra due proporzioni. Una volta generate le distribuzioni a posteriori per le proporzioni di "successi" nei due gruppi, verrà anche generata la quantità

[5] Si veda https://osf.io/5dgmx/.

$$\omega = \frac{\theta_2/(1-\theta_2)}{\theta_1/(1-\theta_1)},$$

ovvero il rapporto tra gli Odds di positività tra i pazienti del gruppo di controllo e gli Odds di positività tra i pazienti del gruppo sperimentale. Se il valore dell'OR è uguale a 1, significa che l'Odds di positività nel gruppo di controllo è uguale all'odds di positività nel gruppo sperimentale, cioè il fattore in esame (somministrazione dell'idrossiclorochina) è ininfluente sulla comparsa della malattia. L'inferenza statistica sull'efficacia dell'idrossiclorochina come terapia per il Covid-19 può dunque essere effettuata esaminando l'intervallo di credibilità al 95% per l'OR: se tale intervallo include il valore 1, allora non vi è evidenza che l'idrossiclorochina sia efficace come terapia per il Covid-19.

Nell'implementazione di questo modello, la quantità di interesse è dunque l'odds ratio; tale quantità viene calcolata nel blocco `generated quantities` del programma Stan. In questo esempio useremo delle distribuzioni a priori vagamente informative per i parametri $\theta_1$ e $\theta_1$.

```
data_list <- list(
  N1 = 18,
  y1 = 10,
  N2 = 16,
  y2 = 14
)
```

```
modelString = "
//  Comparison of two groups with Binomial
data {
  int<lower=0> N1;              // number of experiments in group 1
  int<lower=0> y1;              // number of deaths in group 1
  int<lower=0> N2;              // number of experiments in group 2
  int<lower=0> y2;              // number of deaths in group 2
}
parameters {
  real<lower=0,upper=1> theta1; // probability of death in group 1
  real<lower=0,upper=1> theta2; // probability of death in group 2
}
model {
```

```
  theta1 ~ beta(2, 2);          // prior
  theta2 ~ beta(2, 2);          // prior
  y1 ~ binomial(N1, theta1);    // observation model / likelihood
  y2 ~ binomial(N2, theta2);    // observation model / likelihood
}
generated quantities {
  // generated quantities are computed after sampling
  real oddsratio = (theta2/(1-theta2))/(theta1/(1-theta1));
}
"
writeLines(modelString, con = "code/twoprop1.stan")
```

```
file <- file.path("code", "twoprop1.stan")
```

```
mod <- cmdstan_model(file)
```

```
fit <- mod$sample(
  data = data_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = SEED,
  chains = 4L,
  parallel_chains = 4L,
  refresh = 0,
  thin = 1
)
```

```
stanfit <- rstan::read_stan_csv(fit$output_files())
```

```
print(
  stanfit,
  pars = c("theta1", "theta2", "oddsratio"),
  digits_summary = 3L
)
```

```
#> Inference for Stan model: twoprop1-202201270929-1-5c8c8b.
#> 4 chains, each with iter=6000; warmup=2000; thin=1;
#> post-warmup draws per chain=4000, total post-warmup draws=16000.
#>
#>            mean se_mean    sd  2.5%   25%   50%   75%
#> theta1    0.546   0.001 0.103 0.344 0.475 0.546 0.620
#> theta2    0.798   0.001 0.087 0.601 0.744 0.808 0.862
#> oddsratio 4.721   0.043 4.411 0.906 2.166 3.514 5.698
#>            97.5% n_eff Rhat
#> theta1     0.740 12795    1
#> theta2     0.937 14193    1
#> oddsratio 15.558 10400    1
#>
#> Samples were drawn using NUTS(diag_e) at Gio Gen 27 09:29:15 2022.
#> For each parameter, n_eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor on split chains (at
#> convergence, Rhat=1).
```

L'intervallo di credibilità del 95% per l'OR include il valore di 1.0 (ovvero, il valore che indica che gli odds di positività sono uguali nei due gruppi). In base agli standard correnti, un risultato di questo tipo non viene considerato come evidenza sufficiente per potere concludere che il parametro $\theta$ assume un valore diverso nei due gruppi. In altri termini, se consideriamo tutti i dati, e non solo quelli selezionati dagli autori della ricerca originaria, non vi è evidenza alcuna che l'idrossiclorochina sia efficace come terapia per il Covid-19.

### Commenti e considerazioni finali

Concludiamo questa discussione dicendo che ciò che è stato presentato in questo capitolo è un esercizio didattico: la ricerca di Gautret et al. (2020) include tante altre informazioni che non sono state qui considerate. Tuttavia, notiamo che la semplice analisi statistica che abbiamo qui descritto è stata in grado di replicare le conclusioni a cui sono giunti (per altra via) Hulme et al. (2020).

# Parte II

# Appendici

# A

# *Programmare in Stan*

## A.1 Interfacce e pacchetti

È possibile accedere al linguaggio Stan tramite diverse interfacce:

- `CmdStan`: eseguibile da riga di comando,
- `RStan` - integrazione con il linguaggio R;
- `PyStan` - integrazione con il linguaggio di programmazione Python;
- `MatlabStan` - integrazione con MATLAB;
- `Stan.jl` - integrazione con il linguaggio di programmazione Julia;
- `StataStan` - integrazione con Stata.

Inoltre, vengono fornite interfacce di livello superiore con i pacchetti che utilizzano Stan come backend, principalmente in Linguaggio R:

- `shinystan`: interfaccia grafica interattiva per l'analisi della distribuzione a posteriori e le diagnostiche MCMC;
- `bayesplot`: insieme di funzioni utilizzabili per creare grafici relativi all'analisi della distribuzione a posteriori, ai test del modello e alle diagnostiche MCMC;
- `brms`: fornisce un'ampia gamma di modelli lineari e non lineari specificando i modelli statistici mediante la sintassi usata in R;
- `rstanarm`: fornisce un sostituto per i modelli frequentisti forniti da base R e `lme4` utilizzando la sintassi usata in R per la specificazione dei modelli statistici;
- `edstan`: modelli Stan per la Item Response Theory;
- `cmdstanr`, un'interfaccia R per `CmdStan`.

## A.2 Interfaccia `cmdstanr`

Negli esempi di questa dispensa verrà usata l'interfaccia `cmdstanr`. Il pacchetto `cmdstanr` non è ancora disponibile su CRAN, ma può essere installato come indicato su questo link[1]. Una volta che è stato installato, il pacchetto `cmdstanr` può essere caricato come un qualsiasi altro pacchetto R.

Si noti che `cmdstanr` richiede un'installazione funzionante di `CmdStan`, l'interfaccia shell per Stan. Se `CmdStan` non è installato, `cmdstanr` lo installerà automaticamente se il computer dispone di una *Toolchain* adatta. Stan richiede infatti che sul computer su cui viene installato siano presenti alcuni strumenti necessari per gestire i file C++. Tra le altre ragioni, questo è dovuto al fatto che il codice Stan viene tradotto in codice C++ e compilato. Il modo migliore per ottenere il software necessario per un computer Windows o Mac è quello di installare `RTools`. Per un computer Linux, è necessario installare `build-essential` e una versione recente dei compilatori g++ o clang++. I requisiti sono descritti nella Guida di CmdStan[2].

Per verificare che la Toolchain sia configurata correttamente è possibile utilizzare la funzione `check_cmdstan_toolchain()`:

```
check_cmdstan_toolchain()
```

Se la toolchain è configurata correttamente, `CmdStan` può essere installato mediante la funzione `install_cmdstan()`:

```
# do not run!
# install_cmdstan(cores = 2)
```

La versione installata di `CmdStan` si ottiene con:

```
cmdstan_version()
#> [1] "2.28.2"
```

[1] https://mc-stan.org/docs/2_27/cmdstan-guide/cmdstan-installation.html
[2] https://mc-stan.org/docs/cmdstan-guide/cmdstan-installation.html

## A.3 Codice Stan

Qualunque sia l'interfaccia che viene usata, i modelli sottostanti sono sempre scritti nel linguaggio Stan, il che significa che lo stesso codice Stan è valido per tutte le interfacce possibili. Il codice Stan è costituito da una serie di blocchi che vengono usati per specificare un modello statistico. In ordine, questi blocchi sono: `data`, `transformed data`, `parameters`, `transformed parameters`, `model`, e `generated quantities`.

### A.3.1 "Hello, world" – Stan

Quando si studia un nuovo linguaggio di programmazione si utilizza spesso un programma "Hello, world". Questo è un modo semplice, spesso minimo, per dimostrare alcune delle sintassi di base del linguaggio. In Python, il programme "Hello, world" program è:

```
print("Hello, world.")
```

Qui presentiamo Stan e scriviamo un programma "Hello, world" per Stan.

Prima di scrivere il nostro primo programma "Hello, world" per Stan (che estrarrà campioni dalla distribuzione a posteriori di un modello gaussiano) spendiamo due parole per spiegare cosa fa Stan. Un utente scrive un modello usando il linguaggio Stan. Questo è solitamente memorizzato in un file di testo `.stan`. Il modello viene compilato in due passaggi. Innanzitutto, Stan traduce il modello nel file `.stan` in codice C++. Quindi, quel codice C++ viene compilato in codice macchina. Una volta creato il codice macchina, l'utente può, tramite l'interfaccia CmdStan, campionare la distribuzione definita dal modello ed eseguire altri calcoli con il modello. I risultati del campionamento vengono scritti su disco come file CSV e txt. Come mostrato di seguito, l'utente accede a questi file utilizzando varie funzioni R, senza interagire direttamente con loro.

Per iniziare, possiamo dire che un programma Stan contiene tre "blocchi" obbligatori: blocco `data`, blocco `parameters`, blocco `model`.

### A.3.2 Blocco `data`

Qui vengono dichiarate le variabili che saranno passate a Stan. Devono essere elencati i nomi delle variabili che saranno utilizzate nel programma, il *tipo di dati* da registrare per ciascuna variabile, per esempio:

- *int* = intero,
- *real* = numeri reali (ovvero, numeri con cifre decimali),
- *vector* = sequenze ordinate di numeri reali unidimensionali,
- *matrix* = matrici bidimensionali di numeri reali,
- *array* = sequenze ordinate di dati multidimensionali.

Devono anche essere dichiarate le dimensioni delle variabili e le eventuali restrizioni sulle variabili (es. `upper = 1 lower = 0`, che fungono da controlli per Stan). Tutti i nomi delle variabili assegnate qui saranno anche usati negli altri blocchi del programma.

Per esempio, l'istruzione seguente dichiaria la variabile `Y` – la quale rappresenta, ad esempio, l'altezza di 10 persone – come una variabile di tipo `real[10]`. Ciò significa che specifichiamo un array di lunghezza 10, i cui elementi sono variabili continue definite sull'intervallo dei numeri reali $[-\infty, +\infty]$.

```
data {
  real Y[10]; // heights for 10 people
}
```

Invece, con l'istruzione

```
data {
  int Y[10]; // qi for 10 people
}
```

dichiariamo la variabile `Y` – la quale rappresenta, ad esempio, il QI di 10 persone – come una variabile di tipo `int[10]`, ovvero un array di lunghezza 10, i cui elementi sono numeri naturali, cioè numeri interi non negativi $\{0, +1, +2, +3, +4, ...\}$.

Un altro esempio è

```
data {
  real<lower=0, upper=1> Y[10]; // 10 proportions
}
```

nel quale viene specificato un array di lunghezza 10, i cui elementi sono delle variabili continue definite sull'intervallo dei numeri reali $[0, 1]$ — per esempio, delle proporzioni.

Si noti che i tipi `vector` e `matrix` contengono solo elementi di tipo `real`, ovvero variabili continue, mentre gli `array` possono contenere dati di qualsiasi tipo. I dati passati a Stan devono essere contenuti in un oggetto del tipo `list`.

### A.3.3 Blocco `parameters`

I parametri che vengono stimati sono dichiarati nel blocco `parameters`. Per esempio, l'istruzione

```
parameters {
  real mu; // mean height in population
  real<lower=0> sigma; // sd of height distribution
}
```

dichiaria la variabile `mu` che codifica l'altezza media nella popolazione, che è una variabile continua in un intervallo illimitato di valori, e la deviazione standard `sigma`, che è una variabile continua non negativa. Avremmo anche potuto specificare un limite inferiore di zero su `mu` perché deve essere non negativo.

Per una regressione lineare semplice, ad esempio, devono essere dichiarate le variabili corrispondenti all'intercetta (`alpha`), alla pendenza (`beta`) e alla deviazione standard degli errori attorno alla linea di regressione (`sigma`). In altri termini, nel blocco `parameters` devono essere elencati tutti i parametri che dovranno essere stimati dal modello. Si noti che parametri discreti non sono possibili. Infatti, Stan attualmente non supporta i parametri con valori interi, almeno non direttamente.

### A.3.4 Blocco `model`

Nel blocco `model` vengono elencate le dichiarazioni relative alla verosimiglianza dei dati e alle distribuzioni a priori dei parametri, come ad esempio, nelle istruzioni seguenti.

```
model {
  for(i in 1:10) {
    Y[i] ~ normal(mu, sigma);
  }
  mu ~ normal(170, 15); // prior for mu
  sigma ~ cauchy(0, 20); // prior for sigma
}
```

Mediante l'istruzione all'interno del ciclo `for`, ciascun valore dell'altezza viene concepito come una variable casuale proveniente da una distribuzione Normale di parametri $\mu$ e $\sigma$ (i parametri di interesse nell'inferenza). Il ciclo `for` viene ripetuto 10 volte perché i dati sono costituiti da un array di 10 elementi (ovvero, il campione è costituito da 10 osservazioni).

Le due righe che seguno il ciclo `for` specificano le distribuzioni a priori dei parametri su cui vogliamo effettuare l'inferenza. Per $\mu$ assumiamo una distribuzione a priori Normale di parametri $\mu = 170$ e $\sigma = 15$; per $\sigma$ assumiamo una distribuzione a priori Cauchy(0, 20).

Se non viene definita alcuna distribuzione a priori, Stan utilizzerà la distribuzione a priori predefinita $Unif(-\infty, +\infty)$. Raccomandazioni sulle distribuzioni a priori sono fornite in questo link[3].

La precedente notazione di campionamento può anche essere espressa usando la seguente notazione alternativa:

```
  for(i in 1:10) {
    target += normal_lpdf(Y[i] | mu, sigma);
  }
```

Questa notazione rende trasparente il fatto che, in pratica, Stan esegue un campionamento nello spazio

[3] https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations

$$\log p(\theta \mid y) \propto \log p(y \mid \theta) + \log p(\theta) = \sum_{i=1}^{n} \log p(y_i \mid \theta) + \log p(\theta).$$

Per ogni passo MCMC, viene ottenuto un nuovo valore di $\mu$ e $\sigma$ eviene valutata la log densità a posteriori non normalizzata. Ad ogni passo MCMC, Stan calcola un nuovo valore della densità a posteriori su scala logaritmica partendo da un valore di 0 e incrementandola ogni volta che incontra un'istruzione ~. Quindi, le istruzioni precedenti aumentano la log-densità di una quantità pari a $\log(p(Y[i])) \propto -\frac{1}{2}\log(\sigma^2) - (Y[i] - \mu)^2/2\sigma^2$ per le altezze si ciascuno degli $i = 1 \dots, 10$ individui – laddove la formula esprime, in termini logaritmici, la densità Normale da cui sono stati esclusi i termini costanti.

Oppure, in termini vettorializzati, il modello descritto sopra può essere espresso come

```
model {
  Y ~ normal(mu, sigma);
}
```

dove il termine a sinistra di $\sim$ è un array. Questa notazione più compatta è anche la più efficiente.

### A.3.5 Blocchi opzionali

Ci sono inoltre tre blocchi opzionali:

- Il blocco `transformed data` consente il pre-processing dei dati. È possibile trasformare i parametri del modello; solitamente ciò viene fatto nel caso dei modelli più avanzati per consentire un campionamento MCMC più efficiente.
- Il blocco `transformed parameters` consente la manipolazione dei parametri prima del calcolo della distribuzione a posteriori.
- Il blocco `generated quantities` consente il post-processing riguardante qualsiasi quantità che non fa parte del modello ma può essere calcolata a partire dai parametri del modello, per ogni iterazione dell'algoritmo. Esempi includono la generazione dei campioni a posteriori e le dimensioni degli effetti.

### A.3.6 Sintassi

Si noti che il codice Stan richiede i punti e virgola (;) alla fine di ogni istruzione di assegnazione. Questo accade per le dichiarazioni dei dati, per le dichiarazioni dei parametri e ovunque si acceda ad un elemento di un tipo `data` e lo si assegni a qualcos'altro. I punti e virgola non sono invece richiesti all'inizio di un ciclo o di un'istruzione condizionale, dove non viene assegnato nulla.

In STAN, qualsiasi stringa che segue `//` denota un commento e viene ignorata dal programma.

Stan è un linguaggio estremamente potente e consente di implementare quasi tutti i modelli statistici, ma al prezzo di un certo sforzo di programmazione. Anche l'adattamento di semplici modelli statistici mediante il linguaggio STAN a volte può essere laborioso. Per molti modelli comunemente usati, come i modelli di regressione e multilivello, tale processo può essere semplificato usando le funzioni del pacchetto `brms`. D'altra parte, per modelli veramente complessi, non ci sono molte alternative all'uso di STAN. Per chi è curioso, il manuale del linguaggio Stan è accessibile al seguente link[4].

## A.4 Workflow

Se usiamo `cmdstanr`, dobbiamo prima scrivere il codice con il modello statistico in un file in formato Stan. È necessario poi "transpile" quel file, ovvero tradurre il file in C++ e compilarlo. Ciò viene fatto mediante la funzione `cmdstan_model()`. Possiamo poi eseguire il campionamento MCMC con il metodo `$sample()`. Infine è possibile creare un sommario dei risultati usando, per esempio, usando il metodo `$summary()`.

## A.5 Ciao, Stan

Scriviamo ora il nostro programma Stan "Hello, world" per generare campioni da una distribuzione Normale standard (con media zero e varianza

[4] https://mc-stan.org/docs/2_27/stan-users-guide/index.html

unitaria).

```
modelString = "
parameters {
  real x;
}
model {
  x ~ normal(0, 1);
}
"
writeLines(modelString, con = "code/hello_world.stan")
```

Si noti che ci sono solo due blocchi in questo particolare codice Stan, il blocco parametri e il blocco modello. Questi sono due dei sette blocchi possibili in un codice Stan. Nel blocco parametri, abbiamo i nomi e i tipi di parametri per i quali vogliamo ottenere i campioni. In questo caso, vogliamo ottenere campioni di numeri reale che chiamiamo `x`. Nel blocco modello, abbiamo il nostro modello statistico. Specifichiamo che x, il parametro di cui vogliamo ottenere i campioni, è normalmente distribuito con media zero e deviazione standard unitaria. Ora che abbiamo il nostro codice (che è stato memorizzato in un file chiamato `hello_world.stan`), possiamo usare CmdStan per compilarlo e ottenere `mod`, che è un oggetto R che fornisce l'accesso all'eseguibile Stan compilato.

Leggiamo il file in cui abbiamo salvato il codice Stan

```
file <- file.path("code", "hello_world.stan")
```

compiliamo il modello

```
mod <- cmdstan_model(file)
```

ed eseguiamo il campionamento MCMC:

```
fit <- mod$sample(
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = SEED,
```

```
  chains = 4L,
  refresh = 0,
  thin = 1
)
```

Tasformiamo l'oggetto `fit` nel formato `stanfit` per manipolarlo più facilmente:

```
stanfit <- rstan::read_stan_csv(fit$output_files())
```

Lo esaminiamo

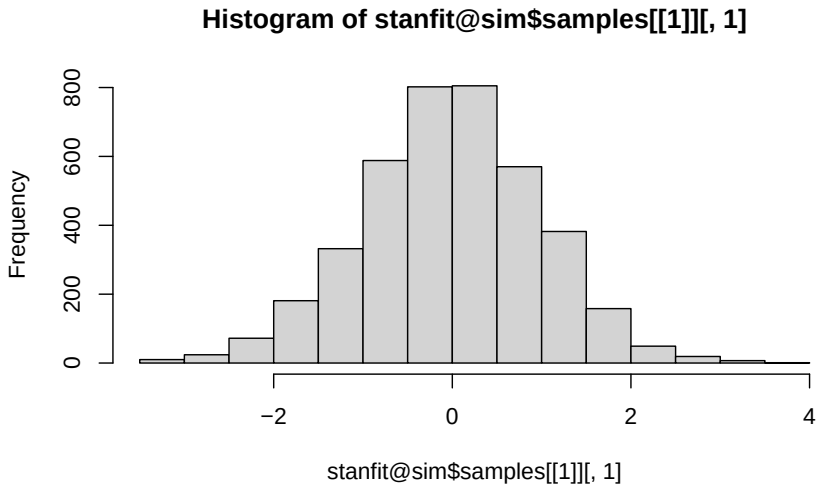
```
length(stanfit@sim$samples)
#> [1] 4
```

Quello che abbiamo ottenuto sono 4 catene di 4000 osservazioni ciascuna, le quali contengono valori casuali estratti dalla gaussiana standardizzata:

```
head(stanfit@sim$samples[[1]])
```

Verifichiamo

```
hist(stanfit@sim$samples[[1]][, 1])
```


Histogram of stanfit@sim$samples[[1]][, 1]
Frequency
0
200
400
600
800
−2
0
2
4
stanfit@sim$samples[[1]][, 1]

# *Bibliografia*


Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1):1–32.

Eckhardt, R. (1987). Stan Ulam, John Von Neumann and the Monte Carlo Method. *Los Alamos Science Special Issue*.

Gautret, P., Lagier, J. C., Parola, P., Meddeb, L., Mailhe, M., Doudier, B., and ... Honoré, S. (2020). Hydroxychloroquine and azithromycin as a treatment of covid-19: Results of an open-label non-randomized clinical trial. *International Journal of Antimicrobial Agents*.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian data analysis*. Chapman and Hall/CRC.

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

Hulme, O. J., Wagenmakers, E. J., Damkier, P., Madelung, C. F., Siebner, H. R., Helweg-Larsen, J., and ... Madsen, K. H. (2020). Reply to gautret et al. 2020: A bayesian reanalysis of the effects of hydroxychloroquine and azithromycin on viral carriage in patients with covid-19. *medRxiv*.

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

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.