Corrado Caudek

Psicometria

Anno Accademico 2024/2025

Indice

Benvenuti

Benvenuti nel sito web dell'insegnamento di Psicometria, parte del Corso di Laurea in Scienze e Tecniche Psicologiche dell'Università degli Studi di Firenze.

Descrizione

Il corso offre una formazione teorico-pratica nell'analisi dei dati psicologici, con particolare attenzione alle applicazioni in R. Vengono affrontati temi come analisi descrittiva, esplorazione dei dati, flusso di lavoro, organizzazione di progetti e modelli statistici di base, sia bayesiani che frequentisti. Grande enfasi è posta sulle buone pratiche di Open Science, promuovendo trasparenza e riproducibilità nelle analisi.

- **Anno Accademico**: 2024-2025
- **Codice Insegnamento**: B000286
- **Orario e Luogo**: Lunedì e Martedì (8:30-10:30), Giovedì (11:30-13:30), Plesso didattico La Torretta.

> **i** Nota
>
> Questo sito web ospita la dispensa ufficiale dell'insegnamento **B000286 - Psicometria**, che include tutte le note e i materiali relativi alle lezioni.

Struttura dell'Insegnamento

Il corso è strutturato per fornire una solida base teorica e pratica nell'analisi dei dati psicologici, combinando approcci metodologici, applicazioni pratiche ed esercitazioni mirate.

- **Introduzione a R**: Approccio pratico all'utilizzo di R per l'analisi dei dati, con enfasi sulla scrittura di script replicabili e sulla creazione di

workflow efficienti.

- **Gestione di Progetti di Analisi**: Strategie per organizzare, documentare e comunicare progetti di data analysis, seguendo le buone pratiche della Open Science e della ricerca trasparente.
- **Statistica Descrittiva**: Esplorazione dei dati attraverso misure descrittive, distribuzioni statistiche e visualizzazioni grafiche.
- **Fondamenti di Probabilità**: Introduzione ai concetti di probabilità, distribuzioni e incertezza, come base essenziale per l'inferenza statistica.
- **Inferenza Frequentista e Bayesiana**: Panoramica sui due principali approcci all'inferenza statistica, con esempi pratici e confronto tra i metodi.
- **Visualizzazione e Comunicazione**: Tecniche avanzate per rappresentare risultati statistici in modo chiaro, efficace e persuasivo, includendo la creazione di report e grafici professionali.

Il corso integra questi argomenti in un percorso didattico coerente, che combina lezioni teoriche, esercitazioni pratiche e momenti di riflessione critica. Gli studenti saranno così preparati ad applicare l'analisi dei dati sia in contesti accademici che pratici.

Syllabus

Consulta il syllabus completo per ulteriori dettagli.

Prefazione

Come possiamo migliorare l'analisi dei dati psicologici per renderla più affidabile e robusta? È possibile affrontare questa sfida semplicemente applicando una serie di algoritmi o procedure standard? L'analisi dei dati in psicologia può davvero essere ridotta a un insieme di "ricette" preconfezionate (**?**)?

Queste domande ci portano a riflettere sulla natura stessa dell'analisi dei dati psicologici. A differenza di ciò che suggerisce l'approccio frequentista del test dell'ipotesi nulla, l'analisi dei dati non è una disciplina che si esaurisce con l'applicazione meccanica di metodi predefiniti. Anzi, considerare l'analisi dei dati come un insieme di procedure automatiche contribuisce a uno dei problemi più gravi della psicologia contemporanea: la crisi della replicabilità dei risultati (**?**).

Ma perché la replicabilità è così cruciale? Se i risultati delle ricerche psicologiche non sono replicabili, significa che la nostra comprensione dei fenomeni psicologici è superficiale e inaffidabile. Questo non è solo un problema teorico o accademico; ha implicazioni dirette sulle applicazioni pratiche della psicologia. Se le basi scientifiche sono incerte, anche le strategie di intervento psicologico rischiano di essere inefficaci o addirittura dannose (**?**; **?**; **?**; **?**).

Perché le pratiche di analisi dei dati derivanti dal frequentismo potrebbero contribuire a questa crisi? In che modo gli incentivi accademici influenzano la qualità della ricerca psicologica? E, soprattutto, quali alternative abbiamo per migliorare l'affidabilità e la validità delle nostre conclusioni?

L'analisi bayesiana emerge come una delle proposte per superare i limiti dell'approccio frequentista (**?**). Tuttavia, è sufficiente abbandonare l'inferenza frequentista per risolvere i problemi della psicologia? Come possiamo integrare metodi robusti e flessibili, come quelli bayesiani, con una comprensione più approfondita e trasparente dei fenomeni psicologici?

In questo corso, esploreremo queste domande, cercando di identificare le "buone pratiche" dell'analisi dei dati psicologici. Discuteremo i limiti delle metodologie attuali, esamineremo le cause sottostanti della crisi della replicabilità e valuteremo come l'adozione di metodi avanzati, come l'inferenza bayesiana e la modellazione causale, possa offrire soluzioni efficaci (**?**; **?**; **?**). Il nostro obiettivo è fornire una visione critica e costruttiva, che non solo identifichi le sfide della ricerca psicologica, ma proponga anche percorsi concreti per migliorare la qualità e l'affidabilità della scienza psicologica.

Bibliografia

Parte I

Calendario

Programma didattico e dettaglio del corso

Il corso comprende un totale di **32 incontri**, pianificati per affrontare tutti gli argomenti del programma. Tre incontri in presenza saranno riservati agli **esami parziali** dedicati agli studenti frequentanti.

Tabella 0.1: Calendario Didattico AA 2024-2025

Incontro	Data	Argomento
1	03 Marzo 2025	Presentazione del corso, obiettivi e fondamenti dell'analisi dei dati
2	04 Marzo 2025	Teoria della misurazione e introduzione a R (parte I)
3	06 Marzo 2025	Introduzione a R (parte II)
4	10 Marzo 2025	Introduzione a R (parte III)
5	11 Marzo 2025	Exploratory Data Analysis (EDA): concetti e applicazioni
6	13 Marzo 2025	Statistica descrittiva: sintesi numeriche e grafiche
7	17 Marzo 2025	Relazioni tra variabili: associazioni e causalità
8	18 Marzo 2025	Elementi di teoria della probabilità: insiemi e calcolo combinatorio
9	20 Marzo 2025	Probabilità condizionata: concetti chiave e applicazioni pratiche
10	24 Marzo 2025	Introduzione alle variabili casuali e alle loro proprietà
11	25 Marzo 2025	Esame parziale 1 (argomenti 1-7)
12	27 Marzo 2025	Il teorema di Bayes: principi e applicazioni
13	31 Marzo 2025	Distribuzioni campionarie: concetti e utilizzi
14	01 Aprile 2025	Distribuzioni di probabilità congiunte e densità marginali
15	03 Aprile 2025	Distribuzioni di massa di probabilità: definizioni e esempi
16	07 Aprile 2025	Distribuzioni di densità di probabilità: utilizzo nell'analisi statistic
17	08 Aprile 2025	La funzione di verosimiglianza: interpretazione e calcolo
18	10 Aprile 2025	Introduzione all'inferenza bayesiana: metodi numerici e approssim
19	14 Aprile 2025	Famiglie coniugate e sintesi a posteriori: esempi pratici
20	15 Aprile 2025	Algoritmo di Metropolis e linguaggi probabilistici (PPL)
21	28 Aprile 2025	Modelli di regressione frequentista: concetti fondamentali
22	29 Aprile 2025	Esame parziale 2 (argomenti 8-19)
23	05 Maggio 2025	Modelli di regressione bayesiana: vantaggi e approcci
24	06 Maggio 2025	Inferenza bayesiana su una media e confronto tra campioni indipe
25	08 Maggio 2025	Analisi della varianza (ANOVA) a una e due vie

26	12 Maggio 2025	Distribuzione campionaria nell'inferenza frequentista
27	13 Maggio 2025	Intervalli di fiducia: costruzione e interpretazione
28	15 Maggio 2025	Test di ipotesi frequentista: metodologia e limiti
29	19 Maggio 2025	Confronto tra medie e proporzioni di gruppi indipendenti
30	20 Maggio 2025	Esercitazione: sviluppo completo di un progetto di analisi dei dati
31	22 Maggio 2025	Crisi della replicazione: cause e implicazioni metodologiche
32	26 Maggio 2025	Open Science: principi, pratiche e strumenti
33	27 Maggio 2025	Esame parziale 3 (argomenti 20-32)

Calendario delle Relazioni in Itinere

Le **relazioni di avanzamento del progetto** per il **Tirocinio Pratico Valutativo (TPV) PSIC-01/C – Psicometria** dovranno essere consegnate rispettando le scadenze previste. Ogni gruppo TPV è tenuto a presentare un unico elaborato, risultato del lavoro collaborativo tra i membri del gruppo.

Tabella 0.2: Calendario delle Relazioni in Itinere

Data di Scadenza	Contenuto della Relazione
24 marzo	Relazione 1: Importazione dei dati, data wrangling, data tidying, dizionario dei dati, statistiche descrittive
28 aprile	Relazione 2: Priori coniugati e metodo basato su griglia
5 maggio	Relazione 3: Regressione lineare, inferenza bayesiana su una media

Ogni relazione costituisce una fase cruciale nello sviluppo del progetto del TPV e fornisce le basi per la **presentazione finale**, che si terrà durante l'esame conclusivo del TPV.

Parte II

Fondamenti

La data science è un campo che si sviluppa all'intersezione tra la statistica e l'informatica. La statistica fornisce una serie di metodologie per analizzare i dati e ottenere informazioni significative, mentre l'informatica si occupa dello sviluppo di software e strumenti per implementare tali metodologie. In questa sezione della dispensa, approfondiremo alcuni concetti fondamentali della statistica e della misurazione psicologica. Considereremo anche in termini generali quali sono gli obiettivi e i limiti dell'analisi dei dati psicologici.

1

Concetti chiave

In questo capitolo apprenderai:

- la definizione di popolazione e campione;
- la distinzione tra variabili indipendenti e dipendenti;
- la struttura e l'importanza della matrice dei dati;
- l'effetto delle variabili all'interno delle analisi statistiche;
- I concetti fondamentali di stima e inferenza;
- il significato e l'applicazione dei modelli psicologici.

Prerequisiti

- Leggere Horoscopes. L'ultimo capitolo di McElreath (**?**) discute il contesto scientifico e culturale della statistica.
- Leggere The Effect: An Introduction to Research Design and Causality. Focalizzati sul capitolo 10 *Treatment Effects*.

1.1 Introduzione

Nella ricerca scientifica, la formulazione di risposte a specifiche domande di indagine avviene attraverso l'applicazione di metodologie rigorose e l'esecuzione di osservazioni accurate e controllate. Le informazioni raccolte mediante diverse tecniche di indagine—come ricerche sul campo, indagini campionarie e protocolli sperimentali—vengono definite con il termine tecnico di **dati**. Questo capitolo introduce i principi fondamentali dell'analisi dei dati, concentrandosi sia sulle caratteristiche dei dati stessi sia sui metodi di raccolta.

Statistica

Il termine "statistica" può assumere diversi significati a seconda del contesto:

- **Primo significato**: La statistica è una scienza che si occupa dello studio e dell'applicazione di metodi per la raccolta, organizzazione, analisi, interpretazione e presentazione dei dati.
- **Secondo significato**: Il termine si riferisce a una misura o valore numerico calcolato a partire da un campione di dati, come la media campionaria, la deviazione standard campionaria o il coefficiente di correlazione campionario.

L'analisi dei dati permette di sintetizzare grandi quantità di informazioni e di verificare le previsioni avanzate dalle teorie. Tuttavia, senza una teoria che dia significato ai dati, le osservazioni rimangono mere descrizioni prive di un contesto esplicativo. È attraverso l'integrazione tra dati e teoria che si raggiunge una comprensione profonda dei fenomeni e si favorisce l'avanzamento scientifico.

1.2 La Spiegazione Scientifica

La scienza non si limita a descrivere o prevedere i fenomeni: il suo obiettivo principale è spiegare il *perché* degli eventi, offrendo una comprensione approfondita delle cause e dei meccanismi che li regolano. La spiegazione scientifica è cruciale per costruire teorie capaci non solo di descrivere e prevedere, ma anche di chiarire le dinamiche causali e le connessioni tra i fenomeni, contribuendo a un controllo più consapevole e informato su di essi.

Consideriamo, ad esempio, il rapporto tra il background familiare e il rendimento scolastico. Numerose ricerche evidenziano una forte correlazione tra il livello di istruzione dei genitori e il successo accademico dei figli. Una prospettiva puramente descrittiva potrebbe limitarsi a constatare che: *"Gli studenti provenienti da famiglie con basso livello di istruzione hanno minori probabilità di conseguire un titolo universitario"*. Tuttavia, la vera sfida scientifica consiste nell'andare oltre questa previsione, ponendosi domande più profonde:

- Quali meccanismi causali determinano questa disparità?
- Quali interventi possono efficacemente ridurre tali disuguaglianze?

Per superare il livello di semplice previsione, la ricerca deve identificare i **fattori causali** alla base del fenomeno, esplorare come l'azione su questi fattori

possa modificare gli esiti e valutare le incertezze e le dinamiche temporali degli interventi.

Nel caso dell'educazione, ciò implica comprendere, ad esempio:

- Se e in che modo il sostegno finanziario possa favorire il percorso degli studenti svantaggiati.
- Quali politiche educative possano produrre effetti positivi sul lungo termine.
- Come i meccanismi sociali e individuali influenzino il processo educativo.

Acquisire una conoscenza approfondita dei meccanismi causali permette di andare oltre la semplice previsione, rendendo possibile la progettazione di interventi mirati e strategici che possano realmente incidere sui fenomeni in modo efficace e duraturo.

1.2.1 Elementi Fondamentali della Spiegazione Scientifica

La filosofia della scienza identifica tre elementi essenziali che caratterizzano una spiegazione scientifica:

- **Explanandum**: il fenomeno da spiegare, ovvero ciò che richiede una comprensione o una giustificazione. Ad esempio, "gli studenti con livelli elevati di ansia da prestazione ottengono punteggi inferiori nei test scolastici rispetto ai loro coetanei."
- **Explanans**: l'insieme di fattori o affermazioni che spiegano il fenomeno osservato. Nel caso dell'ansia da prestazione, l'explanans potrebbe includere: "l'ansia compromette la capacità di concentrazione e memoria di lavoro, influenzando negativamente la prestazione nei test."
- **Legame esplicativo**: i principi o i meccanismi che descrivono come l'explanans produce l'explanandum. Nel nostro esempio, il legame esplicativo potrebbe essere: "elevati livelli di ansia attivano il sistema nervoso simpatico, aumentando lo stress fisiologico e riducendo l'efficienza dei processi cognitivi necessari per svolgere compiti complessi."

Questi tre elementi si integrano nei modelli scientifici, che rappresentano strumenti metodologici per ottenere spiegazioni. I modelli scientifici in psicologia, in particolare, cercano di includere il fenomeno da spiegare, i fattori causali che lo influenzano e i meccanismi sottostanti che collegano cause ed effetti.

Ad esempio, un modello psicologico sull'ansia da prestazione potrebbe incorporare variabili come il livello di ansia percepita, la capacità di regolazione emotiva e la relazione di queste con la memoria di lavoro. Rispetto ai modelli puramente descrittivi o predittivi, tali modelli rispondono a domande causali, permettendo non solo di comprendere il fenomeno, ma anche di progettare interventi per ridurre l'impatto dell'ansia sulla prestazione.

1.3 Modelli Psicologici

Un modello è una rappresentazione matematica e concettuale semplificata di un fenomeno reale. Si basa su un insieme di equazioni e ipotesi che definiscono le relazioni tra variabili e la struttura probabilistica del fenomeno, con l'obiettivo di coglierne gli aspetti essenziali senza includerne ogni dettaglio. Poiché spesso esistono diversi modelli applicabili a uno stesso problema, il compito della ricerca è identificare quello che meglio descrive i dati, rispettando criteri di validità, precisione e parsimonia.

I modelli psicologici, in particolare, sono strumenti teorici per descrivere, spiegare e prevedere il comportamento umano e i processi mentali. Un modello psicologico ben costruito dovrebbe possedere alcune caratteristiche fondamentali:

1. **Coerenza descrittiva**: Il modello deve rappresentare il fenomeno in modo logico e coerente, catturando gli aspetti chiave del processo psicologico e organizzando le osservazioni in una struttura comprensibile.
2. **Capacità predittiva**: Deve essere in grado di fare previsioni accurate su futuri sviluppi del fenomeno, rendendo possibile testare la validità delle sue ipotesi attraverso i dati.
3. **Supporto empirico**: Le ipotesi e le previsioni del modello devono essere confermate da dati raccolti mediante ricerche sistematiche e rigorose.
4. **Falsificabilità**: Un modello scientifico deve poter essere testato e, se necessario, confutato sulla base di osservazioni e risultati sperimentali.
5. **Parsimonia**: Il modello deve spiegare il fenomeno in modo semplice, evitando complessità inutili o ridondanti.
6. **Generalizzabilità**: Deve essere applicabile a una varietà di situazioni e contesti, superando i confini di specifiche condizioni sperimentali.
7. **Utilità pratica**: Deve fornire indicazioni utili per interventi, terapie o applicazioni concrete nel mondo reale.

La modellazione in psicologia presenta sfide uniche a causa della natura soggettiva, complessa e variabile dell'esperienza umana. I ricercatori devono trovare un equilibrio tra la precisione teorica e la flessibilità necessaria per cogliere la complessità dei fenomeni psicologici, tenendo conto anche dei limiti etici della sperimentazione e delle implicazioni sociali.

L'analisi dei dati è uno strumento centrale per valutare i modelli psicologici. Attraverso tecniche statistiche avanzate, si verifica se il modello riesce a

spiegare i dati osservati e a fare previsioni attendibili su dati futuri. Questo processo consente non solo di comprendere meglio i fenomeni psicologici, ma anche di prevedere e, in alcuni casi, influenzare il comportamento e i processi mentali. Un modello valido rappresenta quindi un potente strumento per il progresso teorico e per lo sviluppo di interventi pratici.

1.3.1 Rappresentare i Fenomeni per Ragionare e Comunicare

La spiegazione scientifica non si limita a chiarire i meccanismi causali, ma offre un linguaggio formale per analizzare e condividere conoscenze sui fenomeni. In psicologia, i modelli scientifici rappresentano strumenti fondamentali per descrivere i processi attraverso variabili, funzioni e parametri, fornendo una struttura per identificare relazioni e proprietà essenziali. Un modello efficace semplifica la complessità del fenomeno, rendendo più agevole sia la comunicazione tra studiosi sia la comprensione intuitiva.

I modelli non solo organizzano informazioni, ma stimolano anche intuizioni, generando nuove domande di ricerca. Una rappresentazione chiara consente di formulare ipotesi innovative, collegare concetti apparentemente distanti e trasferire conoscenze tra ambiti disciplinari, ampliando così il campo della ricerca.

1.3.2 Il Ruolo dell'Analisi dei Dati

L'analisi dei dati è cruciale per la scienza, e in psicologia svolge due funzioni principali:

1. **Semplificare e sintetizzare informazioni complesse**
   Attraverso statistiche descrittive, grafici e altre rappresentazioni, l'analisi dei dati aiuta a identificare schemi, tendenze e anomalie. Questo processo facilita la comprensione dei fenomeni psicologici e consente di esplorare le differenze tra individui o gruppi.
2. **Valutare le predizioni teoriche**
   Confrontando le aspettative di un modello con i dati raccolti, l'analisi verifica la validità delle ipotesi sottostanti. Questo confronto è essenziale per sostenere, migliorare o rivedere le teorie, contribuendo al progresso scientifico.

Tuttavia, l'analisi dei dati da sola non è sufficiente per comprendere a fondo un fenomeno. Correlazioni o schemi identificati nei dati, se privi di un contesto teorico, forniscono una visione limitata. È indispensabile un quadro teorico che interpreti e contestualizzi i risultati, proponendo meccanismi causali che diano significato alle osservazioni.

Unire modelli teorici e analisi dei dati permette di andare oltre la descrizione,

offrendo spiegazioni solide e utili per sviluppare interventi o approfondire la conoscenza dei fenomeni psicologici.

1.3.3 Carattere Multidisciplinare dell'Analisi dei Dati

L'analisi dei dati si colloca all'intersezione di tre discipline fondamentali: statistica, teoria della probabilità e informatica. Ciascuna di queste apporta strumenti, metodologie e prospettive indispensabili per comprendere i dati, estrarre conoscenze utili e sviluppare nuove ipotesi scientifiche.

- **Statistica:**
  Fornisce metodi per raccogliere, organizzare e interpretare i dati, consentendo di sintetizzare informazioni, identificare schemi e prendere decisioni basate sull'evidenza.
- **Teoria della probabilità:**
  Costituisce il fondamento matematico della statistica, permettendo di modellare e misurare l'incertezza e comprendere la variabilità delle osservazioni.
- **Informatica:**
  Contribuisce con strumenti per gestire, elaborare e visualizzare grandi quantità di informazioni, rendendo possibile l'applicazione di modelli complessi e l'analisi di dataset di dimensioni considerevoli.

Questa natura multidisciplinare riflette la complessità intrinseca dell'analisi dei dati e la necessità di integrare competenze provenienti da diverse aree per affrontare con successo le sfide scientifiche moderne.

1.4 Concetti Chiave nell'Analisi dei Dati

Per condurre un'analisi dei dati efficace, è fondamentale comprendere alcuni concetti chiave che guidano il processo di indagine, dall'identificazione del fenomeno alla formulazione di inferenze.

1.4.1 Popolazioni e Campioni

L'analisi dei dati inizia con l'identificazione della **popolazione di interesse**, che rappresenta l'insieme completo degli individui o delle entità coinvolte nel fenomeno studiato. Poiché studiare un'intera popolazione è spesso impraticabile, si ricorre ai **campioni**, sottoinsiemi rappresentativi della popolazione. La qualità e la rappresentatività del campione sono cruciali: un campione non rappresentativo può portare a conclusioni errate, limitando la generalizzabilità dei risultati.

1.5 Parametri e Statistiche

Un **parametro** è una caratteristica numerica della popolazione (es. media , deviazione standard). Una **statistica** è una caratteristica numerica calcolata sul campione (es. media campionaria $\bar{x}$, deviazione standard campionaria s). L'inferenza statistica si occupa di stimare i parametri della popolazione a partire dalle statistiche campionarie.

1.5.1 Bias nella Raccolta Dati

I **bias** nella raccolta e interpretazione dei dati possono compromettere l'accuratezza dei risultati. Comprendere **chi ha raccolto i dati, come e con quali scopi** è essenziale per una corretta interpretazione. I dati non sono mai completamente neutri; i metodi e gli obiettivi di raccolta influenzano i risultati. Ad esempio, selezionare partecipanti da una popolazione di studenti universitari potrebbe introdurre un bias sistematico, limitando la generalizzabilità ad altri contesti (**?**; **?**).

1.5.2 Variabili e Costanti

Nell'analisi statistica, le **variabili** rappresentano le caratteristiche osservate che possono assumere diversi valori (numerici o categorici). Le **costanti**, al contrario, rimangono fisse in un determinato contesto. Le variabili si distinguono in:

- **Variabili indipendenti (o predittive)**: influenzano altri fenomeni;
- **Variabili dipendenti**: rappresentano gli esiti di interesse influenzati dalle variabili indipendenti.

Ad esempio, in uno studio sugli effetti della terapia cognitivo-comportamentale, la variabile indipendente potrebbe essere la partecipazione alla terapia, mentre la variabile dipendente sarebbe la riduzione dei sintomi di ansia.

1.5.3 Studi Osservazionali ed Esperimenti

Esistono due principali metodi di raccolta dati:

1. **Esperimenti**: I ricercatori manipolano una o più variabili per valutare il loro effetto su altre variabili, controllando per i fattori confondenti. Ad esempio, per valutare l'efficacia di un trattamento, i partecipanti possono essere assegnati casualmente a un gruppo di

controllo (placebo) e a un gruppo sperimentale (trattamento attivo). La randomizzazione riduce il rischio di bias sistematici.

2. **Studi osservazionali**: I dati vengono raccolti senza interferire con il fenomeno osservato. Ad esempio, un'indagine su come lo stress influenza la produttività lavorativa potrebbe basarsi su questionari senza manipolare lo stress dei partecipanti. Questi studi forniscono correlazioni tra variabili, ma non dimostrano relazioni causali.

1.5.4 Effetti

In statistica, un **effetto** rappresenta il cambiamento osservato nelle variabili dipendenti in relazione alle variabili indipendenti. Questo cambiamento indica una relazione o un'influenza della variabile indipendente sulla variabile dipendente. Ad esempio, una terapia che produce un effetto statisticamente distinguibile dal caso si manifesta con una riduzione dei sintomi tra la fase pre-trattamento e quella post-trattamento (**?**).

1.5.5 Stima e Inferenza Statistica: Dal Campione alla Popolazione

La stima e l'inferenza statistica rappresentano i fondamenti della metodologia quantitativa, poiché permettono di estendere le conclusioni tratte da un campione, ovvero una porzione limitata della popolazione, all'intero insieme di interesse. L'uso dei campioni si rende necessario a causa di vincoli pratici, come il costo, il tempo e le risorse richieste per studiare un'intera popolazione. Tuttavia, l'uso di un campione introduce un'incertezza intrinseca: le **statistiche** calcolate sul campione, chiamate "stime", non corrispondono esattamente ai **parametri** della popolazione, presentando un margine di errore. La teoria degli stimatori e l'inferenza statistica forniscono strumenti per quantificare e gestire questa incertezza, rendendo possibile la generalizzazione dei risultati dal campione alla popolazione.

1.5.6 Stima: Inferire le Caratteristiche della Popolazione

La **stima** è il processo mediante il quale si utilizzano i dati di un campione per dedurre le proprietà della popolazione. Per esempio, la media calcolata sul campione (*media campionaria*) è spesso usata come stima della media della popolazione. Tuttavia, è importante riconoscere che ogni campione rappresenta solo una frazione della popolazione, e campioni diversi possono produrre stime differenti, un fenomeno noto come **variabilità campionaria**. Questa variabilità rappresenta la principale fonte di incertezza nelle stime.

1.5.6.1 Fattori che Influenzano l'Accuratezza

L'accuratezza di una stima dipende da tre fattori principali:

1. **Dimensione del campione:** Campioni più grandi riducono la variabilità campionaria, aumentando la precisione delle stime.
2. **Rappresentatività:** Un campione rappresentativo rispecchia le caratteristiche essenziali della popolazione. Campioni distorti o non rappresentativi possono portare a stime errate.
3. **Variabilità della popolazione:** Popolazioni più eterogenee richiedono campioni più ampi per produrre stime affidabili.

1.5.6.2 Gli Stimatori: Proprietà Fondamentali

Gli **stimatori** sono formule matematiche utilizzate per calcolare le stime. La qualità di uno stimatore dipende dalle sue proprietà:

- **Consistenza:** Uno stimatore è consistente se, aumentando la dimensione del campione, la stima si avvicina al valore vero del parametro.
- **Non distorsione:** Uno stimatore è non distorto quando il suo valore atteso coincide con il parametro della popolazione.
- **Efficienza:** Tra stimatori non distorti, quello con varianza minore è considerato più efficiente.

In sintesi, la stima consente di trasformare i dati limitati del campione in inferenze sulla popolazione, tenendo conto dell'incertezza dovuta alla variabilità campionaria.

1.5.7 Inferenza Statistica

L'**inferenza statistica** è il processo che permette di utilizzare i dati raccolti da un campione per trarre conclusioni sull'intera popolazione. Si tratta di uno strumento fondamentale per rispondere a domande come: *Cosa possiamo dire di un fenomeno generale a partire da osservazioni limitate?*

In questo contesto, l'inferenza statistica affronta tre problemi principali:

1. **Stima dei parametri della popolazione:** Determinare i valori plausibili per parametri come la media, la varianza o le proporzioni che descrivono una popolazione. Questo include non solo identificare un valore centrale (come una media campionaria), ma anche quantificare l'incertezza associata a questa stima.
2. **Valutazione di ipotesi:** Rispondere a domande sulla plausibilità di una particolare affermazione. Per esempio, verificare se due gruppi differiscono rispetto a una variabile di interesse o se esiste una relazione tra due variabili. Questo implica confrontare le ipotesi con i dati osservati per determinare quale sia meglio supportata.
3. **Previsione:** Utilizzare i dati esistenti per anticipare risultati futuri. L'inferenza statistica fornisce strumenti per quantificare quanto

possiamo essere certi rispetto a eventi non ancora osservati, integrando incertezze legate ai parametri e alla variabilità intrinseca dei dati.

Entrambi i principali approcci all'inferenza statistica – frequentista e bayesiano – si occupano di questi problemi, sebbene adottino prospettive diverse per affrontarli. L'obiettivo comune è quello di fornire risposte rigorose e basate sui dati alle domande che emergono nell'analisi statistica.

In sintesi, la stima e l'inferenza statistica sono strumenti essenziali per trasformare i dati campionari in conoscenza generalizzabile, permettendo di esplorare e rispondere a domande fondamentali sui fenomeni di interesse. In psicologia, dove la complessità e la variabilità dei comportamenti sono elevate, l'inferenza statistica riveste un ruolo cruciale nel comprendere i processi mentali e le relazioni tra variabili. Una corretta applicazione di questi strumenti richiede consapevolezza delle loro potenzialità e dei loro limiti, insieme a una scelta ponderata dell'approccio più adatto al problema in esame.

1.6 Le Sfide dell'Inferenza Statistica in Psicologia

L'inferenza statistica, in particolare in psicologia e nelle scienze sociali, si confronta con alcune sfide specifiche, che riflettono la complessità dei fenomeni studiati (**?**):

1. **Generalizzazione dai campioni alla popolazione:** I campioni utilizzati nella ricerca psicologica e sociale spesso presentano limitazioni in termini di rappresentatività della popolazione target. L'uso frequente di campioni di convenienza (ad esempio, studenti universitari) può compromettere la generalizzabilità dei risultati a popolazioni più ampie e diversificate.
2. **Generalizzazione dal trattamento al gruppo di controllo (validità esterna):** Negli studi sperimentali, è fondamentale valutare se gli effetti osservati nel gruppo trattato siano generalizzabili ad altri contesti, popolazioni o condizioni. Questo aspetto è strettamente legato al concetto di validità esterna.
3. **Inferenza su costrutti latenti:** Molti costrutti psicologici (come l'ansia, l'intelligenza o la personalità) non sono direttamente osservabili, ma vengono inferiti attraverso misurazioni indirette (ad esempio, questionari, test, osservazioni comportamentali). L'inferenza statistica deve quindi affrontare la sfida di collegare i dati osservati

ai costrutti teorici sottostanti, tenendo conto dell'errore di misurazione e delle limitazioni degli strumenti di valutazione.

1.6.1 L'Incertezza

Le considerazioni introduttive di questo capitolo fanno capire come un aspetto cruciale della stima e dell'inferenza sia la gestione e la quantificazione dell'**incertezza**. Ogni stima derivata da un campione è intrinsecamente soggetta a errore, in quanto il campione rappresenta solo una parte della popolazione. L'inferenza statistica fornisce gli strumenti per quantificare tale incertezza, ad esempio attraverso gli intervalli di confidenza (nell'approccio frequentista) o le distribuzioni a posteriori (nell'approccio bayesiano), consentendo di esprimere il grado di fiducia nelle conclusioni tratte.

In conclusione, la stima e l'inferenza statistica sono strumenti indispensabili per trasformare i dati empirici in conoscenza utile e rilevante. Tuttavia, è fondamentale applicare queste metodologie con rigore e consapevolezza delle loro limitazioni, prestando particolare attenzione alla rappresentatività del campione, alla validità delle misurazioni e alla corretta interpretazione dei risultati, al fine di evitare generalizzazioni inappropriate e conclusioni errate.

1.7 Riflessioni Conclusive

L'analisi dei dati acquisisce valore solo quando è integrata con una solida teoria scientifica, che fornisce il contesto e il quadro interpretativo necessario per attribuire senso ai risultati. Ad esempio, osservare che un trattamento psicologico riduce i sintomi è un'osservazione empirica che, senza una teoria che chiarisca i meccanismi sottostanti, rimane priva di potere esplicativo. È la teoria che orienta il processo analitico, formulando ipotesi verificabili e offrendo interpretazioni che si inseriscono in un modello più ampio.

In definitiva, la relazione tra teoria e analisi dei dati è intrinsecamente circolare e dinamica: le teorie guidano la raccolta, l'analisi e l'interpretazione dei dati, mentre i dati, a loro volta, stimolano il perfezionamento e l'evoluzione delle teorie. Questo dialogo continuo è ciò che permette un progresso costante nella comprensione dei fenomeni psicologici.

Esercizi

! Problemi

1. Che cos'è una spiegazione scientifica e in che modo si differenzia da una mera descrizione o previsione di un fenomeno?
2. Perché, quando si parla di popolazione e campione, è fondamentale assicurarsi che il campione sia rappresentativo, e quali conseguenze possono derivare da un campione non rappresentativo?
3. Che differenza c'è tra un **parametro** e una **statistica**, e perché in inferenza statistica si cerca di stimare il parametro sconosciuto a partire dalla statistica campionaria?
4. Cosa si intende per **bias** nella raccolta e interpretazione dei dati, e in che modo la consapevolezza dei possibili bias può migliorare la qualità della ricerca?
5. Perché in psicologia e nelle scienze sociali risulta essenziale integrare l'analisi dei dati con un quadro teorico solido e coerente?
6. Qual è la differenza principale tra uno **studio osservazionale** e un **esperimento**, e perché la distinzione è importante per comprendere la causalità?
7. Che ruolo svolgono i **modelli scientifici** in psicologia, e quali caratteristiche fondamentali dovrebbero possedere per essere considerati validi e utili?
8. In che modo l'analisi dei dati aiuta a passare dalle semplici correlazioni o tendenze osservate all'elaborazione di ipotesi e spiegazioni più profonde?
9. Che differenza c'è tra **variabili indipendenti** e **variabili dipendenti**, e perché questa distinzione è cruciale per disegnare uno studio e interpretarne i risultati?
10. Perché parlare di **incertezza** è inevitabile quando si utilizzano i dati di un campione, e come la statistica (frequentista o bayesiana) ci aiuta a gestirla?

Soluzioni

1. Che cos'è una spiegazione scientifica e in che modo si differenzia da una mera descrizione o previsione di un fenomeno?
Una **spiegazione scientifica** mira a individuare le *cause* e i *meccanismi* che generano o influenzano un fenomeno. Non si limita quindi a

descrivere cosa accade o a prevedere ciò che potrebbe accadere (come una semplice correlazione o un modello predittivo), ma cerca di chiarire *perché* il fenomeno si verifica. Ad esempio, dire "i bambini con genitori laureati hanno migliori prestazioni scolastiche" è una descrizione (o previsione) utile; spiegare che ciò avviene a causa di un maggior sostegno nel percorso di studi, di un ambiente più ricco di stimoli culturali, o di un contesto socioeconomico facilitante, fornisce invece una spiegazione che va oltre la pura correlazione statistica.

2. Perché, quando si parla di popolazione e campione, è fondamentale assicurarsi che il campione sia rappresentativo, e quali conseguenze possono derivare da un campione non rappresentativo?

Il **campione** è il sottoinsieme di individui selezionati da una **popolazione** più ampia. Affinché i risultati di uno studio siano validi e generalizzabili, il campione deve rispecchiare le principali caratteristiche della popolazione (ad esempio in termini di età, genere, livello socioeconomico, ecc.). Se il campione non è rappresentativo (per esempio, se si reclutano solo studenti universitari per uno studio su tutta la popolazione italiana), possono emergere **bias di selezione** che rendono impossibile estendere correttamente i risultati a gruppi sociali diversi. Conseguenze tipiche di un campione non rappresentativo includono stime distorte dei parametri d'interesse, conclusioni fuorvianti e ridotta validità esterna della ricerca.

3. Che differenza c'è tra un *parametro* e una *statistica*, e perché in inferenza statistica si cerca di stimare il parametro sconosciuto a partire dalla statistica campionaria?

- Un **parametro** è una caratteristica numerica della *popolazione* (ad esempio la media reale di un determinato tratto o la proporzione di individui con una certa caratteristica).

- Una **statistica** è una misura analoga, ma *calcolata sul campione* (ad esempio la media o la proporzione campionaria).

Poiché in genere è impossibile o molto costoso misurare l'intera popolazione, si raccoglie un campione più piccolo e gestibile. La **statistica** del campione (ad es. la media campionaria) è quindi usata per **stimare** il **parametro** (ad es. la media della popolazione). L'obiettivo dell'inferenza statistica è fornire, insieme a questa stima, una misura dell'incertezza associata (per esempio un intervallo di confidenza), così da comprendere quanto la statistica campionaria potrebbe "avvicinarsi" al vero valore del parametro.

4. Cosa si intende per *bias* nella raccolta e interpretazione dei dati, e in che modo la consapevolezza dei possibili bias può migliorare la qualità della ricerca?

Il **bias** è un errore sistematico che altera i risultati di uno studio in una

direzione specifica, dovuto a scelte o condizioni nel disegno della ricerca, nella selezione del campione, nella misurazione o nell'interpretazione dei dati. Ad esempio, se reclutiamo solo volontari particolarmente motivati a partecipare a una ricerca, potremmo ottenere risultati che sovrastimano un certo fenomeno e non rispecchiano la popolazione generale.
Essere consapevoli di come i bias possano nascere aiuta i ricercatori a *mitigarli* (ad esempio, bilanciando il reclutamento dei partecipanti o rendendo anonima la compilazione di un questionario) e a tenere conto dei loro effetti quando si interpretano i risultati. Così, la ricerca risulta più affidabile e validamente interpretata.

5. Perché in psicologia e nelle scienze sociali risulta essenziale integrare l'analisi dei dati con un quadro teorico solido e coerente?

Nelle scienze sociali e in psicologia, i fenomeni studiati sono spesso complessi e influenzati da molte variabili. I **dati** da soli, senza una teoria, forniscono soltanto una descrizione o una misurazione di ciò che accade in un dato momento. La **teoria** invece permette di:

- Identificare le variabili rilevanti e formulare ipotesi specifiche;
- Interpretare i risultati, attribuendo un *senso* e un *contesto* alle relazioni osservate;
- Comprendere i meccanismi causali e sviluppare spiegazioni che vadano oltre la pura descrizione.

Senza un quadro teorico di riferimento, sarebbe difficile capire *perché* si osservano determinate relazioni e come possano cambiare in contesti diversi o in situazioni sperimentali alternative.

6. Qual è la differenza principale tra uno *studio osservazionale* e un *esperimento*, e perché la distinzione è importante per comprendere la causalità?

- **Studio osservazionale**: Il ricercatore raccoglie i dati senza intervenire né manipolare alcuna variabile. Ad esempio, si misura il livello di stress delle persone e la loro produttività sul lavoro, senza modificare artificialmente il livello di stress. Questi studi mostrano correlazioni, ma è difficile stabilire con certezza relazioni di causa-effetto.
- **Esperimento**: Il ricercatore manipola una o più variabili (variabili indipendenti) e controlla le condizioni, ad esempio assegnando in modo casuale i partecipanti a un gruppo di trattamento e a uno di controllo. Ciò facilita la comprensione di eventuali nessi causali, perché la randomizzazione e il controllo degli altri fattori riducono il rischio che variabili esterne influenzino i risultati.

La distinzione è cruciale perché, nei fenomeni complessi della psicolo-

gia, gli studi osservazionali possono suggerire ipotesi di relazione, ma di solito occorre un disegno sperimentale (quando possibile) per trarre conclusioni più solide sulla causalità.

7. Che ruolo svolgono i *modelli scientifici* in psicologia, e quali caratteristiche fondamentali dovrebbero possedere per essere considerati validi e utili?

I **modelli scientifici** in psicologia forniscono una struttura concettuale e spesso formale (matematica o simulativa) per rappresentare e spiegare processi mentali e comportamentali. Servono a:

- Organizzare osservazioni ed evidenze in un sistema coerente;
- Fare previsioni verificabili empiricamente;
- Guidare l'interpretazione di nuovi dati e la progettazione di futuri studi.

Caratteristiche di un buon modello sono:

1. **Coerenza descrittiva** (rappresenta fedelmente il fenomeno);
2. **Capacità predittiva** (prevede correttamente i risultati di situazioni nuove);
3. **Supporto empirico** (confermato dai dati raccolti rigorosamente);
4. **Falsificabilità** (dev'essere possibile smentirlo con evidenze contrarie);
5. **Parsimonia** (non dev'essere inutilmente complicato);
6. **Generalizzabilità** (applicabile a diversi contesti e situazioni);
7. **Utilità pratica** (fornisce indicazioni utili per interventi o comprensione teorica).

8. In che modo l'analisi dei dati aiuta a passare dalle semplici correlazioni o tendenze osservate all'elaborazione di ipotesi e spiegazioni più profonde?

L'**analisi dei dati** non si limita a segnalare che "due variabili sono associate" (correlazioni), ma offre:

- Strumenti per isolare l'effetto di una variabile sulle altre (regressioni multiple, modelli a effetti misti, ecc.);
- Metodologie per la verifica di ipotesi specifiche sulla direzione e sulla

natura delle relazioni (ad es. test statistici o modelli di mediazione-moderazione in psicologia);

- Indicatori dell'incertezza e della robustezza dei risultati (intervalli di confidenza, analisi della potenza, analisi bayesiane).

Con questi strumenti, i ricercatori possono integrare i risultati quantitativi con le teorie esistenti, sviluppare nuove ipotesi su meccanismi causali e proporre spiegazioni più articolate su *come* e *perché* le variabili si influenzino reciprocamente.

9. Che differenza c'è tra *variabili indipendenti* e *variabili dipendenti*, e perché questa distinzione è cruciale per disegnare uno studio e interpretarne i risultati?

- **Variabile indipendente (VI)**: è quella che si sospetta abbia un *effetto* su un'altra variabile, o che si desidera manipolare in un disegno sperimentale (per esempio, l'introduzione di un nuovo metodo di studio).

- **Variabile dipendente (VD)**: è la variabile che si misura per valutare l'eventuale *effetto* della variabile indipendente (ad esempio, i risultati di un test di apprendimento).

  La distinzione è basilare perché chiarisce la direzione del rapporto di interesse e permette di formulare ipotesi come "VI $\rightarrow$ VD" (es. "il nuovo metodo di studio *migliora* i risultati del test"). Sbagliare a identificare quali sono le variabili indipendenti e dipendenti può portare a disegni di ricerca confusi e interpretazioni errate.

10. Perché parlare di *incertezza* è inevitabile quando si utilizzano i dati di un campione, e come la statistica (frequentista o bayesiana) ci aiuta a gestirla?

Quando raccogliamo dati da un **campione** (necessariamente limitato), non possiamo osservare l'intera popolazione. Questo introduce un margine di *incertezza* su quanto la misura campionaria (statistica) rispecchi il parametro reale della popolazione. Inoltre, possono sempre esserci fattori non controllati o errori di misurazione.

- Nell'**approccio frequentista**, l'incertezza è gestita tramite concetti come gli **intervalli di confidenza** e i **valori *p***, che quantificano la probabilità di osservare determinati risultati assumendo determinate ipotesi (per es. l'ipotesi nulla).

- Nell'**approccio bayesiano**, l'incertezza è modellata tramite **distribuzioni di probabilità** (posteriori) che incorporano sia i dati osservati sia le informazioni pregresse (priors).

Entrambi gli approcci forniscono metodologie per valutare quanto ci si

possa fidare di una data conclusione, riconoscendo il carattere aleatorio e parziale dei dati e rendendo esplicito il grado di incertezza.

Bibliografia

2

Campionamento, metodologia sperimentale e studi osservazionali

! In questo capitolo apprenderai:

- i metodi di campionamento,
- i principi fondamentali degli studi sperimentali e delle ricerche osservazionali.

Prerequisiti

- Consultare *A discipline-wide investigation of the replicability of Psychology papers over the past two decades* (**?**).

2.1 Introduzione

Questo capitolo si propone di approfondire i concetti introdotti in precedenza, concentrandosi in particolare sul processo di campionamento e sull'importanza delle diverse metodologie di ricerca in psicologia. Dopo aver compreso il ruolo fondamentale dei dati nella verifica delle teorie, emerge una questione cruciale: come vengono raccolti questi dati?

La raccolta dei dati non è un'attività neutra o priva di conseguenze metodologiche. I dati, infatti, non hanno lo stesso valore scientifico a seconda di come vengono ottenuti. Alcune modalità di raccolta generano informazioni preziose e utili per testare le teorie, mentre altre possono produrre risultati fuorvianti, distorti o addirittura dannosi per la validità della ricerca.

Il metodo scientifico fornisce un quadro di riferimento che delinea le caratteristiche ideali di un processo di raccolta dati in grado di produrre informazioni valide e affidabili. Tuttavia, le prescrizioni del metodo scientifico sono per loro natura generali e astratte. Tradurre questi principi in procedure concrete

e applicarli efficacemente in un contesto di ricerca specifico rappresenta una sfida significativa.

Questo passaggio, che collega la teoria alla pratica, dipende dalle risorse disponibili, dalla competenza metodologica e dalla creatività del ricercatore. La capacità di ideare e attuare strategie di raccolta dati adeguate al contesto specifico della ricerca è fondamentale per garantire la qualità e la validità dei risultati ottenuti. In questo capitolo, approfondiremo i principi del campionamento e introdurremo i concetti chiave relativi ai disegni di ricerca.

2.2 Popolazioni e Campioni

Nella ricerca scientifica, è essenziale distinguere tra **popolazione** e **campione**.

- **Popolazione**: rappresenta l'insieme completo di unità che condividono una o più caratteristiche specifiche oggetto di studio. La dimensione della popolazione è indicata con N.
- **Campione**: è un sottoinsieme della popolazione, di dimensione n. L'obiettivo del campionamento è ottenere un campione **rappresentativo**, ovvero un sottoinsieme che rifletta accuratamente le caratteristiche della popolazione di riferimento.

2.2.1 Metodi di Campionamento

Esistono diverse strategie per selezionare un campione rappresentativo da una popolazione. Queste strategie si dividono principalmente in due categorie: **campionamento probabilistico** e **campionamento non probabilistico**.

2.2.1.1 Campionamento Probabilistico

Nel campionamento probabilistico, ogni unità della popolazione ha una probabilità nota e non nulla di essere inclusa nel campione. Questo approccio minimizza il rischio di distorsioni sistematiche (bias) e consente di stimare l'errore di campionamento.

1. **Campionamento Casuale Semplice (CCS):**
   Ogni unità della popolazione ha la stessa probabilità di essere inclusa nel campione. Questo metodo richiede una lista completa di tutte le unità della popolazione, nota come *frame di campionamento*. La selezione può avvenire con o senza reinserimento. Il CCS senza reinserimento è il più comune nella pratica, ma nelle ricer-

che psicologiche è raramente utilizzabile a causa della difficoltà di ottenere un frame completo della popolazione.

2. **Campionamento Stratificato:**
   La popolazione viene divisa in strati (H), ovvero sottogruppi omogenei in base a una o più variabili rilevanti (es. età, genere, regione geografica). Da ogni strato h viene estratto un campione casuale semplice di dimensione nh. Questo metodo può essere:

   - **Proporzionale**: la dimensione del campione in ogni strato è proporzionale alla dimensione dello strato nella popolazione.
   - **Non proporzionale**: utilizzato per sovra-campionare gruppi minoritari, consentendo analisi dettagliate di sottogruppi altrimenti poco rappresentati.
     Nelle ricerche psicologiche, il campionamento stratificato è utile per garantire che variabili come il genere o l'età siano adeguatamente rappresentate, ma può essere complesso da implementare.

3. **Campionamento a Grappolo (Cluster Sampling):**
   La popolazione viene suddivisa in grappoli (*cluster*), che rappresentano gruppi eterogenei (es. scuole, ospedali, quartieri). Vengono selezionati casualmente alcuni grappoli, includendo tutte le unità al loro interno. Questo metodo è economico e pratico, specialmente in contesti dove accedere all'intera popolazione è difficile. Tuttavia, la precisione può essere ridotta se i grappoli differiscono notevolmente tra loro.

4. **Campionamento Multistadio:**
   Combinazione di campionamento a grappolo e CCS. Vengono selezionati casualmente alcuni grappoli e, successivamente, all'interno di ciascun grappolo, si estrae un campione casuale di unità. Questo metodo bilancia costi e precisione, risultando particolarmente adatto a studi su larga scala, ad esempio a livello nazionale.

2.2.1.2 Campionamento Non Probabilistico

Nel campionamento non probabilistico, la probabilità di inclusione di ogni unità nel campione non è nota. Questo approccio è spesso adottato per ragioni di praticità o quando non è disponibile un frame di campionamento. Tuttavia, aumenta il rischio di distorsioni e limita la generalizzabilità dei risultati alla popolazione.

1. **Campionamento di Convenienza:**
   È il metodo più diffuso nella ricerca psicologica. I partecipanti vengono selezionati in base alla loro facile accessibilità (es., studenti universitari, volontari). Questo metodo è rapido ed economico, ma

introduce significativi bias di selezione, poiché il campione non è rappresentativo della popolazione generale.

2.3 Il Campionamento nella Ricerca Psicologica

A causa di vincoli pratici, economici e di accessibilità, il campionamento casuale semplice e stratificato sono raramente applicati in modo rigoroso nella ricerca psicologica. Il campionamento di convenienza è molto comune, soprattutto in contesti accademici.

2.3.1 Limiti del Campionamento di Convenienza e Strategie di Mitigazione

- **Bias di Selezione:**
  Il campione non riflette la diversità della popolazione, limitando la generalizzabilità dei risultati.
- **Strategie di Mitigazione:**
  - **Sovra-campionamento:** Includere deliberatamente un numero maggiore di partecipanti appartenenti a gruppi sottorappresentati per consentire analisi più accurate.
  - **Replicazione degli Studi:** Ripetere lo studio con campioni diversi per verificare la robustezza dei risultati e aumentare la generalizzabilità.
  - **Descrizione Dettagliata del Campione:** Fornire informazioni precise sulle caratteristiche del campione (es., età, genere, provenienza geografica, livello di istruzione) per consentire ai lettori di valutare la generalizzabilità dei risultati al loro specifico contesto.

È fondamentale che i ricercatori siano consapevoli dei limiti del metodo di campionamento utilizzato e che interpretino i risultati con cautela, evidenziando le potenziali limitazioni alla generalizzabilità.

2.4 Metodologia Sperimentale

2.4.1 Principi Fondamentali del Disegno Sperimentale

- **Controllo:**
  L'obiettivo è ridurre l'influenza di variabili confondenti (Z) sulla relazione tra la variabile indipendente (X) e quella dipendente (Y). Utilizzare un

gruppo di controllo consente di stabilire un riferimento per valutare l'effetto del trattamento.

- **Randomizzazione:**
  L'assegnazione casuale dei partecipanti ai gruppi sperimentali distribuisce le variabili confondenti in modo equilibrato tra i gruppi, aumentando la credibilità dell'inferenza causale tra X e Y. Questo approccio garantisce che eventuali differenze osservate siano attribuibili al trattamento e non a fattori esterni.

2.4.2 Strategie di Mitigazione dei Bias

- **Cecità (Blinding):**
  Strumento chiave per minimizzare l'influenza di aspettative e pregiudizi, sia nei partecipanti che nei ricercatori. Le principali modalità includono:
  - **Cecità singola:** I partecipanti non sono consapevoli del trattamento assegnato.
  - **Cecità doppia:** Sia i partecipanti che i ricercatori che interagiscono direttamente con loro non conoscono l'assegnazione dei trattamenti.
  - **Cecità tripla:** Né i partecipanti, né i ricercatori, né gli analisti dei dati sono a conoscenza dei trattamenti durante la raccolta e l'analisi dei dati.
- **Gruppo di Controllo:**
  L'introduzione di un gruppo che riceve un trattamento inerte (controllo) consente di isolare gli effetti psicologici legati alle aspettative dei partecipanti, distinguendoli dagli effetti specifici del trattamento.
- **Standardizzazione delle Procedure:**
  Garantire che tutte le condizioni sperimentali, eccetto la variabile manipolata, siano mantenute costanti tra i gruppi. Questo riduce la variabilità non controllata, migliorando la comparabilità dei risultati.

2.4.3 Nota sulla Replicazione

La replicazione degli esperimenti non è una caratteristica intrinseca del metodo sperimentale, ma rappresenta una pratica fondamentale nella scienza per verificare l'affidabilità e la generalizzabilità dei risultati. Si distingue in:

- **Replicazione diretta:** Ripetere lo stesso studio con le stesse condizioni.
- **Replicazione concettuale:** Ripetere lo studio modificando aspetti specifici per testare la robustezza del risultato.

2.4.4 Tipologie di Disegni Sperimentali

- **Disegno a Gruppi Indipendenti (Between-Subjects):**
  I partecipanti vengono assegnati a un unico gruppo e sono esposti a una sola condizione sperimentale. Questo disegno è utile quando l'esposizione multipla potrebbe introdurre confondenti, ma richiede un campione più ampio per raggiungere lo stesso livello di precisione.
- **Disegno a Misure Ripetute (Within-Subjects):**
  Gli stessi partecipanti vengono esposti a tutte le condizioni sperimentali. Questo approccio riduce la variabilità tra soggetti, migliorando la precisione delle stime. Tuttavia, richiede attenzione nel controllare gli effetti di ordine, come affaticamento o apprendimento, spesso attraverso il bilanciamento dell'ordine di presentazione dei trattamenti.

2.5 Studi Osservazionali

Gli studi osservazionali possono essere classificati in:

- **Studi Trasversali (Cross-Sectional):**
  I dati vengono raccolti in un singolo momento. Utili per stimare la prevalenza di una condizione, ma non permettono di stabilire relazioni causali.
- **Studi di Coorte (Cohort Studies):**
  Un gruppo di individui (coorte) viene seguito nel tempo per osservare l'incidenza di un evento. Permettono di studiare la relazione tra esposizione e outcome, ma possono essere costosi e richiedere molto tempo.
- **Studi Caso-Controllo (Case-Control Studies):**
  Vengono confrontati individui con una determinata condizione (casi) con individui senza la condizione (controlli) per identificare possibili fattori di rischio. Utili per studiare malattie rare, ma soggetti a bias di selezione e di ricordo.

Le principali limitazioni degli studi osservazionali sono la presenza di variabili confondenti e la difficoltà di stabilire relazioni causali.

2.6 Riflessioni Conclusive

La comprensione dei metodi di campionamento, dei principi della metodologia sperimentale e delle caratteristiche degli studi osservazionali è essenziale per

interpretare correttamente i risultati riportati nella letteratura scientifica in psicologia.

Questi concetti permettono di valutare la solidità delle conclusioni tratte dagli autori, di identificare potenziali limiti metodologici e di distinguere tra evidenze solide e ipotesi meno supportate. Saper riconoscere l'importanza di aspetti come il controllo, la randomizzazione e il disegno dello studio è indispensabile per leggere la letteratura scientifica con un approccio critico e consapevole.

Esercizi

! Problemi

1. Perché la fase di raccolta dei dati è considerata "non neutrale" e quali conseguenze può avere sull'affidabilità dei risultati di una ricerca psicologica?
2. In che modo i diversi metodi di campionamento (probabilistico vs. non probabilistico) influiscono sulla generalizzabilità delle conclusioni di uno studio e quali situazioni giustificano l'uso dell'uno o dell'altro?
3. Quali sono i principali rischi nell'adottare un campionamento di convenienza in ricerche psicologiche, e quali strategie si possono utilizzare per mitigare questi rischi?
4. In che modo la randomizzazione e l'uso di gruppi di controllo supportano l'inferenza causale in un disegno sperimentale, e perché queste caratteristiche sono spesso più difficili da mantenere negli studi osservazionali?
5. Quali sono i principali criteri da considerare quando si valuta la validità di uno studio in termini di campionamento, disegno di ricerca e replicabilità?

Soluzioni

1. Perché la fase di raccolta dei dati è considerata "non neutrale" e quali conseguenze può avere sull'affidabilità dei risultati di una ricerca psicologica?
La raccolta dei dati non è mai un processo completamente neutrale perché comporta scelte metodologiche e pratiche che possono influenzare la qualità e la natura delle informazioni ottenute. Ad esempio:

- **Selezione del campione**: Scegliere chi includere o escludere nella ricerca incide sulla rappresentatività del campione. Un campione

non rappresentativo può produrre risultati distorti e difficilmente generalizzabili.

- **Modalità di somministrazione degli strumenti**: La maniera in cui vengono somministrati questionari, test o interviste può influire sulle risposte dei partecipanti (per esempio, differenze tra somministrazione online vs. cartacea, questionari anonimi vs. non anonimi, ecc.).

- **Contesto e tempistiche**: Il contesto ambientale (es., rumori, distrazioni) o il momento in cui i dati vengono raccolti (es., in prossimità di esami universitari) possono influenzare lo stato emotivo o motivazionale dei partecipanti.

Come conseguenza, tutte queste variabili possono introdurre **bias** – ossia distorsioni sistematiche – che compromettono l'**affidabilità** e la **validità** dei risultati, rendendo l'interpretazione dei dati più complessa e potenzialmente fuorviante. In altre parole, se la fase di raccolta dati non è accuratamente progettata e condotta, la ricerca potrebbe fornire conclusioni scorrette o di limitata utilità scientifica.

2. In che modo i diversi metodi di campionamento (probabilistico vs. non probabilistico) influiscono sulla generalizzabilità delle conclusioni di uno studio e quali situazioni giustificano l'uso dell'uno o dell'altro?

- **Campionamento probabilistico**:
  - Ogni unità della popolazione ha una probabilità *nota e non nulla* di essere selezionata.
  - Metodi come il campionamento casuale semplice, stratificato o a grappolo forniscono un quadro più solido per stimare l'errore di campionamento e minimizzare i bias.
  - I risultati ottenuti sono, in linea di massima, **più facilmente generalizzabili** all'intera popolazione di riferimento.
  - È preferibile in studi di larga scala, in cui esiste un buon frame di campionamento (lista esaustiva della popolazione) e le risorse (tempo, fondi) consentono di implementare un disegno rigoroso.
- **Campionamento non probabilistico**:
  - La probabilità di inclusione di un'unità non è nota, per cui non si possono calcolare in modo rigoroso le stime di errore.
  - Il metodo più comune è il **campionamento di convenienza**, in cui vengono reclutate persone facilmente accessibili (es., studenti

universitari, volontari, piattaforme online).

- La **generalizzabilità** dei risultati è ridotta, poiché il campione potrebbe non rispecchiare le caratteristiche della popolazione di interesse.

- È spesso utilizzato per studi esplorativi, ricerche a budget limitato o quando non si dispone di un elenco completo della popolazione.

In sintesi, il **campionamento probabilistico** è preferibile quando si mira a ottenere risultati solidi e generalizzabili a una popolazione più ampia e le condizioni logistiche lo consentono. Il **campionamento non probabilistico**, invece, è giustificato in studi preliminari, in situazioni in cui la popolazione non è ben definita o difficilmente accessibile, o quando si hanno **vincoli di risorse** che rendono impraticabile un campionamento probabilistico.

3. Quali sono i principali rischi nell'adottare un campionamento di convenienza in ricerche psicologiche, e quali strategie si possono utilizzare per mitigare questi rischi?

- **Principali rischi**:

  1. **Bias di selezione**: I partecipanti reclutati con metodi di convenienza (ad es. studenti di psicologia) potrebbero non rispecchiare l'eterogeneità della popolazione generale, limitando la generalizzabilità dei risultati.

  2. **Omogeneità del campione**: Se il campione è molto omogeneo (per età, livello di istruzione, contesto culturale), diventa difficile estendere le conclusioni a gruppi con caratteristiche diverse.

  3. **Autoselezione**: I volontari che si offrono di partecipare potrebbero differire sistematicamente da coloro che non partecipano (ad esempio, maggiore interesse per il tema della ricerca o per la ricompensa economica offerta).

- **Strategie di mitigazione**:

  1. **Sovra-campionamento**: Includere deliberatamente più partecipanti appartenenti a gruppi minoritari o sottorappresentati, per disporre di sottocampioni più completi.

  2. **Replicazione**: Ripetere l'esperimento con campioni diversi (per età, contesto geografico, cultura) per verificare se i

risultati si mantengono coerenti.

3. **Descrizione dettagliata del campione**: Fornire informazioni precise sulle caratteristiche sociodemografiche (età, genere, livello di istruzione, ecc.) in modo che altri ricercatori o lettori possano valutare la trasferibilità dei risultati.

4. **Cautela nell'interpretazione**: Esplicitare nelle conclusioni i limiti relativi alla natura del campionamento e invitare a considerare possibili fattori confondenti legati alla non rappresentatività del campione.

4. In che modo la randomizzazione e l'uso di gruppi di controllo supportano l'inferenza causale in un disegno sperimentale, e perché queste caratteristiche sono spesso più difficili da mantenere negli studi osservazionali?

- **Randomizzazione**:
  - Assegna i partecipanti ai gruppi sperimentali (condizione sperimentale vs. condizione di controllo) in maniera casuale.
  - Garantisce che variabili potenzialmente confondenti vengano distribuite equamente tra i gruppi, aumentando la probabilità che eventuali differenze nelle misure di esito (variabile dipendente) siano dovute esclusivamente alla manipolazione sperimentale (variabile indipendente).
  - Questo processo riduce l'influenza di fattori esterni non misurati o non conosciuti, favorendo un'**inferenza causale** più solida.
- **Gruppi di controllo**:
  - Consentono di confrontare i risultati di chi riceve il trattamento/intervento con chi non lo riceve (o riceve un trattamento placebo).
  - Aiutano a isolare l'effetto "vero" del trattamento dalle variazioni dovute a **effetti psicologici** (ad es. effetto placebo), al passare del tempo o a eventi esterni.
- **Difficoltà negli studi osservazionali**:
  - Non prevedono la manipolazione diretta di una variabile indipendente né l'assegnazione casuale dei partecipanti: le persone "si assegnano da sole" alle condizioni.
  - Manca il **controllo sperimentale**: non è sempre possibile includere un gruppo di controllo o applicare procedure di

randomizzazione.

- Le variabili confondenti possono agire in modo non controllabile e compromettere l'interpretazione causale: anche con analisi statistiche sofisticate, è difficile escludere del tutto la presenza di fattori esterni che influenzano la relazione tra esposizione e outcome.

Per questi motivi, gli **studi sperimentali** (con randomizzazione e controllo) rimangono il metodo privilegiato per stabilire nessi di causalità, mentre gli **studi osservazionali** servono principalmente a generare ipotesi, descrivere fenomeni, o analizzare relazioni di associazione più che di causalità.

5. Quali sono i principali criteri da considerare quando si valuta la validità di uno studio in termini di campionamento, disegno di ricerca e replicabilità?

1. **Rappresentatività del campione**:
   - Il campione rispecchia realmente le caratteristiche della popolazione di interesse?
   - È stato usato un metodo di campionamento appropriato (probabilistico vs. non probabilistico)?
2. **Controllo e randomizzazione (validità interna)**:
   - Lo studio ha previsto un **disegno sperimentale** con assegnazione casuale e gruppo di controllo?
   - Quanto è efficace il controllo delle variabili confondenti (bias di selezione, aspettative, effetto placebo, ecc.)?
3. **Generalizzabilità o validità esterna**:
   - I risultati dello studio sono applicabili oltre il contesto specifico in cui è stato condotto?
   - Vi sono limitazioni dovute all'uso di un campione di convenienza o di un contesto culturale molto particolare?
4. **Standardizzazione delle procedure**:
   - Le istruzioni, i tempi di raccolta dati, i materiali utilizzati sono stati gestiti in modo uniforme per tutti i partecipanti?
5. **Replicabilità**:
   - È possibile ripetere lo studio (replicazione diretta o concettuale) e ottenere risultati simili?
   - Gli autori forniscono informazioni sufficienti (materiali, protocolli, analisi) per consentire la replicazione?

6. **Chiarezza nell'esposizione dei limiti**:
   - Gli autori discutono apertamente le limitazioni del metodo di campionamento o del disegno di ricerca e suggeriscono possibili miglioramenti per studi futuri?

Nel complesso, una valutazione critica di uno studio deve integrare tutti questi aspetti (campionamento, disegno, replicabilità) per stabilire in che misura i risultati siano solidi, attendibili e utilmente generalizzabili.

Bibliografia

3

La misurazione in psicologia

! In questo capitolo imparerai a

- conoscere le proprietà delle scale di misura di Stevens;
- comprendere quali operazioni aritmetiche possono essere applicate a ciascun livello di scala e perchè;
- distinguere tra variabili continue e discrete;
- comprendere il concetto di *bias*.

Prerequisiti

- Leggere On the philosophical foundations of psychological measurement (**?**) sui fondamenti filosofici della misurazione psicologica.
- Leggere Psychological Measurement and the Replication Crisis: Four Sacred Cows (**?**). Questo articolo mette in relazione le proprietà delle misure psicologiche con la crisi della replicabilità dei risultati della ricerca.
- Leggere il Capitolo **??** dell'Appendice.

3.1 Introduzione

La scienza si avvale di modelli per interpretare i dati, ma opera sempre con teorie incomplete e misurazioni soggette a errori. Di conseguenza, è fondamentale riconoscere le incertezze quando si cerca di estrarre informazioni dalle misurazioni utilizzando i nostri modelli. Nessuna misurazione, spiegazione o previsione è perfettamente accurata e precisa, e non possiamo mai conoscere con esattezza l'entità dei loro errori. Questo riconoscimento è alla base della teoria della misurazione, che cerca di quantificare e gestire queste incertezze per migliorare la qualità delle nostre conclusioni scientifiche.

Questa incertezza viene catturata in tre equazioni fondamentali. La prima è l'*Equazione di Misurazione*, che riconosce l'errore osservativo:

$$y = z + \varepsilon_y,$$

dove y rappresenta il valore misurato, z il valore reale e ε_y l'errore di misurazione. La seconda è l'*Equazione di Modellazione*, che esprime la presenza di un diverso tipo di errore:

$$z = f(x, \theta) + \varepsilon_{\text{model}},$$

dove f è il modello, x sono le condizioni ambientali per cui eseguiamo il modello, θ sono i valori dei parametri del modello e $\varepsilon_{\text{model}}$ rappresenta l'errore del modello, che sorge perché f, x e θ saranno tutti in qualche misura imprecisi.

Combinando queste due equazioni, otteniamo l'*Equazione della Scienza*:

$$y = f(x, \theta) + \varepsilon_{\text{model}} + \varepsilon_y.$$

La scienza è il tentativo di spiegare le osservazioni y utilizzando un modello f, cercando di minimizzare l'errore di misurazione ε_y e l'errore del modello $\varepsilon_{\text{model}}$, in modo che il modello possa essere utilizzato per fare previsioni sul mondo reale (z). L'approccio bayesiano alla scienza riconosce e quantifica le incertezze su tutti e sei gli elementi dell'Equazione della Scienza: y, f, x, θ, $\varepsilon_{\text{model}}$ e ε_y.

3.2 La teoria della Misurazione

La teoria della misurazione, oggetto di questo capitolo, si concentra sull'errore di misurazione e sull'equazione fondamentale $y = z+\varepsilon_y$. Questa equazione può essere esaminata da tre prospettive distinte. La prima concerne l'affidabilità della misura, rappresentata dal termine ε_y. La psicometria, branca dedicata alla teoria della misurazione psicologica, si occupa di quantificare l'affidabilità delle misure psicologiche attraverso metodi come la Teoria Classica dei Test e la Teoria di Risposta all'Item.

La seconda prospettiva riguarda la validità delle misure psicologiche, ovvero quanto adeguatamente la misura y rappresenti il costrutto z. Questo aspetto, più complesso dell'affidabilità, non può essere risolto meramente con metodi statistici, ma richiede una profonda comprensione delle teorie psicologiche e della loro capacità di descrivere e prevedere i fenomeni psicologici.

La terza prospettiva si concentra sulle procedure di assegnazione dei valori a y, esplorando quali metodi (questionari, interviste, esperimenti) siano più appropriati e come valutarne l'adeguatezza.

3.2.1 Costrutti Psicologici

La teoria della misurazione sottolinea l'importanza di distinguere tra la procedura di misurazione e il costrutto che si intende misurare. Ad esempio, mentre la temperatura è un costrutto, il termometro è lo strumento di misurazione. Analogamente, l'abilità matematica è un costrutto, mentre un test di matematica è la procedura per misurarla.

Nelle scienze psicologiche e sociali, la misurazione presenta sfide uniche rispetto alle scienze fisiche, poiché i costrutti in esame sono spesso astratti e non direttamente osservabili. Ciò richiede una particolare attenzione alla validità e all'affidabilità degli strumenti di misurazione, nonché una costante riflessione sulle limitazioni e le potenziali fonti di errore.

Il capitolo introduce concetti fondamentali relativi alla misurazione quantitativa delle caratteristiche psicologiche, con un focus sulla teoria delle scale di misura di Stevens (1946). Questa teoria fornisce un quadro concettuale per comprendere i diversi tipi di scale di misurazione e le operazioni matematiche appropriate per ciascuna. Inoltre, vengono esplorate alcune procedure di scaling psicologico, ovvero l'assegnazione di numeri all'intensità di fenomeni psicologici.

3.2.2 Scaling Psicologico

Lo scaling psicologico si occupa della trasformazione dei dati empirici raccolti durante uno studio psicologico in misure o punteggi che rappresentino accuratamente le caratteristiche psicologiche oggetto di indagine.

Scaling di Guttman. Uno dei metodi di scaling più noti è lo «Scaling di Guttman», che viene utilizzato per rappresentare relazioni ordinate tra gli elementi di una scala. Ad esempio, in un questionario sui sintomi dell'ansia, le domande possono essere disposte in ordine di intensità crescente dei sintomi. Secondo il modello di Guttman, se un partecipante risponde "sì" a una domanda che riflette un sintomo più intenso, ci si aspetta che abbia risposto "sì" anche a tutte le domande precedenti, che rappresentano sintomi di intensità minore. Questo approccio consente di costruire una scala che riflette in modo sistematico e coerente la gravità dei sintomi.

Scaling Thurstoniano. Lo «Scaling Thurstoniano» è un metodo utilizzato per misurare preferenze o giudizi soggettivi. Ad esempio, per valutare la preferenza tra diversi tipi di cibi, i partecipanti confrontano due cibi alla volta ed esprimono una preferenza. Le risposte vengono poi utilizzate per assegnare punteggi che riflettono la preferenza media per ciascun cibo.

Questionari Likert. I questionari Likert richiedono ai partecipanti di esprimere il loro grado di accordo con una serie di affermazioni su una scala a più livelli, che va da «fortemente in disaccordo» a «fortemente d'accordo». I pun-

teggi ottenuti vengono sommati per rappresentare la posizione complessiva dell'individuo rispetto all'oggetto di studio.

3.2.3 Metodi di Valutazione delle Scale Psicologiche

Per valutare le proprietà delle scale psicologiche, vengono utilizzati vari metodi. Ad esempio, l'affidabilità delle misure può essere analizzata utilizzando il coefficiente alpha di Cronbach o il coefficiente Omega di McDonald, entrambi utilizzati per misurare la coerenza interna delle risposte ai diversi item di un questionario. Inoltre, la validità delle scale può essere esaminata confrontando i risultati ottenuti con misure simili o attraverso analisi statistiche che verificano se la scala cattura accuratamente il costrutto psicologico che si intende misurare. La validità di costrutto è particolarmente cruciale, poiché riguarda la capacità della scala di misurare effettivamente il concetto psicologico che si intende esplorare.

3.2.4 Prospettive Moderne

Negli ultimi anni, il dibattito sulla misurazione psicologica si è arricchito di nuove prospettive, grazie all'avvento di tecnologie avanzate e all'integrazione di approcci interdisciplinari. Ecco alcune delle tendenze più rilevanti.

Teoria della Risposta agli Item. La Teoria della Risposta agli Item (IRT) ha guadagnato popolarità per la sua capacità di fornire stime più precise delle abilità latenti rispetto ai modelli classici. La IRT considera la probabilità che un individuo risponda correttamente a un item in funzione della sua abilità e delle caratteristiche dell'item stesso, offrendo una visione più dettagliata delle proprietà psicometriche degli strumenti di misurazione.

Approcci Bayesiani. Gli approcci bayesiani stanno rivoluzionando il campo della psicometria, permettendo di incorporare informazioni a priori nelle stime e di aggiornare le credenze sulla base di nuovi dati. Questi metodi sono particolarmente utili per affrontare la complessità e l'incertezza inerenti alla misurazione psicologica.

Analisi di Rete. L'analisi di rete è un'altra metodologia emergente che vede i costrutti psicologici non come variabili latenti indipendenti, ma come reti di sintomi interconnessi. Questo approccio può offrire nuove intuizioni sulla struttura delle psicopatologie e sulla dinamica dei sintomi.

3.3 Le scale di misurazione

Le scale di misurazione sono strumenti fondamentali per assegnare numeri ai dati osservati, rappresentando le proprietà psicologiche. La teoria delle scale di Stevens (1946) identifica quattro tipi di scale di misurazione: nominali, ordinali, a intervalli e di rapporti. Ognuna di queste scale consente di effettuare operazioni aritmetiche diverse, poiché ciascuna di esse è in grado di "catturare" solo alcune delle proprietà dei fenomeni psicologici che si intende misurare.

Scale di modalità	Operazioni aritmetiche
nominali	enumerare le classi di equivalenza e/o le frequenze per ciascuna classe di equivalenza
ordinali	enumerare le classi di equivalenza e/o le frequenze per ciascuna classe di equivalenza
intervallari	differenze (rapporti tra differenze)
di rapporti	rapporti diretti tra le misure

Figura 3.1: Scale di misurazione.

3.3.1 Scala nominale

La scala nominale è il livello di misurazione più semplice e corrisponde ad una tassonomia o classificazione delle categorie che utilizziamo per descrivere i fenomeni psicologici. I simboli o numeri che costituiscono questa scala rappresentano i nomi delle categorie e non hanno alcun valore numerico intrinseco. Con la scala nominale possiamo solo distinguere se una caratteristica psicologica è uguale o diversa da un'altra.

I dati raccolti con la scala nominale sono suddivisi in categorie qualitative e mutuamente esclusive, in cui ogni dato appartiene ad una sola categoria. In questa scala, esiste solo la relazione di equivalenza tra le misure delle unità di studio: gli elementi del campione appartenenti a classi diverse sono differenti, mentre tutti quelli della stessa classe sono tra loro equivalenti.

L'unica operazione algebrica consentita dalla scala nominale è quella di contare le unità di studio che appartengono ad ogni categoria e il numero totale di categorie. Di conseguenza, la descrizione dei dati avviene tramite le frequenze assolute e le frequenze relative.

Dalla scala nominale è possibile costruire altre scale nominali equivalenti alla prima, trasformando i valori della scala di partenza in modo tale da cambiare

i nomi delle categorie, ma lasciando inalterata la suddivisione delle unità di studio nelle medesime classi di equivalenza. In altre parole, cambiando i nomi delle categorie di una variabile misurata su scala nominale, si ottiene una nuova variabile esattamente equivalente alla prima.

3.3.2 Scala ordinale

La scala ordinale mantiene la caratteristica della scala nominale di classificare ogni unità di misura all'interno di una singola categoria, ma introduce la relazione di ordinamento tra le categorie. In quanto basata su una relazione di ordine, una scala ordinale descrive solo il rango di ordine tra le categorie e non fornisce informazioni sulla distanza tra di esse. Non ci dice, ad esempio, se la distanza tra le categorie a e b è uguale, maggiore o minore della distanza tra le categorie b e c.

Un esempio classico di scala ordinale è quello della scala Mohs per la determinazione della durezza dei minerali. Per stabilire la durezza dei minerali si usa il criterio empirico della scalfittura. Vengono stabiliti livelli di durezza crescente da 1 a 10 con riferimento a dieci minerali: talco, gesso, calcite, fluorite, apatite, ortoclasio, quarzo, topazio, corindone e diamante. Un minerale appartenente ad uno di questi livelli se scalfisce quello di livello inferiore ed è scalfito da quello di livello superiore.

Mohs Hardness	Mineral	Scratch hardness
1	Talc	.59
2	Gypsum	.61
3	Calcite	3.44
4	Fluorite	3.05
5	Apaptite	5.2
6	Orthoclase Feldspar	37.2
7	Quartz	100
8	Topaz	121
9	Corundum	949
10	Diamond	85,300

Figura 3.2: La scala di durezza dei minerali di Mohs. Un oggetto è considerato più duro di X se graffia X. Sono incluse anche misure di durezza relativa utilizzando uno sclerometro, da cui emerge la non linearità della scala di Mohs (Burchard, 2004).

3.3.3 Scala ad intervalli

La scala ad intervalli di misurazione include le proprietà della scala nominale e della scala ordinale e permette di misurare le distanze tra le coppie di unità statistiche in termini di un intervallo costante, chiamato "unità di misura", a cui viene attribuito il valore "1". L'origine della scala, ovvero il punto zero, è scelta arbitrariamente e non indica l'assenza della proprietà che si

sta misurando. Ciò significa che la scala ad intervalli consente anche valori negativi e lo zero non viene attribuito all'unità statistica in cui la proprietà risulta assente.

La scala ad intervalli equivalenti consente l'esecuzione di operazioni algebriche basate sulla differenza tra i numeri associati ai diversi punti della scala, operazioni algebriche non possibili con le scale di misura nominale o ordinale. Tuttavia, il limite della scala ad intervalli è che non consente di calcolare il rapporto tra coppie di misure. È possibile affermare la differenza tra a e b come la metà della differenza tra c e d o che le due differenze sono uguali, ma non è possibile affermare che a abbia una proprietà misurata in quantità doppia rispetto a b. In altre parole, non è possibile stabilire rapporti diretti tra le misure ottenute. Solo le differenze tra le modalità permettono tutte le operazioni aritmetiche, come la somma, l'elevazione a potenza o la divisione, che sono alla base della statistica inferenziale.

Nelle scale ad intervalli equivalenti, l'unità di misura è arbitraria e può essere cambiata attraverso una dilatazione, ovvero la moltiplicazione di tutti i valori della scala per una costante positiva. Inoltre, la traslazione, ovvero l'aggiunta di una costante a tutti i valori della scala, è ammessa poiché non altera le differenze tra i valori della scala. La scala rimane invariata rispetto a traslazioni e dilatazioni e dunque le uniche trasformazioni ammissibili sono le trasformazioni lineari:

$$y' = a + by, \quad b > 0.$$

Infatti, l'uguaglianza dei rapporti fra gli intervalli rimane invariata a seguito di una trasformazione lineare.

Esempio di scala ad intervalli è la temperatura misurata in gradi Celsius o Fahrenheit, ma non Kelvin. Come per la scala nominale, è possibile stabilire se due modalità sono uguali o diverse: $30°C \neq 20°C$. Come per la scala ordinale è possibile mettere due modalità in una relazione d'ordine: $30°C > 20°C$. In aggiunta ai casi precedenti, però, è possibile definire una unità di misura per cui è possibile dire che tra $30°C$ e $20°C$ c'è una differenza di $30° - 20° = 10°C$. I valori di temperatura, oltre a poter essere ordinati secondo l'intensità del fenomeno, godono della proprietà che le differenze tra loro sono direttamente confrontabili e quantificabili.

Il limite della scala ad intervalli è quello di non consentire il calcolo del rapporto tra coppie di misure. Ad esempio, una temperatura di $80°C$ non è il doppio di una di $40°C$. Se infatti esprimiamo le stesse temperature nei termini della scala Fahrenheit, allora i due valori non saranno in rapporto di 1 a 2 tra loro. Infatti, $20°C = 68°F$ e $40°C = 104°F$. Questo significa che la relazione "il doppio di" che avevamo individuato in precedenza si applicava ai numeri della scala centigrada, ma non alla proprietà misurata (cioè la temperatura). La

decisione di che scala usare (Centigrada vs. Fahrenheit) è arbitraria. Ma questa arbitrarietà non deve influenzare le inferenze che traiamo dai dati. Queste inferenze, infatti, devono dirci qualcosa a proposito della realtà empirica e non possono in nessun modo essere condizionate dalle nostre scelte arbitrarie che ci portano a scegliere la scala Centigrada piuttosto che quella Fahrenheit.

Consideriamo ora l'aspetto invariante di una trasformazione lineare, ovvero l'uguaglianza dei rapporti fra intervalli. Prendiamo in esame, ad esempio, tre temperature: $20°C = 68°F$, $15°C = 59°F$, $10°C = 50°F$.

È facile rendersi conto del fatto che i rapporti fra intervalli restano costanti indipendentemente dall'unità di misura che è stata scelta:

$$\frac{20°C - 10°C}{20°C - 15°C} = \frac{68°F - 50°F}{68°F - 59°F} = 2.$$

3.3.4 Scala di rapporti

Nella scala a rapporti equivalenti, lo zero non è arbitrario e rappresenta l'elemento che ha intensità nulla rispetto alla proprietà misurata. Per costruire questa scala, si associa il numero 0 all'elemento con intensità nulla e si sceglie un'unità di misura u. Ad ogni elemento si assegna un numero a definito come $a = d/u$, dove d rappresenta la distanza dall'origine. In questo modo, i numeri assegnati riflettono le differenze e i rapporti tra le intensità della proprietà misurata.

In questa scala, è possibile effettuare operazioni aritmetiche non solo sulle differenze tra i valori della scala, ma anche sui valori stessi della scala. L'unica scelta arbitraria è l'unità di misura, ma lo zero deve sempre rappresentare l'intensità nulla della proprietà considerata.

Le trasformazioni ammissibili in questa scala sono chiamate trasformazioni di similarità e sono del tipo $y' = by$, dove $b > 0$. In questa scala, i rapporti tra i valori rimangono invariati dopo le trasformazioni. In altre parole, se rapportiamo due valori originali e due valori trasformati, il rapporto rimane lo stesso: $\frac{y_i}{y_j} = \frac{y'_i}{y'_j}$.

3.4 Gerarchia dei livelli delle scale di misurazione

Secondo Stevens (1946), esiste una gerarchia dei livelli delle scale di misurazione, denominati "livelli di scala". Questi livelli sono organizzati in modo gerarchico, in cui la scala nominale rappresenta il livello più basso della misurazione, mentre la scala a rapporti equivalenti rappresenta il livello più alto.

- **Scala nominale**: Classifica le categorie senza un ordine specifico.
- **Scala ordinale**: Classifica le categorie in un ordine specifico, ma senza una misura precisa delle distanze.
- **Scala a intervalli**: Misura le distanze tra le categorie con un intervallo costante, ma senza un punto zero assoluto.
- **Scala di rapporti**: Misura le distanze con un intervallo costante e un punto zero assoluto.

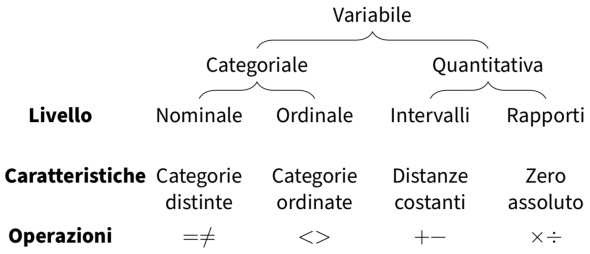


Figura 3.3: Relazioni tra i livelli di misurazione.

Passando da un livello di misurazione ad uno più alto aumenta il numero di operazioni aritmetiche che possono essere compiute sui valori della scala.

3.4.1 Variabili Discrete e Continue

Le variabili possono essere classificate come variabili a livello di intervalli o di rapporti e possono essere sia discrete che continue.

- **Variabili discrete**: Assumono valori specifici ma non possono assumere valori intermedi.
- **Variabili continue**: Possono assumere qualsiasi valore all'interno di un intervallo specificato.

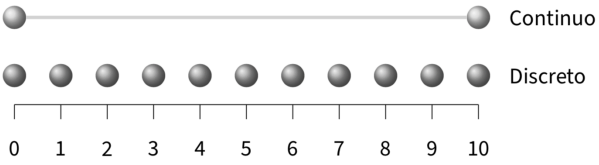


Figura 3.4: Variabili discrete e continue.

3.5 Relazione tra Misurazione e Teoria Sostanziale: Un'Analisi Critica

Un esempio di come condurre una lettura critica di un articolo scientifico è rappresentato dall'analisi di uno studio pubblicato su *Nature* sul tema del **mind-body healing** (**?**). Lo studio presenta risultati empirici che suggeriscono un legame tra pratiche di guarigione mente-corpo e miglioramenti nella salute fisica. Tuttavia, la validità di queste conclusioni è stata fortemente contestata da Andrew Gelman, un noto statistico e ricercatore, in un post sul suo blog Statistical Modeling.

3.5.1 L'Importanza di una Teoria Sostanziale Solida

Gelman sottolinea un aspetto cruciale della ricerca scientifica: l'importanza di una **teoria sostanziale solida e convincente**. Secondo il suo punto di vista, lo studio analizzato presenta una carenza significativa in questo ambito. In assenza di una teoria ben fondata che spieghi in modo plausibile i meccanismi causali attraverso cui le pratiche mente-corpo potrebbero influenzare la salute fisica, i risultati empirici ottenuti rimangono privi di significato. Senza una cornice teorica coerente, lo studio rischia di essere classificato come ciò che Gelman definisce "junk science" – scienza di scarso valore, incapace di offrire vere e proprie intuizioni.

Una teoria sostanziale non deve solo essere plausibile, ma deve anche integrarsi in modo coerente con altre teorie scientifiche consolidate. Quando una ricerca non riesce a inserirsi in una rete più ampia di conoscenze, i suoi risultati diventano inevitabilmente discutibili. È proprio questo il caso dello studio esaminato da Gelman: l'assenza di una teoria chiara e ben articolata rende i risultati empirici poco credibili e difficilmente generalizzabili.

3.5.2 Problemi di Misurazione

Oltre alla debolezza teorica, lo studio presenta anche notevoli problemi legati alla **misurazione**. Le variabili utilizzate per quantificare sia gli interventi mente-corpo che gli esiti sanitari sono soggette a diverse criticità. Ad esempio, la misurazione dell'efficacia delle pratiche mente-corpo potrebbe essere influenzata da fattori confondenti, come le aspettative dei partecipanti o l'effetto placebo. Allo stesso modo, la valutazione degli esiti sanitari può risultare problematica se non vengono impiegate scale di misurazione affidabili e valide.

Per un approfondimento su questi temi, si rimanda alle riflessioni conclusive nel Capitolo **??**, dove vengono discusse criticamente le questioni legate alla misurazione, con particolare attenzione alla distinzione tra precisione e bias.

In generale, uno degli aspetti più critici della misurazione riguarda la validità interna ed esterna negli studi psicologici. La qualità delle misure adottate influisce direttamente sulla solidità delle conclusioni scientifiche: una misurazione imprecisa o distorta può compromettere la validità dei risultati e limitarne la generalizzabilità.

3.5.3 La Necessità di una Valutazione Bilanciata

L'esempio dello studio sul mind-body healing illustra in modo efficace come una lettura critica debba necessariamente passare attraverso due dimensioni fondamentali: la **teoria** e la **misurazione**. Una teoria debole compromette la capacità di attribuire significato ai dati raccolti, riducendo lo studio a una mera raccolta di numeri privi di valore scientifico. Allo stesso tempo, problemi di misurazione possono distorcere i risultati, portando a conclusioni errate o fuorvianti. Solo attraverso una valutazione attenta di entrambi questi aspetti è possibile distinguere tra ricerche scientificamente valide e "junk science".

In definitiva, una lettura critica di un articolo psicologico richiede non solo competenze tecniche, ma anche una profonda consapevolezza dei limiti e delle sfide insite nella misurazione psicologica e nel rapporto tra misurazione e teorie scientifiche. Questo approccio consente di valutare in modo equilibrato la qualità e l'affidabilità delle conclusioni proposte. Solo attraverso una rigorosa analisi di entrambi questi aspetti è possibile distinguere tra ricerche scientificamente valide e quelle che, invece, rischiano di essere considerate "junk science".

3.6 Riflessioni Conclusive

La misurazione in psicologia non è un semplice atto di raccolta di dati, ma un processo fondamentale per garantire che le osservazioni empiriche siano interpretabili alla luce di modelli teorici solidi. Una buona misurazione non si limita a ridurre l'errore, ma consente di attribuire un significato coerente ai punteggi ottenuti, facilitando così il progresso della conoscenza scientifica. Senza strumenti adeguati per la misurazione, il rischio è quello di costruire teorie su basi incerte, compromettendo la validità delle conclusioni tratte.

Due pilastri sostengono una ricerca psicologica rigorosa: la teoria e la misurazione. La teoria fornisce il quadro concettuale entro cui si interpretano i dati, definendo le ipotesi e orientando le analisi. La misurazione, invece, è il ponte tra i costrutti astratti e le osservazioni empiriche, traducendo concetti complessi in variabili operative affidabili. Nessuna delle due componenti può reggersi senza l'altra: una teoria senza misurazione adeguata rischia di

rimanere speculativa, mentre una misurazione priva di un solido fondamento teorico può portare a dati privi di significato.

Nella valutazione di un qualsiasi studio psicologico, un approccio critico richiede quindi di esaminare sia la solidità del quadro teorico sia la qualità degli strumenti di misurazione adottati. Il progresso della ricerca dipende dalla capacità di integrare questi due elementi, attraverso metodologie sempre più sofisticate che riducano l'incertezza e migliorino la precisione delle inferenze. Le moderne tecniche di analisi dei dati, i modelli psicometrici avanzati e le tecnologie digitali stanno ampliando le possibilità di misurazione, offrendo strumenti più sensibili e adattabili alla complessità dei fenomeni psicologici. Tuttavia, la sfida principale rimane la stessa: garantire che la misurazione sia non solo accurata, ma anche teoricamente fondata, affinché le conoscenze acquisite possano davvero contribuire alla comprensione della mente e del comportamento umano.

Esercizi

! Problemi 1

Esercizio 1: Identificazione del Livello di Misurazione
Obiettivo: Comprendere i diversi livelli di misurazione applicati alla psicologia.

1. Identifica il livello di misurazione (nominale, ordinale, intervalli, rapporti) per ciascuna delle seguenti variabili psicologiche:
   - a) Tipo di terapia psicologica (Cognitivo-comportamentale, Psicodinamica, Umanistica)
   - b) Livello di ansia auto-riferito su una scala da 1 a 10
   - c) Numero di episodi depressivi in un anno
   - d) Tempo di reazione in millisecondi in un test cognitivo

Esercizio 2: Confronto tra Scale
Obiettivo: Comprendere le differenze tra le scale di misurazione.

1. Spiega la differenza tra una scala ordinale e una scala a intervalli utilizzando l'esempio della soddisfazione lavorativa.
2. Perché il punteggio QI è misurato su una scala a intervalli e non su una scala a rapporti?
3. In che modo il punteggio di una scala di autostima su una scala Likert differisce da una misurazione su una scala di rapporti?

Esercizio 3: Operazioni Aritmetiche Consentite
Obiettivo: Comprendere le operazioni matematiche consentite per ciascun livello di misurazione.

1. Quali operazioni aritmetiche sono ammissibili per una scala nominale?
2. Può avere senso calcolare la media di punteggi su una scala ordinale? Perché?
3. Se hai misurato il tempo di reazione in secondi, quali operazioni aritmetiche puoi eseguire?

Esercizio 4: Trasformazioni Ammissibili
Obiettivo: Comprendere le trasformazioni possibili per ogni scala di misurazione.

1. Se una variabile è misurata su una scala nominale, quale tipo di trasformazione è consentita?
2. Per una scala a intervalli, quali trasformazioni matematiche sono permesse senza alterare le proprietà della scala?
3. Quale tipo di trasformazione è consentita su una scala di rapporti?

Esercizio 5: Applicazione delle Scale a Dati Psicologici
Obiettivo: Applicare i concetti a contesti psicologici reali.

1. Una scala di ansia clinica fornisce punteggi compresi tra 0 e 100. Quale livello di misurazione è più appropriato e perché?
2. Un esperimento misura la memoria dichiarativa chiedendo ai partecipanti di ricordare un elenco di parole. Come dovrebbe essere misurata la variabile "numero di parole ricordate"?
3. In uno studio sulla personalità, i tratti vengono classificati come "estroverso" e "introverso". Qual è il livello di misurazione?

Esercizio 6: Valutazione della Scala di Misurazione
Obiettivo: Identificare la corretta scala di misurazione per vari fenomeni psicologici.

1. Il livello di aggressività misurato su una scala da 1 a 5 è nominale, ordinale, intervalli o rapporti? Giustifica la tua risposta.
2. Il numero di attacchi di panico in una settimana può essere considerato su scala ordinale? Perché sì o perché no?
3. Un test di intelligenza misura il QI con una media di 100 e una deviazione standard di 15. Qual è il livello di misurazione e quali sono le implicazioni per l'analisi statistica?

Esercizio 7: Costruzione di una Scala Psicologica

Obiettivo: Creare una scala di misurazione per una variabile psicologica.

1. Se dovessi costruire una scala per misurare la resilienza, quale livello di misurazione sceglieresti e perché?
2. Come potresti trasformare una scala nominale di preferenza musicale in una scala ordinale?
3. Un questionario sulla qualità della vita chiede ai partecipanti di valutare la loro felicità su una scala da 1 a 10. È una scala a intervalli o ordinale? Giustifica.

Esercizio 8: Interpretazione Statistica dei Dati
Obiettivo: Collegare il livello di misurazione alle tecniche statistiche appropriate.

1. Perché una mediana è più appropriata della media per dati ordinali?
2. Quale test statistico sarebbe più adatto per confrontare due gruppi su una variabile nominale?
3. Quali analisi possono essere condotte su dati raccolti su una scala a rapporti?

Esercizio 9: Misurazione e Inferenze Psicologiche
Obiettivo: Riflettere su come il livello di misurazione influisce sulle conclusioni di una ricerca.

1. Se un test di personalità usa una scala Likert da 1 a 7, quali precauzioni devono essere prese nell'interpretare le differenze tra punteggi?
2. Un questionario di benessere assegna punteggi tra 0 e 100, ma non ha uno zero assoluto. Quale scala è questa e quali sono le limitazioni?
3. In uno studio sulla depressione, i sintomi vengono codificati come "assenti", "moderati" o "gravi". Che tipo di scala è questa e quali statistiche possono essere usate per analizzarla?

Esercizio 10: Esperimenti Psicologici e Misurazione
Obiettivo: Applicare la teoria della misurazione nella progettazione di esperimenti psicologici.

1. Se un esperimento misura la memoria a breve termine con un compito di richiamo di parole, quale scala di misurazione utilizzeresti?
2. Come la scelta della scala di misurazione può influenzare le inferenze che si possono trarre da un esperimento?
3. Quali tipi di analisi statistica sono appropriati per dati misurati su scala ordinale rispetto a scala di rapporti?

Soluzioni 1

Esercizio 1: Identificazione del Livello di Misurazione
Obiettivo: Comprendere i diversi livelli di misurazione applicati alla psicologia.

1. Identifica il livello di misurazione (nominale, ordinale, intervalli, rapporti) per ciascuna delle seguenti variabili psicologiche:
   - a) **Nominale** (Tipo di terapia psicologica è una classificazione senza ordine)
   - b) **Ordinale** (Scala da 1 a 10, con ordine ma senza distanze uguali)
   - c) **Rapporti** (Numero di episodi depressivi ha uno zero assoluto e si possono fare rapporti tra valori)
   - d) **Rapporti** (Tempo di reazione ha uno zero assoluto e permette operazioni di rapporto)

Esercizio 2: Confronto tra Scale
Obiettivo: Comprendere le differenze tra le scale di misurazione.

1. La scala ordinale fornisce un ordine ma non permette di calcolare differenze precise, mentre la scala a intervalli ha differenze costanti tra i valori. Ad esempio, "soddisfazione lavorativa" su una scala da 1 a 5 è ordinale, mentre il punteggio di un test psicologico è a intervalli.
2. Il punteggio QI è a intervalli perché la differenza tra punteggi è significativa, ma non ha uno zero assoluto che rappresenta l'assenza di intelligenza.
3. Una scala Likert misura il livello di accordo con una dichiarazione, quindi è generalmente considerata ordinale, nonostante sia trattata spesso come una scala a intervalli.

Esercizio 3: Operazioni Aritmetiche Consentite
Obiettivo: Comprendere le operazioni matematiche consentite per ciascun livello di misurazione.

1. Nella scala nominale si può solo contare la frequenza delle categorie (ad es., il numero di partecipanti che usano un tipo di terapia).
2. No, la media su dati ordinali può essere fuorviante perché le distanze tra le categorie non sono necessariamente uguali. Meglio usare la mediana.
3. Sul tempo di reazione si possono eseguire tutte le operazioni aritmetiche, inclusa la media, la moltiplicazione e i rapporti tra valori.

Esercizio 4: Trasformazioni Ammissibili

Obiettivo: Comprendere le trasformazioni possibili per ogni scala di misurazione.

1. Sulla scala nominale, solo le trasformazioni di ricodifica (ad esempio, cambiare i nomi delle categorie) sono permesse.
2. Per una scala a intervalli, si possono effettuare trasformazioni lineari della forma y' = a + by con b > 0.
3. Per una scala di rapporti, sono consentite trasformazioni di similarità della forma y' = by, dove b > 0.

Esercizio 5: Applicazione delle Scale a Dati Psicologici
Obiettivo: Applicare i concetti a contesti psicologici reali.

1. Scala a intervalli, perché ha differenze costanti tra i punteggi ma nessuno zero assoluto.
2. Scala di rapporti, perché il numero di parole ricordate ha uno zero assoluto e consente operazioni di rapporto.
3. Nominale, perché non vi è un ordine gerarchico tra le categorie "estroverso" e "introverso".

Esercizio 6: Valutazione della Scala di Misurazione
Obiettivo: Identificare la corretta scala di misurazione per vari fenomeni psicologici.

1. Ordinale, perché il livello di aggressività segue un ordine, ma le differenze tra i livelli non sono necessariamente uguali.
2. No, perché il numero di attacchi di panico è una variabile discreta e misurabile su scala di rapporti.
3. Intervalli, perché il punteggio QI ha distanze costanti tra i valori, ma non ha uno zero assoluto.

Esercizio 7: Costruzione di una Scala Psicologica
Obiettivo: Creare una scala di misurazione per una variabile psicologica.

1. Ordinale o a intervalli, a seconda della precisione della misurazione della resilienza.
2. Si potrebbe assegnare un valore numerico crescente alle categorie di preferenza musicale per ottenere una scala ordinale.
3. È una scala ordinale, perché la differenza tra livelli non è necessariamente costante.

Esercizio 8: Interpretazione Statistica dei Dati
Obiettivo: Collegare il livello di misurazione alle tecniche statistiche appropriate.

1. Perché la mediana è meno sensibile ai valori estremi rispetto alla media.

2. Un test chi-quadrato è adatto per confrontare frequenze di dati nominali tra gruppi.
3. Si possono calcolare media, deviazione standard e utilizzare test parametrici come t-test o ANOVA.

Esercizio 9: Misurazione e Inferenze Psicologiche
Obiettivo: Riflettere su come il livello di misurazione influisce sulle conclusioni di una ricerca.

1. I punteggi Likert sono ordinali, quindi confronti tra differenze di punteggio devono essere interpretati con cautela.
2. Intervalli, perché non ha uno zero assoluto, il che limita l'uso di operazioni moltiplicative.
3. Ordinale, e si possono usare test non parametrici come il test di Kruskal-Wallis o il test di Mann-Whitney.

Esercizio 10: Esperimenti Psicologici e Misurazione
Obiettivo: Applicare la teoria della misurazione nella progettazione di esperimenti psicologici.

1. Rapporti, perché il numero di parole ricordate è una variabile discreta con uno zero assoluto.
2. Se si usa una scala ordinale, bisogna essere cauti nell'uso della media e della deviazione standard.
3. Scala ordinale → test non parametrici (Mann-Whitney); scala di rapporti → test parametrici (t-test, ANOVA).

! Problemi 2

Esercizio 1 – Teoria Sostanziale e "Junk Science"
Obiettivo: Riconoscere il ruolo di una teoria sostanziale solida e comprendere come la sua assenza possa compromettere uno studio.

1. Leggi la sezione in cui Gelman critica l'assenza di una teoria solida nello studio sulle pratiche mente-corpo.

2. Spiega, in massimo **10 righe**, perché secondo Gelman la mancanza di una teoria coerente rende i risultati del suddetto studio "poco significativi" o addirittura "junk science".

3. Proponi un esempio ipotetico (non correlato al mind-body healing) di uno studio psicologico che, pur presentando dati numerosi e analizzati con metodi statistici sofisticati, risulti privo di una teoria solida. Descrivi sinteticamente perché questo potrebbe rientrare nel concetto di "junk science".

Esercizio 2 – Problemi di Misurazione
Obiettivo: Identificare le criticità più comuni nella misurazione dei fenomeni psicologici.

1. Elenca almeno **tre** possibili fattori confondenti che potrebbero influenzare la misurazione dell'efficacia di un intervento psicologico (ad esempio, l'effetto placebo, le aspettative dei partecipanti, ecc.).

2. Spiega come questi fattori confondenti potrebbero compromettere la **validità interna** dello studio.

3. Indica almeno **due** caratteristiche fondamentali che una buona scala di misurazione (per una variabile psicologica) dovrebbe possedere per essere ritenuta affidabile e valida.

Esercizio 3 – Precisione e Bias
Obiettivo: Chiarire la distinzione tra precisione e distorsione (bias) e come questi aspetti si riflettano nella validità delle conclusioni.

1. Definisci, con parole tue, i concetti di **precisione** e **bias** in ambito psicometrico.
2. Fornisci un esempio concreto di uno strumento di misura **preciso ma distorto** (bias elevato) e di uno strumento **poco preciso ma non distorto** (bias basso).

3. Spiega come la combinazione di scarsa precisione e alto bias possa influire sulla possibilità di trarre conclusioni affidabili in uno studio psicologico.

Esercizio 4 – Validità Interna ed Esterna
Obiettivo: Approfondire come le scelte di misurazione influiscano sulla validità interna ed esterna di uno studio.

1. In riferimento allo studio sul mind-body healing discusso nel capitolo, identifica due fattori che potrebbero compromettere la validità interna e due fattori che potrebbero limitarne la validità esterna.

2. Descrivi in **5-8 righe** le differenze principali tra validità interna e validità esterna, utilizzando esempi presi sia dal contesto della guarigione mente-corpo sia da altri contesti psicologici (ad esempio, studi sull'apprendimento o sulla motivazione).

3. Proponi una modifica al disegno di ricerca (ipotetico) che potrebbe migliorare la validità interna dello studio originale.

Spiega brevemente come questa modifica ne influenzerebbe anche la validità esterna.

Esercizio 5 – Integrare Teoria e Misurazione: Breve Progetto di Ricerca
Obiettivo: Mettere in pratica i concetti di teoria e misurazione attraverso la progettazione di uno studio.

1. Immagina di voler condurre uno studio su un intervento di "training di rilassamento mentale" finalizzato a ridurre l'ansia negli studenti universitari.

2. **Sviluppa una breve traccia di progetto** (massimo **15 righe**) rispondendo ai seguenti punti:
   - **Teoria di base**: Qual è la teoria sostanziale dietro l'efficacia del training di rilassamento? Quali meccanismi psicologici verrebbero attivati?
   - **Ipotesi**: Quale effetto prevedi sull'ansia degli studenti?
   - **Misurazione**: Che tipo di strumento useresti per valutare il livello di ansia e perché (ad esempio, questionari self-report validati, misure fisiologiche come battito cardiaco, ecc.)?
   - **Controllo dei confondenti**: Quali variabili secondarie possono influire sui risultati e come intendi gestirle?
   - **Validità**: Come assicureresti una buona validità interna? Che strategie adotteresti per aumentare la validità esterna?
3. Spiega brevemente in che modo la combinazione di un solido quadro teorico e di una misurazione accurata permette di evitare che lo studio venga etichettato come "junk science".

Soluzioni 2

Esercizio 1 – Teoria Sostanziale e "Junk Science"

1. Perché la mancanza di una teoria solida rende i risultati poco significativi?

- **Gelman** critica lo studio sul mind-body healing perché non vi è un modello teorico convincente che spieghi il **meccanismo causale** tra

pratiche mente-corpo e miglioramenti di salute.

- Senza un quadro teorico robusto, i risultati sono interpretati in modo **esplorativo** e rischiano di essere attribuiti a variabili non controllate (effetto placebo, regressione alla media, ecc.).

- Una teoria ben formulata aiuta a **delimitare le ipotesi**, guidare il disegno di ricerca e interpretare correttamente i dati. In assenza di ciò, i numeri raccolti potrebbero essere viziati da fattori confondenti o da semplici correlazioni spurious.

2. "Junk science" in massimo 10 righe

Esempio di testo in 10 righe (circa)
**Lo studio sul mind-body healing viene talvolta definito "junk science" da Gelman perché, in mancanza di una teoria sostanziale solida, i dati raccolti non forniscono indicazioni chiare sui processi psicologici o fisiologici coinvolti. Una ricerca classificata come "junk science" è priva di rigore metodologico o teorico, e può presentare gravi problemi di replicabilità o di interpretazione dei risultati. In particolare, se non vi è un modello plausibile che colleghi in modo coerente la pratica mente-corpo ai cambiamenti in variabili biologiche e comportamentali, i risultati empirici rischiano di essere semplici coincidenze. L'assenza di un costrutto ben definito e di ipotesi derivanti da una teoria coerente rende difficile capire se i cambiamenti osservati siano reali, casuali o dovuti ad altre cause non considerate (per esempio, l'effetto placebo). Infine, senza un'adeguata cornice teorica, gli studiosi non sanno come interpretare o generalizzare i dati, e la scienza non progredisce realmente.*

3. Esempio di uno studio privo di teoria solida (ipotesi di "junk science")

- **Situazione ipotetica**: Uno studio che raccoglie decine di variabili sulla personalità e sul benessere, poi usa tecniche statistiche sofisticate (analisi di big data, reti neurali, ecc.) per trovare correlazioni fra i tratti di personalità e centinaia di indicatori fisici.
- **Perché "junk science"**: Se lo studio non definisce **a priori** quali ipotesi testare e non ha una teoria chiara che spieghi perché certe caratteristiche di personalità dovrebbero correlarsi con determinati parametri fisici, i risultati trovati potrebbero essere frutto di coincidenze casuali. Inoltre, in assenza di un modello teorico solido, anche risultati statisticamente significativi possono essere privi di significato dal punto di vista psicologico.

Esercizio 2 – Problemi di Misurazione

1. Tre possibili fattori confondenti nell'efficacia di un intervento psicologico

- **Effetto placebo**: I partecipanti migliorano perché si aspettano di migliorare, non per l'effettiva efficacia dell'intervento.

- **Aspettative dei partecipanti**: Se sanno di partecipare a uno studio, potrebbero modificare il proprio comportamento (effetto Hawthorne).

- **Desiderabilità sociale**: I partecipanti forniscono risposte che ritengono socialmente desiderabili, falsando i risultati (ad esempio, sottostimando i livelli di ansia o stress).

2. Come questi fattori confondenti compromettono la validità interna

- La **validità interna** riguarda il grado in cui è possibile concludere che sia effettivamente la **variabile indipendente** (l'intervento) a causare le modifiche osservate nella **variabile dipendente** (es. livelli di ansia).

- Se subentrano l'effetto placebo, aspettative non controllate o tendenze alla desiderabilità sociale, diventa difficile stabilire un nesso causale chiaro. Esiste sempre il dubbio che altri processi cognitivi o sociali (non l'intervento in sé) abbiano prodotto il risultato.

3. Due caratteristiche fondamentali di una buona scala di misurazione

- **Affidabilità**: Capacità dello strumento di fornire misure stabili e coerenti nel tempo (ad esempio, coerenza interna, stabilità test-retest).

- **Validità**: Capacità dello strumento di misurare effettivamente ciò che si propone di misurare (validità di contenuto, di costrutto, di criterio).

Esercizio 3 – Precisione e Bias

1. Definizioni di precisione e bias

- **Precisione**: Indica il grado di dispersione (o variabilità) delle misurazioni. Uno strumento preciso produce misure molto simili fra loro se ripetute nelle stesse condizioni (bassa varianza).

- **Bias (distorsione)**: Indica l'errore sistematico, ossia la tendenza a sovra- o sottostimare sistematicamente il fenomeno in esame. Uno strumento può essere molto coerente nelle misure, ma se è "tarato" male, darà sempre un risultato distorto.

2. Esempio concreto di misura “precisa ma distorta” e “poco precisa ma non distorta”

- **Precisa ma distorta**: Un cronometro che, a causa di un difetto di fabbricazione, parte sempre con 2 secondi di ritardo ma poi misura i tempi con estrema coerenza. Risultato: tutte le misure saranno molto simili (alta precisione), ma sempre sfasate di 2 secondi (alto bias).

- **Poco precisa ma non distorta**: Un termometro vecchio che a volte segna 36,2°C, altre 36,7°C, altre 37,1°C, senza un pattern sistematico. In media potrebbe risultare vicino ai 36,5°C, quindi senza un bias chiaro, ma con un’alta variabilità tra una misurazione e l’altra (bassa precisione).

3. Conseguenze di scarsa precisione e alto bias

- Se uno strumento è **poco preciso** (alta variabilità) e **altamente distorto** (bias elevato), i risultati ottenuti non solo oscillano in modo imprevedibile, ma sono costantemente lontani dal valore “vero”.

- In queste condizioni, le **conclusioni diventano inaffidabili**, poiché è quasi impossibile distinguere l’effetto reale (casuale o causale) dalle deformazioni introdotte dallo strumento e dall’errore di misura.

Esercizio 4 – Validità Interna ed Esterna

1. Due fattori che compromettono la validità interna e due fattori che compromettono la validità esterna (nell’esempio del mind-body healing)

- **Validità interna**:
  - **Assegnazione non casuale ai gruppi**: se i partecipanti scelgono autonomamente di aderire alle pratiche mente-corpo, potrebbero essere più motivati o avere caratteristiche iniziali diverse.

  - **Mancata o inadeguata gestione dell’effetto placebo**: non sapere se l’intervento “mente-corpo” sia stato percepito come particolarmente “speciale” dai partecipanti può introdurre differenze di aspettativa.
- **Validità esterna**:
  - **Campione non rappresentativo**: se lo studio è condotto solo su persone che frequentano un determinato tipo di centro di benessere, i risultati potrebbero non essere generalizzabili all’intera popolazione.

  - **Contesto specifico**: pratiche mente-corpo svolte in un ambiente

estremamente controllato (es. un laboratorio o un ritiro speciale) potrebbero non replicarsi nella vita quotidiana di chiunque.

2. Differenze tra validità interna ed esterna (5-8 righe di esempio)

La validità interna si riferisce alla correttezza del disegno di ricerca nel dimostrare un effetto causale. Un alto livello di validità interna implica che i ricercatori siano ragionevolmente sicuri che l'intervento (ad esempio, una tecnica mente-corpo) abbia causato i risultati osservati (miglioramento della salute). La validità esterna, invece, riguarda la possibilità di generalizzare i risultati a contesti, persone e tempi differenti. Se un intervento è stato testato in condizioni molto specifiche, potrebbe funzionare bene solo in quel contesto e con quel particolare campione. Per esempio, un intervento sul mind-body healing con individui altamente motivati potrebbe non dare gli stessi risultati in una popolazione generalizzata. Allo stesso modo, uno studio sull'apprendimento condotto in un laboratorio altamente controllato potrebbe non riflettere le reali dinamiche di un'aula scolastica.

3. Modifica al disegno di ricerca per migliorare la validità interna e conseguenze sulla validità esterna

- **Proposta**: Introdurre un gruppo di controllo con un intervento placebo o un'attività simile ma priva di contenuto "mente-corpo" (ad es. sessioni di lettura rilassante). In questo modo, si può confrontare l'effetto "specífico" dell'intervento.
- **Come influenza la validità interna**: Con un gruppo di controllo placebo, diventa più semplice escludere che il miglioramento sia dovuto solo alle aspettative dei partecipanti. Questo riduce il rischio di confondenti e aumenta la validità interna.
- **Come influenza la validità esterna**: Potrebbe rendere il contesto dello studio più artificiale (un gruppo fa "meditazione", l'altro legge in silenzio), il che potrebbe ridurre la naturalezza della situazione e potenzialmente limitare la generalizzabilità ad ambienti reali (validità esterna).

Esercizio 5 – Integrare Teoria e Misurazione: Breve Progetto di Ricerca

1. Breve traccia di progetto: "Training di rilassamento mentale per ridurre l'ansia negli studenti universitari"

- **Teoria di base**
  Il training di rilassamento mentale si fonda sul presupposto teorico che le tecniche di riduzione dello stress (es. respirazione consapevole, rilas-

samento muscolare progressivo) possano agire sui livelli di attivazione fisiologica e sui pensieri intrusivi. Riducendo l'iperattivazione del sistema nervoso simpatico e favorendo uno stato di calma, diminuisce l'ansia percepita.

- **Ipotesi**
  Gli studenti che seguono il training di rilassamento per 4 settimane mostreranno una riduzione significativa nei punteggi di ansia, rispetto a un gruppo di controllo che non partecipa al training.
- **Misurazione**
  Utilizzo di una scala validata come lo **STAI (State-Trait Anxiety Inventory)** per misurare il livello di ansia pre e post intervento. Possibile integrazione con misure fisiologiche (battito cardiaco a riposo) per avere dati oggettivi.
- **Controllo dei confondenti**
  - Registrare la storia clinica dei partecipanti (per escludere coloro che assumono farmaci ansiolitici).
  - Richiedere che i partecipanti non modifichino drasticamente le proprie abitudini di studio o di vita durante l'intervento.
  - Assicurarsi che i valutatori non sappiano chi fa parte del gruppo di training o del gruppo di controllo (blinding parziale).
- **Validità**
  - **Validità interna**: Uso di un gruppo di controllo e assegnazione casuale (randomizzazione) per assicurare che i due gruppi siano comparabili.
  - **Validità esterna**: Inclusione di studenti provenienti da diverse facoltà, così da riflettere una maggiore eterogeneità di popolazione.

2. Come teoria solida e misurazione accurata evitano la "junk science"

- Una solida **cornice teorica** spiega i meccanismi psicologici e fisiologici che legano l'intervento (training di rilassamento) all'esito (riduzione dell'ansia).
- Una **misurazione accurata e validata** (STAI, misure fisiologiche) riduce errori e distorsioni. Se le misure sono ripetute nel tempo (pre e post), si possono confrontare i cambiamenti effettivi.
- Integrando teoria e misurazione, i risultati assumono un significato scientifico più robusto. Non basta osservare un miglioramento: occor-

re dimostrare **come e perché** tale miglioramento avvenga, evitando di cadere in semplici correlazioni prive di spiegazione (e quindi potenzialmente "junk science").

! Problemi 3

Esercizio 1 – Trasformazioni in Scala Nominale
Situazione
Un ricercatore vuole indagare la percezione di appartenenza sociale tra studenti universitari di Psicologia. A ciascuno studente viene chiesto di rispondere alla domanda: "Qual è il gruppo studentesco a cui ritieni di appartenere maggiormente?", scegliendo una tra le seguenti categorie:

- A) Gruppo A (focalizzato su ricerca e studio)
- B) Gruppo B (focalizzato su attività ricreative)
- C) Gruppo C (focalizzato su volontariato e progetti sociali)

Istruzioni

1. Identifica la **scala** di misurazione utilizzata per classificare gli studenti (nominale, ordinale, a intervalli o di rapporti).
2. Indica **quali trasformazioni** sono ammissibili su questa scala e spiega perché non è possibile applicare operazioni di tipo aritmetico (somme, differenze, etc.).
3. Proponi un esempio di **nuova scala nominale** equivalente, ossia una nuova denominazione delle categorie che rispetti la suddivisione originale. (Esempio: rinominarle in Gruppo X, Gruppo Y, Gruppo Z, oppure usare colori, animali-simbolo, ecc.). Spiega perché questa trasformazione non altera i risultati dell'indagine.

Esercizio 2 – Trasformazioni in Scala Ordinale
Situazione
In un questionario sul benessere psicologico, agli studenti viene chiesto di classificare il loro **stato di motivazione** allo studio su una scala da 1 (bassa motivazione) a 5 (alta motivazione). Si ottiene così un dato ordinalmente misurato.
Istruzioni

1. Spiega perché tale variabile ("livello di motivazione") rappresenta una **scala ordinale**. Quali proprietà la rendono diversa da una semplice scala nominale?

2. Descrivi in che modo è possibile **ridenominare** i valori della scala (ad esempio, da [1,2,3,4,5] a [“Molto bassa”, “Bassa”, “Media”, “Alta”, “Molto alta”]) senza alterare il **rapporto d’ordine** tra le categorie.

3. Proponi un esempio di **trasformazione non ammissibile**: qual è un’operazione aritmetica che non avrebbe senso applicare su una scala ordinale e perché (ad esempio, calcolare “il doppio di motivazione”)?

Esercizio 3 – Trasformazioni in Scala ad Intervalli
Situazione
Un gruppo di ricercatori in Psicometria vuole confrontare i **punteggi di un test d’intelligenza** (misurati secondo la scala tradizionale del QI, con media 100 e deviazione standard 15) con un nuovo test sperimentale. Come ben noto, la scala del QI è considerata, nelle sue approssimazioni psicometriche, una **scala ad intervalli**.
Istruzioni

1. Spiega in cosa consiste la **trasformazione lineare** ammessa (del tipo $y' = a + by$, con $b > 0$) e perché tale trasformazione preserva le **differenze** tra i punteggi.

2. Fai un esempio concreto di trasformazione lineare: supponi di voler “riscalare” i punteggi del QI in modo che la nuova media sia 50. Definisci i valori di a e b (indicando un’ipotesi di calcolo) e mostra come viene modificato il punteggio di un individuo con QI = 115.

3. Discuta perché, nonostante la somiglianza con le scale ordinale e nominale (puoi comunque distinguere punteggi e ordinarli), una **scala ad intervalli** consente operazioni matematiche più complesse (ad esempio, differenze) che non sarebbero valide negli altri due livelli.

Esercizio 4 – Trasformazioni in Scala di Rapporti
Situazione
Un laboratorio di psicofisiologia misura i **tempi di reazione** (in millisecondi) a uno stimolo luminoso. Poiché il tempo di reazione pari a 0 ms significa realmente assenza di risposta (ovvero, impossibile da misurare in pratica, ma concettualmente corrisponde a intensità nulla del fenomeno “tempo di reazione”), ci troviamo in una **scala di rapporti**.
Istruzioni

1. Spiega perché il tempo di reazione soddisfa i requisiti di una scala di rapporti, inclusa la presenza di uno **zero assoluto**

e la possibilità di confrontare i punteggi con rapporti (ad esempio, "il tempo di reazione del partecipante A è il doppio di quello del partecipante B").

2. Quali sono le **trasformazioni ammissibili** su una scala di rapporti? Fornisci un esempio numerico (per esempio, se moltiplichi tutti i tempi di reazione per 2, che cosa accade al rapporto tra i punteggi di due partecipanti?).

3. Descrivi il motivo per cui è possibile dire che **A ha una latenza doppia di B** usando i millisecondi, ma non è sempre possibile fare asserzioni analoghe usando scale ad intervalli. Fai un parallelo, ad esempio, con le temperature in Celsius.

Esercizio 5 – Riconoscere e Applicare le Trasformazioni nei Quattro Livelli di Scala
Situazione
Un docente di Psicologia sperimentale ha raccolto quattro serie di dati su vari aspetti:

1. **Orientamento politico** (liberale, conservatore, centrista, ecc.).

2. **Classifica di soddisfazione** sul tirocinio (1° posto, 2° posto, 3° posto, etc.).

3. **Punteggi di un test di personalità** su un fattore (con media = 100, deviazione standard = 10) trattato come scala ad intervalli.

4. **Frequenza cardiaca a riposo** misurata in battiti al minuto (bpm).

Istruzioni

1. **Identifica** per ciascuno dei quattro insiemi di dati il **livello di scala** (nominale, ordinale, intervalli, rapporti).

2. Per ognuno dei quattro livelli di scala elenca **almeno una trasformazione ammessa** (ad es. ridenominazione delle categorie per la nominale, traslazione e dilatazione per l'intervalli, ecc.) e una **non ammessa** (esempio: non puoi sommare categorie nominali, non puoi calcolare la radice quadrata di un rango ordinale dandogli significato, ecc.).

3. Rifletti in breve (2-3 righe) su come queste differenze nelle trasformazioni ammissibili incidano sull'**interpretazione** dei dati e sulle **analisi statistiche** che il docente potrà validamente utilizzare (ad esempio, test non parametrici per variabili ordinarie, test parametrici per scale ad intervalli/rapporti).

Soluzioni 3

Esercizio 1 – Trasformazioni in Scala Nominale

1. Identificazione della scala La classificazione degli studenti in "Gruppo A/B/C" è **scala nominale**. Non esiste alcun ordine intrinseco tra le categorie; si tratta semplicemente di etichette qualitative.
2. Trasformazioni ammissibili

- **Trasformazioni ammissibili**: ridenominare o rinominare le categorie senza modificare la partizione del campione (esempio: A → "Studio", B → "Ricreazione", C → "Volontariato").
  - L'unica operazione aritmetica consentita è il **conteggio** delle frequenze nelle varie categorie.

- **Operazioni non consentite**: non è possibile sommare o sottrarre etichette, né confrontare categorie in termini di "più/meno grande" o "rapporto".

3. Esempio di nuova scala nominale equivalente

- Potresti chiamare i gruppi: **"Alpha, Beta, Gamma"** (oppure con colori: "Rosso, Blu, Verde").

- Questa trasformazione non altera la classificazione in sé: tutti gli studenti del Gruppo A rimangono nel "nuovo" gruppo Alpha, e così via.

- Non cambia la struttura dei dati e di conseguenza non altera i risultati della ricerca (restano invariate le frequenze e la suddivisione nelle categorie).

Esercizio 2 – Trasformazioni in Scala Ordinale

1. Perché è una scala ordinale? La variabile "livello di motivazione" da 1 (bassa) a 5 (alta) indica:

- **Classificazione in categorie** (come in una scala nominale).

- **Relazione d'ordine** chiara ($1 < 2 < 3 < 4 < 5$).
- Non fornisce alcuna informazione sulle **distanze reali** tra i punti (non è detto che la differenza tra 1 e 2 sia uguale a quella tra 3 e 4).
  È quindi una **scala ordinale** e non semplicemente nominale.

2. Ridenominazione dei valori mantenendo l'ordine

- Puoi sostituire i numeri con etichette testuali rispettando lo stesso ordine:
  1 → "Molto bassa"
  2 → "Bassa"
  3 → "Media"
  4 → "Alta"
  5 → "Molto alta"
- L'ordine rimane lo stesso: "Molto bassa" < "Bassa" < … < "Molto alta".

3. Esempio di trasformazione non ammissibile

- **Calcolare "il doppio di motivazione"**: dire che la categoria 4 è "il doppio" della categoria 2 non ha senso, perché non c'è un'unità di misura fissa che quantifichi la differenza tra i livelli. Le categorie ordinali servono solo a ordinare, non a quantificare in modo assoluto.

Esercizio 3 – Trasformazioni in Scala ad Intervalli

1. Trasformazione lineare ammessa

- **Forma generale**: $y' = a + by$, con $b > 0$.
- Preserva le **differenze** tra i valori (ad esempio, $(y_2 - y_1) = (y'_2 - y'_1)/b$), perché la traslazione aggiunge una costante a tutti i punteggi e la dilatazione (moltiplicazione per b) mantiene le proporzioni fra gli intervalli.

2. Esempio concreto

- Scala QI: media = 100, deviazione standard = 15.
- Vuoi che la **nuova media** sia 50.
  – Per semplificare, supponiamo di voler "spostare" ogni valore verso una nuova scala centrata a 50, mantenendo una deviazione standard proporzionale.
  – Una possibile trasformazione lineare:

$$y' = (y - 100) + 50 = y - 50.$$

- In questo caso, $a = -50$, $b = 1$.
- Se un individuo ha QI $= 115$, allora $y' = 115 - 50 = 65$.

- Se invece volessi anche cambiare la deviazione standard, potresti usare un fattore $b \neq 1$. Ad esempio, se desideri una deviazione standard $= 10$, potresti usare $b = \frac{10}{15} \approx 0.67$.

3. Differenze rispetto alle scale nominali/ordinali

- Con una **scala ad intervalli** puoi:
  - Ordinare i punteggi.
  - Stabilire **differenze** (es. un individuo A ha 15 punti in più di B).

- Non puoi invece stabilire **rapporti** (es. "A ha il doppio di X rispetto a B" non è lecito), perché lo zero è arbitrario e la distanza "0" non rappresenta l'assenza del fenomeno (come invece avviene nella scala di rapporti).

Esercizio 4 – Trasformazioni in Scala di Rapporti

1. Perché il tempo di reazione è in una scala di rapporti?

- **Zero assoluto**: un tempo di reazione (teoricamente) pari a 0 ms significherebbe nessun tempo trascorso → totale assenza del fenomeno misurato (impossibile nella pratica, ma concettualmente definisce uno zero non arbitrario).

- Puoi confrontare i punteggi con rapporti: "il tempo di reazione di A è il doppio di quello di B" (200 ms vs. 100 ms).

2. Trasformazioni ammissibili

- **Trasformazione di similarità**: $y' = by$ con $b > 0$.

- Se hai due tempi di reazione y_1 e y_2, il rapporto $\frac{y_1}{y_2}$ rimane invariato anche dopo la trasformazione:

$$\frac{y_1'}{y_2'} = \frac{by_1}{by_2} = \frac{y_1}{y_2}.$$

- Esempio numerico: se i tempi di reazione di due partecipanti sono 100 ms e 200 ms, il rapporto è 2. Se moltiplichi entrambi per 2, ottieni 200 ms e 400 ms, e il rapporto rimane 2.

3. Confronto con scala ad intervalli (esempio delle temperature)

- In una scala di rapporti puoi dire "A ha una latenza doppia di B" perché lo zero non è arbitrario.

- Con la temperatura (scala ad intervalli) lo zero (es. 0°C) non rappresenta l'assenza di calore, quindi non ha senso dire che 80°C è "il doppio" di 40°C. Cambiando la scala (ad es. Fahrenheit) il rapporto cambia.

Esercizio 5 – Riconoscere e Applicare le Trasformazioni nei Quattro Livelli di Scala

1. Identificazione del livello di scala

- **Orientamento politico**: scala **nominale** (categorie qualitative prive di ordine).

- **Classifica di soddisfazione** (1°, 2°, 3°, …): scala **ordinale** (c'è un ordine, ma non si conosce la "distanza" fra i posti).

- **Punteggi di un test di personalità** (con media=100, dev.st=10), considerati approssimazione di una scala **ad intervalli** (si assumono le differenze significative, lo zero è arbitrario).

- **Frequenza cardiaca a riposo** (bpm): scala **di rapporti** (zero assoluto e rapporti confrontabili).

2. Trasformazioni ammesse e non ammesse

- **Nominale**:
  - **Ammessa**: cambiare etichette (A → "Liberale", B → "Conservatore" ecc.).
  - **Non ammessa**: sommare categorie, ordinare, calcolare media delle categorie.

- **Ordinale**:
  - **Ammessa**: rietichettare i ranghi (1° → "Migliore", 2° → "Secondo posto"…).
  - **Non ammessa**: calcolare rapporti (il 2° posto non è "il doppio" del 1°), sommare posizioni in modo significativo.

- **A intervalli**:
  - **Ammessa**: trasformazione lineare (traslazione + dilatazione).

  - **Non ammessa**: dire che un punteggio è "tre volte" un altro; lo zero è arbitrario.

- **A rapporti**:
  - **Ammessa**: trasformazione di similarità ($y' = by$), in cui i rapporti rimangono invariati.
  - **Non ammessa**: aggiunta di una costante a tutti i valori (questa sposterebbe lo zero, rendendolo arbitrario e trasformando la scala in una scala ad intervalli).

3. Implicazioni per l'interpretazione e le analisi

- Una **variabile nominale** consente solo **frequenze** e test non parametrici basati su conteggi (es. Chi-quadrato).
- Una **variabile ordinale** permette test di ordinamento (es. test di rank, come il Wilcoxon), ma non calcoli di media con significato forte.
- Una **scala ad intervalli** permette di usare **statistiche parametriche** (calcolo di media, varianza, test come t-test, ANOVA), assumendo che l'interpretazione delle differenze sia coerente.
- Una **scala di rapporti** permette, in più, il confronto di **rapporti** (ad esempio, si possono applicare modelli parametrici che includano il concetto di proporzioni o slope logico su dati che abbiano senso a zero assoluto).

Bibliografia

4

La riforma metodologica in psicologia: dalla crisi alla rivoluzione bayesiana

! In questo capitolo imparerai a

- comprendere il contesto storico e concettuale della trasformazione metodologica in psicologia;
- identificare le principali criticità metodologiche che hanno contribuito alla crisi di replicabilità;
- descrivere l'approccio bayesiano come paradigma statistico alternativo al frequentismo.

Prerequisiti

- Leggere *Statistical Rethinking* (**?**). Focalizzati sui primi capitoli dove si discute della dicotomia tra "small world" e "big world".
- Leggere What has happened down here is the winds have changed (Gelman 2016). Un post sul blog di Andrew Gelman che fornisce una panoramica sulla crisi di replicazione e su come le scienze sociali sono cambiate di conseguenza.
- Leggere Productive Explanation: A Framework for Evaluating Explanations in Psychological Science. L'adozione di teorie formali è essenziale per affrontare la crisi di riproducibilità dei risultati nella ricerca psicologica.
- Per chi fosse interessato a un romanzo su questi temi, sorprendentemente avvincente, consiglio *Quando abbiamo smesso di capire il mondo* di Benjamín Labatut (**?**).

Introduzione

Negli ultimi vent'anni, le scienze sociali e la psicologia hanno vissuto una profonda trasformazione metodologica ed epistemologica. Questo cambiamento,

spesso definito come "Credibility Revolution" (**?**), "Causal Revolution" (**?**) e "Replication Crisis" (**?**; **?**), ha portato a un ripensamento delle pratiche di ricerca, specialmente in psicologia (**?**). Questa transizione verso quella che Munger (**?**) chiama "Science versione 2" è stata motivata dalla consapevolezza di lacune metodologiche passate e ha spinto verso l'adozione di approcci più rigorosi e replicabili.

Le origini di questa riforma risiedono nel riconoscimento di problemi metodologici diffusi, come la proliferazione di falsi positivi (**?**), l'abuso dei "gradi di libertà dei ricercatori" (**?**), e l'inadeguatezza delle pratiche statistiche tradizionali (**?**). Fenomeni come il p-hacking, l'uso di campioni di piccole dimensioni (**?**), e la mancanza di trasparenza nei metodi di ricerca hanno contribuito a minare la credibilità delle scoperte psicologiche (**?**; **?**), portando alla cosiddetta "Replication Crisis" (**?**; **?**) — si veda il Capitolo **??**.

4.1 L'Approccio Bayesiano

In risposta a queste sfide, l'approccio bayesiano è emerso come un paradigma statistico chiave nella "Credibility Revolution". A differenza dell'inferenza frequentista, che si basa sul Test dell'Ipotesi Nulla, la statistica bayesiana offre un framework più flessibile e intuitivo per l'analisi dei dati e l'inferenza causale. Il principio fondamentale dell'approccio bayesiano è l'aggiornamento delle distribuzioni di probabilità a priori (priors) alla luce di nuove evidenze, un processo che si allinea con l'obiettivo di una scienza cumulativa e autocorrettiva.

L'adozione di metodi bayesiani in psicologia presenta numerosi vantaggi:

1. **Quantificazione dell'incertezza**: L'inferenza bayesiana fornisce distribuzioni di probabilità posteriori complete per i parametri di interesse, offrendo una rappresentazione più ricca e sfumata dell'incertezza rispetto agli intervalli di confidenza frequentisti.
2. **Incorporazione di conoscenze pregresse**: Le priors bayesiane permettono di integrare formalmente conoscenze precedenti nel processo inferenziale, promuovendo un approccio cumulativo alla ricerca.
3. **Robustezza alle pratiche di ricerca discutibili**: I metodi bayesiani sono meno suscettibili a pratiche come il p-hacking, poiché l'inferenza si basa sull'intera distribuzione posteriore piuttosto che su soglie arbitrarie di significatività.

4.2 L'Approccio Bayesiano nella Ricerca Psicologica

L'uso delle statistiche bayesiane nella ricerca psicologica offre vantaggi significativi rispetto ai metodi statistici tradizionali, come il test di significatività dell'ipotesi nulla. Uno dei punti di forza principali è la sua indipendenza dalla teoria dei grandi campioni, rendendolo particolarmente adatto per studi che spesso si basano su campioni di dimensioni ridotte (**?**).

La ricerca psicologica è spesso caratterizzata da campioni limitati, dovuti a fattori come la bassa prevalenza di determinate condizioni, difficoltà nel reclutamento dei partecipanti e complessità nelle procedure di valutazione. Questi campioni di piccole dimensioni sono intrinsecamente soggetti a una maggiore eterogeneità, che si manifesta nella variabilità del fenotipo comportamentale delle condizioni psicologiche esaminate e nella discrepanza tra le stime degli effetti in diversi studi. Tale eterogeneità può portare a stime degli effetti distorte e scarsamente riproducibili.

L'approccio bayesiano offre una soluzione efficace a queste problematiche. In primo luogo, consente di valutare l'adeguatezza della dimensione del campione attraverso un'analisi della sensibilità dei risultati rispetto alla specificazione delle distribuzioni a priori. In secondo luogo, permette di ottenere risultati precisi anche con campioni ridotti, a condizione che le conoscenze a priori siano accurate e ben definite.

Un ulteriore vantaggio è la capacità dell'approccio bayesiano di ottimizzare l'uso dei campioni di partecipanti, favorendo un'inclusione equa delle popolazioni diversificate. Questo è particolarmente rilevante per gruppi spesso sottorappresentati, come le minoranze etniche. Le statistiche bayesiane aiutano a superare questa sfida evitando di esercitare una pressione eccessiva su questi gruppi per aumentarne la partecipazione, permettendo così una ricerca più equa e rappresentativa.

4.3 Modellazione Formale e Data Science

La "Credibility Revolution" ha favorito l'integrazione della Data Science nelle pratiche di ricerca psicologica. L'adozione di pipeline di analisi dei dati riproducibili, l'uso di controllo di versione e la condivisione di dati e codice sono diventati standard de facto nella comunità scientifica. Questi strumenti non solo migliorano la trasparenza e la replicabilità della ricerca, ma facilitano anche la collaborazione e l'accumulo di conoscenze nel campo.

Parallelamente, si è osservato un rinnovato interesse per la modellazione for-

male in psicologia, che consente non solo la verifica ma anche lo sviluppo di modelli dei meccanismi sottostanti ai fenomeni psicologici (**?**; **?**). Questo approccio supera la mera descrizione delle associazioni tra variabili, tipica della pratica dominante dell'ANOVA nel contesto pre-riforma.

La modellazione bayesiana si presta particolarmente bene a questo approccio, offrendo un framework unificato per la specificazione di modelli formali, l'incorporazione di incertezza parametrica e la valutazione dell'evidenza empirica. Attraverso tecniche come il confronto tra modelli bayesiano e l'analisi di sensibilità, i ricercatori possono valutare rigorosamente la plausibilità relativa di diverse teorie psicologiche.

4.4 Riflessioni Epistemologiche

L'adozione di metodi bayesiani e della Data Science in psicologia deve essere accompagnata da una profonda riflessione epistemologica. Come sottolineato da George Box:

> Tutti i modelli sono sbagliati, ma alcuni sono utili.

Questa massima risuona particolarmente nel contesto della ricerca psicologica, dove i fenomeni di interesse sono spesso complessi e multifattoriali.

L'approccio bayesiano, con la sua enfasi sull'aggiornamento iterativo delle credenze alla luce di nuove evidenze, si allinea naturalmente con una visione della scienza come processo di apprendimento continuo piuttosto che come ricerca di verità assolute. Questa prospettiva riconosce i limiti intrinseci dei nostri modelli e delle nostre teorie, pur valorizzandone l'utilità euristica e predittiva.

In particolare, McElreath (**?**) sottolinea l'importanza di riconoscere la dualità tra il "mondo del modello" e il mondo reale più ampio che cerchiamo di comprendere. Questa consapevolezza è cruciale per evitare la reificazione dei nostri modelli statistici e per mantenere una prospettiva critica sulle nostre inferenze.

4.5 Riflessioni Conclusive

L'integrazione dell'approccio bayesiano e della Data Science nella ricerca psicologica rappresenta una risposta promettente alle sfide poste dalla "Replication Crisis". Offrendo un framework coerente per la modellazione formale, l'inferenza statistica e l'incorporazione di conoscenze pregresse, questi approc-

ci promettono di elevare il rigore e la credibilità della ricerca psicologica. Tuttavia, è fondamentale che l'adozione di questi metodi sia accompagnata da una adeguata consapevolezza metodologica ed epistemologica.

Esercizi

! Problemi

1. Quali sono i principali fattori che hanno portato alla "Credibility Revolution" in psicologia e in che modo le nuove metodologie — in particolare l'approccio bayesiano e le buone pratiche di Data Science — mirano a superare i limiti che hanno contribuito alla "Replication Crisis"?
2. In che modo il paradigma bayesiano differisce dall'approccio frequentista nella gestione dell'incertezza e nell'aggiornamento della conoscenza, e quali vantaggi pratici offre quando si lavora con campioni di piccole dimensioni e popolazioni eterogenee?
3. Qual è il ruolo delle distribuzioni a priori nelle analisi bayesiane e come ci si assicura che l'uso di priors non introduca bias indesiderati, specialmente in un contesto come quello psicologico dove le teorie e i dati pregressi possono essere incompleti o controversi?
4. Perché la modellazione formale e le buone pratiche di Data Science (ad es. condivisione di codice, controllo di versione, pipeline riproducibili) risultano fondamentali per una scienza cumulativa e per mitigare gli errori sistematici nella ricerca psicologica?
5. Dal punto di vista epistemologico, in che modo il principio "tutti i modelli sono sbagliati, ma alcuni sono utili" influenza la costruzione, la valutazione e l'interpretazione dei modelli in psicologia, e come si integra con la prospettiva bayesiana di scienza come processo iterativo di apprendimento?

Soluzioni

1. Quali sono i principali fattori che hanno portato alla "Credibility Revolution" in psicologia e in che modo le nuove metodologie — in particolare l'approccio bayesiano e le buone pra-

tiche di Data Science — mirano a superare i limiti che hanno contribuito alla "Replication Crisis"?
Negli ultimi decenni, la psicologia ha attraversato una "Replication Crisis" a causa di diverse pratiche di ricerca problematiche, tra cui l'utilizzo di campioni di piccole dimensioni, l'uso eccessivo di test di significatività frequentisti con $p < .05$ come soglia rigida, il fenomeno del *p-hacking* (cioè l'adattamento delle analisi per ottenere risultati "significativi"), e la carenza di trasparenza e condivisione dei dati. Questi fattori hanno portato alla pubblicazione di molti falsi positivi e a un'erosione della fiducia nelle conclusioni psicologiche.
La "Credibility Revolution" nasce dalla presa di coscienza di questi problemi e dall'introduzione di nuove metodologie che offrono maggior rigore e trasparenza. L'approccio bayesiano, in particolare, consente di superare alcuni limiti della statistica frequentista, poiché fornisce distribuzioni posteriori di plausibilità per i parametri e non si affida a soglie arbitrarie di significatività. Inoltre, la Data Science ha promosso la diffusione di *pipeline* analitiche riproducibili (tramite il controllo di versione, la condivisione del codice e dei dati), contribuendo a ridurre errori, bias e a favorire la replicabilità. Insieme, queste innovazioni mirano a creare una scienza più aperta, solida e cumulativa.
2. In che modo il paradigma bayesiano differisce dall'approccio frequentista nella gestione dell'incertezza e nell'aggiornamento della conoscenza, e quali vantaggi pratici offre quando si lavora con campioni di piccole dimensioni e popolazioni eterogenee?
Il paradigma frequentista si basa sull'idea di ripetizione ipotetica degli esperimenti e sull'applicazione di test di significatività, focalizzandosi su p-value e intervalli di confidenza che rispondono a domande "se si ripetesse infinite volte l'esperimento, in media cosa accadrebbe?". L'inferenza bayesiana, al contrario, concepisce la probabilità come uno stato di conoscenza (o di credenza) e integra le informazioni precedenti (priors) con i dati osservati per produrre una distribuzione posteriore. Ciò consente un aggiornamento continuo e iterativo delle ipotesi alla luce delle nuove evidenze.
Questa impostazione risulta particolarmente utile quando i campioni sono ridotti e le popolazioni indagate sono eterogenee. In tali condizioni, il paradigma frequentista rischia di generare stime instabili o intervalli di confidenza molto ampi. L'approccio bayesiano, invece, permette di incorporare informazioni pregresse e di ottenere stime più precise, a patto che le priors siano giustificate e non eccessivamente informative. L'attenzione alla distribuzione posteriore rende inoltre più chiaro il grado di incertezza associato ai parametri di interesse e favorisce la formulazione di inferenze più calibrate.
3. Qual è il ruolo delle distribuzioni a priori nelle analisi bayesiane e come ci si assicura che l'uso di priors non introduca bias

indesiderati, specialmente in un contesto come quello psicologico dove le teorie e i dati pregressi possono essere incompleti o controversi?
Nell'approccio bayesiano, le distribuzioni a priori (priors) rappresentano la conoscenza o le ipotesi iniziali di cui si dispone sui parametri in esame prima di raccogliere i dati. Se ben specificate, contribuiscono a rendere l'analisi più informativa, soprattutto quando il campione è di piccole dimensioni. Tuttavia, un uso improprio delle priors può introdurre bias, poiché priors troppo "forti" (ossia eccessivamente vincolanti) possono spingere i risultati verso determinate conclusioni.
Per evitare questi rischi, i ricercatori devono adottare *priors* ben motivate e trasparenti. In psicologia, dove le teorie possono essere ancora in fase di sviluppo e i dati pregressi non sempre affidabili, è spesso utile iniziare con priors non informative o debolmente informative, per ridurre il rischio di forzare troppo il modello. È anche fondamentale effettuare **analisi di sensibilità**: testare differenti specificazioni di *priors* per verificare se i risultati sono robusti oppure fortemente dipendenti da particolari assunzioni iniziali. La documentazione delle scelte fatte (e le relative ragioni) è parte integrante di una buona pratica di ricerca trasparente.
4. Perché la modellazione formale e le buone pratiche di Data Science (ad es. condivisione di codice, controllo di versione, pipeline riproducibili) risultano fondamentali per una scienza cumulativa e per mitigare gli errori sistematici nella ricerca psicologica?
La modellazione formale consente di superare la mera descrizione delle relazioni tra variabili (tipica dell'ANOVA o dei modelli lineari tradizionali) e di entrare nel merito dei meccanismi sottostanti i fenomeni psicologici, costruendo teorie più articolate e fondate. Questa prospettiva rende più esplicite le assunzioni e le ipotesi su cui si basa la ricerca, permettendo un confronto chiaro tra diverse spiegazioni concorrenti.
Parallelamente, l'adozione di strumenti e pratiche di Data Science come la condivisione di codice (in repository pubblici), l'uso del controllo di versione (es. Git) e pipeline analitiche riproducibili riducono gli errori e favoriscono la verifica indipendente dei risultati. Ciò aumenta la trasparenza, poiché altri ricercatori possono ispezionare i passaggi compiuti, e consente la replicazione degli studi. In una scienza cumulativa, infatti, la possibilità di riprodurre, criticare e migliorare i risultati di lavori precedenti è essenziale per costruire un corpus di conoscenze solido e affidabile.
5. Dal punto di vista epistemologico, in che modo il principio "tutti i modelli sono sbagliati, ma alcuni sono utili" influenza la costruzione, la valutazione e l'interpretazione dei modelli in

psicologia, e come si integra con la prospettiva bayesiana di scienza come processo iterativo di apprendimento?
La celebre frase di George Box ("tutti i modelli sono sbagliati, ma alcuni sono utili") evidenzia come i modelli statistici e teorici non possano mai rappresentare perfettamente la realtà, specialmente in un campo complesso come la psicologia, in cui le variabili spesso interagiscono in modo non lineare e multi-dimensionale. Ciò non significa che i modelli siano inutili; anzi, se ben costruiti, possono offrire uno strumento potente per interpretare e prevedere i fenomeni.
Nell'ottica bayesiana, la costruzione dei modelli è un processo iterativo in cui le ipotesi iniziali (priors) vengono continuamente aggiornate alla luce di nuove evidenze (la distribuzione posteriore). Questo ciclo di apprendimento riflette un'idea di scienza non come ricerca della "verità assoluta", ma come progressivo affinamento delle teorie. Il principio di Box ricorda ai ricercatori che qualsiasi modello deve essere costantemente messo alla prova, confrontato con altri modelli, e aggiornato o abbandonato se i dati non lo supportano più. In questo senso, la prospettiva bayesiana favorisce una mentalità flessibile e aperta, pronta a rivedere i propri presupposti e a migliorare progressivamente la comprensione dei fenomeni psicologici.

Bibliografia

Parte III

R

La programmazione richiede un approccio strutturato e logico per risolvere problemi, articolato in due livelli interconnessi: **algoritmico** e **sintattico**.

Livello Algoritmico

Si inizia con l'analisi del problema e la descrizione astratta della soluzione, indipendente dal linguaggio di programmazione. Ad esempio, per calcolare la media di un insieme di valori:

- **Input**: Valori numerici.
- **Output**: Media aritmetica.
- **Algoritmo**: Somma dei valori divisa per il numero di osservazioni:

$$\text{media} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Strutture dati come i vettori sono ideali per memorizzare e accedere ai valori in modo ordinato.

Livello Sintattico

Si traduce l'algoritmo in un linguaggio specifico. Ad esempio:

- In **R**:

```r
media <- sum(x) / length(x)
```

- In **Python**:

```python
media = sum(x) / len(x)
```

La logica rimane invariata, ma la sintassi dipende dal linguaggio scelto.

In conclusione, la programmazione richiede due competenze chiave:

1. **Pensiero algoritmico**: Capacità di astrarre e definire la logica computazionale.
2. **Conoscenza sintattica**: Traduzione della logica in codice eseguibile.

Il livello algoritmico è fondamentale, poiché la logica è universale, mentre la sintassi può essere appresa o automatizzata. Questo approccio è particolarmente rilevante nell'era dell'AI, dove la chiarezza nella descrizione algoritmica è essenziale per formulare dei prompt efficaci (**?**).

Questa distinzione riflette il framework di Marr (**?**) sulla visione artificiale, che separa:

1. **Livello computazionale**: Cosa e perché.
2. **Livello algoritmico**: Come.
3. **Livello implementativo**: Realizzazione pratica.

Nella programmazione, il livello algoritmico rappresenta il fondamento per risolvere problemi in modo efficace, a prescindere dal linguaggio di programmazione scelto. In questo insegnamento, l'attenzione sarà focalizzata principalmente sul **livello algoritmico**, piuttosto che su quello sintattico.

R: Uno Strumento per l'Analisi dei Dati

Per trovare la soluzione concreta a un problema di analisi dei dati, è necessario implementare l'algoritmo desiderato in un linguaggio di programmazione. In questo insegnamento, utilizzeremo **R**, uno dei linguaggi più utilizzati per l'analisi dei dati, apprezzato per la sua flessibilità, potenza e il supporto offerto da una vasta comunità di utenti e sviluppatori.

Perché R?

- **Nato per l'analisi statistica**: R è stato concepito specificamente per rispondere alle esigenze di analisi statistica e visualizzazione grafica, diventando rapidamente uno strumento essenziale nel panorama accademico e scientifico.
- **Gestione dei dati**: R offre strumenti avanzati per gestire, manipolare e analizzare grandi quantità di dati, coprendo un'ampia gamma di tecniche statistiche, dalla modellazione lineare all'analisi delle serie temporali.
- **Visualizzazione grafica**: Con pacchetti come `ggplot2` e `plotly`, R permette di creare grafici e visualizzazioni di alta qualità, fondamentali per comunicare risultati in modo efficace.
- **Comunità e pacchetti**: L'ecosistema di R è arricchito da una vasta libreria di pacchetti, che estendono le capacità del linguaggio per soddisfare necessità specifiche e settoriali.

R in Psicologia e Scienze Sociali

In psicologia e nelle scienze sociali, R è particolarmente utile grazie alle sue capacità avanzate di analisi statistica e visualizzazione. Permette di affrontare analisi sofisticate, come modelli di regressione, analisi fattoriale e metodi per dati longitudinali, rendendolo uno strumento indispensabile per la ricerca.

In conclusione, imparare R non significa solo acquisire competenze tecniche, ma anche aprire le porte a nuove possibilità di analisi e ricerca. Tuttavia, è fondamentale ricordare che la vera sfida nella programmazione non è padroneggiare la sintassi di un linguaggio specifico, ma comprendere la **logica algoritmica** che sta alla base della soluzione di un problema. L'intelligenza artificiale può aiutarci a trovare la sintassi corretta, ma spetta a noi decidere quale algoritmo implementare. Pertanto, i nostri sforzi devono essere rivolti a capire la logica del problema, piuttosto che concentrarci esclusivamente sull'implementazione sintattica.

Bibliografia

5

Data Science e R: un approccio moderno all'analisi dei dati

! In questo capitolo imparerai a:

- installare R e RStudio;
- creare e gestire progetti in RStudio;
- manipolare oggetti e vettori in R;
- utilizzare funzioni e lavorare con dati mancanti;
- estrarre e gestire sottoinsiemi di dati;
- apprezzare l'importanza di rendere riproducibile l'analisi dei dati, condividendone ogni passaggio.

Prerequisiti

- Studiare il Capitolo **??** dell'Appendice.
- Leggere *Analysis of open data and computational reproducibility in registered reports in psychology* (**?**).
- Leggere il capitolo *Getting Started with Data in R* di Statistical Inference via Data Science: A ModernDive into R and the Tidyverse (Second Edition), capitoli 1.1-1.3.
- Consultare Introduction to Data Science: Data Wrangling and Visualization with R (**?**)
- Consultare R for Data Science (2e) (**?**).
- Consultare R Programming for Data Science, capitoli 3-4.

Preparazione del Notebook

```
here::here("code", "_common.R") |>
  source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(tidyr)
```

5.1 Introduzione

Negli ultimi anni, la data science è emersa come uno dei domini più rilevanti nel panorama tecnologico e scientifico. Questa crescita è guidata dalla necessità sempre maggiore di analizzare grandi quantità di dati per supportare processi decisionali più efficaci. In questo contesto, R si è affermato come uno degli strumenti più potenti e versatili, particolarmente apprezzato da statistici e data scientist.

R è un linguaggio di programmazione specializzato nel calcolo statistico, che eccelle nell'analisi dei dati e nel data mining. La sua piattaforma offre capacità straordinarie per l'analisi, l'elaborazione e la visualizzazione dei dati, rendendolo uno strumento fondamentale nel panorama della data science moderna.

5.2 Trasparenza e Riproducibilità

La scelta di R come strumento per la data science assume particolare rilevanza nel contesto della "crisi della replicabilità" in psicologia, un problema che ha evidenziato come molte analisi dei dati non siano facilmente riproducibili (**?**). R affronta questa sfida offrendo tre vantaggi fondamentali:

1. **Documentazione Trasparente attraverso lo Scripting**
   - Ogni analisi viene documentata attraverso script espliciti
   - Il codice può essere salvato, condiviso e rivisto
   - Le analisi possono essere facilmente replicate con nuovi dati
2. **Flessibilità e Personalizzazione**
   - Ambiente adattabile a diverse esigenze analitiche
   - Vasta collezione di pacchetti per metodi statistici avanzati
   - Possibilità di sviluppare soluzioni personalizzate
3. **Reporting Integrato**
   - Integrazione con strumenti come R Markdown e Quarto
   - Creazione di documenti dinamici che combinano codice e risultati
   - Generazione automatica di report aggiornabili

A differenza dei software con interfacce grafiche, l'approccio basato sul codice di R elimina le ambiguità e aumenta la verificabilità delle analisi. Questo non solo migliora la qualità della ricerca, ma contribuisce anche a costruire una comunità scientifica più trasparente e affidabile.

L'utilizzo di R rappresenta quindi non solo una scelta tecnica, ma l'adozione di una metodologia che promuove rigore scientifico, trasparenza e riproducibilità nella ricerca moderna.

5.3 Installare R e RStudio

1. **Scarica e installa R**
   Vai al sito ufficiale di CRAN (https://cran.r-project.org/), scegli la versione per il tuo sistema operativo (Windows, Mac o Linux) e segui le istruzioni di installazione.
2. **Scarica e installa RStudio**
   Dopo aver installato R, scarica RStudio dal sito ufficiale (https://posit.co/download/rstudio-desktop/). Scegli la versione gratuita "RStudio Desktop" e segui le istruzioni per il tuo sistema operativo.

Una spiegazione dettagliata del processo di installazione di R e RStudio è disponibile in Okoye & Hosseini (**?**).

5.4 Panoramica sull'interfaccia di RStudio

RStudio rende l'uso di R più intuitivo grazie alla sua interfaccia divisa in quattro pannelli principali:

- **Pannello degli script**: Qui puoi scrivere e modificare i tuoi script, cioè sequenze di comandi salvabili per analisi ripetibili e organizzate.
- **Console**: Esegue i comandi scritti direttamente o lanciati dagli script, mostrando risultati, messaggi e errori.
- **Pannello dell'ambiente**: Mostra i dataset, le variabili e gli oggetti caricati nella sessione di lavoro, permettendoti di gestire facilmente i dati.
- **Pannello grafici/aiuto/file**: Visualizza grafici, fornisce accesso alla documentazione di R e consente di navigare tra file e cartelle sul tuo sistema.

5.5 Creare un Nuovo Progetto in RStudio

Avviare un nuovo progetto
Dal menu di RStudio, seleziona **File > New Project...** per creare un nuovo progetto. I progetti in RStudio sono uno strumento efficace per organizzare il lavoro relativo a una specifica analisi o domanda di ricerca. All'interno di un progetto puoi raccogliere script, file di dati e output, mantenendo tutto ben strutturato.

Scegliere la posizione del progetto
Puoi creare una nuova directory dedicata al progetto oppure associare il progetto a una directory esistente. Organizzare i progetti in cartelle dedicate aiuta a mantenere i file in ordine e a utilizzare percorsi relativi, rendendo il tuo lavoro più facile da condividere con collaboratori e più portabile tra diversi sistemi.

Questa organizzazione è particolarmente utile per evitare confusione e assicurarsi che tutti i file necessari siano facilmente accessibili e collegati al progetto corretto.

5.6 Concetti di Base nella Programmazione in R

Iniziare a usare R, soprattutto per chi si avvicina per la prima volta a questo linguaggio nel contesto della psicologia, significa comprendere i concetti fondamentali che ne costituiscono la base. Questo capitolo introduce i principi essenziali della programmazione in R, tra cui:

- La comprensione della **sintassi di R**.
- La familiarizzazione con i principali **tipi di dati** e **strutture**.
- L'acquisizione delle **operazioni di base**.

Questi concetti sono fondamentali per manipolare efficacemente i dati e condurre analisi statistiche, rappresentando il punto di partenza per sfruttare al meglio le potenzialità di R.

5.7 Oggetti in R

In R, tutto è un oggetto: dai numeri o stringhe di testo semplici, fino a strutture più complesse come grafici, riassunti di analisi statistiche o script che

eseguono compiti specifici. Creare e assegnare valori agli oggetti è fondamentale per lavorare in R.

5.7.1 Creare oggetti

Per creare un oggetto, basta assegnargli un nome e un valore usando l'operatore di assegnazione `<-`:

```
my_obj <- 48
```

In questo esempio, abbiamo creato un oggetto chiamato `my_obj` e gli abbiamo assegnato il valore 48. Anche l'operatore `=` può essere usato, ma è considerato una cattiva pratica.

Per visualizzare il valore di un oggetto, basta scriverne il nome:

```
my_obj
#> [1] 48
```

Gli oggetti creati vengono memorizzati nell'ambiente di lavoro. In RStudio, puoi visualizzarli nella scheda **Environment** e ottenere dettagli come tipo, lunghezza e valore.

È possibile assegnare a un oggetto anche una stringa di testo, racchiudendola tra virgolette:

```
my_obj2 <- "R è fantastico"
my_obj2
#> [1] "R è fantastico"
```

Se dimentichi le virgolette, R mostrerà un errore.

Per modificare il valore di un oggetto esistente, basta riassegnarlo:

```
my_obj2 <- 1024
```

Ora il tipo di `my_obj2` è cambiato da carattere a numerico. È anche possibile usare oggetti per crearne di nuovi:

```
my_obj3 <- my_obj + my_obj2
my_obj3
#> [1] 1072
```

Se provi a sommare oggetti di tipo diverso, R restituirà un errore:

```
char_obj <- "ciao"
char_obj2 <- "mondo"
char_obj3 <- char_obj + char_obj2
#> Error in char_obj + char_obj2 : non-numeric argument to binary operator
```

Quando incontri errori come questo, chiedi a AI la spiegazione del messaggio,

per esempio: "non-numeric argument to binary operator error + r". Un errore comune è anche:

```
my_obj <- 48
my_obj4 <- my_obj + no_obj
#> Error: object 'no_obj' not found
```

R segnala che `no_obj` non è stato definito e, di conseguenza, l'oggetto `my_obj4` non è stato creato.

5.7.2 Nomi degli Oggetti

Attribuire nomi agli oggetti potrebbe sembrare un dettaglio secondario, ma è fondamentale scegliere nomi **brevi** e **informativi**. Un buon nome migliora la leggibilità del codice e ne facilita la manutenzione. È importante adottare uno stile coerente, come uno dei seguenti:

- **Snake case**: `output_summary`
- **Dot case**: `output.summary`
- **Camel case**: `outputSummary`

In questo corso useremo lo stile più diffuso, **Snake Case**, che separa le parole con il carattere di sottolineatura `_`.

Ci sono alcune regole fondamentali da rispettare nella scelta dei nomi:

1. Non possono iniziare con un numero (ad esempio, `2my_variable` non è valido).
2. Non possono contenere caratteri speciali come `&`, `^`, `/`, ecc.
3. Evita di usare parole riservate (ad esempio, `TRUE`, `NA`) o nomi di funzioni esistenti (ad esempio, `data`).

Esempio di cosa **non fare**:

```
data <- read.table("mydatafile", header = TRUE) # `data` è già una funzione!
```

5.7.3 Commenti

I commenti sono uno strumento essenziale per rendere il codice più chiaro e comprensibile, sia per te stesso sia per altri. Nel linguaggio R, i commenti iniziano con il simbolo `#`, e tutto ciò che lo segue sulla stessa riga viene **ignorato dall'interprete** durante l'esecuzione.

5.7.3.1 Perché commentare?

I commenti servono a spiegare **perché** il codice è scritto in un certo modo, non solo **come** funziona (questo è evidente leggendo il codice). Una buona pratica consiste nel commentare le decisioni o i passaggi che non risultano immediatamente evidenti.

Ad esempio, invece di scrivere un commento ridondante come:

```
# Assegno 42 alla variabile x
x <- 42
```

è più utile fornire un contesto:

```
# Valore iniziale scelto per semplificare i calcoli successivi
x <- 42
```

5.7.3.2 Vantaggi

Commentare in modo appropriato aiuta a:

- **Ridurre il tempo** necessario per comprendere o modificare il codice, anche mesi o anni dopo averlo scritto.
- Facilitare la collaborazione con altri, rendendo il codice leggibile e accessibile.
- Migliorare la **manutenibilità** e il riutilizzo del codice.

Un codice ben commentato non è solo più facile da leggere, ma anche più professionale e robusto nel lungo termine.

5.7.4 Usare R come Calcolatore

R può essere utilizzato come un semplice calcolatore digitando direttamente nella console numeri e operatori aritmetici per eseguire operazioni come somma, sottrazione, moltiplicazione e divisione (`+`, `-`, `*`, `/`). Questo lo rende uno strumento immediato e versatile per calcoli di base e avanzati.

Esempio 5.1. La *Satisfaction With Life Scale* (SWLS) contiene 5 item, ciascuno valutato con una scala Likert a 7 punti, dove:

1 = “completamente in disaccordo” e 7 = “completamente d’accordo”.

Gli item sono:

1. Per la maggior parte, la mia vita si avvicina al mio ideale.
2. Le mie condizioni di vita sono eccellenti.

3. Sono soddisfatto della mia vita.
4. Fino ad ora, ho ottenuto le cose importanti che voglio nella vita.
5. Se potessi vivere la mia vita di nuovo, non cambierei quasi nulla.

Supponiamo che un individuo risponda nel seguente modo:

- Item 1: 5
- Item 2: 3
- Item 3: 4
- Item 4: 2
- Item 5: 2

Il punteggio totale sulla SWLS si calcola sommando i punteggi di ciascun item:

```
sogg1 <- 5 + 3 + 4 + 2 + 2
sogg1
#> [1] 16
```

Esempio 5.2. Il **Body Mass Index (BMI)** si calcola dividendo il peso, in chilogrammi, per il quadrato dell'altezza, in metri.
La formula è:

$$\text{BMI} = \frac{\text{Peso (kg)}}{\text{Altezza (m)}^2}.$$

Supponiamo che un individuo pesi **79000 grammi** (79 kg) e sia alto **176 cm**.
Il calcolo in R sarà:

```
bmi <- (79000 / 1000) / (176 / 100)^2
bmi
#> [1] 25.5
```

Nota. L'uso di parentesi è fondamentale per garantire che le operazioni vengano eseguite nell'ordine corretto. In R, come in matematica, le operazioni racchiuse tra parentesi hanno la precedenza rispetto ad altre operazioni. Ad esempio, nel calcolo del BMI, abbiamo usato le parentesi per calcolare prima la conversione dei valori nell'unità di misura appropriata.

5.8 Ordine di precedenza degli operatori

Le operazioni algebriche vengono eseguite in una particolare sequenza in R, nota come **ordine di precedenza degli operatori**. Questo ordine determina quali operazioni vengono eseguite per prime quando un'espressione include più operatori. In assenza di parentesi, l'ordine di precedenza è il seguente (dal più alto al più basso):

1. **Parentesi**: Le operazioni racchiuse tra parentesi `()` vengono eseguite per prime. Questo permette di sovrascrivere l'ordine naturale delle operazioni.

```
result <- (2 + 3) * 4  # Risultato: 20
```

2. **Esponenziazione**: L'operatore `^` viene eseguito dopo le parentesi.

```
result <- 2^3  # Risultato: 8
```

3. **Segni unari**: Il segno meno `-` o più `+` applicato a un singolo valore.

```
result <- -3 + 5  # Risultato: 2
```

4. **Moltiplicazione, divisione e modulo**: Gli operatori `*`, `/`, `%/%` (divisione intera) e `%%` (resto) hanno la stessa precedenza e vengono eseguiti da sinistra a destra.

```
result <- 10 / 2 * 3  # Risultato: 15
result <- 10 %% 3     # Risultato: 1
```

5. **Addizione e sottrazione**: Gli operatori `+` e `-` vengono eseguiti dopo quelli di moltiplicazione/divisione.

```
result <- 5 + 3 - 2  # Risultato: 6
```

6. **Operatori di assegnazione**: Gli operatori `<-`, `->`, `=`, che assegnano valori a variabili, vengono valutati per ultimi.

```
x <- 2 + 3 * 4  # Risultato: 14
```

Note importanti:

- **Associazione a sinistra**: La maggior parte degli operatori in R viene valutata da sinistra a destra (ad esempio, `+`, `*`, `/`).

- **Uso delle parentesi**: Quando l'ordine di precedenza non è immediatamente chiaro o si vuole assicurare un ordine specifico, è sempre buona pratica usare le parentesi.

Capire l'ordine di precedenza è fondamentale per evitare errori logici e garantire che il codice funzioni come previsto.

5.9 Usare le funzioni in R

Fino ad ora abbiamo creato oggetti semplici assegnando loro direttamente un valore. Con l'aumento dell'esperienza in R, potresti voler creare oggetti più complessi. Per aiutarti, R offre numerose funzioni già disponibili nella sua installazione di base, e altre possono essere aggiunte installando pacchetti. Una funzione è un insieme di istruzioni che eseguono un compito specifico. Inoltre, è possibile creare funzioni personalizzate.

5.9.1 La funzione `c()` per creare vettori

La prima funzione utile da imparare è `c()`, che serve a concatenare valori in un *vettore*. Ad esempio:

```
my_vec <- c(2, 3, 1, 6, 4, 3, 3, 7)
```

Questo codice crea un oggetto chiamato `my_vec` che contiene una sequenza di numeri. Alcuni concetti fondamentali sulle funzioni in R:

1. **Nome e parentesi**: Le funzioni in R sono sempre seguite da parentesi tonde `()`.
2. **Argomenti**: Gli elementi passati alla funzione (tra le parentesi) ne personalizzano il comportamento e sono separati da virgole.

Per vedere il contenuto del vettore:

```
my_vec
#> [1] 2 3 1 6 4 3 3 7
```

5.9.2 Funzioni per analizzare vettori

Puoi utilizzare altre funzioni per calcolare statistiche sul vettore:

```
mean(my_vec)    # Media
#> [1] 3.625
```

```r
var(my_vec)     # Varianza
#> [1] 3.982
```

```r
sd(my_vec)      # Deviazione standard
#> [1] 1.996
```

```r
length(my_vec)  # Numero di elementi
#> [1] 8
```

Puoi anche salvare i risultati in nuovi oggetti per riutilizzarli:

```r
vec_mean <- mean(my_vec)
vec_mean
#> [1] 3.625
```

Concetto Chiave

La varianza e la deviazione standard sono misure statistiche descrittive che sintetizzano in un unico valore numerico la variabilità di un insieme di dati. Questi indici, che verranno approfonditi nel Capitolo **??**, forniscono informazioni su quanto i valori di un dataset siano simili o diversi tra loro.
In particolare:

- la varianza e la deviazione standard sono pari a 0 quando tutti i valori nel dataset sono identici, indicando assenza di variabilità;
- assumono valori più elevati all'aumentare delle differenze tra i dati, segnalando una maggiore dispersione.

Per i nostri scopi attuali, è sufficiente comprendere che queste misure descrivono il grado di diversità o omogeneità dei dati.

5.9.3 Creare sequenze regolari

Per creare sequenze di numeri in passi regolari, puoi usare i seguenti comandi.

Simbolo `:` per sequenze semplici:

```r
my_seq <- 1:10
my_seq
#>  [1]  1  2  3  4  5  6  7  8  9 10
```

Funzione `seq()` per maggiore controllo:

```r
my_seq2 <- seq(from = 1, to = 5, by = 0.5)
my_seq2
#> [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

5.9.4 Ripetere valori

Puoi ripetere valori o sequenze con la funzione `rep()`.

Ripetere un valore:

```
my_seq3 <- rep(2, times = 10)
my_seq3
#>  [1] 2 2 2 2 2 2 2 2 2 2
```

Ripetere una sequenza:

```
my_seq5 <- rep(1:5, times = 3)
```

Ripetere ogni elemento di una sequenza:

```
my_seq6 <- rep(1:5, each = 3)
my_seq6
#>  [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
```

5.9.5 Annidare funzioni

È possibile combinare funzioni per creare comandi più complessi, come nell'esempio:

```
my_seq7 <- rep(c(3, 1, 10, 7), each = 3)
my_seq7
#>  [1]  3  3  3  1  1  1 10 10 10  7  7  7
```

Per maggiore leggibilità, puoi separare i passaggi:

```
in_vec <- c(3, 1, 10, 7)
my_seq7 <- rep(in_vec, each = 3)
my_seq7
#>  [1]  3  3  3  1  1  1 10 10 10  7  7  7
```

Questa pratica facilita la comprensione del codice e lo rende più chiaro.

5.10 Lavorare con i vettori in R

In R, i vettori sono uno degli elementi fondamentali per manipolare, riassumere e ordinare i dati. Qui trovi una panoramica su come estrarre, sostituire, ordinare, lavorare con dati mancanti e sfruttare la vettorizzazione dei vettori.

5.10.1 Estrarre elementi da un vettore

Puoi estrarre uno o più elementi da un vettore usando le parentesi quadre `[ ]`.

Per posizione: Specifica la posizione degli elementi.

```
my_vec <- c(2, 3, 1, 6, 4, 3, 3, 7)
my_vec[3]  # Terzo elemento
#> [1] 1
```

```
my_vec[c(1, 5, 6)]  # Elementi 1°, 5° e 6°
#> [1] 2 4 3
```

```
my_vec[3:8]  # Da 3° a 8°
#> [1] 1 6 4 3 3 7
```

Con condizioni logiche: Usa espressioni logiche per selezionare elementi.

```
my_vec[my_vec > 4]  # Elementi > 4
#> [1] 6 7
```

```
my_vec[my_vec <= 4]  # Elementi  4
#> [1] 2 3 1 4 3 3
```

```
my_vec[my_vec != 4]  # Elementi diversi da 4
#> [1] 2 3 1 6 3 3 7
```

Operatori logici: Combina condizioni con `&` (AND) e `|` (OR).

```
my_vec[my_vec > 2 & my_vec < 6]  # Tra 2 e 6
#> [1] 3 4 3 3
```

5.10.2 Sostituire elementi in un vettore

Puoi modificare i valori di un vettore usando `[ ]` e l'operatore `<-`.

Un singolo elemento:

```
my_vec[4] <- 500  # Cambia il 4° elemento
my_vec
#> [1]   2   3   1 500   4   3   3   7
```

Più elementi:

```
my_vec[c(6, 7)] <- 100  # Cambia il 6° e 7° elemento
my_vec
#> [1]   2   3   1 500   4 100 100   7
```

Con condizioni logiche:

```
my_vec[my_vec <= 4] <- 1000  # Cambia valori   4
my_vec
#> [1] 1000 1000 1000  500 1000  100  100    7
```

5.10.3 Ordinare un vettore

Dal più piccolo al più grande:

```
vec_sort <- sort(my_vec)
vec_sort
#> [1]    7  100  100  500 1000 1000 1000 1000
```

Dal più grande al più piccolo:

```
vec_sort2 <- sort(my_vec, decreasing = TRUE)
vec_sort2
#> [1] 1000 1000 1000 1000  500  100  100    7
```

Ordinare un vettore in base a un altro:

```
height <- c(180, 155, 160, 167, 181)
p.names <- c("Joanna", "Charlotte", "Helen", "Karen", "Amy")
height_ord <- order(height)
names_ord <- p.names[height_ord]
names_ord
#> [1] "Charlotte" "Helen"     "Karen"     "Joanna"    "Amy"
```

5.11 Operazioni Vettoriali e Vettorizzazione in R

La vettorializzazione è una delle caratteristiche più potenti di R, che consente di applicare operazioni o funzioni direttamente a tutti gli elementi di un vettore in modo simultaneo, senza dover ricorrere a cicli espliciti. Questo approccio rende il codice più conciso, leggibile ed efficiente, sfruttando al meglio le capacità intrinseche del linguaggio.

5.11.1 Operazioni Aritmetiche su Vettori

Le operazioni algebriche in R, come addizione, sottrazione, moltiplicazione e divisione, sono vettorizzate. Questo significa che ogni operazione viene applicata "elemento per elemento" al vettore.

Consideriamo ad esempio il seguente vettore:

```
my_vec <- c(3, 5, 7, 1, 9, 20)
```

Se vogliamo moltiplicare ciascun elemento di `my_vec` per 5, possiamo scrivere:

```
my_vec * 5
#> [1]  15  25  35   5  45 100
```

Analogamente, possiamo effettuare altre operazioni algebriche, come divisione o elevamento a potenza:

```
my_vec / 2
#> [1]  1.5  2.5  3.5  0.5  4.5 10.0
```

```
my_vec^2
#> [1]   9  25  49   1  81 400
```

Queste operazioni vengono applicate automaticamente a ciascun elemento del vettore, senza dover iterare su di essi.

5.11.2 Operazioni Elemento per Elemento tra Due Vettori

La vettorializzazione consente anche di eseguire operazioni tra due vettori, applicandole elemento per elemento. Supponiamo di avere un secondo vettore:

```
my_vec2 <- c(17, 15, 13, 19, 11, 0)
```

Se vogliamo sommare i due vettori, possiamo scrivere:

```
my_vec + my_vec2
#> [1] 20 20 20 20 20 20
```

In questo caso, il primo elemento di `my_vec` viene sommato al primo elemento di `my_vec2`, il secondo elemento al secondo, e così via.

Esempio 5.3. Di seguito mostriamo come calcolare i punteggi totali per 10 individui che hanno risposto ai 5 item della *Satisfaction With Life Scale* (SWLS), utilizzando le formule e l'aritmetica vettorializzata di R.

Step 1: Definiamo i punteggi per ciascun item. Ogni vettore contiene i punteggi dati dai 10 individui a uno specifico item della scala:

```
# Punteggi dei 10 individui per ciascun item
item1 <- c(5, 4, 6, 7, 3, 2, 5, 6, 4, 7)
item2 <- c(3, 2, 4, 6, 2, 1, 4, 5, 3, 6)
item3 <- c(4, 5, 6, 5, 3, 2, 5, 7, 4, 5)
item4 <- c(2, 3, 4, 3, 2, 1, 3, 4, 2, 5)
item5 <- c(2, 2, 3, 4, 1, 1, 3, 3, 2, 4)
```

I valori 5, 3, 4, 2, 2 sono i punteggi del primo individuo sui 5 item; i punteggio 4, 2, 5, 3, 2 sono i punteggi del secondo individuo sui 5 item, e così via.

Step 2: Sommiamo i punteggi per calcolare il totale. Il punteggio totale di ciascun individuo è la somma dei punteggi relativi ai 5 item. Formalmente, per l'individuo i ($i = 1, 2, \ldots, 10$), il punteggio totale è calcolato come:

$$\text{PunteggioTotale}_i = \text{item1}_i + \text{item2}_i + \text{item3}_i + \text{item4}_i + \text{item5}_i.$$

In R, possiamo sommare i vettori direttamente grazie all'aritmetica vettorializzata:

```
# Calcolo dei punteggi totali per ciascun individuo
total_scores <- item1 + item2 + item3 + item4 + item5
total_scores
#>  [1] 16 16 23 25 11  7 20 25 15 27
```

Il risultato è un vettore con i punteggi totali per ciascun individuo.

Step 3: Mostriamo i risultati. Per organizzare meglio i dati, creiamo una tabella che associa i punteggi totali agli individui:

```
# Creiamo una tabella con i punteggi totali
individui <- paste("Individuo", 1:10)
risultati_swls <- data.frame(Individuo = individui, PunteggioTotale = total_scores)
print(risultati_swls)
#>       Individuo PunteggioTotale
#> 1   Individuo 1              16
#> 2   Individuo 2              16
#> 3   Individuo 3              23
#> 4   Individuo 4              25
#> 5   Individuo 5              11
#> 6   Individuo 6               7
#> 7   Individuo 7              20
#> 8   Individuo 8              25
#> 9   Individuo 9              15
#> 10 Individuo 10              27
```

Spiegazione delle operazioni:

1. Ogni vettore contiene i punteggi di 10 individui per un dato item. Ad esempio, il vettore `item1` contiene i punteggi relativi al primo item, e così via.

2. Grazie all'aritmetica vettorializzata, quando sommiamo i vettori item1, item2, item3, item4, item5, R somma elemento per elemento:

$$\text{total\_scores}_i = \text{item1}_i + \text{item2}_i + \text{item3}_i + \text{item4}_i + \text{item5}_i$$

3. Questa tecnica consente di calcolare rapidamente i punteggi totali per tutti gli individui senza dover scrivere un ciclo esplicito, rendendo il codice più semplice e leggibile.

Questo esempio illustra come R semplifichi operazioni complesse grazie al calcolo vettorializzato, migliorando l'efficienza e la chiarezza del codice.

5.11.3 Attenzione al Riciclo dei Vettori

Se i due vettori hanno lunghezze diverse, R applicherà il meccanismo di **riciclo**: gli elementi del vettore più corto verranno ripetuti ciclicamente per abbinarsi alla lunghezza del vettore più lungo. Questo comportamento, sebbene utile, richiede attenzione per evitare risultati inattesi.

Ad esempio:

```
short_vec <- c(1, 2)
my_vec + short_vec
#> [1]  4  7  8  3 10 22
```

In questo caso, gli elementi di `short_vec` vengono riciclati per abbinarsi alla lunghezza di `my_vec`. Il risultato è:

```
(3+1, 5+2, 7+1, 1+2, 9+1, 20+2)
```

5.11.4 Applicazione di Funzioni su Vettori

La vettorializzazione non si limita alle operazioni algebriche, ma si estende anche all'uso di funzioni. Supponiamo di voler calcolare il logaritmo naturale di ciascun elemento di un vettore:

```
log(my_vec)
#> [1] 1.099 1.609 1.946 0.000 2.197 2.996
```

La funzione `log()` viene applicata automaticamente a ogni elemento del vettore. Analogamente, possiamo utilizzare altre funzioni predefinite di R, come:

```
sqrt(my_vec)  # Calcola la radice quadrata di ciascun elemento
#> [1] 1.732 2.236 2.646 1.000 3.000 4.472
exp(my_vec)   # Eleva e alla potenza specificata da ciascun elemento
#> [1] 2.009e+01 1.484e+02 1.097e+03 2.718e+00 8.103e+03 4.852e+08
```

In conclusione, la vettorializzazione in R rappresenta un approccio elegante ed efficiente per gestire calcoli su vettori. Che si tratti di operazioni algebriche, operazioni tra vettori o applicazione di funzioni, la possibilità di evitare cicli espliciti migliora la leggibilità e la velocità del codice. Tuttavia, è importante prestare attenzione al riciclo dei vettori per evitare errori non intenzionali.

5.12 Gestire dati mancanti (NA)

R rappresenta i dati mancanti con `NA`. La gestione dei dati mancanti dipende dalla funzione utilizzata.

Calcolo con dati mancanti:

```
temp <- c(7.2, NA, 7.1, 6.9, 6.5, 5.8, 5.8, 5.5, NA, 5.5)
mean(temp)  # Restituisce NA
#> [1] NA
```

```
mean(temp, na.rm = TRUE)  # Ignora i valori mancanti
#> [1] 6.287
```

Nota: `na.rm = TRUE` è un argomento comune per ignorare i `NA`, ma non tutte le funzioni lo supportano. Consulta la documentazione della funzione per verificare come gestisce i dati mancanti.

In conclusione, manipolare vettori è un'abilità essenziale in R. Dalla selezione e modifica degli elementi all'ordinamento e gestione di dati mancanti, queste tecniche sono alla base dell'analisi dei dati in R.

5.13 I dati in R

In R, i dati possono essere rappresentati in diversi tipi e strutture. Comprendere come gestirli è fondamentale per manipolare, analizzare e riassumere i dataset più complessi.

5.13.1 Tipi di dati in R

R supporta diversi tipi di dati:

1. **Numeric**: Numeri decimali (es. `2.5`).
2. **Integer**: Numeri interi (es. `3`).
3. **Logical**: Valori booleani (`TRUE` o `FALSE`) e `NA` per dati mancanti.
4. **Character**: Stringhe di testo (es. `"hello"`).
5. **Factor**: Variabili categoriche (es. livelli come `"low"`, `"medium"`, `"high"`).

Puoi verificare il tipo di un oggetto con `class()` e controllare se appartiene a un tipo specifico con funzioni come `is.numeric()`. È anche possibile convertire un tipo in un altro con funzioni come `as.character()`.

5.13.2 Strutture di dati in R

Vettori: Contengono dati dello stesso tipo (es. numeri, stringhe o logici).

```
my_vec <- c(1, 2, 3)
my_vec
#> [1] 1 2 3
```

Matrici e array: Strutture bidimensionali (matrici) o multidimensionali (array) con dati dello stesso tipo.

Creare una matrice:

```
my_mat <- matrix(1:12, nrow = 3, byrow = TRUE)
my_mat
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    2    3    4
#> [2,]    5    6    7    8
#> [3,]    9   10   11   12
```

Operazioni utili:

- **Trasposizione**: `t(my_mat)`
- **Diagonale**: `diag(my_mat)`
- **Moltiplicazione matriciale**: `mat1 %*% mat2`

Liste: Possono contenere elementi di tipi diversi, inclusi vettori, matrici o altre liste.

```
my_list <- list(
  numbers = c(1, 2),
  text = "hello",
  mat = matrix(1:4, nrow = 2)
)
my_list$numbers  # Accedi agli elementi con il nome
#> [1] 1 2
```

Data frame: Strutture bidimensionali che possono contenere colonne di tipi diversi. Ideale per dataset strutturati.

Creare un data frame:

```
height <- c(180, 155, 160)
weight <- c(65, 50, 52)
names <- c("Joanna", "Charlotte", "Helen")

dataf <- data.frame(height = height, weight = weight, names = names)
str(dataf)  # Mostra la struttura del data frame
#> 'data.frame':    3 obs. of  3 variables:
#>  $ height: num  180 155 160
```

```
#>  $ weight: num  65 50 52
#>  $ names : chr  "Joanna" "Charlotte" "Helen"
```

Per convertire le stringhe in fattori durante la creazione:

```
dataf <- data.frame(
  height = height,
  weight = weight,
  names = names,
  stringsAsFactors = TRUE
)

dataf
#>   height weight     names
#> 1    180     65    Joanna
#> 2    155     50 Charlotte
#> 3    160     52     Helen
```

5.13.3 Operazioni utili sui data frame

- **Verificare dimensioni**: `dim(dataf)`
- **Visualizzare struttura**: `str(dataf)`
- **Accedere a colonne**: `dataf$height`

5.14 Operazioni di Base in R

5.14.1 Operazioni Aritmetiche

Come abbiamo visto in precedenza, R supporta le classiche operazioni aritmetiche come somma (`+`), sottrazione (`-`), moltiplicazione (`*`), divisione (`/`) ed esponenziazione (`^`).

5.14.2 Operazioni Logiche

Le operazioni logiche in R includono:

- **&**: "and" logico
- |: "or" logico
- **!**: "not" logico

- $>$: maggiore di
- $<$: minore di
- $==$: uguale a
- $!=$: diverso da

Per esempio:

```
# Maggiore di
3 > 2
#> [1] TRUE
# Uguale a
3 == 2
#> [1] FALSE
```

Esempio 5.4. Consideriamo l'esempio precedente, in cui abbiamo calcolato i punteggi totali dei 10 individui sulla Satisfaction With Life Scale (SWLS). Ora vogliamo determinare la proporzione di individui nel campione che ha ottenuto un punteggio totale maggiore di 15.

I punteggi totali dei 10 individui sono memorizzati nella colonna `PunteggioTotale` del data frame `risultati_swls`:

```
risultati_swls$PunteggioTotale
#>  [1] 16 16 23 25 11  7 20 25 15 27
```

Possiamo creare un vettore logico che indica, per ciascun individuo, se il suo punteggio totale supera 15. In R, l'operatore di confronto `>` restituisce un valore TRUE se la condizione è soddisfatta e FALSE altrimenti:

```
risultati_swls$PunteggioTotale > 15
#>  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
```

In R, i valori TRUE e FALSE possono essere trattati come numeri: TRUE equivale a 1 e FALSE equivale a 0. Questo ci permette di sommare i valori logici per contare quante volte la condizione è soddisfatta. Per calcolare la proporzione, dividiamo questa somma per il numero totale di individui:

```
sum(risultati_swls$PunteggioTotale > 15) / length(risultati_swls$PunteggioTotale)
#> [1] 0.7
```

La proporzione è calcolata come:

$$\text{Proporzione} = \frac{\sum_{i=1}^{n} I(\text{PunteggioTotale}_i > 15)}{n},$$

dove:

- n è il numero totale di individui (in questo caso, 10),
- $I(\text{PunteggioTotale}_i > 15)$ è una funzione indicatrice che vale 1 se il punteggio dell'individuo i è maggiore di 15, e 0 altrimenti.

Nel nostro esempio, la proporzione degli individui con punteggio totale maggiore di 15 è 0.7, cioè il 70% del campione soddisfa questa condizione.

5.15 Estrazione di Sottoinsiemi di Oggetti in R

In R esistono tre operatori principali per estrarre sottoinsiemi di oggetti:

1. **Operatore `[ ]`**
   Questo operatore restituisce sempre un oggetto della stessa classe dell'originale. È utile per selezionare più elementi da un oggetto. È importante chiudere l'estrazione con `]`.
2. **Operatore `[[ ]]`**
   Questo operatore viene utilizzato per estrarre elementi da liste o data frame. A differenza di `[ ]`, permette di estrarre un solo elemento alla volta e la classe dell'oggetto restituito non sarà necessariamente una lista o un data frame. L'estrazione va chiusa con `]]`.
3. **Operatore `$`**
   Come visto in precedenza, questo operatore serve per estrarre elementi da una lista o un data frame utilizzando il loro nome letterale. Il comportamento semantico è simile a quello di `[[ ]]`.

5.15.1 Gli Indici di un Data Frame in R

In R, gli indici di un **data frame** sono utilizzati per selezionare righe e colonne. La sintassi generale è:

```
df[i, j]
```

dove:

- `i` rappresenta l'indice o gli indici delle righe,
- `j` rappresenta l'indice o gli indici delle colonne.

Se uno degli indici viene omesso, si considerano **tutte le righe** o **tutte le colonne**, a seconda della dimensione omessa.

5.15.1.1 Esempi Pratici

1. **Selezione di righe specifiche su tutte le colonne**
   Se vogliamo estrarre solo alcune righe, possiamo specificare gli indici delle righe nel primo argomento e lasciare vuoto il secondo. Ad esempio:

   ```
   df[c(2, 3, 5), ]
   ```

   Questo seleziona la **seconda**, **terza** e **quinta** riga del data frame `df`, includendo **tutte le colonne**.

2. **Selezione di colonne specifiche su tutte le righe**
   Per selezionare solo alcune colonne, specifichiamo i loro indici nel secondo argomento e lasciamo vuoto il primo. Ad esempio:

   ```
   df[, c(2, 3, 5)]
   ```

   Questo seleziona la **seconda**, **terza** e **quinta** colonna del data frame `df`, includendo **tutte le righe**.

3. **Selezione di righe e colonne specifiche**
   Possiamo combinare gli indici per selezionare una sotto-matrice specifica. Ad esempio:

   ```
   df[c(2, 4), c(1, 3)]
   ```

   Questo seleziona le righe **2** e **4** e le colonne **1** e **3**.

5.15.1.2 Ulteriori Dettagli

- **Selezione singola di riga o colonna**
  Se vogliamo estrarre una singola riga o colonna, possiamo specificare un solo valore per `i` o `j`. Ad esempio:

  ```
  df[1, ]  # Prima riga, tutte le colonne
  df[, 2]  # Seconda colonna, tutte le righe
  ```

- **Uso di nomi invece di indici**
  Se il data frame ha **nomi** per righe o colonne, possiamo utilizzarli per la selezione. Ad esempio:

  ```
  df["nome_riga", ]        # Seleziona la riga con nome "nome_riga"
  df[, "nome_colonna"]     # Seleziona la colonna con nome "nome_colonna"
  ```

- **Selezione logica**
  Possiamo utilizzare un vettore logico per selezionare righe o colonne. Ad

esempio, per selezionare le righe dove il valore nella prima colonna è maggiore di 10:

```
df[df[, 1] > 10, ]
```

5.15.1.3 Sintesi Visiva

Sintassi	Descrizione
`df[i, ]`	Seleziona la riga `i` con tutte le colonne.
`df[, j]`	Seleziona la colonna `j` con tutte le righe.
`df[c(i1, i2), c(j1, j2)]`	Seleziona righe e colonne specifiche.
`df[i, j]`	Seleziona l'intersezione di righe e colonne.
`df[ , ]`	Restituisce l'intero data frame.

Questa flessibilità rende l'indicizzazione dei data frame in R potente ed efficace per manipolare e analizzare i dati.

Esempio 5.5. Consideriamo il data frame `iris` incluso di default in `base` R.

```
iris |> head()
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          5.1         3.5          1.4         0.2  setosa
#> 2          4.9         3.0          1.4         0.2  setosa
#> 3          4.7         3.2          1.3         0.2  setosa
#> 4          4.6         3.1          1.5         0.2  setosa
#> 5          5.0         3.6          1.4         0.2  setosa
#> 6          5.4         3.9          1.7         0.4  setosa
```

Usiamo la funzione `head()` per stampare le prime 6 righe del data frame.

L'istruzione seguente restituisce le prime tre colonne del dataset `iris`.

```
iris[, 1:3] |> head()
#>   Sepal.Length Sepal.Width Petal.Length
#> 1          5.1         3.5          1.4
#> 2          4.9         3.0          1.4
#> 3          4.7         3.2          1.3
#> 4          4.6         3.1          1.5
#> 5          5.0         3.6          1.4
#> 6          5.4         3.9          1.7
```

Selezione di colonne specifiche per nome:

```
iris[, c('Sepal.Length', 'Petal.Length')] |>
  head()
#>   Sepal.Length Petal.Length
```

```
#> 1          5.1          1.4
#> 2          4.9          1.4
#> 3          4.7          1.3
#> 4          4.6          1.5
#> 5          5.0          1.4
#> 6          5.4          1.7
```

Selezione di una singola colonna:

```
iris[, 'Petal.Length']
#>   [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3
#>  [18] 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4
#>  [35] 1.5 1.2 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7
#>  [52] 4.5 4.9 4.0 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1
#>  [69] 4.5 3.9 4.8 4.0 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5
#>  [86] 4.5 4.7 4.4 4.1 4.0 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1
#> [103] 5.9 5.6 5.8 6.6 4.5 6.3 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9
#> [120] 5.0 5.7 4.9 6.7 4.9 5.7 6.0 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1
#> [137] 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9 5.7 5.2 5.0 5.2 5.4 5.1
```

oppure

```
iris$Petal.Length
#>   [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3
#>  [18] 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4
#>  [35] 1.5 1.2 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7
#>  [52] 4.5 4.9 4.0 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1
#>  [69] 4.5 3.9 4.8 4.0 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5
#>  [86] 4.5 4.7 4.4 4.1 4.0 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1
#> [103] 5.9 5.6 5.8 6.6 4.5 6.3 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9
#> [120] 5.0 5.7 4.9 6.7 4.9 5.7 6.0 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1
#> [137] 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9 5.7 5.2 5.0 5.2 5.4 5.1
```

Per selezionare righe specifiche, definiamo gli indici corrispondenti. Per esempio, l'istruzione seguente restituisce le righe 1 e 3.

```
iris[c(1, 3), ]
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          5.1         3.5          1.4         0.2  setosa
#> 3          4.7         3.2          1.3         0.2  setosa
```

Filtraggio logico di righe:

```
iris[iris$Species == 'versicolor', ] |> head()
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 51          7.0         3.2          4.7         1.4 versicolor
#> 52          6.4         3.2          4.5         1.5 versicolor
#> 53          6.9         3.1          4.9         1.5 versicolor
```

```
#> 54          5.5         2.3          4.0         1.3 versicolor
#> 55          6.5         2.8          4.6         1.5 versicolor
#> 56          5.7         2.8          4.5         1.3 versicolor
```

Restituisce le righe con `Species` uguale a "versicolor". Numero di righe e colonne del sottoinsieme:

```
dim(iris[iris$Species == 'versicolor', ])
#> [1] 50  5
```

5.15.2 Filtraggio Avanzato con Operatori Logici

Gli operatori logici `&` (AND), `|` (OR) e `!` (NOT) permettono un filtraggio più sofisticato.

Esempio: Filtrare le osservazioni di specie "versicolor" con lunghezza del sepalo non superiore a 5.0:

```
iris[(iris$Species == 'versicolor') & (iris$Sepal.Length <= 5.0), ]
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 58          4.9         2.4          3.3           1 versicolor
#> 61          5.0         2.0          3.5           1 versicolor
#> 94          5.0         2.3          3.3           1 versicolor
```

Numero di osservazioni trovate:

```
dim(iris[(iris$Species == 'versicolor') & (iris$Sepal.Length <= 5.0), ])
#> [1] 3 5
```

5.16 Riflessioni Conclusive

R non è soltanto un linguaggio di programmazione per la statistica, ma rappresenta una filosofia che si fonda su tre principi chiave: **apertura**, **collaborazione** e **avanzamento della conoscenza scientifica**.

Per chi si avvicina a R, sia nel campo della comunicazione sia in altri ambiti, cogliere questa filosofia è essenziale per apprezzarne appieno il valore. R promuove non solo competenze tecniche, ma anche un impegno verso pratiche di **ricerca trasparente e riproducibile**, che costituiscono un pilastro fondamentale per una scienza rigorosa e affidabile.

Open Source
R è un software open source, liberamente accessibile a tutti. Questo significa che chiunque può visualizzarne, modificarne e distribuirne il codice sorgente, promuovendo un ambiente trasparente e collaborativo. Essendo gratuito, R

garantisce accessibilità a ricercatori di tutto il mondo, indipendentemente dal budget o dal supporto istituzionale. Inoltre, grazie alla sua natura aperta, R beneficia del contributo collettivo di una comunità globale eterogenea.

Contributi della Comunità
La comunità di R è uno dei suoi punti di forza principali. Statistici, ricercatori e data scientist di diverse discipline arricchiscono continuamente R sviluppando pacchetti: raccolte di funzioni, dati e codice che ampliano le sue funzionalità. Questa collaborazione ha portato alla creazione di migliaia di pacchetti che coprono tecniche statistiche, metodi grafici e strumenti per la manipolazione dei dati, rendendo R uno strumento sempre più versatile e adatto a un'ampia gamma di esigenze di ricerca.

Ricerca Riproducibile
La ricerca riproducibile consiste nel condurre studi in modo tale che altri possano replicarne i risultati utilizzando gli stessi dati e seguendo la stessa metodologia. Questo approccio è cruciale per la validazione delle scoperte scientifiche, permettendo la verifica dei risultati e la costruzione di nuove conoscenze su basi solide.

R facilita la ricerca riproducibile grazie a:

- Un ecosistema completo di pacchetti per l'analisi dei dati e la generazione di report dinamici.

- Strumenti come R Markdown e Quarto, che permettono di integrare testo descrittivo e codice R in un unico documento. Questa integrazione consente di documentare ogni fase del processo di ricerca—dalla pulizia dei dati all'analisi e alla presentazione dei risultati—garantendo trasparenza e replicabilità.

In conclusione, comprendere la filosofia open source di R e il suo ruolo nella promozione della ricerca riproducibile fornisce un quadro chiaro del motivo per cui R è diventato uno strumento essenziale per ricercatori e statistici di diverse discipline. Per chi opera in psicologia, sfruttare le potenzialità di R significa produrre risultati di ricerca più trasparenti, replicabili e credibili, contribuendo alla robustezza e affidabilità della conoscenza scientifica nel settore.

Esercizi

! Problemi 1

Svolgere gli esercizi da 1 a 38, sia in modo **manuale** che utilizzando **R**. Gli esercizi sono disponibili al seguente link:
Esercizi su Summation Notation.

! Problemi 2

In questo esercizio, lavorerai con i dati della **Satisfaction With Life Scale (SWLS)** raccolti da ciascuno degli studenti del gruppo TPV di appartenenza.
Istruzioni SWLS: Di seguito sono riportate alcune affermazioni con cui puoi descrivere la tua soddisfazione rispetto alla tua vita. Indica quanto sei d'accordo con ciascuna affermazione utilizzando la scala di risposta fornita.

- Il più delle volte la mia vita è vicina al mio ideale di vita.
- Le condizioni della mia vita sono eccellenti.
- Sono soddisfatto/a della mia vita.
- Finora ho ottenuto le cose importanti che voglio dalla vita.
- Se io potessi rivivere la mia vita, non cambierei quasi nulla.

La SWLS utilizza una scala Likert a 7 punti, con i seguenti ancoraggi:

1. Fortemente in disaccordo
2. Disaccordo
3. Leggermente in disaccordo
4. Né d'accordo né in disaccordo
5. Leggermente d'accordo
6. D'accordo
7. Fortemente d'accordo

Il tuo compito sarà analizzare i dati raccolti **sia manualmente su carta** che utilizzando **R**.
Parte 1: Calcolo Manuale

1. **Calcolo del punteggio totale**
   - La SWLS è composta da 5 item, ciascuno valutato su una scala Likert da 1 a 7.
   - Somma i punteggi dei 5 item per ciascun partecipante per ottenere il punteggio totale.
   - Registra i punteggi totali su carta.

2. **Determinazione della media del campione**
   - Calcola la media aritmetica dei punteggi totali dei 10 studenti.
   - Scrivi il calcolo e il risultato.
3. **Calcolo della deviazione standard**
   - Calcola la deviazione standard dei punteggi totali manualmente utilizzando la formula:

$$s^2 = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}.$$

   - Registra il risultato.

Parte 2: Analisi con R

4. **Creazione del dataset in R**
   - Inserisci i dati in R come un vettore chiamato `swls_scores`.
5. **Calcolo della media e della deviazione standard in R**
   - Usa le funzioni `mean()` e `sd()` per ottenere la media e la deviazione standard dei punteggi totali.
6. **Visualizzazione dei dati**
   - Crea un istogramma per visualizzare la distribuzione dei punteggi totali utilizzando `hist()`. Se non conosci l'istogramma, fai una ricerca su web; commenta il risultato ottenuto.
7. **Identificazione dei punteggi superiori a 20**
   - Utilizza un'operazione logica per contare quanti partecipanti hanno un punteggio totale maggiore di 20.
8. **Filtraggio dei dati**
   - Estrai e visualizza solo i punteggi superiori alla media del campione.
9. **Esportazione dei risultati**
   - Salva i punteggi totali in un file CSV utilizzando la funzione `write.csv()`.

Consegna

- Scrivi le risposte della **Parte 1** su carta.
- Scrivi il codice e i risultati della **Parte 2** in un file `.R` e invialo come consegna.

Soluzioni 2

Parte 1: Calcolo Manuale

1. **Calcolo del punteggio totale**
   - Supponiamo che i punteggi per 10 studenti siano:

Studente	Item 1	Item 2	Item 3	Item 4	Item 5	Totale
1	5	4	6	3	2	20
2	4	2	5	3	2	16
3	6	4	6	4	3	23
4	7	6	5	3	4	25
5	3	2	3	2	1	11
6	2	1	2	1	1	7
7	5	4	5	3	3	20
8	6	5	7	4	3	25
9	4	3	4	2	2	15
10	7	6	5	5	4	27

2. **Determinazione della media**
   - Media:

$$\bar{x} = \frac{20 + 16 + 23 + 25 + 11 + 7 + 20 + 25 + 15 + 27}{10} = 18.9$$

3. **Calcolo della deviazione standard**
   - La deviazione standard è:

$$s = \sqrt{\frac{1}{9} \sum (x_i - 18.9)^2} \approx 6.56$$

Parte 2: Analisi con R

4. **Creazione del dataset in R**

```r
swls_scores <- c(20, 16, 23, 25, 11, 7, 20, 25, 15, 27)
```

5. **Calcolo della media e della deviazione standard**

```r
mean(swls_scores)  # Media
sd(swls_scores)    # Deviazione standard
```

6. **Visualizzazione dei dati**

```r
hist(swls_scores, main="Distribuzione SWLS", xlab="Punteggi", col="lightblue", bor
```

7. **Identificazione dei punteggi superiori a 20**

```r
sum(swls_scores > 20)  # Numero di studenti con punteggio > 20
```

8. **Filtraggio dei dati**

```r
swls_scores[swls_scores > mean(swls_scores)]
```

9. **Esportazione dei risultati**

```r
write.csv(data.frame(Student=1:10, Score=swls_scores), "swls_results.csv", row.nam
```

Conclusione Questi esercizi hanno permesso di confrontare il calcolo manuale con l'automatizzazione tramite R, facilitando l'analisi statistica della SWLS.

Informazioni sull'Ambiente di Sviluppo

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] thematic_0.1.6   MetBrewer_0.2.0  ggokabeito_0.1.0 see_0.10.0
#>  [5] gridExtra_2.3    patchwork_1.3.0  bayesplot_1.11.1 psych_2.4.12
#>  [9] scales_1.3.0     markdown_1.13    knitr_1.49       lubridate_1.9.4
#> [13] forcats_1.0.0    stringr_1.5.1    dplyr_1.1.4      purrr_1.0.4
#> [17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1
#> [21] tidyverse_2.0.0  rio_1.2.3        here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] generics_0.1.3    stringi_1.8.4     lattice_0.22-6    hms_1.1.3
#>  [5] digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3    grid_4.4.2
#>  [9] timechange_0.3.0  fastmap_1.2.0     rprojroot_2.0.4   jsonlite_1.8.9
#> [13] mnormt_2.1.1      cli_3.6.4         rlang_1.1.5       munsell_0.5.1
#> [17] withr_3.0.2       yaml_2.3.10       tools_4.4.2       parallel_4.4.2
#> [21] tzdb_0.4.0        colorspace_2.1-1  pacman_0.5.1      vctrs_0.6.5
#> [25] R6_2.6.1          lifecycle_1.0.4   pkgconfig_2.0.3   pillar_1.10.1
#> [29] gtable_0.3.6      glue_1.8.0        xfun_0.50         tidyselect_1.2.1
#> [33] rstudioapi_0.17.1 farver_2.1.2      htmltools_0.5.8.1 nlme_3.1-167
#> [37] rmarkdown_2.29    compiler_4.4.2
```

Bibliografia

6

Utility functions in R

In questo capitolo imparerai a

- conoscere e sapere utilizzare le principali funzioni di utilità di R.

Prerequisiti

- Consultare Introduction to Data Science: Data Wrangling and Visualization with R (**?**)
- Leggere R for Data Science (2e) (**?**).

Preparazione del Notebook

```
here::here("code", "_common.R") |>
  source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(tidyr)
```

6.1 Introduzione

In questo capitolo esamineremo le principali funzioni di utilità di R per raccogliere statistiche descrittive e altre informazioni generali sui data frame.

6.1.1 Funzioni principali e loro utilizzo

Funzione	Descrizione
`summary()`	Restituisce statistiche descrittive di base per ogni colonna di un data frame. Per le colonne numeriche, calcola valori come il minimo, massimo, media, mediana, primo e terzo quartile, e il numero di valori mancanti (se presenti). Per le colonne non numeriche, restituisce il tipo di dati (carattere, logico) e il conteggio delle categorie. **Esempio:** `summary(iris)` restituisce una sintesi delle colonne del dataset `iris`.
`str()` e `glimpse()`	Forniscono una rappresentazione sintetica delle informazioni di un data frame, come dimensione, nomi delle colonne, tipi di dati e valori iniziali. La funzione `str()` fa parte della configurazione base di R (pacchetto `utils`), mentre `glimpse()` è inclusa in `dplyr` (pacchetto `tidyverse`). **Esempio:** `str(mtcars)` o `glimpse(mtcars)`.
`head()` e `tail()`	Permettono di visualizzare rispettivamente le prime o ultime righe di un data frame. Utile per una rapida ispezione del contenuto. Si può specificare il numero di righe da mostrare (es. `head(df, 10)`), altrimenti il valore predefinito è sei righe. **Esempio:** `head(iris)` per vedere le prime righe del dataset `iris`.
`View()` e `view()`	Visualizzano un data frame in una finestra grafica tipo foglio di calcolo all'interno di RStudio. La funzione `View()` è parte della configurazione base di R, mentre `view()` è un alias fornito da `tibble` (pacchetto `tidyverse`). Utile per piccoli data frame, ma poco pratico per dataset di grandi dimensioni. **Esempio:** `View(iris)` apre il dataset `iris` nel visualizzatore di RStudio.

Funzione	Descrizione
`unique()`	Restituisce i valori unici presenti in una colonna o in un vettore. **Esempio:** `unique(iris$Species)` restituisce le specie uniche nel dataset `iris`.
`names()`	Restituisce i nomi delle colonne di un data frame. **Esempio:** `names(mtcars)` restituisce i nomi delle colonne del dataset `mtcars`.
`class()`	Indica il tipo di dato di un oggetto in R, come `numeric`, `character`, `logical`, o `data.frame`. **Esempio:** `class(iris)` restituisce `data.frame`.
`length()`	Restituisce il numero di elementi di un oggetto. Per i data frame, restituisce il numero di colonne. **Esempio:** `length(iris)` restituisce 5 (colonne).
`nrow()` e `ncol()`	Restituiscono rispettivamente il numero di righe e colonne di un data frame. **Esempio:** `nrow(iris)` restituisce 150 (righe), mentre `ncol(iris)` restituisce 5 (colonne).

Questi strumenti rappresentano la base per esplorare rapidamente i dati e comprenderne la struttura prima di passare a manipolazioni più avanzate.

6.2 Un Esempio Concreto

Importiamo il dataset `msleep` contenuto nel pacchetto `ggplot2` che abbiamo caricato in precedenza:

```
data(msleep)
```

Esaminiamo la struttura del data frame usando `str()`:

```
str(msleep)
#> tibble [83 x 11] (S3: tbl_df/tbl/data.frame)
#>  $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tai
#>  $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...
#>  $ vore        : chr [1:83] "carni" "omni" "herbi" "omni" ...
```

```
#>  $ order       : chr [1:83] "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
#>  $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
#>  $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
#>  $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
#>  $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
#>  $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
#>  $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
#>  $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
```

oppure usando `glimpse()`:

```
glimpse(msleep)
#> Rows: 83
#> Columns: 11
#> $ name         <chr> "Cheetah", "Owl monkey", "Mountain beaver", "Greater ~
#> $ genus        <chr> "Acinonyx", "Aotus", "Aplodontia", "Blarina", "Bos", ~
#> $ vore         <chr> "carni", "omni", "herbi", "omni", "herbi", "herbi", "~
#> $ order        <chr> "Carnivora", "Primates", "Rodentia", "Soricomorpha", ~
#> $ conservation <chr> "lc", NA, "nt", "lc", "domesticated", NA, "vu", NA, "~
#> $ sleep_total  <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4, 8.7, 7.0, 10.1, 3.~
#> $ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2, 1.4, NA, 2.9, NA, 0.6, 0~
#> $ sleep_cycle  <dbl> NA, NA, NA, 0.1333, 0.6667, 0.7667, 0.3833, NA, 0.333~
#> $ awake        <dbl> 11.9, 7.0, 9.6, 9.1, 20.0, 9.6, 15.3, 17.0, 13.9, 21.~
#> $ brainwt      <dbl> NA, 0.01550, NA, 0.00029, 0.42300, NA, NA, NA, 0.0700~
#> $ bodywt       <dbl> 50.000, 0.480, 1.350, 0.019, 600.000, 3.850, 20.490, ~
```

Stampiamo le prime righe del data frame:

```
head(msleep)
#> # A tibble: 6 x 11
#>   name                genus      vore  order        conservation sleep_total
#>   <chr>               <chr>      <chr> <chr>        <chr>              <dbl>
#> 1 Cheetah             Acinonyx   carni Carnivora    lc                  12.1
#> 2 Owl monkey          Aotus      omni  Primates     <NA>                17
#> 3 Mountain beaver     Aplodontia herbi Rodentia     nt                  14.4
#> 4 Greater short-tail~ Blarina    omni  Soricomorpha lc                  14.9
#> 5 Cow                 Bos        herbi Artiodactyla domesticated         4
#> 6 Three-toed sloth    Bradypus   herbi Pilosa       <NA>                14.4
#> # i 5 more variables: sleep_rem <dbl>, sleep_cycle <dbl>, awake <dbl>,
#> #   brainwt <dbl>, bodywt <dbl>
```

oppure le ultime righe:

```
tail(msleep)
#> # A tibble: 6 x 11
#>   name                genus    vore  order        conservation sleep_total
#>   <chr>               <chr>    <chr> <chr>        <chr>              <dbl>
```

```
#> 1 Tenrec               Tenrec   omni  Afrosoricida <NA>              15.6
#> 2 Tree shrew           Tupaia   omni  Scandentia   <NA>               8.9
#> 3 Bottle-nosed dolphin Tursiops carni Cetacea      <NA>               5.2
#> 4 Genet                Genetta  carni Carnivora    <NA>               6.3
#> 5 Arctic fox           Vulpes   carni Carnivora    <NA>              12.5
#> 6 Red fox              Vulpes   carni Carnivora    <NA>               9.8
#> # i 5 more variables: sleep_rem <dbl>, sleep_cycle <dbl>, awake <dbl>,
#> #   brainwt <dbl>, bodywt <dbl>
```

Esaminiamo le modalità della variabile qualitativa `vore`:

```
unique(msleep$vore)
#> [1] "carni"   "omni"    "herbi"   NA        "insecti"
```

Se vogliamo la numerosità di ciascuna categoria, possiamo usare `table()`:

```
table(msleep$vore)
#>
#>   carni   herbi insecti    omni
#>      19      32       5      20
```

Si noti che `table()` esclude i dati mancanti.

Stampiamo i nomi delle colonne del data frame:

```
names(msleep)
#>  [1] "name"         "genus"        "vore"         "order"
#>  [5] "conservation" "sleep_total"  "sleep_rem"    "sleep_cycle"
#>  [9] "awake"        "brainwt"      "bodywt"
```

Esaminiamo il tipo di variabile della colonna `vore`:

```
class(msleep$vore)
#> [1] "character"
```

Le dimensioni del data frame sono date da:

```
dim(msleep)
#> [1] 83 11
```

laddove il primo valore è il numero di righe e il secondo valore è il numero di colonne.

Il numero di elementi di un vettore è dato da:

```
length(msleep$vore)
#> [1] 83
```

In alternativa, possiamo usare `nrow()`

```
nrow(msleep)
#> [1] 83
```

per il numero di righe e `ncol()`

```
ncol(msleep)
#> [1] 11
```

per il numero di colonne. In maniera equivalente:

```
dim(msleep)[2]
#> [1] 11
```

6.3 Esercizi

Esercizio

In questo esercizio, utilizzerai R per esplorare i dati raccolti con il questionario **Satisfaction With Life Scale (SWLS)** dagli studenti del tuo gruppo TPV. L'obiettivo è familiarizzare con le funzioni di base di R per caricare, visualizzare e manipolare i dati.

Parte 1: Operazioni Manuali

1. **Creazione e gestione degli oggetti in R**
   - Scrivi su carta i comandi R che creerebbero un oggetto chiamato `swls_scores` contenente i punteggi di 10 studenti.
   - Quali sono le regole per assegnare un nome a un oggetto in R?
2. **Visualizzazione dei dati**
   - Scrivi il comando R per visualizzare il contenuto dell'oggetto `swls_scores`.
   - Come puoi visualizzare solo i primi 5 valori del vettore?
3. **Esplorazione della struttura dei dati**
   - Scrivi i comandi R per verificare il tipo di dati contenuti in `swls_scores`.
   - Come puoi verificare quanti elementi contiene?

Parte 2: Esecuzione in R

4. **Creazione del dataset in R**
   - Inserisci i dati in un oggetto chiamato `swls_scores` in R.
5. **Verifica della struttura dei dati**

- Usa le funzioni `str()`, `class()`, `length()`, `nrow()`, `ncol()` su `swls_scores`.
- Annota i risultati e spiega a parole loro significato.

6. **Visualizzazione dei dati**
   - Usa `head()` e `tail()` per esplorare i dati.
   - Qual è la differenza tra le due funzioni?
7. **Identificazione dei valori unici**
   - Usa `unique(swls_scores)` per individuare i punteggi distinti.
8. **Creazione di una tabella con i dati**
   - Trasforma `swls_scores` in un data frame con una colonna `"Punteggio"` e una colonna `"Studente"` (numerata da 1 a 10).
9. **Esportazione dei dati**
   - Salva il data frame in un file CSV chiamato `"swls_data.csv"` usando `write.csv()`.

Consegna

- Scrivi le risposte della **Parte 1** su carta.
- Scrivi il codice e i risultati della **Parte 2** in un file `.R` e invialo come consegna.

Soluzione

Parte 1: Operazioni Manuali

1. **Creazione e gestione degli oggetti in R**
   - Consideriamo dei valori di risposta arbitrari. Il comando per creare l'oggetto `swls_scores` è:

     ```
     swls_scores <- c(20, 16, 23, 25, 11, 7, 20, 25, 15, 27)
     ```

   - Regole per assegnare un nome a un oggetto in R:
     – Non può iniziare con un numero.
     – Non può contenere spazi o caratteri speciali (tranne _ e .).
     – Non deve avere lo stesso nome di funzioni già esistenti.
2. **Visualizzazione dei dati**
   - Per visualizzare il contenuto:

     ```
     swls_scores
     ```

   - Per visualizzare solo i primi 5 valori:

```
head(swls_scores, 5)
```

3. **Esplorazione della struttura dei dati**
   - Per verificare il tipo di dati:

     ```
     class(swls_scores)
     ```

   - Per verificare il numero di elementi:

     ```
     length(swls_scores)
     ```

Parte 2: Esecuzione in R

4. **Creazione del dataset in R**

   ```
   swls_scores <- c(20, 16, 23, 25, 11, 7, 20, 25, 15, 27)
   ```

5. **Verifica della struttura dei dati**

   ```
   str(swls_scores)
   class(swls_scores)
   length(swls_scores)
   ```

   - `str()` mostra che `swls_scores` è un vettore numerico.
   - `class()` conferma che è di tipo `"numeric"`.
   - `length()` indica che il vettore ha 10 elementi.

6. **Visualizzazione dei dati**

   ```
   head(swls_scores)
   tail(swls_scores)
   ```

   - `head()` mostra i primi 6 elementi, `tail()` gli ultimi 6.

7. **Identificazione dei valori unici**

   ```
   unique(swls_scores)
   ```

   - Restituisce: `7, 11, 15, 16, 20, 23, 25, 27`.

8. **Creazione di una tabella con i dati**

   ```
   df_swls <- data.frame(Studente = 1:10, Punteggio = swls_scores)
   df_swls
   ```

9. **Esportazione dei dati**

   ```
   write.csv(df_swls, "swls_data.csv", row.names=FALSE)
   ```

Conclusione

Questi esercizi hanno introdotto i comandi di base per creare, visualizzare e manipolare dati in R.

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] thematic_0.1.6   MetBrewer_0.2.0  ggokabeito_0.1.0 see_0.10.0
#>  [5] gridExtra_2.3    patchwork_1.3.0  bayesplot_1.11.1 psych_2.4.12
#>  [9] scales_1.3.0     markdown_1.13    knitr_1.49       lubridate_1.9.4
#> [13] forcats_1.0.0    stringr_1.5.1    dplyr_1.1.4      purrr_1.0.4
#> [17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1
#> [21] tidyverse_2.0.0  rio_1.2.3        here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] utf8_1.2.4        generics_0.1.3    stringi_1.8.4     lattice_0.22-6
#>  [5] hms_1.1.3         digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3
#>  [9] grid_4.4.2        timechange_0.3.0  fastmap_1.2.0     rprojroot_2.0.4
#> [13] jsonlite_1.8.9    mnormt_2.1.1      cli_3.6.4         rlang_1.1.5
#> [17] munsell_0.5.1     withr_3.0.2       tools_4.4.2       parallel_4.4.2
#> [21] tzdb_0.4.0        colorspace_2.1-1  pacman_0.5.1      vctrs_0.6.5
#> [25] R6_2.6.1          lifecycle_1.0.4   pkgconfig_2.0.3   pillar_1.10.1
#> [29] gtable_0.3.6      glue_1.8.0        xfun_0.50         tidyselect_1.2.1
#> [33] rstudioapi_0.17.1 farver_2.1.2      htmltools_0.5.8.1 nlme_3.1-167
#> [37] rmarkdown_2.29    compiler_4.4.2
```

Bibliografia

7

Programmazione in R

! In questo capitolo imparerai a

- Conoscere e sapere utilizzare le funzioni, le istruzioni condizionali e i cicli.

Prerequisiti

- Consultare Introduction to Data Science: Data Wrangling and Visualization with R (**?**)
- Leggere R for Data Science (2e).

Preparazione del Notebook

```
here::here("code", "_common.R") |>
  source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(tidyr)
```

7.1 Introduzione

In questo capitolo esploreremo tre strumenti fondamentali per la scrittura di codice in R: le funzioni, le istruzioni condizionali e i cicli. Questi elementi costituiscono la base per sviluppare script flessibili, efficienti e riutilizzabili, essenziali per ogni programmatore o analista che utilizza R.

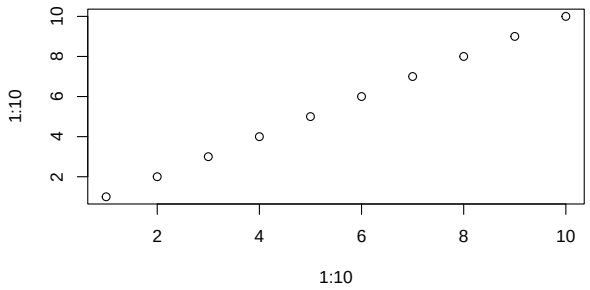
7.2 Funzioni

R offre un'ampia gamma di funzioni integrate per supportare l'analisi statistica, la manipolazione dei dati e la visualizzazione grafica, rendendolo uno strumento estremamente versatile per diverse esigenze.

Esempi di funzioni comuni includono:

```
# Sommare numeri
sum(1, 2, 3)  # Restituisce la somma dei numeri
#> [1] 6
```

```
# Creare un grafico semplice
plot(1:10, 1:10)  # Crea un grafico a dispersione dei valori
```



In sostanza, una funzione è un blocco di codice progettato per svolgere un'operazione specifica. Puoi pensare a una funzione come a una "black box": fornisci un input (i dati), la funzione elabora l'informazione attraverso le sue istruzioni e restituisce un output (il risultato). Questo approccio modulare semplifica il lavoro, permettendo di riutilizzare e combinare facilmente diverse operazioni.

7.2.1 Creare Funzioni Personalizzate

La creazione di funzioni personalizzate in R è uno strumento essenziale per migliorare la programmazione, soprattutto per gestire operazioni ripetitive o complesse. Le funzioni consentono di rendere il codice più leggibile, efficiente e riutilizzabile, promuovendo un approccio organizzato e chiaro alla risoluzione dei problemi.

7.2.1.1 Vantaggi delle Funzioni Personalizzate

L'uso di funzioni personalizzate offre numerosi benefici:

- **Chiarezza e leggibilità**: Un nome descrittivo permette di comprendere immediatamente lo scopo della funzione, anche a distanza di tempo o per altri utenti che leggono il codice.
- **Manutenzione semplificata**: Modificare il codice all'interno di una funzione aggiorna automaticamente tutte le sue occorrenze, riducendo il rischio di errori e semplificando il debugging.
- **Riduzione degli errori**: Si evitano gli errori tipici del copia-e-incolla, come omissioni o incoerenze nei programmi complessi.
- **Riutilizzabilità**: Una funzione ben progettata può essere utilizzata in più contesti o progetti, risparmiando tempo e sforzi.

7.2.1.2 Quando Creare una Funzione?

Un buon criterio per decidere se creare una funzione è osservare se il medesimo blocco di codice viene copiato più volte. Se ti trovi a ripetere lo stesso codice più di due volte, probabilmente è il momento di creare una funzione. Questo aiuta a scrivere codice più pulito, scalabile e professionale, migliorando anche la sostenibilità del lavoro a lungo termine.

7.2.2 Sintassi di una Funzione

La struttura base di una funzione in R è la seguente:

```
nome_funzione <- function(argomenti) {
  # Corpo della funzione
  codice
  return(risultato)  # Facoltativo: restituisce il valore calcolato
}
```

- `nome_funzione`: Nome della funzione, scelto per descrivere chiaramente la sua finalità.
- `argomenti`: Parametri necessari per eseguire le operazioni all'interno della funzione.
- `codice`: Le istruzioni che definiscono il comportamento della funzione.
- `risultato`: Il valore restituito dalla funzione. Se non si usa `return()`, R restituisce l'ultimo valore calcolato.

Esempio 7.1. Immaginiamo di voler creare una funzione per sommare due numeri.

```
somma_due <- function(a, b) {
  a + b  # Restituisce la somma dei due numeri
}
```

Per utilizzarla, basta richiamarla specificando i parametri:

```
somma_due(5, 3)  # Restituisce 8
```

Questo approccio aiuta a scrivere codice più leggibile e facile da gestire. Ad esempio, se in futuro volessi modificare il comportamento della somma (ad esempio, aggiungere un messaggio di log), basterà intervenire solo all'interno della funzione.

Esempio 7.2. Immaginiamo di avere un dataset con i punteggi di 10 individui su 3 subscale di un test psicometrico. L'obiettivo è:

1. Creare una funzione per calcolare il punteggio totale di un individuo.
2. Creare una funzione per trovare il massimo punteggio totale nel campione.
3. Creare una funzione per individuare chi ha ottenuto il massimo punteggio.

Passo 1: Simulazione dei Dati. Simuliamo i punteggi di 10 individui su 3 subscale:

```
# Simulazione dei punteggi
set.seed(123)
punteggi <- data.frame(
  individuo = paste("Individuo", 1:10),
  subscale1 = sample(30:50, 10, replace = TRUE),
  subscale2 = sample(40:60, 10, replace = TRUE),
  subscale3 = sample(35:55, 10, replace = TRUE)
)
print(punteggi)
#>       individuo subscale1 subscale2 subscale3
#> 1   Individuo 1        44        44        48
#> 2   Individuo 2        48        58        51
#> 3   Individuo 3        43        48        45
#> 4   Individuo 4        32        42        41
#> 5   Individuo 5        39        47        55
#> 6   Individuo 6        47        46        46
#> 7   Individuo 7        40        49        49
#> 8   Individuo 8        34        48        44
```

```
#> 9   Individuo 9        49        58        47
#> 10 Individuo 10        43        43        41
```

La funzione `sample()` in R è utilizzata per estrarre casualmente un sottoinsieme di valori da un vettore. Nell'esempio sopra, `sample()` viene utilizzata per generare casualmente i punteggi delle subscale dei test psicometrici.

Nell'istruzione `subscale1 <- sample(30:50, 10, replace = TRUE)`

- `30:50`: Rappresenta il vettore di numeri interi da cui vengono estratti i punteggi (valori possibili tra 30 e 50).
- `10`: Indica che vogliamo estrarre 10 valori.
- `replace = TRUE`: Consente che lo stesso valore possa essere estratto più volte (estrazione con ripetizione).

Passo 2: Creazione delle Funzioni.

1. **Calcolo del punteggio totale per ogni individuo**
Questa funzione somma i punteggi delle subscale di un individuo:

```
calcola_totale <- function(subscale1, subscale2, subscale3) {
  return(subscale1 + subscale2 + subscale3)
}
```

2. **Trovare il punteggio massimo nel campione**
Questa funzione accetta un vettore di punteggi totali e restituisce il valore massimo:

```
trova_massimo <- function(punteggi_totali) {
  return(max(punteggi_totali))
}
```

3. **Individuare l'individuo con il punteggio massimo**
Questa funzione accetta un data frame con i punteggi e restituisce il nome dell'individuo con il punteggio più alto:

```
trova_individuo_massimo <- function(punteggi) {
  punteggi_totali <- rowSums(punteggi[, c("subscale1", "subscale2", "subscale3")])
  indice_massimo <- which.max(punteggi_totali)
  return(punteggi$individuo[indice_massimo])
}
```

La funzione `which.max()` restituisce l'indice della posizione in cui si trova il valore massimo in un vettore.

Passo 3: Applicazione delle Funzioni

1. **Calcolo dei punteggi totali per ogni individuo**
Applichiamo la funzione ai dati simulati:

```
punteggi$punteggio_totale <- with(
  punteggi, calcola_totale(subscale1, subscale2, subscale3)
 )
print(punteggi)
#>       individuo subscale1 subscale2 subscale3 punteggio_totale
#> 1   Individuo 1        44        44        48              136
#> 2   Individuo 2        48        58        51              157
#> 3   Individuo 3        43        48        45              136
#> 4   Individuo 4        32        42        41              115
#> 5   Individuo 5        39        47        55              141
#> 6   Individuo 6        47        46        46              139
#> 7   Individuo 7        40        49        49              138
#> 8   Individuo 8        34        48        44              126
#> 9   Individuo 9        49        58        47              154
#> 10 Individuo 10        43        43        41              127
```

2. **Troviamo il punteggio massimo nel campione**

```
massimo <- trova_massimo(punteggi$punteggio_totale)
print(massimo)
#> [1] 157
```

3. **Troviamo chi ha il punteggio massimo**

```
individuo_massimo <- trova_individuo_massimo(punteggi)
print(individuo_massimo)
#> [1] "Individuo 2"
```

7.2.3 Stile

È consigliato di usare nomi di funzioni chiari e descrittivi, preferibilmente verbi (es. `compute_mean()`). Inoltre, è importante mantenere una struttura leggibile, con spazi coerenti e indentazione.

7.3 Istruzioni Condizionali in R

Le istruzioni condizionali permettono di introdurre logica nel tuo codice. Ad esempio, l'operazione `x * y` si limita a moltiplicare i valori di `x` e `y`, senza

alcuna logica aggiunta. Con le istruzioni condizionali, puoi dire al programma di eseguire diverse operazioni a seconda che una condizione sia vera (`TRUE`) o falsa (`FALSE`).

L'istruzione condizionale più comune in R è `if`. Può essere letta come: **"Se la condizione è vera, esegui un'azione"**. Con `else`, si estende la logica: **"Se la condizione è vera, fai qualcosa; altrimenti fai qualcos'altro"**.

La struttura generale è questa:

```
if (condizione) {
  # Codice eseguito se la condizione è TRUE
} else {
  # Codice eseguito se la condizione è FALSE
}
```

Immagina questa situazione:

- "Se un partecipante al test psicologico riporta un punteggio elevato sulla scala di ansia (es. > 15), consigliagli un esercizio di rilassamento. Altrimenti, non è necessario."

Vediamo come rappresentare questa situazione in R.

```
anxiety_score <- 18 # Punteggio riportato dal partecipante

if (anxiety_score > 15) {
    exercise <- "rilassamento"
} else {
    exercise <- "nessun esercizio"
}

exercise
#> [1] "rilassamento"
```

Se il punteggio è maggiore di 15, il risultato sarà:

```
[1] "rilassamento"
```

Se il punteggio è inferiore o uguale a 15, il risultato sarà:

```
[1] "nessun esercizio"
```

7.3.1 Uso di `ifelse()`

Un'alternativa più compatta a `if` e `else` è la funzione `ifelse()`, utile soprattutto per vettori. Ad esempio, supponiamo di avere i punteggi di ansia di un gruppo di partecipanti e vogliamo decidere se assegnare un esercizio di rilassamento a ciascuno:

```
anxiety_scores <- c(12, 18, 9, 22, 15)
exercises <- ifelse(anxiety_scores > 15, "rilassamento", "nessun esercizio")
```

Il risultato sarà:

```
exercises
#> [1] "nessun esercizio" "rilassamento"     "nessun esercizio"
#> [4] "rilassamento"     "nessun esercizio"
```

7.3.2 Creare una Funzione con Istruzioni Condizionali

Le istruzioni condizionali possono essere racchiuse in una funzione per rendere il codice più flessibile e riutilizzabile. Ad esempio, supponiamo di voler personalizzare un feedback per un partecipante in base al punteggio ottenuto in un questionario:

```
feedback <- function(score) {
    if (score > 15) {
        "Consigliamo un esercizio di rilassamento."
    } else if (score > 10) {
        "Monitoriamo la situazione, ma non è necessario alcun intervento."
    } else {
        "Nessun intervento necessario."
    }
}
```

```
feedback(18)
#> [1] "Consigliamo un esercizio di rilassamento."
```

```
feedback(12)
#> [1] "Monitoriamo la situazione, ma non è necessario alcun intervento."
```

```
feedback(8)
#> [1] "Nessun intervento necessario."
```

In conclusione, le istruzioni condizionali come `if`, `else` e `ifelse()` sono strumenti fondamentali per introdurre logica e controllo nel tuo codice. Puoi usarle per prendere decisioni, gestire errori e rendere il tuo codice più flessibile ed efficiente. Creare funzioni che incorporano queste istruzioni è un passo fondamentale per scrivere codice ordinato e riutilizzabile in contesti psicologici e non solo.

7.3.3 Combinare Operatori Logici in R

Finora abbiamo creato funzioni abbastanza semplici e mirate. Ora proviamo a realizzare una funzione leggermente più complessa. Immaginiamo di voler

determinare se una persona ha avuto **una buona giornata** basandoci su due criteri:

1. **Livello di stress**: basso (`TRUE`) o alto (`FALSE`).
2. **Livello di supporto sociale percepito**: alto (`TRUE`) o basso (`FALSE`).

Vogliamo creare una funzione che prenda questi due fattori e restituisca un messaggio che descrive come potrebbe essere stata la giornata della persona.

Ecco come possiamo costruire la funzione:

```
good_day <- function(low_stress, high_support) {
    if (low_stress == TRUE && high_support == TRUE) {
        "Giornata fantastica! Ti senti calmo e supportato."
    } else if (low_stress == FALSE && high_support == TRUE) {
        "Il supporto sociale ti aiuta a gestire lo stress elevato."
    } else if (low_stress == TRUE && high_support == FALSE) {
        "Nonostante lo stress sia basso, la mancanza di supporto sociale pesa."
    } else if (low_stress == FALSE && high_support == FALSE) {
        "Giornata difficile: stress elevato e poco supporto sociale."
    }
}
```

Esempi di utilizzo.

Caso 1: Stress basso e supporto sociale alto

```
good_day(low_stress = TRUE, high_support = TRUE)
#> [1] "Giornata fantastica! Ti senti calmo e supportato."
```

Caso 2: Stress elevato e supporto sociale alto.

```
good_day(FALSE, TRUE)
#> [1] "Il supporto sociale ti aiuta a gestire lo stress elevato."
```

Caso 3: Stress basso e supporto sociale basso.

```
good_day(TRUE, FALSE)
#> [1] "Nonostante lo stress sia basso, la mancanza di supporto sociale pesa."
```

Caso 4: Stress elevato e supporto sociale basso.

```
good_day(FALSE, FALSE)
#> [1] "Giornata difficile: stress elevato e poco supporto sociale."
```

La funzione considera tutte le combinazioni di stress e supporto sociale:

1. **Stress basso e supporto alto**: giornata ideale.

2. **Stress elevato e supporto alto**: il supporto aiuta a mitigare lo stress.
3. **Stress basso e supporto basso**: la mancanza di supporto rovina una situazione potenzialmente buona.
4. **Stress elevato e supporto basso**: la situazione peggiore.

Nell'esempio abbiamo usato i seguenti operatori logici:

- **`&&` (AND logico)**: Entrambe le condizioni devono essere vere.
- **`==` (uguale a)**: Verifica se una variabile è vera o falsa.

Ad esempio, questa condizione:

```
if (low_stress == TRUE && high_support == TRUE)
```

verifica se il livello di stress è basso **e** il supporto sociale è alto.

In conclusione, questa funzione dimostra come combinare condizioni logiche complesse utilizzando operatori logici come `&&` (AND) e `||` (OR). Grazie a questi strumenti, possiamo gestire facilmente logiche più articolate, mantenendo il codice leggibile e funzionale.

7.3.4 Gli operatori Logici in R

Gli operatori logici sono essenziali per definire le condizioni nelle istruzioni `if`. Ecco una tabella riassuntiva con i principali operatori:

Operatore	Descrizione tecnica	Significato	Esempio
`&&`	AND logico	Entrambe le condizioni devono essere vere	`if(cond1 == test && cond2 == test)`
`||`	OR logico	Almeno una condizione deve essere vera	`if(cond1 == test || cond2 == test)`
`<`	Minore di	X è minore di Y	`if(X < Y)`
`>`	Maggiore di	X è maggiore di Y	`if(X > Y)`
`<=`	Minore o uguale a	X è minore o uguale a Y	`if(X <= Y)`
`>=`	Maggiore o uguale a	X è maggiore o uguale a Y	`if(X >= Y)`
`==`	Uguale a	X è uguale a Y	`if(X == Y)`
`!=`	Diverso da	X è diverso da Y	`if(X != Y)`

7.4 Cicli in R

R è particolarmente efficace nell'eseguire attività ripetitive. Quando dobbiamo ripetere un'operazione più volte, possiamo utilizzare un **ciclo**. I cicli eseguono un insieme di istruzioni per un numero specifico di volte o fino a quando una determinata condizione non è soddisfatta.

In R esistono tre tipi principali di cicli:

1. **Ciclo `for`**: ripete un'operazione per un numero definito di iterazioni.
2. **Ciclo `while`**: continua a eseguire le istruzioni fino a quando una condizione logica è soddisfatta.
3. **Ciclo `repeat`**: itera indefinitamente fino a quando non viene esplicitamente interrotto con un'istruzione `break`.

I cicli sono strumenti essenziali in tutti i linguaggi di programmazione, ma in R il loro utilizzo dovrebbe essere valutato attentamente, poiché spesso esistono alternative più efficienti come le funzioni della famiglia `apply`.

7.4.1 Il ciclo `for`

Il ciclo `for` è il più utilizzato per eseguire un'operazione un numero definito di volte. Ecco un esempio base:

```
for (i in 1:5) {
    print(i)
}
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 4
#> [1] 5
```

Come funziona?

- L'indice `i` prende il primo valore della sequenza `1:5` (cioè 1).
- Il corpo del ciclo, ovvero il codice tra `{ }`, viene eseguito.
- Al termine di ogni iterazione, `i` assume il valore successivo nella sequenza, e il processo si ripete fino all'ultimo valore (5 in questo caso).

Aggiungere logica nel corpo del ciclo

Possiamo aggiungere operazioni all'interno del ciclo, come ad esempio sommare 1 a ogni valore:

```r
for (i in 1:5) {
    print(i + 1)
}
#> [1] 2
#> [1] 3
#> [1] 4
#> [1] 5
#> [1] 6
```

7.4.2 Il ciclo `while`

Il ciclo `while` continua a eseguire le istruzioni fino a quando una condizione logica è soddisfatta. Ecco un esempio:

```r
i <- 0
while (i <= 4) {
    i <- i + 1
    print(i)
}
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 4
#> [1] 5
```

Come funziona?

- La condizione logica (`i <= 4`) viene verificata prima di ogni iterazione.
- Se la condizione è vera, il ciclo esegue il codice tra `{ }`.
- Quando la condizione diventa falsa (`i > 4`), il ciclo si interrompe.

7.4.3 Ciclo `repeat`

Il ciclo `repeat` esegue il codice indefinitamente, a meno che non venga interrotto con un'istruzione `break`:

```r
i <- 0
repeat {
    i <- i + 1
    print(i)
    if (i >= 5) {
        break
    }
}
#> [1] 1
#> [1] 2
```

```
#> [1] 3
#> [1] 4
#> [1] 5
```

Quando usarlo?

Il ciclo `repeat` è raro e viene utilizzato solo in situazioni molto particolari. Nella maggior parte dei casi, `for` o `while` sono più adatti.

7.4.4 Evitare i cicli: la famiglia di funzioni `apply`

I cicli in R sono relativamente lenti, specialmente con dataset di grandi dimensioni. Quando possibile, è preferibile usare funzioni della famiglia `apply` per ottenere lo stesso risultato in modo più efficiente e con meno rischi di errore.

7.4.4.1 La funzione `lapply()`

`lapply()` esegue una funzione su ciascun elemento di una lista o vettore e restituisce una lista con i risultati.

Esempio:

```
lapply(0:4, function(a) {
    a + 1
})
#> [[1]]
#> [1] 1
#>
#> [[2]]
#> [1] 2
#>
#> [[3]]
#> [1] 3
#>
#> [[4]]
#> [1] 4
#>
#> [[5]]
#> [1] 5
```

7.4.4.2 La funzione `sapply()`

`lapply()` restituisce una lista, ma se vuoi un vettore come output, usa `sapply()`:

```
sapply(0:4, function(a) {
    a + 1
```

```
})
#> [1] 1 2 3 4 5
```

7.4.5 Quando usare i cicli?

I cicli sono utili quando:

- Devi simulare modelli complessi (es. modelli ricorsivi).
- Hai bisogno di operazioni che dipendono dai risultati delle iterazioni precedenti.

In tutti gli altri casi, considera alternative come `apply()`, `lapply()` o funzioni simili per un codice più efficiente e meno soggetto a errori.

7.5 Linee Guida per Scrivere Codice

Di seguito trovi alcune linee guida per scrivere codice chiaro, conciso e riutilizzabile:

1. **Evita di ripeterti**: Segui il principio *Don't Repeat Yourself* (DRY). Scrivi funzioni e utilizza funzioni come `map` (per applicare un pezzo di codice iterativamente a tutti gli elementi di un oggetto) per evitare di copiare e incollare variazioni minime dello stesso codice in più parti del progetto.
2. **Segui uno stile coerente**: Adotta una guida di stile per mantenere uniformità nel tuo codice. Per R, raccomandiamo la guida di stile del "tidyverse", scritta da Hadley Wickham. Questa guida, derivata dalla Google R Style Guide, fornisce istruzioni dettagliate su sintassi del codice, nomi delle variabili, spaziature, indentazioni, commenti, convenzioni per scrivere funzioni, utilizzo delle pipe (metodo per concatenare funzioni), e altro ancora.
3. **Commenta abbondantemente**: Usa i commenti (ad esempio, con `#`) per spiegare perché ogni parte del codice è necessaria e cosa fa. I commenti rendono il codice più leggibile e facilitano la manutenzione futura.
4. **Testa il tuo codice**: Ogni volta che scrivi codice, verifica che funzioni come previsto. Puoi farlo scrivendo funzioni di test specifiche o controllando manualmente che l'output corrisponda alle aspettative. Abituati a pensare a eventuali *edge cases* (casi limite) in cui il tuo codice potrebbe non comportarsi come previsto.

5. **Esegui una revisione del codice**: Quando possibile, fai revisionare il tuo codice da un'altra persona per individuare errori e incoerenze. Se non hai nessuno a disposizione, puoi rivedere il tuo codice autonomamente: rileggendo con attenzione, è sorprendente il numero di errori che si possono individuare!

Seguendo queste linee guida, potrai scrivere codice più robusto, leggibile e facile da mantenere nel tempo.[1]

7.6 Riflessioni Conclusive

Scrivere funzioni è un passaggio essenziale per migliorare la leggibilità, l'efficienza e la riutilizzabilità del codice. Funzioni ben progettate semplificano le modifiche, riducono errori e rendono il lavoro più chiaro, sia per te stesso che per i collaboratori futuri. Se trovi che stai copiando e incollando codice più volte, è il momento di pensare a creare una funzione.

Le istruzioni condizionali, come `if`, `else` e `ifelse()`, sono fondamentali per introdurre logica e controllo nel codice. Permettono di gestire scenari diversi e prendere decisioni dinamiche, migliorando la flessibilità e l'efficienza dei tuoi script. Combinando queste istruzioni con operatori logici come `&&` e `||`, puoi affrontare situazioni complesse con un codice chiaro e leggibile.

I cicli sono potenti strumenti per eseguire operazioni ripetitive, ma in R il loro utilizzo dovrebbe essere limitato ai casi in cui non esistono alternative più efficienti. Le funzioni `apply()` e simili rappresentano spesso un'opzione migliore per manipolare dati in modo più rapido e leggibile.

7.7 Esercizi

Esercizio

In questo esercizio, utilizzerai R per praticare la creazione di funzioni, l'uso delle istruzioni condizionali e l'applicazione dei cicli. L'obiettivo è comprendere come scrivere codice più strutturato, riutilizzabile ed efficiente.

[1] Per un approfondimento sull'approccio frequentista alla regressione, si veda, per esempio, Caudek & Luccio (**?**).

Parte 1: Comprensione Teorica

1. **Cos'è una funzione in R?**
   - Descrivi con parole tue cosa fa una funzione e perché è utile.
2. **Sintassi delle funzioni**
   - Scrivi la struttura generale di una funzione in R.
3. **Uso di istruzioni condizionali**
   - Qual è la differenza tra `if`, `else` e `ifelse()`? Fornisci un esempio per ciascuno.
4. **Cicli in R**
   - Qual è la differenza tra `for`, `while` e `repeat`?

Parte 2: Creazione ed Esecuzione in R

5. **Creazione di una funzione per calcolare il punteggio totale SWLS**
   - Scrivi una funzione in R chiamata `calcola_SWLS()` che accetta un vettore con 5 punteggi SWLS e restituisce il totale.
6. **Condizione per determinare la soddisfazione**
   - Scrivi una funzione `valuta_soddisfazione()` che prende un punteggio SWLS totale e restituisce:
     - `"Alta soddisfazione"` se il punteggio è sopra 24.
     - `"Soddisfazione moderata"` se è tra 15 e 24.
     - `"Bassa soddisfazione"` se è inferiore a 15.
7. **Applicare una funzione a più individui**
   - Scrivi un ciclo `for` che calcola la soddisfazione per un gruppo di 5 persone e stampa il risultato.
8. **Uso di `ifelse()`**
   - Usa `ifelse()` per determinare rapidamente se i punteggi di 5 individui indicano soddisfazione alta (`> 24`) o bassa (24).
9. **Ciclo `while` per controllare input**
   - Scrivi un ciclo `while` che continua a chiedere all'utente di inserire un punteggio SWLS fino a quando non inserisce un valore valido (compreso tra 5 e 35).
10. **Esportazione dei dati**

- Salva in un file CSV `"swls_results.csv"` un data frame contenente i punteggi SWLS e la valutazione della soddisfazione.

Consegna

- Scrivi le risposte della **Parte 1** su carta.
- Scrivi il codice e i risultati della **Parte 2** in un file `.R` e invialo come consegna.

Soluzione

Parte 1: Comprensione Teorica

1. **Cos'è una funzione in R?**
   - Una funzione è un blocco di codice che esegue un'operazione specifica. Permette di scrivere codice riutilizzabile e più organizzato.
2. **Sintassi delle funzioni**

```
nome_funzione <- function(argomenti) {
  # Corpo della funzione
  return(risultato)
}
```

3. **Uso di istruzioni condizionali**
   - `if`: Controlla una condizione e esegue codice solo se è vera.

```
if (x > 10) { print("Maggiore di 10") }
```

   - `else`: Esegue codice alternativo se la condizione è falsa.

```
if (x > 10) { print("Maggiore di 10") } else { print("10 o meno") }
```

   - `ifelse()`: Alternativa vettorializzata a `if`.

```
y <- ifelse(x > 10, "Alto", "Basso")
```

4. **Cicli in R**
   - `for`: Itera su una sequenza.

```
for (i in 1:5) { print(i) }
```

   - `while`: Continua fino a quando una condizione è vera.

```
i <- 1
while (i <= 5) { print(i); i <- i + 1 }
```

   - `repeat`: Ripete fino a un `break`.

```
i <- 1
repeat { print(i); i <- i + 1; if (i > 5) break }
```

Parte 2: Creazione ed Esecuzione in R

5. **Creazione della funzione per il punteggio totale SWLS**

```r
calcola_SWLS <- function(punteggi) {
  return(sum(punteggi))
}
```

6. **Condizione per determinare la soddisfazione**

```r
valuta_soddisfazione <- function(score) {
  if (score > 24) {
    return("Alta soddisfazione")
  } else if (score >= 15) {
    return("Soddisfazione moderata")
  } else {
    return("Bassa soddisfazione")
  }
}
```

7. **Applicazione della funzione a più individui**

```r
punteggi_lista <- list(c(25, 27, 22, 24, 28), c(18, 20, 17, 16, 19))
for (punteggi in punteggi_lista) {
  print(valuta_soddisfazione(calcola_SWLS(punteggi)))
}
```

8. **Uso di `ifelse()`**

```r
punteggi_totali <- c(28, 19, 15, 10, 25)
soddisfazione <- ifelse(punteggi_totali > 24, "Alta", "Bassa")
print(soddisfazione)
```

9. **Ciclo `while` per controllare input**

```r
score <- 0
while (score < 5 || score > 35) {
  score <- as.numeric(readline(prompt = "Inserisci un punteggio SWLS (5-35): "))
}
```

10. **Esportazione dei dati**

```r
df <- data.frame(Punteggio = punteggi_totali, Soddisfazione = soddisfazione)
write.csv(df, "swls_results.csv", row.names = FALSE)
```

Conclusione
Questi esercizi hanno mostrato come scrivere funzioni, utilizzare condizioni e cicli per strutturare meglio il codice in R.

Informazioni sull'Ambiente di Sviluppo

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] thematic_0.1.6   MetBrewer_0.2.0  ggokabeito_0.1.0 see_0.10.0
#>  [5] gridExtra_2.3    patchwork_1.3.0  bayesplot_1.11.1 psych_2.4.12
#>  [9] scales_1.3.0     markdown_1.13    knitr_1.49       lubridate_1.9.4
#> [13] forcats_1.0.0    stringr_1.5.1    dplyr_1.1.4      purrr_1.0.4
#> [17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1
#> [21] tidyverse_2.0.0  rio_1.2.3        here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] generics_0.1.3    stringi_1.8.4     lattice_0.22-6    hms_1.1.3
#>  [5] digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3    grid_4.4.2
#>  [9] timechange_0.3.0  fastmap_1.2.0     rprojroot_2.0.4   jsonlite_1.8.9
#> [13] mnormt_2.1.1      cli_3.6.4         rlang_1.1.5       munsell_0.5.1
#> [17] withr_3.0.2       tools_4.4.2       parallel_4.4.2    tzdb_0.4.0
#> [21] colorspace_2.1-1  pacman_0.5.1      vctrs_0.6.5       R6_2.6.1
#> [25] lifecycle_1.0.4   pkgconfig_2.0.3   pillar_1.10.1     gtable_0.3.6
#> [29] glue_1.8.0        xfun_0.50         tidyselect_1.2.1  rstudioapi_0.17.1
#> [33] farver_2.1.2      htmltools_0.5.8.1 nlme_3.1-167      rmarkdown_2.29
#> [37] compiler_4.4.2
```

Bibliografia

8

Pacchetti in R

Prerequisiti

- Consultare Introduction to Data Science: Data Wrangling and Visualization with R (**?**)
- Leggere R for Data Science (2e).

Concetti e competenze chiave

- Comprendere cosa sono i pacchetti in R e la loro utilità.
- Acquisire la capacità di installare e caricare pacchetti in R.

8.1 Introduzione

I pacchetti R sono estensioni del linguaggio di programmazione statistica R. Questi pacchetti forniscono una raccolta di risorse che possono essere utilizzate per ampliare le funzionalità di base di R. Ogni pacchetto generalmente include:

- **Codice**: funzioni e script scritti in R (e talvolta in altri linguaggi come C++ o Fortran) che implementano specifiche analisi o strumenti.
- **Dati**: dataset di esempio o utili per testare e dimostrare le funzionalità del pacchetto.
- **Documentazione**: file descrittivi che spiegano come utilizzare il pacchetto, spesso in formato manuale o vignette (tutorial pratici).

I pacchetti R sono distribuiti e installati attraverso repository centralizzati, il più noto dei quali è CRAN (Comprehensive R Archive Network). CRAN garantisce la qualità e l'affidabilità dei pacchetti, sottoponendoli a controlli rigorosi prima della pubblicazione.

La vasta disponibilità di pacchetti è una delle ragioni principali della popolarità di R.

8.2 Installare i Pacchetti R

Quando installi R, vengono installati automaticamente alcuni pacchetti base. Tuttavia, hai la possibilità di aggiungere ulteriori pacchetti che trovi utili per i tuoi scopi. Questi pacchetti sono memorizzati sui server di R (mirror), e l'installazione di un nuovo pacchetto richiede una connessione internet al mirror CRAN che hai scelto durante l'installazione di R.

Per installare un pacchetto, utilizza il comando:

```
install.packages("<nome_pacchetto>")
```

Sostituisci `<nome_pacchetto>` con il nome del pacchetto che desideri installare. Ad esempio, se vuoi installare il pacchetto **rio** (utile per importare i dati in R), puoi digitare:

```
install.packages("rio")   # Non dimenticare le virgolette!
```

8.3 Caricamento di un pacchetto

Ogni volta che avvii una nuova sessione di R, se desideri utilizzare un pacchetto, devi caricarlo manualmente. Questo si fa con il comando `library()`. Ad esempio, dopo aver installato **rio**, per utilizzarlo digita:

```
library(rio)   # Nota: le virgolette non sono necessarie, ma puoi usarle se preferisci.
```

8.4 Utilizzo delle funzioni di un pacchetto senza caricarlo

Se hai bisogno di utilizzare una funzione specifica di un pacchetto, ma sai che la userai solo una volta, puoi evitare di caricare l'intero pacchetto con `library()`. Ad esempio, invece di scrivere:

```
library(nome_pacchetto)
funzione_specifica(x = 2, sd = 3)
```

puoi accedere direttamente alla funzione usando l'operatore `::`, come indicato di segtuito:

```
nome_pacchetto::funzione_specifica(x = 2, sd = 3)
```

Questo approccio è utile per funzioni che usi raramente o una sola volta. Personalmente, utilizzo `::` anche quando ho già caricato il pacchetto, per ricordare ad un "me futuro" da quale pacchetto proviene una determinata funzione. Questo può rendere il codice più leggibile e comprensibile nel tempo.

Bibliografia

9

Introduzione a `dplyr`

! In questo capitolo imparerai a

- utlizzare le principali funzioni del pacchetto `dplyr`.

Prerequisiti

- Leggere R for Data Science (2e).
- Consultare Data cleaning for social scientists.
- Leggere il capitolo *Data Wrangling* di Statistical Inference via Data Science: A ModernDive into R and the Tidyverse (Second Edition).
- Consultare Introduction to Data Science: Data Wrangling and Visualization with R (**?**)

Preparazione del Notebook

```
here::here("code", "_common.R") |> source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(tidyr, mice, missForest)
```

9.1 Introduzione

L'obiettivo di questo capitolo è fornire un'introduzione alle funzioni principali del pacchetto `dplyr` per le operazioni di *data wrangling*, cioè per il preprocessing e la pulizia dei dati. In R, queste operazioni sono strettamente legate al concetto di "data tidying", che si riferisce all'organizzazione sistematica dei dati per facilitare l'analisi.

Per comprendere meglio il concetto di "data tidying", possiamo rifarci a una citazione tratta dal testo di riferimento *R for Data Science (2e)*:

> "Happy families are all alike; every unhappy family is unhappy in its own way." — Leo Tolstoy
>
> "Tidy datasets are all alike, but every messy dataset is messy in its own way." — Hadley Wickham

L'essenza del "data tidying" è organizzare i dati in un formato che sia facile da gestire e analizzare. Anche se gli stessi dati possono essere rappresentati in vari modi, non tutte le rappresentazioni sono ugualmente efficienti o facili da usare. Un dataset "tidy" segue tre principi fondamentali che lo rendono particolarmente pratico:

1. **Ogni variabile è una colonna**: ogni colonna nel dataset rappresenta una singola variabile.
2. **Ogni osservazione è una riga**: ogni riga nel dataset rappresenta un'unica osservazione.
3. **Ogni valore è una cella**: ogni cella del dataset contiene un singolo valore.

Il pacchetto R *{dplyr}* e gli altri pacchetti del tidyverse sono progettati specificamente per lavorare con dati in formato "tidy", permettendo agli utenti di eseguire operazioni di manipolazione e visualizzazione in modo più intuitivo ed efficiente.

9.2 Pipe

Il pacchetto `dplyr`, così come l'intero ecosistema `tidyverse`, fa largo uso dell'operatore pipe, che consente di concatenare una sequenza di operazioni in modo leggibile ed efficiente. In R, esistono due principali notazioni per il pipe:

1. `|>`: introdotto nativamente a partire dalla versione 4.1.0 di R.
2. `%>%`: introdotto dal pacchetto `magrittr`, ed è una delle componenti centrali del `tidyverse`.

Entrambi gli operatori permettono di ottenere risultati simili e, per la maggior parte degli utilizzi, possono essere considerati intercambiabili. Tuttavia, è importante sottolineare alcune differenze:

- `|>` è integrato nel linguaggio R e non richiede pacchetti aggiuntivi.

- **%>%**, essendo parte di `magrittr`, richiede che il pacchetto sia installato e caricato (`library(magrittr)` o automaticamente tramite `tidyverse`).

Consideriamo l'esempio seguente (che anticipa l'uso della funzione `filter()` che descriveremo in seguito). Un'operazione comune è filtrare un data frame e calcolare la media di una colonna. Con il pipe, questa sequenza di operazioni diventa più leggibile:

```
# Usando %>%
iris %>%
  dplyr::filter(Species == "setosa") |>
  summarise(
    mean_sepal_length = mean(Sepal.Length)
  )
#>   mean_sepal_length
#> 1             5.006
```

```
# Usando |>
iris |>
  dplyr::filter(Species == "setosa") |>
  summarise(
    mean_sepal_length = mean(Sepal.Length)
  )
#>   mean_sepal_length
#> 1             5.006
```

9.2.1 Cosa Fa la Pipe?

La pipe è uno strumento potente che permette di collegare in modo diretto l'output di una funzione come input della funzione successiva. Questo approccio:

- Riduce la necessità di creare variabili intermedie.
- Migliora la leggibilità del codice.
- Rende il flusso delle operazioni più chiaro e lineare.

Ogni funzione applicata con la pipe riceve automaticamente l'output della funzione precedente come suo primo argomento. Ciò consente di scrivere sequenze di operazioni in un formato compatto e intuitivo.

Ecco un altro esempio:

```
# Utilizzo della pipe per trasformare un dataset
df <- data.frame(
  id = 1:5,
  value = c(10, 20, 30, 40, 50)
)
```

```
# Filtra i dati, seleziona colonne e calcola nuovi valori
df_clean <- df |>
  dplyr::filter(value > 20) |>
  dplyr::select(id, value) |>
  mutate(squared_value = value^2)
```

In questa sequenza, il dataset originale `df` viene filtrato, le colonne desiderate vengono selezionate e viene aggiunta una nuova colonna con il valore al quadrato.

```
head(df_clean)
#>   id value squared_value
#> 1  3    30           900
#> 2  4    40          1600
#> 3  5    50          2500
```

In sintesi, la pipe è uno strumento fondamentale per scrivere codice R moderno e leggibile, indipendentemente dal fatto che si utilizzi `|>` o `%>%`.

9.3 Verbi

Le funzioni principali ("verbi) di `dplyr` sono le seguenti:

Verbo dplyr	Descrizione
`select()`	Seleziona colonne
`filter()`	Filtra righe
`arrange()`	Riordina o organizza le righe
`mutate()`	Crea nuove colonne
`summarise()`	Riassume i valori
`group_by()`	Consente di eseguire operazioni di gruppo

I verbi di `dplyr` sono suddivisi in quattro gruppi, in base all'elemento su cui operano: righe, colonne, gruppi o tabelle.

Inoltre, le diverse funzioni `bind_` e `_joins` permettono di combinare più tibbles (ovvero, data frame) in uno solo.

Per fare un esempio prarico, usiamo nuovamente il dataset `msleep`.

```
data(msleep)
dim(msleep)
#> [1] 83 11
```

Esaminiamo i dati:

```
glimpse(msleep)
#> Rows: 83
#> Columns: 11
#> $ name         <chr> "Cheetah", "Owl monkey", "Mountain beaver", "Greater ~
#> $ genus        <chr> "Acinonyx", "Aotus", "Aplodontia", "Blarina", "Bos", ~
#> $ vore         <chr> "carni", "omni", "herbi", "omni", "herbi", "herbi", "~
#> $ order        <chr> "Carnivora", "Primates", "Rodentia", "Soricomorpha", ~
#> $ conservation <chr> "lc", NA, "nt", "lc", "domesticated", NA, "vu", NA, "~
#> $ sleep_total  <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4, 8.7, 7.0, 10.1, 3.~
#> $ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2, 1.4, NA, 2.9, NA, 0.6, 0~
#> $ sleep_cycle  <dbl> NA, NA, NA, 0.1333, 0.6667, 0.7667, 0.3833, NA, 0.333~
#> $ awake        <dbl> 11.9, 7.0, 9.6, 9.1, 20.0, 9.6, 15.3, 17.0, 13.9, 21.~
#> $ brainwt      <dbl> NA, 0.01550, NA, 0.00029, 0.42300, NA, NA, NA, 0.0700~
#> $ bodywt       <dbl> 50.000, 0.480, 1.350, 0.019, 600.000, 3.850, 20.490, ~
```

Le colonne, nell'ordine, corrispondono a quanto segue:

Nome colonna	Descrizione
name	Nome comune
genus	Rango tassonomico
vore	Carnivoro, onnivoro o erbivoro?
order	Rango tassonomico
conservation	Stato di conservazione del mammifero
sleep_total	Quantità totale di sonno, in ore
sleep_rem	Sonno REM, in ore
sleep_cycle	Durata del ciclo di sonno, in ore
awake	Quantità di tempo trascorso sveglio, in ore
brainwt	Peso del cervello, in chilogrammi
bodywt	Peso corporeo, in chilogrammi

9.4 Righe

I verbi più importanti che operano sulle righe di un dataset sono `filter()`, che seleziona le righe da includere senza modificarne l'ordine, e `arrange()`, che cambia l'ordine delle righe senza alterare la selezione delle righe presenti.

```
msleep |>
  dplyr::filter(sleep_total < 4) |>
  arrange(sleep_total)
#> # A tibble: 9 x 11
```

```
#>   name             genus         vore  order          conservation
#>   <chr>            <chr>         <chr> <chr>          <chr>
#> 1 Giraffe          Giraffa       herbi Artiodactyla   cd
#> 2 Pilot whale      Globicephalus carni Cetacea        cd
#> 3 Horse            Equus         herbi Perissodactyla domesticated
#> 4 Roe deer         Capreolus     herbi Artiodactyla   lc
#> 5 Donkey           Equus         herbi Perissodactyla domesticated
#> 6 African elephant Loxodonta     herbi Proboscidea    vu
#> 7 Caspian seal     Phoca         carni Carnivora      vu
#> 8 Sheep            Ovis          herbi Artiodactyla   domesticated
#> 9 Asian elephant   Elephas       herbi Proboscidea    en
#> # i 6 more variables: sleep_total <dbl>, sleep_rem <dbl>,
#> #   sleep_cycle <dbl>, awake <dbl>, brainwt <dbl>, bodywt <dbl>
```

Possiamo usare `filter()` speficicano più di una condizione logica.

```
msleep |>
  dplyr::filter((sleep_total < 4 & bodywt > 100) | brainwt > 1) |>
  arrange(sleep_total)
#> # A tibble: 7 x 11
#>   name             genus         vore  order          conservation
#>   <chr>            <chr>         <chr> <chr>          <chr>
#> 1 Giraffe          Giraffa       herbi Artiodactyla   cd
#> 2 Pilot whale      Globicephalus carni Cetacea        cd
#> 3 Horse            Equus         herbi Perissodactyla domesticated
#> 4 Donkey           Equus         herbi Perissodactyla domesticated
#> 5 African elephant Loxodonta     herbi Proboscidea    vu
#> 6 Asian elephant   Elephas       herbi Proboscidea    en
#> 7 Human            Homo          omni  Primates       <NA>
#> # i 6 more variables: sleep_total <dbl>, sleep_rem <dbl>,
#> #   sleep_cycle <dbl>, awake <dbl>, brainwt <dbl>, bodywt <dbl>
```

9.5 Colonne

Esistono quattro verbi principali che modificano le colonne di un dataset senza cambiare le righe:

- `relocate()` cambia la posizione delle colonne;
- `rename()` modifica i nomi delle colonne;
- `select()` seleziona le colonne da includere o escludere;
- `mutate()` crea nuove colonne a partire da quelle esistenti.

```
msleep2 <- msleep |>
  mutate(
    rem_prop = sleep_rem / sleep_total * 100
  ) |>
  dplyr::select(name, vore, rem_prop, sleep_total) |>
  arrange(desc(rem_prop))

glimpse(msleep2)
#> Rows: 83
#> Columns: 4
#> $ name        <chr> "European hedgehog", "Thick-tailed opposum", "Giant ar~
#> $ vore        <chr> "omni", "carni", "insecti", "omni", "carni", "omni", "~
#> $ rem_prop    <dbl> 34.65, 34.02, 33.70, 29.21, 28.71, 27.22, 26.37, 26.21~
#> $ sleep_total <dbl> 10.1, 19.4, 18.1, 8.9, 10.1, 18.0, 9.1, 10.3, 12.5, 8.~
```

In questo esempio, utilizziamo `mutate()` per creare una nuova colonna `rem_prop` che rappresenta la percentuale di sonno REM sul totale del sonno. Successivamente, `select()` viene utilizzato per scegliere solo alcune colonne del dataset, e infine `desc(rem_prop)` ordina i valori di `rem_prop` in ordine decrescente, dal valore maggiore a quello minore.

Per cambiare il nome di una colonna possiamo usare `rename()`. Inoltre, possiamo cambiare l'ordine delle variabili con `relocate()`.

```
msleep2 |>
  rename(rem_perc = rem_prop) |>
  relocate(rem_perc, .before = name)
#> # A tibble: 83 x 4
#>    rem_perc name                  vore    sleep_total
#>       <dbl> <chr>                 <chr>         <dbl>
#>  1     34.7 European hedgehog     omni           10.1
#>  2     34.0 Thick-tailed opposum  carni          19.4
#>  3     33.7 Giant armadillo       insecti        18.1
#>  4     29.2 Tree shrew            omni            8.9
#>  5     28.7 Dog                   carni          10.1
#>  6     27.2 North American Opossum omni           18
#>  7     26.4 Pig                   omni            9.1
#>  8     26.2 Desert hedgehog       <NA>           10.3
#>  9     25.6 Domestic cat          carni          12.5
#> 10     25   Eastern american mole insecti         8.4
#> # i 73 more rows
```

9.6 Gruppi

Il verbo `group_by()` viene utilizzato per suddividere un dataset in gruppi, in base a una o più variabili, che siano rilevanti per l'analisi. Questo permette di eseguire operazioni di sintesi su ciascun gruppo separatamente, ottenendo informazioni aggregate.

Ad esempio, nel codice seguente:

```
msleep |>
  group_by(order) |>
  summarise(
    avg_sleep = mean(sleep_total),
    min_sleep = min(sleep_total),
    max_sleep = max(sleep_total),
    total = n()
  ) |>
  arrange(desc(avg_sleep))
#> # A tibble: 19 x 5
#>    order           avg_sleep min_sleep max_sleep total
#>    <chr>               <dbl>     <dbl>     <dbl> <int>
#>  1 Chiroptera          19.8       19.7      19.9     2
#>  2 Didelphimorphia     18.7       18        19.4     2
#>  3 Cingulata           17.8       17.4      18.1     2
#>  4 Afrosoricida        15.6       15.6      15.6     1
#>  5 Pilosa              14.4       14.4      14.4     1
#>  6 Rodentia            12.5        7        16.6    22
#>  7 Diprotodontia       12.4       11.1      13.7     2
#>  8 Soricomorpha        11.1        8.4      14.9     5
#>  9 Primates            10.5        8        17      12
#> 10 Erinaceomorpha      10.2       10.1      10.3     2
#> 11 Carnivora           10.1        3.5      15.8    12
#> 12 Scandentia           8.9        8.9       8.9     1
#> 13 Monotremata          8.6        8.6       8.6     1
#> 14 Lagomorpha           8.4        8.4       8.4     1
#> 15 Hyracoidea           5.67       5.3       6.3     3
#> 16 Artiodactyla         4.52       1.9       9.1     6
#> 17 Cetacea              4.5        2.7       5.6     3
#> 18 Proboscidea          3.6        3.3       3.9     2
#> 19 Perissodactyla       3.47       2.9       4.4     3
```

1. `group_by(order)` suddivide il dataset `msleep` in gruppi, ciascuno corrispondente a un valore distinto della variabile `order`.

2. Successivamente, `summarise()` calcola diverse statistiche per ogni gruppo:
    - `avg_sleep` è la media del totale del sonno (`sleep_total`) all'interno di ciascun gruppo.
    - `min_sleep` è il valore minimo di `sleep_total` in ogni gruppo.
    - `max_sleep` è il valore massimo di `sleep_total` in ogni gruppo.
    - `total` è il numero di osservazioni (o righe) per ciascun gruppo, calcolato con la funzione `n()`.
3. Infine, `arrange(desc(avg_sleep))` ordina i risultati in ordine decrescente in base alla media del sonno totale (`avg_sleep`), mostrando prima i gruppi con la media di sonno più alta.

Questo tipo di approccio è utile quando si vuole analizzare come cambiano le caratteristiche dei dati a seconda dei gruppi specifici, fornendo una visione più dettagliata e utile.

9.7 Considerazioni Conclusive

Il *data wrangling* è una delle fasi più importanti in qualsiasi pipeline di analisi dei dati. In questo capitolo abbiamo introdotto l'uso del pacchetto `tidyverse` di R per la manipolazione dei dati e il suo utilizzo in scenari di base. Tuttavia, il tidyverse è un ecosistema ampio e qui abbiamo trattato solo gli elementi fondamentali. Per approfondire, si consiglia di consultare ulteriori risorse come quelle disponibili sul sito web del tidyverse e il libro *R for Data Science (2e)*, di cui esiste anche una traduzione italiana.

9.8 Esercizi

Esercizio

In questo esercizio, utilizzerai il pacchetto `dplyr` per imparare a manipolare e trasformare i dati della SWLS (Satisfaction With Life Scale). Gli esercizi ti aiuteranno a consolidare la conoscenza dei principali verbi di `dplyr`, inclusi `filter()`, `select()`, `mutate()`, `arrange()` e `group_by()`.

Parte 1: Comprensione Teorica

1. **Cos'è un dataset "tidy"?**
   - Descrivi con parole tue cosa significa avere un dataset "tidy" e quali sono le sue tre caratteristiche principali.
2. **Cos'è la pipe (`%>%` o `|>`) e perché è utile?**
   - Spiega a cosa serve l'operatore pipe e fornisci un esempio di utilizzo.
3. **Quali sono i verbi principali di `dplyr`?**
   - Elenca e spiega brevemente i sei verbi principali di `dplyr` per la manipolazione dei dati.
4. **Cosa fa il verbo `group_by()`?**
   - Spiega il suo scopo e come viene utilizzato in combinazione con `summarise()`.

Parte 2: Applicazione Pratica con i Dati SWLS

5. **Caricamento dei dati SWLS**
   - Crea un data frame in R contenente i punteggi SWLS che hai raccolto.
6. **Selezione delle colonne**
   - Usa `select()` per mantenere solo le colonne con i punteggi degli item.
7. **Filtraggio dei dati**
   - Usa `filter()` per selezionare solo gli individui che hanno un punteggio totale superiore a 20.
8. **Creazione di una nuova colonna**
   - Usa `mutate()` per calcolare il punteggio totale della SWLS per ciascun individuo e salvarlo in una nuova colonna chiamata `punteggio_totale`.
9. **Riordinamento dei dati**
   - Usa `arrange()` per ordinare il dataset in base al punteggio totale, dal più alto al più basso.
10. **Raggruppamento e sintesi dei dati**

- Usa `group_by()` e `summarise()` per calcolare la media e la deviazione standard del punteggio SWLS totale nel dataset.

Consegna

- Scrivi le risposte della **Parte 1** su carta.
- Scrivi il codice e i risultati della **Parte 2** in un file `.R` e invialo come consegna.

Soluzione

Parte 1: Comprensione Teorica

1. **Cos'è un dataset "tidy"?**
   - Un dataset "tidy" è un dataset organizzato in modo sistematico per facilitare l'analisi. Le sue tre caratteristiche principali sono:
     1. Ogni variabile è una colonna.
     2. Ogni osservazione è una riga.
     3. Ogni valore è una cella.
2. **Cos'è la pipe (`%>%` o `|>`) e perché è utile?**
   - La pipe (`%>%` o `|>`) permette di concatenare più operazioni di manipolazione dati in modo leggibile ed efficiente.
   - Esempio:

```
df |>
  filter(score > 20) |>
  select(name, score)
```

3. **Quali sono i verbi principali di `dplyr`?**
   - `select()`: Seleziona colonne.
   - `filter()`: Filtra righe.
   - `arrange()`: Riordina le righe.
   - `mutate()`: Crea nuove colonne.
   - `summarise()`: Riassume i dati.
   - `group_by()`: Permette di raggruppare i dati.
4. **Cosa fa il verbo `group_by()`?**
   - `group_by()` suddivide i dati in gruppi, permettendo di applicare funzioni di aggregazione con `summarise()`.
   - Esempio:

```
df |>
  group_by(gruppo) |>
  summarise(media = mean(score), sd = sd(score))
```

Parte 2: Applicazione Pratica con i Dati SWLS

5. **Caricamento dei dati SWLS** Per svolgere l'esercizio, simuliamo i dati di 10 individui su 5 item (numeri casuali da 1 a 7):

```
set.seed(123)
swls <- data.frame(
  id = 1:10,
  item1 = sample(1:7, 10, replace = TRUE),
  item2 = sample(1:7, 10, replace = TRUE),
  item3 = sample(1:7, 10, replace = TRUE),
  item4 = sample(1:7, 10, replace = TRUE),
  item5 = sample(1:7, 10, replace = TRUE)
)
print(swls)
```

6. **Selezione delle colonne**

```
swls_selected <- swls |> select(item1:item5)
```

7. **Filtraggio dei dati**

```
swls_filtered <- swls |> filter(rowSums(select(swls, item1:item5)) > 20)
```

8. **Creazione di una nuova colonna**

```
swls <- swls |> mutate(punteggio_totale = rowSums(select(swls, item1:item5)))
```

9. **Riordinamento dei dati**

```
swls_sorted <- swls |> arrange(desc(punteggio_totale))
```

10. **Raggruppamento e sintesi dei dati**

```
swls_summary <- swls |>
  summarise(media = mean(punteggio_totale), sd = sd(punteggio_totale))
```

Conclusione
Questi esercizi hanno mostrato come usare `dplyr` per manipolare dati in modo efficace e leggibile.

Informazioni sull'Ambiente di Sviluppo

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
```

```
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] missForest_1.5   mice_3.17.0      thematic_0.1.6   MetBrewer_0.2.0
#>  [5] ggokabeito_0.1.0 see_0.10.0       gridExtra_2.3    patchwork_1.3.0
#>  [9] bayesplot_1.11.1 psych_2.4.12     scales_1.3.0     markdown_1.13
#> [13] knitr_1.49       lubridate_1.9.4  forcats_1.0.0    stringr_1.5.1
#> [17] dplyr_1.1.4      purrr_1.0.4      readr_2.1.5      tidyr_1.3.1
#> [21] tibble_3.2.1     ggplot2_3.5.1    tidyverse_2.0.0  rio_1.2.3
#> [25] here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] tidyselect_1.2.1     farver_2.1.2         fastmap_1.2.0
#>  [4] pacman_0.5.1         digest_0.6.37        rpart_4.1.24
#>  [7] timechange_0.3.0     lifecycle_1.0.4      survival_3.8-3
#> [10] magrittr_2.0.3       compiler_4.4.2       rngtools_1.5.2
#> [13] rlang_1.1.5          tools_4.4.2          utf8_1.2.4
#> [16] doRNG_1.8.6.1        mnormt_2.1.1         withr_3.0.2
#> [19] itertools_0.1-3      nnet_7.3-20          grid_4.4.2
#> [22] jomo_2.7-6           colorspace_2.1-1     iterators_1.0.14
#> [25] MASS_7.3-64          cli_3.6.4            rmarkdown_2.29
#> [28] reformulas_0.4.0     generics_0.1.3       rstudioapi_0.17.1
#> [31] tzdb_0.4.0           minqa_1.2.8          splines_4.4.2
#> [34] parallel_4.4.2       vctrs_0.6.5          boot_1.3-31
#> [37] glmnet_4.1-8         Matrix_1.7-2         jsonlite_1.8.9
#> [40] hms_1.1.3            mitml_0.4-5          foreach_1.5.2
#> [43] glue_1.8.0           nloptr_2.1.1         pan_1.9
#> [46] codetools_0.2-20     stringi_1.8.4        shape_1.4.6.1
#> [49] gtable_0.3.6         lme4_1.1-36          munsell_0.5.1
#> [52] pillar_1.10.1        htmltools_0.5.8.1    randomForest_4.7-1.2
#> [55] R6_2.6.1             Rdpack_2.6.2         rprojroot_2.0.4
```

```
#> [58] evaluate_1.0.3       lattice_0.22-6       rbibutils_2.3
#> [61] backports_1.5.0      broom_1.0.7          Rcpp_1.0.14
#> [64] nlme_3.1-167         xfun_0.50            pkgconfig_2.0.3
```

Bibliografia

10

Quarto

Prerequisiti

- Leggere *Reproducibility in the Classroom* (**?**).
- Leggere An Introduction to R.
- Leggere R for Data Science (2e).

Concetti e competenze chiave

Preparazione del Notebook

```
here::here("code", "_common.R") |> source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(tidyr)
```

10.1 Introduzione

La *crisi della riproducibilità* scientifica rappresenta una delle sfide più importanti della ricerca contemporanea. Con questo termine ci si riferisce alla difficoltà, riscontrata in diverse discipline, di replicare i risultati degli studi scientifici. Sebbene le definizioni di riproducibilità varino tra i diversi ambiti, un'interpretazione ampiamente condivisa la identifica come la capacità di ottenere gli stessi risultati utilizzando i medesimi dati di input e seguendo gli stessi passaggi computazionali nei metodi e nelle analisi.

La pratica scientifica è profondamente radicata nella formazione accademica: ciò che viene insegnato nelle aule universitarie si riflette direttamente nel lavoro svolto nei laboratori, sul campo e nell'analisi dei dati. Riconoscendo questo stretto legame tra didattica e ricerca, molti studiosi sostengono l'importanza di integrare i metodi di riproducibilità nei corsi universitari di data science, sia a livello undergraduate che graduate (**?**). L'educazione alla data science che incorpora la riproducibilità nell'analisi dei dati viene infatti considerata la "controffensiva statistica" alla crisi della riproducibilità.

In questo contesto si inserisce Quarto, uno strumento innovativo che affronta direttamente le sfide della crisi della riproducibilità. Quarto si colloca nella tradizione del *literate programming*, un approccio pioneristico introdotto da Donald Knuth negli anni '80. Questa metodologia nasce dalla visione di unificare codice e testo descrittivo in un unico documento, rendendo i programmi non solo eseguibili ma anche comprensibili agli esseri umani. L'obiettivo è superare la tradizionale separazione tra codice e documentazione, permettendo di spiegare non solo il funzionamento tecnico di un programma, ma anche le ragioni delle scelte implementative.

Questa filosofia risulta particolarmente pertinente nell'ambito della data science e dell'analisi statistica, dove riproducibilità e trasparenza sono requisiti imprescindibili. Quarto eccelle in questo contesto, offrendo la possibilità di integrare in codice, risultati e narrazione. La sua versatilità si manifesta nella capacità di produrre diversi tipi di output - dai report agli articoli scientifici, dalle presentazioni ai documenti tecnici - in vari formati come HTML, PDF e Word, combinando efficacemente testo interpretativo, risultati numerici e visualizzazioni grafiche.

Un punto di forza distintivo di Quarto è la sua flessibilità nel supportare molteplici linguaggi di programmazione, tra cui R, Python e Julia. Lo strumento può essere utilizzato secondo tre modalità principali: per presentare conclusioni condividendo i risultati senza esporre il codice sottostante; per documentare il processo analitico includendo sia il codice che i risultati, garantendo così piena trasparenza e riproducibilità; e per annotare l'analisi, integrando interpretazioni e motivazioni delle decisioni prese durante il processo analitico.

Nonostante Quarto sia tecnicamente uno strumento da riga di comando (CLI), l'integrazione con RStudio ne semplifica notevolmente l'utilizzo, rendendo l'installazione e l'operatività accessibili anche agli utenti meno esperti nell'uso del terminale. Questa caratteristica, unita alle sue potenti funzionalità, rende Quarto una naturale evoluzione del literate programming, offrendo un ambiente di lavoro che bilancia efficacemente praticità d'uso e rigore scientifico. In questo modo, Quarto si configura come una risposta concreta alle sfide della riproducibilità nella ricerca contemporanea, fornendo gli strumenti necessari per una scienza più trasparente e verificabile. L'obiettivo di questo capitolo è quello di fornire un'introduzione pratica a Quarto.

10.1.1 Creare un documento Quarto

Un file Quarto ha estensione `.qmd` e segue questa struttura:

````
---
title: "ggplot2 demo"
author: "Norah Jones"
date: "5/22/2021"
format:
  html:
    code-fold: true
---

## Air Quality

@fig-airquality further explores the impact of temperature on ozone level.

```{r}
#| label: fig-airquality
#| fig-cap: "Temperature and ozone level."
#| warning: false

library(ggplot2)

ggplot(airquality, aes(Temp, Ozone)) +
  geom_point() +
  geom_smooth(method = "loess")
```
````

Questo file include:

1. Un'intestazione YAML (metadati del documento).
2. Blocchi di codice delimitati da "‘.
3. Testo scritto in Markdown con formattazioni semplici come titoli (`# Titolo`), corsivi (`*testo*`), ecc.

10.1.2 Editor visivo e sorgente

- **Editor visivo**: simile a Google Docs, offre un'interfaccia WYSIWYM (What You See Is What You Mean). Consente di inserire facilmente immagini, tabelle, citazioni e altro.
- **Editor sorgente**: consente un controllo diretto sul Markdown, utile per debug e personalizzazioni avanzate.

10.1.3 Blocchi di codice

I blocchi di codice (chiamati "chunks") eseguono codice e visualizzano i risultati. Ogni chunk è delimitato da "‘ e può includere opzioni specifiche:

```
#| label: esempio
#| echo: false
1 + 1
```

Le opzioni più comuni includono:

- `echo: false` (nasconde il codice nel report),
- `eval: false` (non esegue il codice),
- `message: false` e `warning: false` (nasconde messaggi o avvisi).

10.1.4 Figure

Le figure possono essere generate tramite codice (es. `ggplot()`) o inserite come file esterni. Le opzioni più comuni per il controllo delle dimensioni sono:

- `fig-width` e `fig-height` (dimensioni della figura in pollici),
- `out-width` (percentuale di larghezza del documento),
- `fig-asp` (rapporto d'aspetto, es. 0.618 per il rapporto aureo).

Esempio:

```
#| fig-width: 6
ggplot(data, aes(x, y)) + geom_point()
```

10.1.5 Tabelle

Le tabelle possono essere stampate direttamente o personalizzate con funzioni come `knitr::kable()` o pacchetti come `gt`:

```
knitr::kable(head(mtcars))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

10.1.6 Caching

Per velocizzare i documenti con calcoli complessi, Quarto supporta la memorizzazione dei risultati:

- `cache: true` salva i risultati di un chunk, evitando di ricalcolarli se il codice non cambia.
- `dependson` specifica dipendenze tra chunk.

10.1.7 Gestione delle Citazioni e delle Bibliografie in Quarto

Quarto offre un supporto avanzato per la generazione automatica di citazioni e bibliografie, consentendo l'applicazione di formati personalizzati come lo stile APA. Per includere riferimenti bibliografici, è necessario creare un file `.bib` (ad esempio, `references.bib`) contenente le citazioni nel formato BibTeX. Queste citazioni possono essere ottenute direttamente da Google Scholar o altri database accademici.

Ecco un esempio di una citazione in formato BibTeX:

```
@article{ceccarini2024age,
  title={Age-dependent changes in the anger superiority effect: Evidence from a visual sea
  author={Ceccarini, Francesco and Colpizzi, Ilaria and Caudek, Corrado},
  journal={Psychonomic Bulletin \& Review},
  pages={1--10},
  year={2024},
  publisher={Springer}
}
```

Questa citazione deve essere inserita in un file `.bib`, ad esempio, `references.bib`. Tale file dovrà poi essere specificato nell'intestazione del documento Quarto.

10.1.7.1 Configurazione dell'Intestazione YAML

Nel file `.qmd`, è necessario aggiungere le seguenti righe all'intestazione YAML per collegare il file `references.bib` e configurare lo stile della bibliografia:

```
bibliography: references.bib
biblio-style: apalike
csl: apa.csl
```

- **`bibliography`**: Specifica il percorso del file `.bib`. In questo esempio, si assume che il file si trovi nella stessa cartella del documento Quarto.
- **`biblio-style`**: Imposta lo stile delle citazioni. Ad esempio, `apalike` è uno stile simile allo stile APA.
- **`csl`**: Consente di utilizzare uno stile di citazione personalizzato, come `apa.csl`. Puoi scaricare facilmente questi stili dal Zotero Style Repository.

10.1.7.2 Esempio Completo

Di seguito è riportato un esempio completo di un documento Quarto che include una citazione e genera automaticamente la bibliografia:

```
---
title: "Articolo di Esempio"
author: "Autore di Esempio"
```

```
date: "2025-02-22"
bibliography: references.bib
biblio-style: apalike
csl: apa.csl
---

## Introduzione

In questo articolo, discutiamo i cambiamenti dipendenti dall'età nell'anger-superiority ef

## Risultati

I risultati mostrano che...

## Riferimenti
```

In questo esempio, l'identificatore `@ceccarini2024age` viene utilizzato per fare riferimento alla citazione contenuta nel file `references.bib`. Al momento della compilazione, Quarto genererà automaticamente la lista dei riferimenti bibliografici in base al formato specificato.

10.1.7.3 Citazioni Inline

All'interno di un documento `.qmd`, le citazioni vengono aggiunte utilizzando il simbolo `@` seguito dall'identificativo della citazione specificato nel file `.bib`. Ad esempio:

```
... come evidenziato da @ceccarini2024age, si osserva che...
```

Quarto genera automaticamente la bibliografia, includendo solo i riferimenti effettivamente citati nel documento. La bibliografia viene aggiunta alla fine del file renderizzato (ad esempio, in formato HTML o PDF).

Ad esempio, nel caso di un documento `.qmd`, il testo sopra sarà visualizzato così:

… come evidenziato da Ceccarini et al. (2024), si osserva che…

La citazione completa sarà inclusa automaticamente nella bibliografia, posizionata alla fine della pagina web o del documento finale. Si noti che Quarto gestisce automaticamente la formattazione e la posizione della bibliografia, garantendo coerenza e precisione.

Esempio 10.1. Per fare un esempio pratico, possiamo inserire la citazione `@ceccarini2024age` direttamente nel file `.qmd` di questa pagina web. Quando il documento viene compilato, Quarto renderà la citazione in modo appropriato, come mostrato qui: Ceccarini et al. (**?**).

Si noti che, in fondo a questa pagina web, è presente un riferimento bibliogra-

fico corrispondente. Questo riferimento è stato aggiunto automaticamente da Quarto in risposta all'uso della citazione `@ceccarini2024age` nel testo del documento. Questo processo automatizzato semplifica la gestione delle citazioni e garantisce che tutti i riferimenti siano correttamente inclusi e formattati.

10.2 Riflessioni Conclusive

Quarto è uno strumento potente per la creazione di documenti riproducibili e ben strutturati, integrando codice, risultati e testo descrittivo in un unico file. Questa introduzione dovrebbe essere sufficiente per iniziare a lavorare con Quarto, ma c'è ancora molto da imparare. Il modo migliore per rimanere aggiornati è consultare il sito ufficiale di Quarto: https://quarto.org.

Un argomento importante che non abbiamo trattato qui riguarda i dettagli di come comunicare in modo accurato le proprie idee agli altri. Per migliorare le proprie capacità di scrittura, Wickham et al. (**?**) consigliano due libri: *Style: Lessons in Clarity and Grace* di Joseph M. Williams & Joseph Bizup, e *The Sense of Structure: Writing from the Reader's Perspective* di George Gopen. Una serie di brevi articoli sulla scrittura sono offerti da George Gopen e sono disponibili su https://www.georgegopen.com/litigation-articles.html.

10.3 Esercizi

Esercizio

In questo esercizio esplorerai l'importanza della riproducibilità nella scienza dei dati e le funzionalità principali di Quarto.

1. Concetti di base sulla riproducibilità

1. Cos'è la *crisi della riproducibilità* e perché è rilevante nella scienza dei dati?
2. In che modo Quarto può aiutare ad affrontare la crisi della riproducibilità?
3. Spiega il concetto di *literate programming* e come si collega a Quarto.

2. Struttura di un file Quarto

4. Qual è l'estensione di un file Quarto e quali sono le sue tre sezioni principali?

5. Qual è la differenza tra *editor visivo* ed *editor sorgente* in Quarto?
6. Qual è la funzione dell'intestazione YAML in un file `.qmd`?

3. Blocchi di codice e opzioni

7. Come si scrive un blocco di codice in Quarto?
8. Quali opzioni puoi utilizzare nei blocchi di codice per controllare l'esecuzione e la visualizzazione del codice e dei risultati?
9. Scrivi un blocco di codice Quarto che calcola la media di un vettore di numeri e stampa il risultato senza mostrare il codice.

4. Figure e Tabelle

10. Quali opzioni di formattazione delle figure offre Quarto?
11. Come puoi creare una tabella formattata in Quarto usando `knitr::kable()`?

5. Citazioni e Bibliografia

12. Come si aggiunge una citazione bibliografica in Quarto?
13. Quali file devono essere inclusi per gestire una bibliografia in Quarto?
14. Scrivi un esempio di citazione in formato BibTeX e mostra come collegarla a un documento `.qmd`.

6. Considerazioni Finali

15. Quali sono i vantaggi di usare Quarto rispetto a strumenti più tradizionali come Word per la creazione di report scientifici?

Soluzione

1. Concetti di base sulla riproducibilità

1. La *crisi della riproducibilità* è il fenomeno per cui molti studi scientifici non possono essere replicati con gli stessi metodi e dati. Questo mina la fiducia nella scienza e può portare a risultati non affidabili.
2. Quarto aiuta la riproducibilità integrando codice, testo e risultati in un unico documento, rendendo più semplice verificare e riprodurre le analisi.
3. *Literate programming* è un approccio introdotto da Donald Knuth che combina codice e spiegazioni testuali nello stesso file, migliorando la comprensione e documentazione delle analisi. Quarto segue questa filosofia.

2. Struttura di un file Quarto

4. L'estensione di un file Quarto è `.qmd`. Le tre sezioni principali sono:
   - L'intestazione YAML (metadati),
   - Il codice (chunks),
   - Il testo scritto in Markdown.
5. L'*editor visivo* è un'interfaccia intuitiva simile a Google Docs, mentre l'*editor sorgente* permette di scrivere direttamente in Markdown e codice.
6. L'intestazione YAML definisce le proprietà del documento come titolo, autore, formato di output e opzioni di rendering.

3. Blocchi di codice e opzioni

7. Un blocco di codice in Quarto si scrive con tripli backtick (“‘) e un linguaggio specificato:

```
#| echo: true
print("Esempio di codice in Quarto")
```

8. Alcune opzioni utili nei blocchi di codice sono:
   - `echo: false` per nascondere il codice,
   - `eval: false` per non eseguire il codice,
   - `warning: false` e `message: false` per nascondere messaggi e avvisi.
9. Esempio di blocco di codice che calcola una media senza mostrare il codice:

```
#| echo: false
mean(c(1, 2, 3, 4, 5))
```

10.4 4. Figure e Tabelle

10. Le opzioni principali per le figure includono:
    - `fig-width` e `fig-height` per le dimensioni,
    - `out-width` per la larghezza relativa,
    - `fig-asp` per il rapporto d'aspetto.
11. Per creare una tabella formattata con `knitr::kable()`:

```
knitr::kable(head(mtcars))
```

5. Citazioni e Bibliografia

12. Le citazioni in Quarto si aggiungono usando il simbolo `@` seguito dal riferimento BibTeX (es. `@ceccarini2024age`).
13. Per gestire la bibliografia in Quarto servono:
    - Un file `.bib` con le citazioni,
    - Un'intestazione YAML che collega il file `.bib` e specifica lo stile (`csl`).
14. Esempio di citazione BibTeX e collegamento in YAML:

```
@article{ceccarini2024age,
  title={Age-dependent changes in the anger superiority effect},
  author={Ceccarini, Francesco et al.},
  journal={Psychonomic Bulletin & Review},
  year={2024}
}
```

YAML:

```
bibliography: references.bib
biblio-style: apalike
```

6. Considerazioni Finali

15. I vantaggi di Quarto rispetto a Word includono:
    - maggiore riproducibilità e trasparenza,
    - possibilità di integrare codice ed esecuzione in un unico documento,
    - facilità di gestione delle citazioni automatiche,
    - supporto per diversi formati di output (HTML, PDF, Word).

Informazioni sull'Ambiente di Sviluppo

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
```

```
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] thematic_0.1.6   MetBrewer_0.2.0  ggokabeito_0.1.0 see_0.10.0
#>  [5] gridExtra_2.3    patchwork_1.3.0  bayesplot_1.11.1 psych_2.4.12
#>  [9] scales_1.3.0     markdown_1.13    knitr_1.49       lubridate_1.9.4
#> [13] forcats_1.0.0    stringr_1.5.1    dplyr_1.1.4      purrr_1.0.4
#> [17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1
#> [21] tidyverse_2.0.0  rio_1.2.3        here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] generics_0.1.3    stringi_1.8.4     lattice_0.22-6    hms_1.1.3
#>  [5] digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3    grid_4.4.2
#>  [9] timechange_0.3.0  fastmap_1.2.0     rprojroot_2.0.4   jsonlite_1.8.9
#> [13] mnormt_2.1.1      cli_3.6.4         rlang_1.1.5       munsell_0.5.1
#> [17] withr_3.0.2       yaml_2.3.10       tools_4.4.2       parallel_4.4.2
#> [21] tzdb_0.4.0        colorspace_2.1-1  pacman_0.5.1      vctrs_0.6.5
#> [25] R6_2.6.1          lifecycle_1.0.4   pkgconfig_2.0.3   pillar_1.10.1
#> [29] gtable_0.3.6      glue_1.8.0        xfun_0.50         tidyselect_1.2.1
#> [33] rstudioapi_0.17.1 farver_2.1.2      htmltools_0.5.8.1 nlme_3.1-167
#> [37] rmarkdown_2.29    compiler_4.4.2
```

Bibliografia

11

L'ambiente di programmazione in R

! In questo capitolo imparerai a

- le nozioni di base dell'ambiente R.

Prerequisiti

- Leggere il capitolo 3, "Setting Up Your Data Science Project", del libro Veridical Data Science (**?**).
- Leggere il Capitolo **??** della dispensa.

Preparazione del Notebook

```
here::here("code", "_common.R") |>
  source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(tidyr)
```

11.1 Introduzione

Programmare presuppone necessariamente un ambiente di programmazione. Una cattiva gestione di questo ambiente può causare il malfunzionamento degli script, rendendo fondamentale imparare a gestirlo correttamente.

Nota: In R, il termine "ambiente" ha un significato specifico, riferendosi allo spazio di lavoro in cui gli oggetti vengono memorizzati durante una sessione. In questo capitolo, tuttavia, il termine si riferisce allo stato complessivo del tuo computer mentre programmi, inclusa l'organizzazione dei file, la versione di R che stai usando e altre impostazioni.

L'ambiente può influenzare persino il comportamento delle funzioni più basilari. Considera il seguente esempio:

```
print(1:9)
```

Questo potrebbe produrre il seguente output:

```
[1] 1 2 3 4 5 6 7 8 9
```

Tuttavia, modificando l'opzione di larghezza in R, il comportamento cambia:

```
# Dopo aver modificato options(width)
options(width = 10)
print(1:9)
```

In questo caso, l'output potrebbe essere:

```
[1] 1 2 3
[4] 4 5 6
[7] 7 8 9
```

La differenza è dovuta a un'opzione dell'ambiente, `width`, che regola il numero massimo di caratteri visualizzati per ogni riga.

11.2 File system

Prima di iniziare a organizzare un progetto in R, è fondamentale seguire alcune linee guida per strutturare e nominare i file in modo efficace. Spesso si tende a sottovalutare l'importanza di una buona organizzazione, ma adottare un sistema coerente può far risparmiare tempo prezioso nella ricerca e gestione dei progetti passati. Danielle Navarro ha creato una presentazione sulla struttura dei progetti, nella quale propone tre principi fondamentali per la gestione dei file:

- essere gentili con le macchine;
- essere gentili con gli esseri umani;
- facilitare l'ordinamento e la ricerca.

11.2.1 Essere gentili con le macchine

Le macchine possono confondersi con spazi, caratteri speciali (come `^.*?+|$"`), e lettere accentate. Per evitare problemi:

- usa solo lettere minuscole, numeri, trattini `_` o `-`;
- evita caratteri speciali e spazi nei nomi dei file;
- evita le lettere accentate;

- usa estensioni coerenti, come `.R` per gli script R.

Esempi:

```
# Buono
progetto01_analisi_dati.R

# Cattivo
Progetto "Analisi Dati".R
```

11.2.2 Essere gentili con gli umani

Gli esseri umani hanno bisogno di contesto. Evita nomi vaghi e usa descrizioni significative.

```
# Buono
analisi01_statistiche_descrittive.R
note02_intro_modello.docx

# Cattivo
01.R
appunti.docx
```

> **!** Importante
>
> Evitate categoricamente l'uso di spazi nei nomi di file, cartelle o oggetti in R. Anche se il sistema operativo potrebbe consentirlo, questa pratica può generare problemi futuri, complicare il debugging e rendere il codice meno leggibile e portabile. Per evitare questi inconvenienti, adottate sempre nomi privi di spazi, preferendo separatori come trattini bassi (_) o trattini (-).

11.2.3 Facilitare l'ordinamento e la ricerca

Se i nomi dei file includono date, usa sempre il formato `YYYY-MM-DD` per permettere un ordinamento automatico.

```
# Buono
2024-01-01_analisi.R
2024-02-15_riassunto.docx

# Cattivo
1-gennaio-2024.R
riassunto-15-02-2024.docx
```

Se devi ordinare i file in base a qualcosa di diverso dalle date, usa numeri con lo zero iniziale per mantenere l'ordine.

```
reading01_shakespeare_romeo-and-juliet.docx
reading02_shakespeare_romeo-and-juliet.docx
...
reading11_shakespeare_romeo-and-juliet.docx
notes01_shakespeare_romeo-and-juliet.docx
...
```

11.3 Versioni di R e pacchetti

Aggiornare regolarmente R e i pacchetti è essenziale per evitare bug e sfruttare le nuove funzionalità. Ecco alcune buone pratiche:

1. Esegui `update.packages()` ogni poche settimane per aggiornare i pacchetti.
2. Aggiorna la versione di R ogni pochi mesi. Su Windows puoi usare il pacchetto `installr`, mentre su altri sistemi puoi scaricare l'ultima versione dal sito ufficiale di R.
3. Mantieni aggiornato anche il sistema operativo.

11.4 Progetti in R

Se hai seguito i consigli finora, avrai creato una cartella per tutti i tuoi progetti di programmazione e la tua installazione di R sarà aggiornata. Ora è il momento di organizzare i tuoi progetti in R.

11.4.1 Percorsi Assoluti e Relativi

Un percorso assoluto parte dalla directory principale del tuo computer (ad esempio, `/` su Linux/MacOS o `C:/` su Windows) e indica in modo completo e univoco la posizione di un file o di una cartella. Un percorso relativo, invece, parte dalla directory corrente del progetto o dalla directory di lavoro impostata e descrive la posizione di un file in relazione a questa.

Ad esempio, il percorso assoluto del file utilizzato per generare questa pagina HTML potrebbe essere ottenuto così:

```
fs::path_abs("05_environment.qmd")
#> /Users/corrado/_repositories/psicometria-r/chapters/R/05_environment.qmd
```

11.5 Funzione `here()`

Se il progetto si trova nella cartella `psicometria-r`, possiamo utilizzare la funzione `here()` del pacchetto **here** per indicare la posizione del file `05_environment.qmd` in modo relativo. Ecco un esempio:

```
file.exists(here::here("chapters", "R", "07_environment.qmd"))
#> [1] TRUE
```

In questo caso, il file `07_environment.qmd` è contenuto nella cartella `chapters/R`, che si trova all'interno della directory principale del progetto. Grazie a `here()`, non è necessario specificare manualmente la posizione del progetto: questa funzione identifica automaticamente la directory principale e consente di indicare solo il percorso relativo del file rispetto ad essa. In altre parole, puoi riferirti al file `07_environment.qmd` semplicemente fornendo il percorso relativo all'interno della struttura del progetto, lasciando a `here()` il compito di gestire il contesto globale.

11.5.1 Perché preferire i percorsi relativi?

L'utilizzo di percorsi relativi con `here()` offre numerosi vantaggi:

1. **Portabilità**: Il codice diventa più semplice da condividere, poiché non dipende dalla struttura delle directory specifica del computer su cui è stato scritto.
2. **Organizzazione**: Favorisce una struttura chiara e coerente all'interno del progetto, rendendo più facile individuare e accedere ai file.
3. **Affidabilità**: Riduce il rischio di errori dovuti a percorsi assoluti errati, soprattutto quando il progetto viene spostato o condiviso.

11.5.2 Buone pratiche

- **Usare sempre percorsi relativi**: Questo assicura che il progetto sia facilmente eseguibile su altri sistemi senza necessità di modifiche ai percorsi.
- **Impostare una struttura coerente del progetto**: Organizzare i file in cartelle ben definite (ad esempio, `data`, `scripts`, `outputs`) facilita l'uso di percorsi relativi.

In sintesi, specificare i percorsi relativi rispetto alla directory principale del progetto è una buona pratica essenziale per garantire portabilità, organizzazione e riproducibilità del lavoro.

11.5.3 Creare un progetto in R

Un progetto R è semplicemente una cartella con un file `.Rproj`. Puoi crearne uno con RStudio o con il pacchetto `usethis`.

In RStudio:

1. Vai su `File > New Project`.
2. Seleziona `New Directory > New Project`.
3. Dai un nome al progetto e scegli la sua posizione.

Con `usethis`:

```
usethis::create_project("path/alla/cartella")
```

Esempio:

```
usethis::create_project("/Users/corrado/_repositories/psicometria-r")
```

Vedremo nel Capitolo **??** come organizzare i file all'interno di un progetto.

Informazioni sull'Ambiente di Sviluppo

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
```

```
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] thematic_0.1.6   MetBrewer_0.2.0  ggokabeito_0.1.0 see_0.10.0
#>  [5] gridExtra_2.3    patchwork_1.3.0  bayesplot_1.11.1 psych_2.4.12
#>  [9] scales_1.3.0     markdown_1.13    knitr_1.49       lubridate_1.9.4
#> [13] forcats_1.0.0    stringr_1.5.1    dplyr_1.1.4      purrr_1.0.4
#> [17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1
#> [21] tidyverse_2.0.0  rio_1.2.3        here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] generics_0.1.3    stringi_1.8.4     lattice_0.22-6    hms_1.1.3
#>  [5] digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3    grid_4.4.2
#>  [9] timechange_0.3.0  fastmap_1.2.0     rprojroot_2.0.4   jsonlite_1.8.9
#> [13] mnormt_2.1.1      cli_3.6.4         crayon_1.5.3      rlang_1.1.5
#> [17] munsell_0.5.1     withr_3.0.2       yaml_2.3.10       tools_4.4.2
#> [21] parallel_4.4.2    tzdb_0.4.0        colorspace_2.1-1  pacman_0.5.1
#> [25] vctrs_0.6.5       R6_2.6.1          lifecycle_1.0.4   fs_1.6.5
#> [29] pkgconfig_2.0.3   pillar_1.10.1     gtable_0.3.6      glue_1.8.0
#> [33] xfun_0.50         tidyselect_1.2.1  rstudioapi_0.17.1 farver_2.1.2
#> [37] htmltools_0.5.8.1 nlme_3.1-167      rmarkdown_2.29    compiler_4.4.2
```

Bibliografia

12

Utilizzo di strumenti AI per la programmazione

12.1 Introduzione

Il panorama della programmazione sta attraversando una trasformazione radicale, guidata dall'avvento degli strumenti di intelligenza artificiale (AI). Questi assistenti innovativi stanno rivoluzionando il modo in cui sviluppatori, ricercatori e studenti scrivono, comprendono e ottimizzano il codice. Grazie a una nuova generazione di strumenti che vanno oltre i tradizionali ambienti di sviluppo, l'intelligenza artificiale sta aprendo possibilità inedite. Piattaforme come **ChatGPT**, **Google Gemini**, **Claude.ai**, **DeepSeek** e **Qwen** sono in grado di generare codice, spiegare concetti complessi e fornire supporto agli sviluppatori in modi che, fino a poco tempo fa, erano impensabili.

12.2 Potenzialità e Sfide dell'AI nella Programmazione

Gli strumenti di intelligenza artificiale (AI) stanno trasformando radicalmente il modo in cui si programma, inclusa l'elaborazione di codice in linguaggi come R. Tuttavia, accanto alle immense potenzialità, emergono anche sfide significative. I modelli di linguaggio di grandi dimensioni (LLM, Large Language Models) possono ottimizzare e accelerare i flussi di lavoro, ma non sono privi di limitazioni. Tra queste, la possibilità di generare codice impreciso, introdurre bias involontari o produrre output che richiedono una verifica approfondita da parte dell'utente.

Nonostante queste sfide, gli strumenti di AI offrono un supporto prezioso in diverse aree chiave:

- **Supporto Concettuale:**
  Gli LLM si dimostrano particolarmente efficaci nel rispondere a domande complesse su metodi statistici, algoritmi e tecniche di analisi dei dati. La

qualità delle risposte migliora notevolmente quando le domande sono formulate in modo chiaro, specifico e dettagliato.

- **Generazione e Completamento del Codice:**
  Questi strumenti possono aiutare gli sviluppatori a scrivere codice più rapidamente, suggerendo completamenti automatici, identificando potenziali errori e persino generando interi script a partire da descrizioni testuali. Questo riduce il tempo dedicato alla scrittura manuale e permette di concentrarsi su aspetti più creativi o complessi del progetto.

12.3 Panoramica Comparativa dei Principali Strumenti AI

Come scegliere il modello linguistico più adatto per un determinato compito? Nell'articolo di Gibney (**?**), ricercatori condividono i loro strumenti preferiti attualmente in uso, offrendo una guida pratica a chi ha bisogno di orientarsi tra le varie opzioni.

12.3.1 o3-mini (il ragionatore)

OpenAI ha lanciato **o3-mini**, un modello di ragionamento gratuito per gli utenti registrati, sviluppato in risposta alla crescente concorrenza di DeepSeek. Questo modello si distingue per l'utilizzo di un processo di **ragionamento a catena** (chain-of-thought reasoning), che gli permette di affrontare problemi complessi in ambito matematico e scientifico con precisione. Oltre a eccellere nell'analisi tecnica e nella riformattazione dei dati, o3-mini è particolarmente efficace nel scomporre concetti intricati in passaggi più semplici. Tuttavia, nonostante le sue capacità avanzate, non è ancora in grado di eguagliare il ragionamento umano in contesti che richiedono creatività o intuizione profonda.

12.3.2 DeepSeek (il tuttofare)

DeepSeek-R1 è un modello open-weight paragonabile a o1 di OpenAI, ma disponibile a un costo inferiore attraverso API. La sua natura trasparente lo rende particolarmente attraente per i ricercatori, che possono adattarlo ai propri progetti specifici. DeepSeek è utile per generare ipotesi, migliorare la diagnostica medica e supportare attività di ricerca avanzate. Tuttavia, presenta alcuni limiti: il suo processo di ragionamento è più lento rispetto ad altri modelli e offre meno filtri contro output potenzialmente dannosi. Inoltre, OpenAI ha sollevato dubbi sulla legittimità del suo processo di addestramento, alimentando un dibattito sulla trasparenza e l'etica degli LLM.

12.3.3 Llama (il cavallo di battaglia)

Sviluppato da Meta, **Llama** è uno dei modelli LLM più utilizzati nella ricerca grazie alla sua natura **open-weight**, che consente agli scienziati di personalizzarlo e impiegarlo in ambienti controllati. È stato applicato con successo in una vasta gamma di ambiti, dalla predizione delle strutture cristalline ai calcoli quantistici, dimostrando una grande versatilità. Tuttavia, l'accesso a Llama richiede un'autorizzazione specifica, rendendolo meno immediato rispetto ad altri modelli open-source emergenti che sono disponibili senza restrizioni.

12.3.4 Claude (lo sviluppatore)

Claude 3.5 Sonnet, prodotto da Anthropic, è particolarmente apprezzato per la sua capacità di scrivere codice e interpretare dati visivi. Questo modello si distingue per la sua abilità nel mantenere il significato tecnico anche durante la semplificazione del linguaggio, rendendolo ideale per redigere proposte di ricerca, annotare codice e supportare attività di sviluppo software. Tuttavia, l'accesso completo alle sue funzionalità richiede un'API a pagamento, il che lo rende meno competitivo rispetto ai modelli open-source in rapida crescita, soprattutto per utenti con budget limitati.

12.3.5 OLMo (il veramente open)

OLMo 2 rappresenta un passo avanti nella trasparenza degli LLM. Questo modello non solo fornisce i pesi del modello, ma anche i **dati di addestramento** e il **codice di sviluppo**, offrendo una visione completa del suo funzionamento. Questa apertura lo rende ideale per ricercatori e sviluppatori che desiderano analizzare bias, ottimizzare le prestazioni o comprendere a fondo il processo di creazione di un LLM. L'unico svantaggio è che richiede competenze tecniche avanzate per l'implementazione, sebbene il numero di risorse educative e tutorial disponibili stia crescendo rapidamente.

Ogni modello presenta punti di forza e limiti specifici, rendendoli adatti a contesti diversi. Mentre **o3-mini** e **DeepSeek** si concentrano su ragionamento e analisi tecnica, **Llama** e **OLMo** offrono maggiore flessibilità e trasparenza per la ricerca. **Claude**, d'altra parte, si distingue per le sue capacità di sviluppo e interpretazione di dati complessi. La scelta del modello dipende dalle esigenze specifiche dell'utente, dal budget disponibile e dalle competenze tecniche.

12.4 Considerazioni Etiche e Pratiche

L'adozione di strumenti di intelligenza artificiale (AI) solleva questioni etiche e pratiche che richiedono un'attenta riflessione e un approccio responsabile.

- **Trasparenza**
  I dataset utilizzati per addestrare i modelli di AI sono spesso poco documentati e opachi, rendendo difficile valutarne la qualità e l'equità. Questo solleva interrogativi sulla presenza di bias involontari e sulla rappresentatività dei dati, con notevoli implicazioni per l'affidabilità dei risultati.
- **Equità di Accesso**
  Nonostante le potenzialità rivoluzionarie degli strumenti di AI, l'accesso a queste tecnologie non è uniformemente distribuito. Disparità economiche, geografiche e infrastrutturali possono creare disuguaglianze, limitando l'adozione di queste risorse in contesti meno privilegiati e ampliando il divario digitale.
- **Responsabilità**
  Uno dei dilemmi più complessi riguarda l'attribuzione della responsabilità per i risultati generati dai sistemi di AI. In caso di errori, bias o conseguenze indesiderate, non è sempre chiaro chi debba assumersi la responsabilità: gli sviluppatori del modello, gli utenti o le organizzazioni che lo implementano.

12.5 Riflessioni Conclusive

Gli strumenti di intelligenza artificiale stanno rivoluzionando il mondo della programmazione, offrendo un supporto senza precedenti per la risoluzione di problemi complessi, la generazione di codice e l'ottimizzazione dei flussi di lavoro. Tuttavia, il loro utilizzo deve essere accompagnato da un approccio critico e consapevole. È essenziale verificare i risultati, valutare le implicazioni etiche e garantire che l'adozione di queste tecnologie avvenga in modo equo e responsabile.

L'intelligenza artificiale non sostituirà gli sviluppatori, ma si affermerà come un alleato indispensabile, ampliando la creatività e le competenze umane. Questa collaborazione tra uomo e macchina ridefinirà il modo in cui affrontiamo le sfide del futuro, aprendo nuove opportunità e trasformando il panorama della tecnologia e della ricerca (**?**; **?**).

Bibliografia

Parte IV

EDA

Dopo aver acquisito un dataset, è fondamentale comprendere a fondo le caratteristiche dei dati in esso contenuti. Sebbene le statistiche descrittive e altre misure numeriche siano spesso efficaci per ottenere una visione d'insieme, talvolta è un'immagine a valere più di mille parole.

In questa sezione della dispensa, esploreremo alcuni concetti chiave della statistica descrittiva. Oltre a fornire definizioni teoriche, presenteremo istruzioni pratiche in R per condurre analisi statistiche su dati reali. Il capitolo si concluderà con una riflessione critica sui limiti di un approccio epistemologico basato esclusivamente sull'analisi delle associazioni tra variabili, evidenziando l'importanza di indagare le cause sottostanti ai fenomeni osservati.

13

Le fasi del progetto di analisi dei dati

In questo capitolo imparerai a

- gestire il ciclo di vita di un progetto di analisi: dalla definizione della domanda di ricerca alla comunicazione dei risultati
- organizzare i file del progetto per garantirne portabilità e condivisione

Prerequisiti

- Leggere Veridical Data Science (**?**) focalizzandoti sul primo capitolo, che introduce le problematiche della data science, e sul quarto capitolo, che fornisce le linee guida dettagliate sull'organizzazione di un progetto di analisi dei dati.

Preparazione del Notebook

```
# Carica il file _common.R per impostazioni di pacchetti e opzioni
here::here("code", "_common.R") |>
  source()
```

13.1 Introduzione

Una gestione accurata ed efficace dei dati è fondamentale, soprattutto in discipline come la psicologia, dove l'analisi di dataset complessi è una componente centrale della ricerca. Assicurare che i dati siano raccolti con precisione, organizzati in modo chiaro e facilmente accessibili per analisi e verifiche è essenziale per preservare l'integrità del lavoro scientifico e promuovere la sua riproducibilità. Una gestione rigorosa dei dati garantisce qualità e affidabilità durante tutte le fasi del progetto, dalla raccolta alla documentazione dei processi di elaborazione e delle eventuali modifiche apportate.

Dati ben organizzati e documentati non solo semplificano e rendono più effi-

ciente il processo di analisi, ma riducono anche il rischio di errori, migliorando l'utilizzabilità e l'interpretazione delle informazioni. Questo aspetto è particolarmente rilevante quando si lavora con dataset provenienti da fonti eterogenee o con strutture complesse. Inoltre, la trasparenza e la completezza nella gestione dei dati rappresentano una condizione imprescindibile per garantire la riproducibilità della ricerca, pilastro fondamentale della scienza.

La possibilità per altri ricercatori di replicare i risultati utilizzando gli stessi dati e metodi rafforza la credibilità delle conclusioni e contribuisce a costruire un progresso scientifico condiviso e solido. Una gestione dei dati responsabile, dunque, non è solo una buona pratica, ma una necessità per la produzione di conoscenze affidabili e sostenibili.

13.2 Capacità di Gestione dei Dati in R

R è uno strumento potente e versatile, progettato per supportare ogni fase del ciclo di vita dei dati, dalla raccolta alla documentazione, rendendolo indispensabile per chiunque lavori con dati complessi.

- **Importazione ed esportazione dei dati**: Con pacchetti come `readr` e `rio`, R facilita l'importazione di dati da diverse fonti, tra cui file CSV, database e API web, e consente l'esportazione in formati adatti a diversi utilizzi.
- **Pulizia e preparazione dei dati**: Pacchetti come `dplyr`, `tidyr` e `stringr` offrono strumenti intuitivi e potenti per manipolare, trasformare e preparare i dati in modo efficiente, rendendoli pronti per l'analisi.
- **Esplorazione e sintesi**: R, attraverso pacchetti come `dplyr` e `ggplot2`, permette di calcolare statistiche descrittive, individuare pattern significativi e visualizzare distribuzioni e relazioni in modo chiaro e informativo.
- **Documentazione dinamica**: Grazie a strumenti come R Markdown e Quarto, è possibile creare documenti interattivi che integrano codice, analisi, testo esplicativo e risultati, promuovendo la riproducibilità e la trasparenza del lavoro.
- **Controllo delle versioni**: L'integrazione di Git in RStudio offre un sistema di gestione delle versioni che consente di monitorare modifiche, collaborare con altri e garantire la tracciabilità del processo analitico.

Queste funzionalità rendono R uno strumento essenziale per gestire i dati in modo organizzato, trasparente e ottimale, facilitando analisi rigorose e riproducibili.

13.3 Configurare l'Ambiente R

Per sfruttare al meglio le potenzialità di R, è essenziale configurare correttamente RStudio e integrare i pacchetti fondamentali per la gestione dei dati. Una configurazione adeguata favorisce la riproducibilità e l'organizzazione del lavoro.

13.3.1 Workspace e Cronologia

Accedi a **Tools > Global Options > General** e modifica le seguenti impostazioni:

- Disabilita l'opzione *Restore .RData into workspace at startup*.
- Imposta *Save workspace to .RData on exit* su **Never**.

Queste configurazioni incoraggiano una gestione basata sugli script, rendendo il lavoro più trasparente e riducendo conflitti tra sessioni diverse. Ogni analisi sarà così chiaramente documentata nello script, evitando dipendenze da file temporanei o precedenti sessioni.

13.3.2 Pacchetti Essenziali

R è composto da un modulo base che fornisce le funzionalità fondamentali del linguaggio, ma la sua potenza deriva dall'enorme ecosistema di pacchetti aggiuntivi. Questi pacchetti possono essere caricati secondo necessità. Per questo corso, utilizzeremo regolarmente i seguenti pacchetti:

- `here`: per una gestione ordinata dei percorsi relativi, evitando problemi legati ai percorsi assoluti.
- `tidyverse`: una raccolta di pacchetti per manipolazione, analisi e visualizzazione dei dati, che include `dplyr`, `tidyr`, `ggplot2` e altri strumenti indispensabili.

Assicurati di installarli e caricarli all'inizio di ogni script con i comandi:

```
# install.packages(c("here", "tidyverse"))
# è solo necessario installarli una volta
library(here)
library(tidyverse)
```

13.3.3 Gestione dei Progetti

I progetti in RStudio rappresentano uno strumento chiave per mantenere il lavoro organizzato. Utilizzare i progetti consente di lavorare in ambienti separati, dove ogni progetto ha la propria directory dedicata.

Per creare un nuovo progetto:

1. Vai su **File > New Project**.
2. Seleziona una directory specifica per il progetto.
3. Salva tutti i file correlati (script, dati, risultati) all'interno della directory del progetto.

Questo approccio facilita la navigazione e previene errori derivanti dall'uso di file non correlati. Ogni progetto diventa così un'unità autonoma, ideale per mantenere ordine e coerenza nel lavoro.

13.4 Il Ciclo di Vita di un Progetto di Data Science

I progetti di analisi dei dati seguono solitamente queste fasi (**?**):

1. **Formulazione del problema e raccolta dei dati**: Definizione delle domande di ricerca e acquisizione dei dataset.
2. **Pulizia, preprocessing e analisi esplorativa**: Preparazione dei dati per l'analisi attraverso trasformazioni e sintesi.
3. **Analisi predittiva e/o inferenziale**: (Opzionale) Modelli statistici o predittivi per rispondere alle domande di ricerca.
4. **Valutazione dei risultati**: Interpretazione e verifica delle conclusioni tratte.
5. **Comunicazione dei risultati**: Presentazione dei risultati in forma visiva e narrativa.

Non tutti i progetti includono la fase 3, ma quasi tutti attraversano le altre fasi, rendendo essenziale un approccio organizzato e ben strutturato.

13.4.1 Fase 1: Formulazione del Problema e Raccolta dei Dati

La formulazione di una domanda di ricerca chiara e precisa rappresenta il punto di partenza fondamentale per qualsiasi progetto di data science. È essenziale definire con attenzione l'ambito in cui si intende condurre l'analisi dei dati. Questo ambito può assumere due principali direzioni:

1. **Ambito Applicativo**
   In questo contesto, l'obiettivo è spesso quello di produrre un report descrittivo che sintetizzi i risultati di un intervento specifico. Un esempio potrebbe essere la valutazione dell'efficacia di un programma di educazione sessuale rivolto agli adolescenti nelle scuole. In questo caso, l'analisi si concentra sulla descrizione dei risultati ottenuti, senza necessariamente approfondire le basi teoriche sottostanti.

2. **Ambito della Ricerca Psicologica**
   In questo secondo caso, l'approccio è più rigoroso e complesso. Non è sufficiente descrivere i risultati, ma è necessario giustificare le scelte metodologiche, come la selezione degli strumenti di misurazione e la formulazione delle domande di ricerca, basandosi su teorie consolidate e sulla letteratura scientifica esistente. Questo aspetto è stato approfondito nel Capitolo **??** del materiale didattico.

Caratteristiche di una Buona Domanda di Ricerca
La domanda di ricerca deve essere formulata in modo da poter essere risolta attraverso l'analisi dei dati disponibili. Spesso, la domanda iniziale può risultare troppo vaga o ambiziosa, rendendo necessario un processo di revisione per adattarla alle informazioni effettivamente accessibili. Questo approccio garantisce che i dati raccolti o utilizzati siano appropriati e sufficienti per rispondere in modo efficace al problema di ricerca.

Raccolta dei Dati
La fase di raccolta dei dati è altrettanto cruciale per il successo del progetto. Esistono due principali scenari:

1. **Utilizzo di Dati Esistenti**:
   In alcuni casi, è possibile sfruttare dataset già disponibili, provenienti da repository pubblici, database interni o esperimenti precedenti.

2. **Nuova Raccolta di Dati**:
   In altri casi, è necessario procedere con una raccolta dati ex novo. In entrambe le situazioni, è fondamentale pianificare con cura le analisi statistiche prima di avviare la raccolta. Una mancata pianificazione può portare a dataset incompleti, privi di informazioni essenziali o non conformi alle assunzioni richieste dai modelli statistici previsti.

Aspetti Chiave della Raccolta Dati
Per garantire l'affidabilità dei risultati, è essenziale comprendere a fondo i processi e le metodologie utilizzate per acquisire i dati. Questo include:

- **Documentazione delle Tecniche e Procedure**:
  Descrivere in dettaglio le tecniche di raccolta, le procedure seguite e i

potenziali bias che potrebbero influenzare i risultati.

- **Valutazione delle Limitazioni**:
  Identificare e considerare eventuali limitazioni dei dati, in modo da interpretare i risultati in modo critico e consapevole.

Questo approccio metodologico contribuisce a garantire che le misure ottenute siano affidabili e che le analisi siano condotte in modo rigoroso e trasparente.

13.4.2 Fase 2: Pulizia dei Dati e Analisi Esplorativa

13.4.2.1 Importazione ed Esportazione dei dati in R

Importazione dei Dati

In R, i dati vengono solitamente caricati in un **data frame**. Per facilitare questo processo, il pacchetto `rio` fornisce la funzione universale `import()`, che consente di caricare dati da formati diversi (come `.csv`, `.xlsx`, `.json`, ecc.) senza dover usare funzioni specifiche per ogni tipo di file.

Per garantire un'organizzazione ottimale dei file, è utile:

1. Creare una struttura di progetto chiara, con cartelle dedicate (es. `data/raw` per i dati grezzi e `data/processed` per quelli elaborati).

2. Usare percorsi relativi, che rendono i progetti portabili e più facilmente replicabili.

Il pacchetto **here** semplifica la gestione dei percorsi relativi, assicurando che il codice funzioni correttamente indipendentemente dalla posizione del progetto.

Supponiamo di avere un progetto chiamato `my_project` con la seguente struttura:

```
my_project/
  my_project.Rproj
  data/
     raw/
        my_data.csv
     processed/
```

Per importare il file `my_data.csv` nella memoria di lavoro di R, possiamo usare:

```r
library(here)
library(rio)

# Importazione del file
df <- import(here("data", "raw", "my_data.csv"))
```

Esportazione dei Dati

Dopo aver elaborato i dati, è importante salvarli in modo organizzato. Con il pacchetto `rio`, l'esportazione si effettua tramite `export()`. Ad esempio, per salvare un file elaborato nella cartella `data/processed`:

```
# Esportazione del file elaborato
export(df, here("data", "processed", "my_data_processed.csv"))
```

Vantaggi di un Workflow Organizzato

1. **Portabilità**: Usando percorsi relativi con `here`, il progetto può essere spostato su diversi computer o condiviso senza necessità di modificare il codice.

2. **Semplificazione**: `rio` elimina la necessità di ricordare funzioni specifiche per diversi tipi di file.

3. **Riproducibilità**: Una struttura chiara e un codice ben organizzato favoriscono la replicabilità dei risultati.

13.4.2.2 Pulizia dei Dati

Dopo aver definito la domanda della ricerca e avere raccolto i dati rilevanti, è il momento di pulire i dati. Un dataset pulito è ordinato, formattato in modo appropriato e ha voci non ambigue. La fase iniziale di pulizia dei dati consiste nell'identificare problemi con i dati (come formattazioni anomale e valori non validi) e modificarli in modo che i valori siano validi e formattati in modo comprensibile sia per il computer che per noi. La pulizia dei dati è una fase estremamente importante di un progetto di data science perché non solo aiuta a garantire che i dati siano interpretati correttamente dal computer, ma aiuta anche a sviluppare una comprensione dettagliata delle informazioni contenute nei dati e delle loro limitazioni.

L'obiettivo della pulizia dei dati è creare una versione dei dati che rifletta nella maniera più fedele possibile la realtà e che sia interpretata correttamente dal computer. Per garantire che il computer utilizzi fedelmente le informazioni contenute nei dati, è necessario modificare i dati (scrivendo codice, non modificando il file dati grezzo stesso) in modo che siano in linea con ciò che il computer "si aspetta". Tuttavia, il processo di pulizia dei dati è necessariamente soggettivo e comporta fare assunzioni sulle quantità reali sottostanti misurate e decisioni su quali modifiche siano le più sensate.

13.4.2.3 Preprocessing

Il preprocessing si riferisce al processo di modifica dei dati puliti per soddisfare i requisiti di un algoritmo specifico che si desidera applicare. Ad esempio, se si utilizza un algoritmo che richiede che le variabili siano sulla stessa scala,

potrebbe essere necessario trasformarle, oppure, se si utilizza un algoritmo che non consente valori mancanti, potrebbe essere necessario imputarli o rimuoverli. Durante il preprocessing, potrebbe essere utile anche definire nuove caratteristiche/variabili utilizzando le informazioni esistenti nei dati, se si ritiene che queste possano essere utili per l'analisi.

Come per la pulizia dei dati, non esiste un unico modo corretto per preelaborare un dataset, e la procedura finale comporta tipicamente una serie di decisioni che dovrebbero essere documentate nel codice e nei file di documentazione.

13.4.2.4 Analisi Esplorativa dei Dati

Dopo l'acquisizione dei dati, si procede con un'Analisi Esplorativa dei Dati (EDA - Exploratory Data Analysis). Questa fase iniziale mira a far familiarizzare il ricercatore con il dataset e a scoprire pattern nascosti. Si realizza attraverso:

- La costruzione di tabelle di frequenza e contingenza.
- Il calcolo di statistiche descrittive (come indici di posizione, dispersione e forma della distribuzione).
- La creazione di rappresentazioni grafiche preliminari.

L'EDA permette di generare ipotesi sui dati e di guidare le successive analisi statistiche.

13.4.3 Fase 3: Analisi Predittiva e Inferenziale

Molte domande nella data science si presentano come problemi di inferenza e/o previsione, in cui l'obiettivo principale è utilizzare dati osservati, passati o presenti, per descrivere le caratteristiche di una popolazione più ampia o per fare previsioni su dati futuri non ancora disponibili. Questo tipo di analisi è spesso orientato a supportare decisioni nel mondo reale.

13.4.4 Fase 4: Valutazione dei Risultati

In questa fase, i risultati ottenuti vengono analizzati alla luce della domanda di ricerca iniziale. Si procede a una valutazione sia quantitativa, attraverso l'applicazione di tecniche statistiche appropriate, sia qualitativa, attraverso un'attenta riflessione critica.

13.4.5 Fase 5: Comunicazione dei Risultati

L'ultima fase di un progetto di analisi dei dati consiste nel condividere i risultati con un pubblico più ampio, il che richiede la preparazione di materiali comunicativi chiari e concisi. L'obiettivo è trasformare i risultati dell'analisi in informazioni utili per supportare il processo decisionale. Questo può inclu-

dere la stesura di un articolo scientifico, la creazione di un report per un team di lavoro, o la preparazione di una presentazione con diapositive.

La comunicazione deve essere adattata al pubblico di riferimento. Non si deve dare per scontato che il pubblico abbia familiarità con il progetto: è fondamentale spiegare l'analisi e le visualizzazioni in modo chiaro e dettagliato. Anche se per il ricercatore il messaggio principale di una figura o diapositiva può sembrare ovvio, è sempre una buona pratica guidare il pubblico nella sua interpretazione, evitando l'uso di gergo tecnico complesso.

13.5 Organizzazione del Progetto

Un requisito fondamentale per un progetto di analisi dei dati è organizzare in modo efficiente i file sul proprio computer. Questo include i file dei dati, il codice e la documentazione del progetto. Tutti questi elementi dovrebbero essere raccolti all'interno di una singola cartella dedicata al progetto.

13.5.1 Home Directory

In RStudio, è possibile creare un file chiamato `nome_del_progetto.Rproj`, che consente di configurare automaticamente la *home directory* del progetto, ovvero la cartella principale da cui R avvia il lavoro relativo al progetto. Per utilizzare questa funzionalità, è sufficiente aprire RStudio cliccando direttamente sul file `nome_del_progetto.Rproj`.

La *home directory* rappresenta il punto di riferimento principale per tutte le operazioni del progetto, come il caricamento di file, il salvataggio degli output e la gestione delle risorse.

Grazie a questa configurazione, è possibile utilizzare percorsi relativi per accedere ai file all'interno del progetto. I percorsi relativi si basano sempre sulla cartella principale del progetto, il che rende il codice più portabile e adattabile. In pratica, chiunque scarichi il tuo progetto sarà in grado di eseguirlo senza dover modificare manualmente i percorsi dei file. Questo approccio migliora la condivisione e garantisce una maggiore riproducibilità del tuo lavoro.

13.5.2 Struttura di un Progetto

Un'adeguata organizzazione della struttura di un progetto è fondamentale per garantire chiarezza, riproducibilità e portabilità. Yu & Barter (**?**) propone il seguente template per la gestione di un progetto:

```
data/
  data_documentation/
    codebook.txt
  data.csv
dslc_documentation/
  01_cleaning.qmd
  02_eda.qmd
  03_clustering.qmd
  04_prediction.qmd
  functions/
    cleanData.R
    preProcessData.R
README.md
```

13.5.2.1 Cartelle Principali

1. **`data/`:**
   Questa cartella raccoglie i dataset utilizzati nel progetto, suddivisi in:

   - **`raw/`**: per i dati grezzi, che non devono mai essere modificati. Esempio: `data/raw/data.csv`.

   - **`processed/`**: per i dati elaborati, cioè quelli puliti e trasformati per le analisi successive.

   È indispensabile includere un **codebook** nella sottocartella `raw` per documentare variabili, unità di misura e ogni altra informazione necessaria per comprendere i dati. Questo migliora la trasparenza e facilita la condivisione del progetto.

2. **`dslc_documentation/`:**
   Contiene tutti i file necessari per documentare e condurre le analisi.

   - File principali: possono essere in formato `.qmd` (Quarto per R) o `.ipynb` (Jupyter Notebook per Python), ordinati cronologicamente tramite un prefisso numerico (ad esempio, `01_data_cleaning.qmd`, `02_analysis.qmd`).

   - **`functions/`:** una sottocartella dedicata a script con funzioni personalizzate, utili per diverse fasi del progetto. Ad esempio, `.R` per R o `.py` per Python.

13.5.2.2 File Supplementari

- **`README.md`:**
  Un file fondamentale che descrive la struttura del progetto e offre una breve panoramica del contenuto di ogni cartella e file. Deve includere:

- Lo scopo del progetto.
- Una spiegazione della struttura delle cartelle.
- I passaggi necessari per replicare le analisi.

13.5.2.3 Vantaggi della Struttura Proposta

1. **Organizzazione Chiara:**
   Separare i dati grezzi da quelli elaborati riduce il rischio di modificare accidentalmente le versioni originali.
2. **Riproducibilità:**
   La documentazione completa (codebook, file analitici e README) facilita la comprensione e la riproduzione del progetto anche a distanza di tempo o da parte di altri utenti.
3. **Portabilità:**
   Utilizzando percorsi relativi (ad esempio, con il pacchetto **here** in R), l'intero progetto può essere facilmente trasferito tra utenti o computer senza modificare il codice.
4. **Efficienza:**
   La sottocartella `functions/` consente di riutilizzare codice in più parti del progetto, migliorando la modularità e riducendo la duplicazione di script.

In conclusione, l'organizzazione del progetto secondo questo template garantisce non solo un flusso di lavoro ordinato e ben documentato, ma anche la possibilità di condividere facilmente il progetto con colleghi o team. Adottare questa struttura è un primo passo essenziale verso un'analisi dati rigorosa e riproducibile.

13.6 Riflessioni Conclusive

La forza e la bellezza del codice risiedono nella sua riusabilità: una volta scritto, può essere applicato infinite volte per ottenere risultati coerenti. Se configurato correttamente, lo stesso codice applicato agli stessi dati produrrà sempre gli stessi risultati. Questo principio, noto come *riproducibilità computazionale*, è fondamentale per garantire la trasparenza e l'affidabilità del lavoro scientifico.

La riproducibilità offre numerosi vantaggi:

- **Monitorare le modifiche del progetto**
  La possibilità di riprodurre il lavoro semplifica il monitoraggio delle evoluzioni e dei cambiamenti nel progetto. Questo consente di comprendere come il progetto si è sviluppato nel tempo, facilitando il confronto tra diverse versioni o approcci.
- **Riprodurre il proprio lavoro**
  Il primo beneficiario della riproducibilità sei tu stesso. Essere in grado di replicare i propri risultati è essenziale, soprattutto se in futuro sarà necessario rivedere o approfondire il lavoro. La riproducibilità garantisce che le analisi possano essere riprese con facilità e senza ambiguità.
- **Costruire su basi solide**
  Altri ricercatori possono utilizzare il tuo lavoro come punto di partenza, espandendolo o applicandolo a nuovi contesti. La condivisione di codice riproducibile non solo favorisce la collaborazione, ma contribuisce a creare un corpo di conoscenze più robusto e condiviso.

Tuttavia, rendere il codice riproducibile non è sempre semplice. Richiede attenzione nella documentazione, un'organizzazione chiara del progetto e strumenti adeguati per garantire che l'ambiente di lavoro sia stabile e replicabile. In questo capitolo, abbiamo esplorato alcune strategie e tecniche per raggiungere questi obiettivi.

> **Nota**
>
> Un problema cruciale nella psicologia contemporanea è la *crisi di replicabilità*, che evidenzia come molti risultati di ricerca non siano replicabili (**?**). La *riproducibilità computazionale*, pur avendo un obiettivo più ristretto, si concentra sulla possibilità di ottenere gli stessi risultati applicando lo stesso codice agli stessi dati. Questo approccio, sebbene non risolva interamente la crisi, rappresenta un passo fondamentale verso una scienza più trasparente e rigorosa.

13.7 Esercizi

> **Esercizio**
>
> L'obiettivo di questo esercizio è comprendere il ciclo di vita di un progetto di analisi dei dati, l'organizzazione del progetto e la gestione della riproducibilità.

1. **Gestione del progetto di analisi**
   - Quali sono le fasi principali di un progetto di analisi dei dati secondo Yu (2024)?
   - Spiega il ruolo della fase di formulazione del problema e raccolta dei dati.
2. **Organizzazione del workspace in R**
   - Quali impostazioni devono essere modificate in RStudio per favorire la riproducibilità?
   - Perché è importante usare percorsi relativi nei progetti in RStudio?
   - Descrivi il ruolo del pacchetto `here` nella gestione dei percorsi dei file.
3. **Struttura dei progetti in R**
   - Quali sono i vantaggi dell'utilizzo dei progetti in RStudio?
   - Quali sono le cartelle principali in una struttura organizzata di un progetto?
   - Perché è utile separare i dati grezzi dai dati processati?
4. **Importazione ed esportazione dei dati**
   - Quali pacchetti di R possono essere utilizzati per importare ed esportare dati?
   - Scrivi un esempio di codice per importare un file CSV usando `rio` e il pacchetto `here`.
   - Come puoi esportare un dataset modificato in una cartella dedicata ai dati processati?
5. **Pulizia e preprocessing dei dati**
   - Qual è la differenza tra pulizia e preprocessing dei dati?
   - Quali strumenti di `dplyr` sono comunemente usati per pulire e trasformare i dati?
6. **Analisi esplorativa dei dati (EDA)**
   - Quali sono alcuni strumenti utilizzati in R per effettuare un'analisi esplorativa dei dati?
   - Scrivi un breve esempio di codice in R per calcolare statistiche descrittive di base su un dataset.
7. **Riproducibilità e comunicazione dei risultati**
   - Perché la riproducibilità è un elemento chiave nella scienza dei dati?
   - Quali strumenti offre Quarto per la documentazione e la condivisione dei risultati di un'analisi?

Soluzione

1. Gestione del progetto di analisi

- **Fasi principali del progetto di analisi dei dati (Yu, 2024):**
  1. Formulazione del problema e raccolta dei dati
  2. Pulizia, preprocessing e analisi esplorativa
  3. Analisi predittiva e/o inferenziale (se applicabile)
  4. Valutazione dei risultati
  5. Comunicazione dei risultati
- **Ruolo della fase di formulazione del problema:**
  Aiuta a definire gli obiettivi dell'analisi e a selezionare le fonti di dati adeguate. Una domanda di ricerca ben definita garantisce che i dati siano pertinenti e che le analisi siano mirate.

2. Organizzazione del workspace in R

- **Impostazioni da modificare in RStudio:**
  - Disabilitare *Restore .RData into workspace at startup*
  - Impostare *Save workspace to .RData on exit* su "Never"
- **Importanza dei percorsi relativi:**
  Permettono di rendere il progetto portabile e riproducibile, evitando problemi di percorsi assoluti specifici per un computer.
- **Ruolo del pacchetto `here`**
  Aiuta a gestire i percorsi relativi all'interno del progetto senza dover specificare percorsi assoluti.

3. Struttura dei progetti in R

- **Vantaggi dell'uso dei progetti in RStudio:**
  Mantengono ambienti separati, organizzano i file e facilitano la riproducibilità.
- **Cartelle principali in una struttura organizzata:**
  - `data/raw/` → Dati grezzi
  - `data/processed/` → Dati elaborati
  - `dslc_documentation/` → Documentazione e script
  - `functions/` → Funzioni personalizzate
- **Separare dati grezzi da dati processati:**
  Evita di modificare accidentalmente i dati originali, garantendo riproducibilità.

4. Importazione ed esportazione dei dati

- **Pacchetti per importazione/esportazione:**
  - `rio`: unifica funzioni di import/export
  - `readr`: specifico per CSV e altri formati di testo
  - `here`: gestisce percorsi relativi
- **Esempio di codice per importare dati CSV:**

```
library(here)
library(rio)
df <- rio::import(here("data", "raw", "my_data.csv"))
```

- **Esportare dati modificati:**

```
rio::export(df, here("data", "processed", "my_data_processed.csv"))
```

5. Pulizia e preprocessing dei dati

- **Differenza tra pulizia e preprocessing:**
  - Pulizia: rimozione di errori, gestione dei dati mancanti, formattazione
  - Preprocessing: trasformazione dei dati per adattarli a modelli specifici
- **Strumenti di `dplyr` per pulizia e trasformazione:**
  - `mutate()`, `filter()`, `select()`, `rename()`, `relocate()`

6. Analisi esplorativa dei dati (EDA)

- **Strumenti comuni:**
  - `summary()`, `str()`, `glimpse()`, `ggplot2` per visualizzazione
- **Esempio di codice per statistiche descrittive:**

```
summary(df)
```

7. Riproducibilità e comunicazione dei risultati

- **Importanza della riproducibilità:**
  - Facilita la verifica e il miglioramento degli studi
  - Previene errori accidentali
  - Consente a terzi di replicare e costruire su ricerche precedenti
- **Strumenti di Quarto:**
  - Permette di combinare testo, codice e output in documenti riproducibili
  - Supporta citazioni automatiche e gestione delle bibliografie

Informazioni sull'Ambiente di Sviluppo

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dyl
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dyl
#>
#> locale:
#> [1] C/UTF-8/C/C/C/C
#>
#> time zone: Europe/Rome
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] thematic_0.1.6   MetBrewer_0.2.0  ggokabeito_0.1.0 see_0.10.0
#>  [5] gridExtra_2.3    patchwork_1.3.0  bayesplot_1.11.1 psych_2.4.12
#>  [9] scales_1.3.0     markdown_1.13    knitr_1.49       lubridate_1.9.4
#> [13] forcats_1.0.0    stringr_1.5.1    dplyr_1.1.4      purrr_1.0.4
#> [17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1
#> [21] tidyverse_2.0.0  rio_1.2.3        here_1.0.1
#>
#> loaded via a namespace (and not attached):
#>  [1] generics_0.1.3    stringi_1.8.4     lattice_0.22-6    hms_1.1.3
#>  [5] digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3    grid_4.4.2
#>  [9] timechange_0.3.0  fastmap_1.2.0     rprojroot_2.0.4   jsonlite_1.8.9
#> [13] mnormt_2.1.1      cli_3.6.4         rlang_1.1.5       munsell_0.5.1
#> [17] withr_3.0.2       tools_4.4.2       parallel_4.4.2    tzdb_0.4.0
#> [21] colorspace_2.1-1  pacman_0.5.1      vctrs_0.6.5       R6_2.6.1
#> [25] lifecycle_1.0.4   pkgconfig_2.0.3   pillar_1.10.1     gtable_0.3.6
#> [29] glue_1.8.0        xfun_0.50         tidyselect_1.2.1  rstudioapi_0.17.1
#> [33] farver_2.1.2      htmltools_0.5.8.1 nlme_3.1-167      rmarkdown_2.29
#> [37] compiler_4.4.2
```

Bibliografia

14

Flusso di lavoro per la pulizia dei dati

In questo capitolo imparerai a

- verificare, pulire e trasformare i dati per l'analisi
- garantire riservatezza eliminando informazioni sensibili
- documentare il dataset con un dizionario e note esplicative
- assicurare validità, unicità e organizzazione dei dati
- applicare regole coerenti per denominazione e codifica

Prerequisiti

- Leggere Cleaning sample data in standardized way di Crystal Lewis.
- Leggere Getting Started Creating Data Dictionaries: How to Create a Shareable Data Set di Buchanan et al. (**?**).
- Consultare il capitolo Documentation di *Data Management in Large-Scale Education Research*.
- Consultare How to Make a Data Dictionary.
- Consultare data dictionary template.

Preparazione del Notebook

```
here::here("code", "_common.R") |>
  source()

# Load packages
if (!requireNamespace("pacman")) install.packages("pacman")
pacman::p_load(mice, labelled, haven, pointblank, mice)
```

14.1 Introduzione

Nonostante la fase più interessante di un progetto di analisi dei dati sia quella in cui si riesce a rispondere alla domanda che ha dato avvio all'indagine, gran parte del tempo di un analista è in realtà dedicata a una fase preliminare: la pulizia e il preprocessing dei dati, operazioni che vengono svolte ancor prima dell'analisi esplorativa.

In questo capitolo, esamineremo un caso concreto di *data cleaning* e preprocessing, seguendo il tutorial di Crystal Lewis. Il problema viene presentato come segue:

> I am managing data for a longitudinal randomized controlled trial (RCT) study. For this RCT, schools are randomized to either a treatment or control group. Students who are in a treatment school receive a program to boost their math self-efficacy. Data is collected on all students in two waves (wave 1 is in the fall of a school year, and wave 2 is collected in the spring). At this point in time, we have collected wave 1 of our student survey on a paper form and we set up a data entry database for staff to enter the information into. Data has been double-entered, checked for entry errors, and has been exported in a csv format ("w1_mathproj_stu_svy_raw.csv") to a folder (called "data") where it is waiting to be cleaned.

Crystal Lewis elenca i seguenti passaggi da seguire nel processo di data cleaning:

1. Revisione dei dati.
2. Regolazione del numero di casi.
3. De-identificazione dei dati.
4. Eliminazione delle colonne irrilevanti.
5. Divisione delle colonne, se necessario.
6. Ridenominazione delle variabili.
7. Trasformazione/normalizzazione delle variabili.
8. Standardizzazione delle variabili.
9. Aggiornamento dei tipi di variabili, se necessario.
10. Ricodifica delle variabili.
11. Creazione di eventuali variabili necessarie.
12. Gestione dei valori mancanti, se necessario.
13. Aggiunta di metadati, se necessario.
14. Validazione dei dati.
15. Fusione e/o unione dei dati, se necessario.
16. Trasformazione dei dati, se necessario.
17. Salvataggio dei dati puliti.

Sebbene l'ordine di questi passaggi sia flessibile e possa essere adattato alle esigenze specifiche, c'è un passaggio che non dovrebbe mai essere saltato: il primo, ovvero la revisione dei dati. Senza una revisione preliminare, l'analista rischia di sprecare ore a pulire i dati per poi scoprire che mancano dei partecipanti, che i dati non sono organizzati come previsto o, peggio ancora, che si sta lavorando con i dati sbagliati.

14.2 Tutorial

Questo tutorial segue i passaggi descritti da Crystal Lewis per illustrare le buone pratiche nella gestione e pulizia dei dati.

14.2.1 Organizzazione dei Dati

Un principio fondamentale nella gestione dei dati è preservare l'integrità dei dati grezzi. I dati originali **non devono mai essere modificati direttamente**. È quindi consigliabile strutturare i dati in una directory denominata `data`, suddivisa in due sottocartelle:

- `raw`: contiene i dati originali, mantenuti inalterati.
- `processed`: destinata ai dati ripuliti e preprocessati.

Ad esempio, importiamo i dati da un file denominato `w1_mathproj_stu_svy_raw.csv` per avviare il processo di pulizia. Tutte le operazioni dovranno essere effettuate utilizzando percorsi relativi alla home directory del progetto, che definiremo come primo passo.

14.2.2 Passaggi del Tutorial

14.2.2.1 Importare e Esaminare i Dati

Importiamo i dati utilizzando la funzione `import()` della libreria `rio` e visualizziamo i primi valori di ciascuna colonna per verificarne la corretta importazione:

```
# Importa i dati
svy <- rio::import(here::here("data", "w1_mathproj_stu_svy_raw.csv"))

# Esamina la struttura del dataset
glimpse(svy)
#> Rows: 6
#> Columns: 7
#> $ stu_id      <int> 1347, 1368, 1377, 1387, 1347, 1399
```

```
#> $ svy_date    <IDate> 2023-02-13, 2023-02-13, 2023-02-13, 2023-02-13, 2023-0~
#> $ grade_level <int> 9, 10, 9, 11, 9, 12
#> $ math1       <int> 2, 3, 4, 3, 2, 4
#> $ math2       <chr> "1", "2", "\n4", "3", "2", "1"
#> $ math3       <int> 3, 2, 4, NA, 4, 3
#> $ math4       <int> 3, 2, 4, NA, 2, 1
```

Per controllare visivamente i dati, possiamo esaminare le prime e le ultime righe del `data frame`:

```
# Visualizza le prime righe
svy |>
  head()
#>   stu_id   svy_date grade_level math1 math2 math3 math4
#> 1   1347 2023-02-13           9     2     1     3     3
#> 2   1368 2023-02-13          10     3     2     2     2
#> 3   1377 2023-02-13           9     4   \n4     4     4
#> 4   1387 2023-02-13          11     3     3    NA    NA
#> 5   1347 2023-02-14           9     2     2     4     2
#> 6   1399 2023-02-14          12     4     1     3     1

# Visualizza le ultime righe
svy |>
  tail()
#>   stu_id   svy_date grade_level math1 math2 math3 math4
#> 1   1347 2023-02-13           9     2     1     3     3
#> 2   1368 2023-02-13          10     3     2     2     2
#> 3   1377 2023-02-13           9     4   \n4     4     4
#> 4   1387 2023-02-13          11     3     3    NA    NA
#> 5   1347 2023-02-14           9     2     2     4     2
#> 6   1399 2023-02-14          12     4     1     3     1
```

14.2.2.2 Individuare e Rimuovere i Duplicati

In questa fase, eseguiamo alcune modifiche necessarie al `data frame`, come rimuovere duplicati e ordinare i dati:

- **Verifica duplicati**: controlliamo i record duplicati nel dataset.
- **Rimuovi duplicati**: manteniamo solo la prima occorrenza.
- **Ordina per data**: organizziamo i record in ordine crescente rispetto alla variabile `svy_date`.
- **Esamina i dati puliti**: controlliamo il risultato delle modifiche.

```
# Identifica i duplicati basati su 'stu_id'
duplicates <-
  svy[duplicated(svy$stu_id) | duplicated(svy$stu_id, fromLast = TRUE), ]
```

```
# Visualizza i duplicati trovati
duplicates
#>   stu_id   svy_date grade_level math1 math2 math3 math4
#> 1   1347 2023-02-13           9     2     1     3     3
#> 5   1347 2023-02-14           9     2     2     4     2
```

```
# Ordina per 'svy_date' in ordine crescente
svy <- svy[order(svy$svy_date), ]
```

```
# Rimuove i duplicati mantenendo la prima occorrenza
svy <- svy[!duplicated(svy$stu_id), ]
```

```
# Esamina il dataset finale
print(svy)
#>   stu_id   svy_date grade_level math1 math2 math3 math4
#> 1   1347 2023-02-13           9     2     1     3     3
#> 2   1368 2023-02-13          10     3     2     2     2
#> 3   1377 2023-02-13           9     4   \n4     4     4
#> 4   1387 2023-02-13          11     3     3    NA    NA
#> 6   1399 2023-02-14          12     4     1     3     1
```

Verifichiamo le dimensioni del dataset pulito per assicurarci che le operazioni siano state eseguite correttamente:

```
# Controlla il numero di righe e colonne
svy |>
  dim()
#> [1] 5 7
```

14.2.2.3 De-identificazione dei Dati

```
# Rimuovi la colonna 'svy_date'
svy <- svy |>
  dplyr::select(-svy_date)

# Mostra i nomi delle colonne rimaste
names(svy)
#> [1] "stu_id"      "grade_level" "math1"       "math2"       "math3"
#> [6] "math4"
```

14.2.2.4 Rimuovere le Colonne non Necessarie

Nel caso presente, la rimozione di colonne non è necessaria. Tuttavia, in molti progetti di analisi dei dati, soprattutto quando i dati vengono raccolti utilizzando software di terze parti o strumenti specifici per esperimenti psicologici, è comune trovarsi con colonne che non sono pertinenti allo studio in corso.

Queste colonne possono includere dati come identificatori interni, timestamp generati automaticamente, informazioni di debug, o variabili che non sono rilevanti per l'analisi che si intende condurre. Quando tali colonne sono irrilevanti per la ricerca, possono essere rimosse per semplificare il dataset e ridurre il rischio di confusione o errori durante l'analisi. Rimuovere le colonne non necessarie non solo rende il dataset più gestibile, ma aiuta anche a focalizzare l'analisi sulle variabili che realmente importano per rispondere alle domande di ricerca.

14.2.2.5 Dividere le Colonne Secondo Necessità

Nel caso presente, questa operazione non è necessaria. Tuttavia, se si lavora con un dataset che include una colonna chiamata "NomeCompleto", contenente sia il nome che il cognome di uno studente, per esempio, è buona pratica separare questa colonna in due colonne distinte, "Nome" e "Cognome". Questa suddivisione facilita l'analisi e la manipolazione dei dati, rendendoli più organizzati e accessibili.

14.2.2.6 Rinominare le Colonne

È importante assegnare nomi chiari alle colonne del dataset. Utilizzare nomi di variabili comprensibili aiuta a rendere l'analisi dei dati più intuitiva e a ridurre il rischio di errori interpretativi.

Esempi di buone pratiche:

- Evita nomi di colonne come "x" o acronimi incomprensibili. Questi possono creare confusione durante l'analisi, specialmente se il dataset viene condiviso con altri ricercatori o se viene ripreso dopo un lungo periodo di tempo.
- Invece, cerca di utilizzare nomi di variabili che descrivano chiaramente il contenuto della colonna. Ad esempio, invece di "x1" o "VAR123", un nome come "ansia_base" o "liv_autoefficacia" è molto più comprensibile e immediato.
- Per i nomi composti, utilizza un separatore come il trattino basso `_`. Ad esempio, se stai lavorando con dati relativi a un test psicologico, potresti avere colonne chiamate "test_ansia_pre" e "test_ansia_post" per indicare i risultati del test di ansia prima e dopo un intervento.

Esempi di nomi di colonne ben scelti:

- *Nome generico:* `TS`, `AE`
  - *Nome migliore:* `tempo_studio`, `auto_efficacia`
- *Nome generico:* `S1`, `S2`
  - *Nome migliore:* `stress_situazione1`, `stress_situazione2`
- *Nome generico:* `Q1`, `Q2`
  - *Nome migliore:* `qualità_sonno_sett1`, `qualità_sonno_sett2`

14.2.2.7 Trasformare le Variabili

Nel caso presente non si applica, ma è un passo importante in molte analisi dei dati.

Esempi di trasformazione delle variabili:

- *Logaritmo di una variabile:* Immaginiamo di avere una variabile che misura i tempi di reazione dei partecipanti a un esperimento. Se i tempi di reazione hanno una distribuzione fortemente asimmetrica (con alcuni valori molto elevati), potrebbe essere utile applicare una trasformazione logaritmica per rendere la distribuzione più simmetrica e migliorare l'interpretabilità dei risultati.
- *Codifica delle variabili categoriche:* Se è presente una variabile categorica come il "tipo di intervento" con valori come "cognitivo", "comportamentale" e "farmacologico", potrebbe essere necessario trasformare questa variabile in variabili dummy (ad esempio, `intervento_cognitivo`, `intervento_comportamentale`, `intervento_farmacologico`), dove ogni variabile assume il valore 0 o 1 a seconda della presenza o meno di quel tipo di intervento. Questo è utile quando si utilizzano tecniche di regressione.

14.2.2.8 Standardizzazione delle Variabili

La standardizzazione è utile quando si desidera rendere comparabili variabili misurate su scale diverse, ad esempio per confronti tra gruppi o per inclusione in modelli di regressione.

Esempio. Supponiamo di avere una variabile che misura il livello di ansia su una scala da 0 a 100. Per standardizzarla:

1. **Sottrai la media** del campione dalla variabile.
2. **Dividi per la deviazione standard**.

Il risultato è una variabile con media pari a 0 e deviazione standard pari a 1.

Vantaggi della standardizzazione:

- facilita l'interpretazione dei coefficienti in un modello di regressione;
- permette un confronto diretto tra variabili che hanno unità di misura diverse.

14.2.2.9 Normalizzazione delle Variabili

La normalizzazione consiste nel ridimensionare i dati su una scala predefinita, spesso compresa tra **0 e 1**. Questo processo è particolarmente utile in analisi multivariate, dove variabili con scale molto diverse potrebbero influenzare in modo sproporzionato i risultati.

Esempio. Hai dati su:

- **Ore di sonno** (misurate in ore, da 0 a 24).
- **Livello di stress** (misurato su una scala da 1 a 50).
- **Auto-efficacia** (misurata su una scala da 0 a 100).

Per garantire che ogni variabile abbia lo stesso peso nell'analisi, puoi normalizzarle usando una formula come:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}.$$

Perché Standardizzare o Normalizzare?

Trasformare le variabili è cruciale per:

1. **Gestire scale diverse**: Riduce il rischio che variabili con valori numericamente più grandi dominino i risultati.
2. **Garantire validità e interpretabilità**: Facilita il confronto tra dati provenienti da fonti o gruppi diversi.
3. **Evitare problemi numerici nei modelli statistici**: Alcuni algoritmi di machine learning o tecniche di ottimizzazione richiedono che i dati siano su scale simili per funzionare correttamente.

In conclusione, la scelta tra standardizzazione e normalizzazione dipende dal contesto dell'analisi e dagli obiettivi specifici. Entrambi i processi sono strumenti indispensabili per garantire che i dati siano trattati in modo adeguato, portando a risultati robusti e facilmente interpretabili.

14.2.2.10 Aggiornare i Tipi delle Variabili

Nel caso presente non è necessario. Supponiamo invece di avere una colonna in un dataset psicologico che contiene punteggi di un questionario, ma i dati sono stati importati come stringhe (testo) invece che come numeri. Per eseguire calcoli statistici, sarà necessario convertire questa colonna da stringa a numerico.

In `R`, si potrebbe usare il seguente codice:

```
# Supponiamo di avere un data frame chiamato 'df' con una colonna 'punteggio' importata co
df$punteggio <- as.numeric(df$punteggio)

# Ora la colonna 'punteggio' è stata convertita in un tipo numerico ed è possibile eseguir
```

In questo esempio, la funzione `as.numeric()` viene utilizzata per convertire la colonna `punteggio` in un formato numerico, permettendo di eseguire analisi quantitative sui dati.

Un altro caso molto comune si verifica quando si importano dati da file Excel. Spesso capita che, all'interno di una cella di una colonna che dovrebbe contenere solo valori numerici, venga inserito erroneamente uno o più caratteri alfanumerici. Di conseguenza, l'intera colonna viene interpretata come di tipo alfanumerico, anche se i valori dovrebbero essere numerici. In questi casi, è fondamentale individuare la cella problematica, correggere manualmente il valore errato, e poi riconvertire l'intera colonna da alfanumerica a numerica.

14.2.2.11 Ricodificare le Variabili

Anche se in questo caso non è necessario, la ricodifica delle variabili è una pratica molto comune nelle analisi dei dati psicologici.

Per esempio, consideriamo una variabile categoriale con modalità descritte da stringhe poco comprensibili, che vengono ricodificate con nomi più chiari e comprensibili.

Supponiamo di avere un DataFrame chiamato `df` con una colonna `tipo_intervento` che contiene le modalità `"CT"`, `"BT"`, e `"MT"` per rappresentare rispettivamente "Terapia Cognitiva", "Terapia Comportamentale" e "Terapia Mista". Queste abbreviazioni potrebbero non essere immediatamente chiare a chiunque analizzi i dati, quindi decidiamo di ricodificarle con nomi più espliciti. Ecco come farlo in `R`:

```
# Tibble chiamato 'df' con una colonna 'tipo_intervento'
df <- tibble(tipo_intervento = c("CT", "BT", "MT", "CT", "BT"))
df
#> # A tibble: 5 x 1
#>   tipo_intervento
#>   <chr>
#> 1 CT
#> 2 BT
#> 3 MT
#> 4 CT
#> 5 BT
```

```
# Ricodifica delle modalità della variabile 'tipo_intervento' in nomi più comprensibili
df <- df %>%
  mutate(tipo_intervento_ricodificato = dplyr::recode(
    tipo_intervento,
    "CT" = "Terapia Cognitiva",
    "BT" = "Terapia Comportamentale",
    "MT" = "Terapia Mista"
  ))
```

```
# Mostra il tibble con la nuova colonna ricodificata
df
#> # A tibble: 5 x 2
#>   tipo_intervento tipo_intervento_ricodificato
#>   <chr>           <chr>
#> 1 CT              Terapia Cognitiva
#> 2 BT              Terapia Comportamentale
#> 3 MT              Terapia Mista
#> 4 CT              Terapia Cognitiva
#> 5 BT              Terapia Comportamentale
```

14.2.2.12 Aggiungere Nuove Variabili nel Data Frame

Nel caso presente non è richiesto, ma aggiungere nuove variabili a un DataFrame è un'operazione comune durante l'analisi dei dati. Un esempio è il calcolo dell'indice di massa corporea (BMI).

Supponiamo di avere un DataFrame chiamato `df` che contiene le colonne `peso_kg` (peso in chilogrammi) e `altezza_m` (altezza in metri) per ciascun partecipante a uno studio psicologico. Per arricchire il dataset, possiamo calcolare il BMI per ogni partecipante e aggiungerlo come una nuova variabile.

Il BMI si calcola con la formula:

$$\text{BMI} = \frac{\text{peso in kg}}{\text{altezza in metri}^2}.$$

Ecco come aggiungere la nuova colonna.

```
# Supponiamo di avere un tibble chiamato 'df' con le colonne 'peso_kg' e 'altezza_m'
df <- tibble(
  peso_kg = c(70, 85, 60, 95),
  altezza_m = c(1.75, 1.80, 1.65, 1.90)
)
df
#> # A tibble: 4 x 2
#>   peso_kg altezza_m
#>     <dbl>     <dbl>
#> 1      70      1.75
#> 2      85      1.8
#> 3      60      1.65
#> 4      95      1.9
```

```
# Calcola il BMI e aggiungilo come una nuova colonna 'BMI'
df <- df %>%
  mutate(BMI = peso_kg / (altezza_m^2))
```

```
# Mostra il tibble con la nuova variabile aggiunta
df
#> # A tibble: 4 x 3
#>   peso_kg altezza_m   BMI
#>     <dbl>     <dbl> <dbl>
#> 1      70      1.75  22.9
#> 2      85      1.8   26.2
#> 3      60      1.65  22.0
#> 4      95      1.9   26.3
```

14.2.3 Affrontare il Problema dei Dati Mancanti

I dati mancanti sono un problema comune nelle ricerche psicologiche e in molte altre discipline. Quando mancano delle informazioni in un dataset, possono verificarsi gravi problemi per l'analisi statistica, come risultati distorti, riduzione della precisione delle stime o, in alcuni casi, l'impossibilità di applicare alcuni algoritmi.

14.2.3.1 Perché i Dati Mancanti Sono un Problema?

Immagina di voler capire il rendimento medio di una classe in un test, ma alcuni studenti hanno lasciato delle risposte in bianco. Se ignoriamo le risposte mancanti o eliminiamo gli studenti con dati incompleti, rischiamo di ottenere una stima che non rappresenta correttamente la realtà. Questo succede perché:

- **Bias dei risultati**: Se i dati mancanti non sono casuali (ad esempio, i dati mancanti sono presenti più spesso in studenti con basso rendimento), le conclusioni possono essere errate.
- **Riduzione della potenza statistica**: Eliminare dati incompleti riduce il numero totale di osservazioni, rendendo più difficile trovare risultati significativi.
- **Impossibilità di applicare alcuni metodi**: Molti algoritmi statistici richiedono dati completi e non funzionano con valori mancanti.

14.2.3.2 Come Gestire i Dati Mancanti?

Ci sono diversi modi per affrontare i dati mancanti. Vediamo prima i metodi più semplici e poi quelli più avanzati.

Esaminiamo il data frame iniziale:

```
svy |>
  summary()
#>      stu_id      grade_level       math1        math2
#>  Min.   :1347   Min.   : 9.0   Min.   :2.0   Length:5
#>  1st Qu.:1368   1st Qu.: 9.0   1st Qu.:3.0   Class :character
```

```
#>  Median :1377   Median :10.0   Median :3.0   Mode  :character
#>  Mean   :1376   Mean   :10.2   Mean   :3.2
#>  3rd Qu.:1387   3rd Qu.:11.0   3rd Qu.:4.0
#>  Max.   :1399   Max.   :12.0   Max.   :4.0
#>
#>      math3          math4
#>  Min.   :2.00   Min.   :1.00
#>  1st Qu.:2.75   1st Qu.:1.75
#>  Median :3.00   Median :2.50
#>  Mean   :3.00   Mean   :2.50
#>  3rd Qu.:3.25   3rd Qu.:3.25
#>  Max.   :4.00   Max.   :4.00
#>  NA's   :1      NA's   :1
```

Si noti la presenza di dati mancanti sulle variabili `math3` e `math4`.

```
dim(svy)
#> [1] 5 6
```

1. **Esclusione dei Casi Incompleti (Complete Case Analysis).** Un approccio comune, ma spesso non ideale, è quello di analizzare solo i casi completi, eliminando tutte le righe con valori mancanti. In R, questo si può fare con il seguente comando:

   ```
   svy_comp <- svy |>
     drop_na()
   ```

   In questo modo abbiamo escluso tutte le righe nelle quali sono presenti dei dati mancanti (in questo caso, una sola riga).

   ```
   dim(svy_comp)
   #> [1] 4 6
   ```

   Limite: Questo metodo può introdurre bias se i dati mancanti non sono casuali e riduce il campione, compromettendo l'affidabilità delle analisi.

2. **Imputazione Semplice**
   Sostituire i valori mancanti con stime semplici:

   - **Media o mediana**: Per le variabili numeriche, sostituire i valori mancanti con la media o la mediana. Questo metodo è facile da implementare, ma può ridurre la variabilità nei dati.
   - **Moda**: Per le variabili categoriche, sostituire i valori mancanti con il valore più frequente. Tuttavia, può introdurre distorsioni se i dati sono molto eterogenei.

3. **Imputazione Multipla**
   Un approccio più avanzato è l'imputazione multipla, che utilizza modelli statistici per stimare i valori mancanti in modo iterativo. L'idea di base è semplice: ogni valore mancante viene stimato tenendo conto delle relazioni con tutte le altre variabili.

   Vantaggi:

   - Mantiene la variabilità dei dati.
   - Preserva le relazioni tra le variabili.
   - Riduce il rischio di bias rispetto ai metodi semplici.

14.2.3.3 Applicazione Pratica: Imputazione Multipla con `mice` in R

Supponiamo di avere un dataset con alcune colonne numeriche che contengono valori mancanti. Possiamo utilizzare il pacchetto `mice` per imputare i dati mancanti.

Selezioniamo le colonne numeriche da imputare.

```
numeric_columns <- c("math1", "math2", "math3", "math4")
svy <- svy %>%
  mutate(across(all_of(numeric_columns), as.numeric))
```

Eseguiamo l'imputazione multipla.

```
imputed <- mice(
  svy[numeric_columns],
  m = 1,
  maxit = 10,
  method = "norm.predict",
  seed = 123
)
#>
#>  iter imp variable
#>   1   1  math3  math4
#>   2   1  math3  math4
#>   3   1  math3  math4
#>   4   1  math3  math4
#>   5   1  math3  math4
#>   6   1  math3  math4
#>   7   1  math3  math4
#>   8   1  math3  math4
#>   9   1  math3  math4
#>   10   1  math3  math4
```

Ottieniamo il dataset con i valori imputati.

```
svy_imputed <- complete(imputed)
```

Arrotondiamo i valori imputati (se necessario).

```
svy_imputed <- svy_imputed %>%
  mutate(across(everything(), round))
```

Sostituiamo i valori imputati nel dataset originale.

```
svy[numeric_columns] <- svy_imputed
```

Esaminiamo il risultato ottenuto.

```
svy |>
  summary()
#>      stu_id      grade_level       math1          math2          math3
#>  Min.   :1347   Min.   : 9.0   Min.   :2.0   Min.   :1.0   Min.   :2.0
#>  1st Qu.:1368   1st Qu.: 9.0   1st Qu.:3.0   1st Qu.:1.0   1st Qu.:3.0
#>  Median :1377   Median :10.0   Median :3.0   Median :2.0   Median :3.0
#>  Mean   :1376   Mean   :10.2   Mean   :3.2   Mean   :2.2   Mean   :3.2
#>  3rd Qu.:1387   3rd Qu.:11.0   3rd Qu.:4.0   3rd Qu.:3.0   3rd Qu.:4.0
#>  Max.   :1399   Max.   :12.0   Max.   :4.0   Max.   :4.0   Max.   :4.0
#>      math4
#>  Min.   :1.0
#>  1st Qu.:2.0
#>  Median :3.0
#>  Mean   :2.8
#>  3rd Qu.:4.0
#>  Max.   :4.0
```

14.2.3.4 Come Funziona l'Imputazione Multipla?

1. Ogni variabile con valori mancanti viene modellata come funzione delle altre variabili.
2. I valori mancanti vengono stimati iterativamente. In ogni iterazione, si utilizza l'output precedente come input per migliorare le stime.
3. Dopo un numero sufficiente di iterazioni, le stime si stabilizzano (convergenza).

In conclusione, l'imputazione multipla è una tecnica avanzata che permette di gestire i dati mancanti preservando la qualità delle analisi. Rispetto ai metodi semplici, consente di mantenere la variabilità e ridurre il rischio di bias, rendendola una scelta ideale per analisi psicologiche e di ricerca.

14.2.3.5 Aggiungere i Metadati

I *metadati* sono informazioni che descrivono i dati stessi, come etichette di variabili, etichette di valori, informazioni sull'origine dei dati, unità di misura

e altro ancora. Questi metadati sono essenziali per comprendere, documentare e condividere correttamente un dataset.

In R, i metadati sono gestiti in modo molto dettagliato e strutturato attraverso pacchetti come `haven`, `labelled`, e `Hmisc`. Questi pacchetti consentono di associare etichette ai dati, come etichette di variabili e di valori, e persino di gestire i valori mancanti con etichette specifiche.

- *Etichette di variabili*: Si possono aggiungere direttamente alle colonne di un DataFrame usando funzioni come `labelled::set_variable_labels()`.
- *Etichette di valori*: Possono essere aggiunte a variabili categoriali utilizzando `labelled::labelled()`.
- *Valori mancanti*: In R, è possibile etichettare specifici valori come mancanti usando `labelled::na_values<-`.

Questi strumenti rendono molto facile documentare un dataset all'interno del processo di analisi, assicurando che tutte le informazioni critiche sui dati siano facilmente accessibili e ben documentate.

Esaminiamo un esempio pratico. Consideriamo nuovamente il data set `svy`:

```
glimpse(svy)
#> Rows: 5
#> Columns: 6
#> $ stu_id      <int> 1347, 1368, 1377, 1387, 1399
#> $ grade_level <int> 9, 10, 9, 11, 12
#> $ math1       <dbl> 2, 3, 4, 3, 4
#> $ math2       <dbl> 1, 2, 4, 3, 1
#> $ math3       <dbl> 3, 2, 4, 4, 3
#> $ math4       <dbl> 3, 2, 4, 4, 1
```

Si noti che la variabile `math2` contiene un valore inamissibile, probabilmente un errore di battitura. Questo fa in modo che `math2` sia di tipo `char` mentre dovrebbe essere una variabile numerica. Correggiamo.

```
# Correzione di `math2`: Rimuovi valori non validi e converti in numerico
svy$math2 <- gsub("\\n", "", svy$math2)  # Rimuovi caratteri non validi come '\n'
svy$math2 <- as.numeric(svy$math2)      # Converte la variabile in numerico

# Visualizzazione del dataset corretto
print(svy)
#>   stu_id grade_level math1 math2 math3 math4
#> 1   1347           9     2     1     3     3
#> 2   1368          10     3     2     2     2
#> 3   1377           9     4     4     4     4
#> 4   1387          11     3     3     4     4
#> 6   1399          12     4     1     3     1
```

Definiamo le etichette di valore per le variabili `math1:math4`.

```
value_labels_math <- set_names(
  as.numeric(names(c(
    `1` = "strongly disagree",
    `2` = "disagree",
    `3` = "agree",
    `4` = "strongly agree"
  ))),
  c("strongly disagree", "disagree", "agree", "strongly agree")
)
```

Aggiungiamo le etichette di valore alle colonne `math1:math4`.

```
svy <- svy %>%
  mutate(across(starts_with("math"), ~ labelled(., labels = value_labels_math)))
```

Verifica delle etichette.

```
val_labels(svy$math1)
#> strongly disagree          disagree             agree    strongly agree
#>                 1                 2                 3                 4
```

```
svy
#>   stu_id grade_level math1 math2 math3 math4
#> 1   1347           9     2     1     3     3
#> 2   1368          10     3     2     2     2
#> 3   1377           9     4     4     4     4
#> 4   1387          11     3     3     4     4
#> 6   1399          12     4     1     3     1
```

14.2.3.5.1 Utilizzo delle Etichette in R con Variabili Numeriche

Le etichette dei valori (*value labels*) vengono utilizzate per rendere più leggibili e interpretabili le variabili numeriche, associando ad ogni valore un'etichetta descrittiva. Questo approccio è particolarmente utile in ambiti come la ricerca psicologica, dove le risposte ai questionari sono spesso codificate con numeri (ad esempio, 1 = "Strongly Disagree", 2 = "Disagree", ecc.) ma rappresentano concetti qualitativi.

Vantaggi dell'Uso delle Etichette

1. **Chiarezza nelle Analisi**: Le etichette descrittive rendono i dati più facilmente comprensibili senza dover ricordare il significato numerico di ciascun valore.

2. **Documentazione Integrata**: Permettono di incorporare metada-

ti direttamente nelle variabili, migliorando la trasparenza e riducendo il rischio di interpretazioni errate.

3. **Compatibilità con Software Statistici**: Molti strumenti (ad esempio, SPSS o Stata) utilizzano etichette di valori. Il pacchetto `haven` in R consente di gestire facilmente i dati etichettati esportati/importati da questi software.

Manipolazione di Variabili Etichettate

Anche se una variabile numerica è etichettata con `labelled` (ad esempio, tramite il pacchetto **`haven`**), essa conserva la sua natura numerica e può essere utilizzata in calcoli, modelli statistici, e trasformazioni. Le etichette non alterano il valore sottostante, ma lo arricchiscono con informazioni aggiuntive.

In conclusione, le etichette dei valori migliorano l'interpretabilità dei dati senza comprometterne la manipolabilità. Questo approccio è ideale per mantenere le variabili numeriche pienamente funzionali per analisi statistiche, mentre le etichette descrittive forniscono un contesto chiaro e leggibile.

14.2.3.6 Validazione dei Dati

La validazione dei dati è un passaggio fondamentale per garantire che il dataset soddisfi i criteri previsti e sia pronto per le analisi successive. Questo processo include il controllo della coerenza e della correttezza dei dati in base a specifiche regole definite dal dizionario dei dati. Alcune verifiche comuni includono:

- **Unicità delle righe**: Assicurarsi che ogni riga sia unica, verificando l'assenza di ID duplicati.
- **Validità degli ID**: Controllare che gli ID rientrino in un intervallo previsto (es. numerico).
- **Valori accettabili nelle variabili categoriali**: Verificare che variabili come `grade_level`, `int` e le colonne `math` contengano esclusivamente valori appartenenti a un set di valori validi.

Il pacchetto pointblank fornisce strumenti flessibili e intuitivi per eseguire verifiche di validazione e generare report dettagliati. Questo pacchetto consente di:

- **Definire le regole di validazione**: Specificare controlli come unicità, intervalli di valori e appartenenza a insiemi predefiniti.
- **Eseguire i controlli**: Applicare le regole di validazione su un dataset per identificare eventuali discrepanze.
- **Generare report interattivi**: Creare un riepilogo chiaro e visivo dei controlli, evidenziando eventuali errori o anomalie.

Con `pointblank`, è possibile integrare la validazione dei dati come parte di