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

Un modello di analisi fattoriale basato sulla distribuzione binomiale negativa per dati di scRNA-seq

Relatore Prof. Davide Risso
Dipartimento di Scienze Statistiche

Laureanda Clara Bagatin
Matricola N 1182889

Anno Accademico 2018/2019

So di non sapere

— Socrate

Indice

Elenco delle tabelle

Elenco delle figure

Introduzione

Negli ultimi anni il sequenziamento del DNA e dell'RNA è diventato sempre più rilevante negli ambiti della ricerca medica.
Si provi a pensare alla ricerca contro i tumori. Al giorno d'oggi le terapie sono sempre più personalizzate e specifiche per la mutazione genetica del singolo individuo e spesso per lo stesso tumore variano da paziente a paziente.
Oppure si provi a pensare agli studi sulle malattie genetiche che sono resi possibili solo grazie alla decodifica del genoma.
Senza questa tecnologia non ci sarebbero stati i grandi passi avanti compiuti dalla medicina negli ultimi anni.
Tuttavia, il sequenziamento è solo il primo tassello di un grande puzzle. Infatti, i biologi e i medici non traggono nessuna informazione utile dall'infinita sequenza di lettere che produce il sequenziamento, come è illustrato nella simpatica vignetta in Figura 1.
Ed è a questo punto che entrano in azione gli statistici. Servono tecniche statistiche costruite ad hoc per analizzare questa nuova tipologia di dati per poter estrarre le informazioni necessarie, affinché l'utente finale riesca a trarre delle conclusioni corrette.
Le tecniche per il sequenziamento sono in rapida evoluzione, infatti in pochi anni si è passati dai *microarrays* [1], all'RNA-seq, fino ad arrivare all'RNA-seq

[1] I microarrays sono dei supporti solidi sui quali vengono fissate delle piccole sequenze di DNA. Questa tecnica è stata sviluppata negli anni 90

Figura 1: Vignetta sul sequenziamento del genoma

a singola cellula (scRNA-seq). In particolare, quest'ultima tecnologia è abbastanza recente ed è molto diversa dal suo predecessore.

Quindi, risulta evidente che anche la statistica debba stare al passo con le nuove tecnolgie.

Proprio per questa serie di motivi si è deciso di incentrare questa tesi sull'analisi dei dati del sequenziamento dell'RNA a singola cellula.

Recentemente è stata proposta una nuova tecnologia che limita gli eventi di dropout, ovvero geni che non vengono rilevati anche se sono espressi nel campione. Questa tecnologia è detta UMI (Unique Molecular Identifiers) e permette di identificare in modo univoco le sequenze di RNA prima della fase di amplificazione.

Lo scopo di questa tesi è proporre un modello di analisi fattoriale per analizzare i dati di scRNA-seq con tecnologia UMI e confrontare i risultati con i metodi statistici più recenti.

Di seguito si descrive brevemente la struttura di questa tesi.

Nel Capitolo 1 si offre una panoramica generale sul sequenziamento dell'R-

NA: partendo dal concetto di espressione genica, si passa alla descrizione delle tecniche statistiche utilizzate per analizzare i dati di RNA-seq, fino ad arrivare a parlare dei dati di RNA-seq a singola cellula.
Successivamente, nel Capitolo 2 si illustrano tre modelli per l'analisi dei dati di RNA-seq a singola cellula. Inizialmente si presenta il modello binomiale negativo con inflazione di zeri (ZINB-WaVE). Si prosegue con il modello GLM-PCA, ovvero un'estensione dell'analisi delle componenti principali alla famiglia esponenziale, proposto per i dati scRNA-seq con tecnologia UMI. Ed infine, si descrive il modello binomiale negativo proposto per la prima volta in questa tesi.
Nel Capitolo 3 si confrontano i modelli appena citati tramite delle simulazioni: si verifica l'accuratezza del modello binomiale negativo e binomiale negativo con inflazione di zeri nella stima dei fattori latenti; successivamente si testano le capacità dei modelli nell'ambito della riduzione della dimensionalità dei dati in un'applicazione di clustering.
Inoltre, si confrontano i tempi computazionali impiegati dai modelli per stiamare i parametri.
Infine, nel Capitolo 4 si applica il modello binomiale negativo ad un dataset reale.
Le analisi presentate in questa tesi sono state effettuate con il software R.

Capitolo 1

Da RNA-seq a scRNA-seq

In questo capitolo si vuole illustrare brevemente in cosa consiste il sequenziamento dell'RNA, le principali tecniche statistiche sviluppate appositamente per analizzate i dati genomici e le nuove tecnologie proposte in questo ambito.

1.1 L'espressione genica

Sorge spontaneo chiedersi perché si parla di sequenziamento dell'RNA e non di sequenziamento del DNA.
All'interno del nucleo di tutte le cellule di un organismo vi è una copia identica di DNA: uguale sia nell'aspetto chimico-fisico sia nella quantità. Di conseguenza, analizzare il DNA di una cellula qualsiasi equivale ad analizzare il DNA di tutte le cellule di un organismo. È chiaro, allora, che ciò che differenzia le cellule non è il DNA.
Il DNA è suddiviso in sequenze, dette geni, che impartiscono gli ordini per la produzione delle proteine. Tuttavia, il DNA non può uscire dal nucleo e

per soddisfare il fabbisogno cellulare gli organismi hanno ideato un sistema molto efficacie: producono molte copie dei geni, le quali possono uscire dal nucleo con le informazioni necessarie per fabbricare le proteine. Queste copie prendono il nome di RNA.
Ma cosa differenzia una cellula da un'altra se il DNA è uguale per tutte?
La risposta è l'espressione genica: ogni cellula produce copie di RNA solo dei geni che le servono per vivere e per svolgere la propria funzione. Ad esempio, le cellule del pancreas "esprimono" il gene dell'insulina, mentre non esprimono il gene dell'emoglobina che è espresso nei globuli rossi.
Analizzando, quindi, l'RNA si cerca di capire l'attività cellulare sia in stati fisiologici che patologici. Lo scopo della ricerca è scoprire quali e quante copie di RNA vengono prodotte per ogni gene in una data cellula.
Questo assume un ruolo fondamentale, per esempio, nello studio dei tumori. Infatti, analizzando l'RNA, si studiano i meccanismi che portano ad una proliferazione incontrollata, alla metastasi, alla resistenza alle cure, ecc..

1.2 RNA-seq

Per poter misurare l'espressione genica si ricorre all'RNA-seq, ovvero al sequenziamento dell'RNA.
Sia il DNA che l'RNA sono una lunga sequenza alternata di basi azotate (Adenina, Guanina, Citosina, Timina/Uracile): una specifica sequenza identifica un gene. Lo scopo del sequenziamento è, quindi, decodificare la sequenza.
Nessuna tecnologia di sequenziamento è in grado di leggere l'intera sequenza di un gene, perciò l'RNA viene suddiviso in piccole sequenze (lunghe circa 100 basi) dette *reads*.
Proprio per questo motivo, dopo il sequenziamento, è necessario ricorrere

all'allineamento, ovvero si cerca di capire da quale gene proviene una determinata read.
Dopo aver effettuato l'allineamento, si procede con il conteggio del numero di reads che mappano su ogni gene. Questo conteggio rappresenta il livello di espressione del gene: più il numero è grande, più il gene è espresso.

1.3 Normalizzazione

Il sequenziamento dà come risultato una matrice di dati $n \times J$ (dove n indica il numero dei campioni e J il numero dei geni) che deve essere analizzata con tecniche statistiche specifiche per poter trarre delle conclusioni.
Tuttavia, prima di procedere con l'inferenza è opportuno normalizzare i dati, in quanto contengono errori sistematici e distorsioni dovuti alla tecnologia di sequenziamento. Ad esempio il numero di reads totale può essere diverso fra campioni oppure ci possono essere differenze nei protocolli di sequenziamento.
La normalizzazione può essere interna o tra campioni.
La normalizzazione interna non è sempre necessaria e va effettuata quando il contenuto di GC (percentuale di Guanina e Citosina nel campione) è diversa fra le reads. Il GC bias descrive la relazione tra il contenuto di GC e la copertura delle reads di un genoma. Cioè, una regione genomica con un contenuto GC più elevato tende ad avere più reads che coprono quella regione (Chen et al. 2013).
La normalizzazione tra campioni è sempre necessaria, in quanto il numero totale di reads per campione influenza molto i risultati (più reads vengono sequenziate, più i conteggi dei geni in quel campione saranno elevati).

1.4 Modelli e metodi statistici

Lo scopo dell'inferenza è identificare i geni differenzialmente espressi tra due o più condizioni. Questo si traduce in un test d'ipotesi per ogni gene:

H_0: Il gene j è espresso allo stesso modo tra le diverse condizioni

H_1: Il gene j è differenzialmente espresso tra le diverse condizioni.

1.4.1 Test multipli: False Discovery Rate

Si è appena detto che bisogna testare un'ipotesi per ogni gene. Siamo in presenza di test multipli indipendenti ed è quindi opportuno utilizzare delle tecniche che tengano in considerazione la molteplicità.
Tuttavia, le procedure che controllano il Family Wise Error Rate (FWER), come la correzione di Bonferroni o il metodo di Holm, sono spesso troppo conservative, ovvero tendono a rifiutare poche ipotesi nulle. Questo dipende dalla definizione di FWER, infatti controllare la probabilità di osservare almeno un falso positivo [1] tra i test effettuati è una condizione troppo stringente.
Si può pensare, allora, di controllare la probabilità di ottenere una certa proporzione di falsi positivi. Il False Discovery Rate è la frazione attesa di falsi positivi tra le ipotesi nulle che sono state dichiarate significative (ovvero che sono state rifiutate): $E[\frac{FP}{R}]$ (Tabella 1.1).
Una procedura che controlla il FDR è il metodo di Benjamini-Hochberg (Benjamini e Hochberg 1995). Supponendo di avere J ipotesi nulle $H_1, ..., H_J$ con i corrispettivi p-value $p_1, ..., p_J$, la procedura è composta dai seguenti punti:

1. Si ordinano i p-value in ordine crescente: $p_{(1)}, ..., p_{(J)}$.

[1] Non rifiutare l'ipotesi nulla quando in realtà è falsa.

Tabella 1.1: Definizione di Veri Positivi, Veri negativi, Falsi Positivi e Falsi Negativi

	Ipotesi non rifiutate	Ipotesi rifiutate	**Totale**
Ipotesi nulle	VN	FP Errore di I tipo	J_0
Ipotesi alternative	FN Errore di II tipo	VP	$J - J_0$
Totale	J-R	R	J

2. Per un livello di significatività α si considera il più grande k tale che $p_{(k)} \leq \frac{k}{J}\alpha$.

3. Si rifiutano le ipotesi $p_{(1)}, ..., p_{(k)}$ e non si rifiutano le altre.

Se si vuole ottenere un valore di significatività, oltre a sapere quante e quali ipotesi rifiutare, non si può osservare il p-value del test, ma è necessario correggerlo. Il p-value aggiustato secondo il metodo di Benjamini-Hocberg si ottiene moltiplicando il p-value per il numero totale di test effettuati e dividendolo per il rango del j-esimo p-value:

$$\tilde{p}_j = \min\left\{1, \frac{J}{r_j} p_j\right\}.$$

1.4.2 Descrizione dei dati

Dal Paragrafo 1.2 si può intuire che i dati da analizzare sono dati di conteggio. Il modello di regressione di Poisson è un modello parametrico per dati di conteggio. Tuttavia, va ricordato che il modello di Poisson assume che

la varianza sia uguale alla media. Solitamente questo non è verificato per i dati di RNA-seq, perché presentano sovra-dispersione (Figura 1.1).

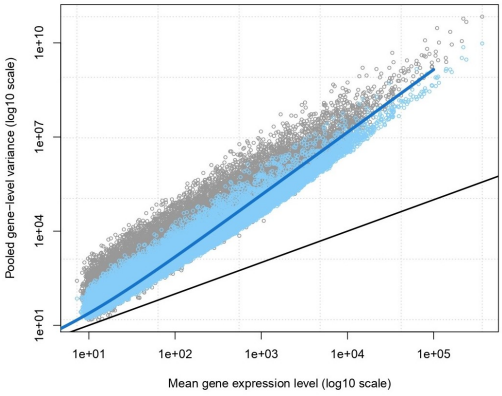


Figura 1.1: Sovra-dispersione dati RNA-seq

Di seguito vengono presentati alcuni modelli proposti per i dati di RNA-seq.

1.4.3 Modello lineare generalizzato: Binomiale Negativa

La distribuzione binomiale negativa è una generalizzazione della distribuzione di Poisson che può essere utilizzata per dati di conteggio sovra-dispersi. La binomiale negativa è una mistura Gamma-Poisson: se $Y|\lambda \sim Poi(\lambda)$ e $\lambda \sim Ga(\phi^{-1}, (\phi\mu)^{-1})$, allora $Y \sim NB(\mu, \phi)$, con $E[Y] = \mu$ e $Var(Y) = \mu + \mu^2\phi$. Quando ϕ=0 si ricade nella distribuzione di Poisson.
Si definisce Y_{ij} il numero di reads che mappano sul gene j=1,...,J nel campione i=1,...,n. Per ogni gene j si stima un modello di regressione in cui si assume che $Y_{ij} \sim NB(\mu_{ij}, \phi_j)$ e si definisce $\log E[Y_{ij}|X] = \log \mu_{ij} = X\beta_j$.
Se la normalizzazione venisse effettuata prima dell'inferenza, si perderebbe la natura di conteggio dei dati; di conseguenza, si aggiunge un *offset* al predittore lineare per normalizzare i dati ($\log \mu_{ij} = X\beta_j + O_{ij}$).
É importante ricordare che la distribuzione binomiale negativa è una famiglia

di dispersione esponenziale solo se il parametro di dispersione (ϕ) è noto. Inoltre, i modelli lineari generalizzati sono definiti solo per famiglie esponenziali. Perciò, è necessario stimare il parametro di dispersione prima di proseguire con la stima dei coefficienti di regressione con un GLM.
Una volta stimati i parametri di interesse β_j, per identificare i geni differenzialmente espressi si può testare l'ipotesi di nullità dei coefficienti con il test di Wald o il test del rapporto di verosimiglianza.

Stima del parametro di dispersione

Il parametro di dispersione ϕ è gene-specifico, perciò occorre stimare J parametri, spesso con una numerosità campionaria limitata. Sono stati proposti diversi metodi, di seguito ne vengono proposti due.
La stima di massima verosimiglianza profilo approssimata consiste in una media pesata tra la verosimiglianza profilo per il gene j e la verosimiglianza profilo globale (Robinson e Smyth 2007).
La stima Bayesiana empirica basata sulla regressione sulla media schiaccia le stime di ϕ_j verso una curva che modella il trend media-varianza; l'approccio Bayesiano empirico è adottato per scegliere quanto devono essere schiacciate (Love, Huber e Anders 2014).

1.4.4 Modello lineare: Metodo *voom*

Il modello di regressione binomiale negativo presenta alcuni problemi. Ad esempio, le distribuzioni della statistica di Wald e del rapporto di verosimiglianza sono asintotiche e va ricordato che siamo in presenza di campioni piccoli. Inoltre, non si tiene conto dell'incertezza della stima del parametro di dispersione, ma si considera noto durante la stima del modello.
Si è proposto, quindi, di modellare il logaritmo dei valori normalizzati inve-

ce dei conteggi, così da poter utilizzare le statistiche test esatte del modello lineare.

Tuttavia, la distribuzione normale non è adatta a causa della forte eteroschedasticità presente nei dati di RNA-seq (anche dopo la trasformazione logaritmica) e la relazione quadratica tra media e varianza.

Il metodo *voom* incorpora la relazione media-varianza in una procedura basata sui modelli lineari (Figura 1.2). Di seguito vengono riportati i passaggi dell'algoritmo (Law et al. 2014):

1. Si stima la relazione media-varianza in modo non parametrico (ad esempio con una regressione *loess*).

2. Questa relazione è usata per stimare la varianza di ogni osservazione.

3. L'inverso della varianza è usato come peso per ogni osservazione.

4. Si stima un modello lineare pesato: $\hat{\beta} = (X^T W X)^{-1} X^T W Y$, dove W è la matrice dei pesi.

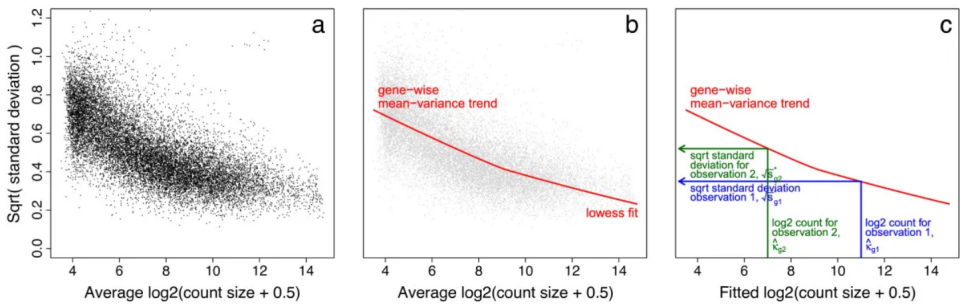


Figura 1.2: Stima della relazione media-varianza metodo *voom*

1.4.5 Metodo non parametrico

I metodi non parametrici sono un'alternativa ai modelli parametrici illustrati precedentemente. Un esempio è l'approccio basato sul test di Wilcoxon o sul test di Kruskal-Wallis (se i gruppi sono più di due) in cui si calcola il rango per ogni gene j, indipendentemente dal gruppo di appartenenza (J. Li e Tibshirani 2013).
Tuttavia, questo metodo è applicabile solo se i campioni hanno lo stesso numero totale di reads. Affinché tutti i campioni abbiano lo stesso numero totale di reads, si ricorre al ricampionamento: per B volte e per ogni campione si generano nuovi conteggi $(Y_{ij}^{'})$ da una Poisson con media $(\bar{N}/N_i)Y_{ij}$, dove N_i è il numero totale di reads per il campione i e $\bar{N} = (\prod_{i=1}^{n} N_{ij})^{\frac{1}{n}}$.
Supponendo che ci siano K gruppi ognuno con n_k campioni, indicando con C_k l'insieme delle osservazioni appartenenti al gruppo k e con R_{tj} il rango del conteggio ricampionato per il gene j, la statistica test è data dalla media di tutte le B statistiche test dei ricampionamenti:

$$T_j = \frac{1}{B}\sum_{b=1}^{B}\left(\frac{12}{n(n+1)}\sum_{k=1}^{K}\frac{(\sum_{t\in C_k} R_{tj}(Y'^b))^2}{n_k} - 3(n+1)\right).$$

Calcolo del FDR

A causa del ricampionamento, la distribuzione delle statistiche test T_j non è nota e quindi non è possibile calcolare il p-value nel modo in cui si ottiene per il test di Wilcoxon o di Kruskal-Wallis e poi ricavare il FDR.
Perciò, si utilizza una procedura basata sulle permutazioni per stimare il FDR:

1. Si calcolano $T_1, ..., T_J$.

2. Si permutano i valori di Y'_{ij} L volte e per la l-esima permutazione si calcolano $T_1^l, ..., T_J^l$.

3. Per una serie di valori di cutoff C, si calcolano

$$\widehat{FP} = \frac{1}{L}\sum_{j=1}^{J}\sum_{l=1}^{L} I_{(|T_j^l|>C)} \quad \text{e} \quad \hat{R} = \sum_{j=1}^{J} I_{(|T_j|>C)}\,.$$

4. Si stima il FDR per il cutoff C come $\widehat{FDR}_C = \pi_0 \frac{\widehat{FP}}{\hat{R}}$.
   π_0 è la vera proporzione di ipotesi nulle non rifiutate, stimata da $\hat{\pi}_0 = 2\sum_{j=1}^{J} I_{(|T_j|\leq q)}/J$, dove q è la mediana di tutti i valori $|T_j^l|$.

1.5 Batch effects

I dati ad alta dimensionalità spesso soffrono di variabilità indesiderata. Questa può essere causata dalle diverse condizioni che possono cambiare all'interno di un esperimento o tra esperimenti diversi, come ad esempio il giorno e l'ora in cui si esegue il sequenziamento, il tecnico che aziona il macchinario, i reagenti utilizzati, la temperatura del laboratorio.
La variabilità indesiderata complica le analisi e può portare a conclusioni errate: dichiarare differenzialmente espressi geni che in realtà non lo sono e viceversa.
I batch effects sono quindi quelle variabili (osservate e non) che influenzano i valori dell'esperimento senza riflettere un segnale biologico.
Si potrebbe pensare che la normalizzazione (Paragrafo 1.3) sia sufficiente ad eliminare gli effetti di batch, ma va sottolineato che questi possono agire solo su un sottoinsieme di geni o in modo diverso su geni diversi.
Agendo a livello di disegno sperimentale e durante l'analisi dei dati, si può ridurre l'effetto di queste variabili confondenti.

Di fatto, costruire un disegno sperimentale che tenga conto dei possibili confondenti non è sempre facile: ad esempio quando analizziamo dei dati pubblici, oppure quando non possiamo randomizzare i campioni tra i vari batch, o in aggiunta si possono verificare situazioni che vanno al di sopra del nostro controllo. Per questo diventano molto importanti le tecniche statistiche che si utilizzano.

1.5.1 Rimozione dei batch effects noti

Se la variabile confondente è osservata e il suo effetto è lineare, si può includere come covariata nel modello.
Tuttavia, se l'effetto non è lineare si può utilizzare il metodo *ComBat* (Combating Batch Effects When Combining Batches of Gene Expression Microarray Data). Questo modello include un effetto additivo e uno moltiplicativo, cosicché i dati abbiano stessa media e stessa varianza in ogni batch. Il modello specificato è:

$$Y_{ijk} = \alpha_j + X\beta_j + \gamma_{jk} + \delta_{jk}\epsilon_{ijk},$$

dove

- α_j è la media globale di espressione del gene j;
- X è la matrice del disegno;
- β_j è il vettore dei coefficienti di regressione del gene j;
- γ_{jk} e δ_{jk} rappresentano i batch effects additivi e moltiplicativi;
- $\epsilon_{ijk} \sim N(0, \sigma_j^2)$ sono gli errori.

L'algoritmo *ComBat* è strutturato nel modo seguente (Johnson, C. Li e Rabinovic 2007):

1. Si standardizzano i dati in modo che ogni gene abbia stessa media e varianza: si ottengono le stime OLS di α_j, β_j e γ_{jk} indipendentemente per ogni gene con il vincolo $\sum_k n_k \hat{\gamma}_{jk} = 0$. Successivamente si stima $\hat{\sigma}_j^2 = \frac{1}{N} \sum_{ik} \left(Y_{ijk} - \hat{\alpha}_j - X\hat{\beta}_j - \hat{\gamma}_{jk}\right)^2$ e si standardizzano i dati

$$Z_{ijk} = \frac{Y_{ijk} - \hat{\alpha}_j - X\hat{\beta}_j}{\hat{\sigma}_j} .$$

2. Si stimano i parametri del modello Bayesiano empirico: si assume che $Z_{ijk} \sim N(\gamma_{jk}, \delta_{jk}^2)$ e che le distribuzioni a priori dei parametri siano $\gamma_{jk} \sim N(\gamma_k, \tau_k^2)$ e $\delta_{jk}^2 \sim IG(\lambda_k, \theta_k)$, dove IG$(\cdot)$ è la distribuzione Gamma Inversa e gli iperparametri sono stimati con il metodo dei momenti.

3. Si rimuovono gli effetti di batch:

$$Y_{ijk}^* = \hat{\sigma}_j \frac{Z_{ijk} - \hat{\gamma}_{jk}^*}{\hat{\delta}_{jk}^*} + \hat{\alpha}_j + X\hat{\beta}_j ,$$

   dove $\hat{\gamma}_{jk}^*$ e $\hat{\delta}_{jk}^*$ sono le stime dei parametri a posteriori.

Tuttavia, con questo approccio si corre il rischio di eliminare anche la variabilità di interesse biologico, soprattutto se è correlata con la variabilità indesiderata. Infatti, i dati vengono usati due volte: una per stimare le quantità che servono per "aggiustare" i dati per gli effetti di batch e l'altra per stimare il modello, portando ad una sovra-stima dei gradi di libertà del modello finale.

1.5.2 Rimozione dei batch effects latenti

In alcuni casi non si hanno a disposizione le informazioni riguardo le variabili di batch, o perché non sono state registrate o perché le variabili che abbiamo a disposizione non spiegano la variabilità indesiderata.

Uno dei metodi proposti per risolvere questo problema è il metodo *RUV* (Remove Unwanted Variation). L'assunzione che sta alla base è che esistono geni, detti controlli negativi, la cui espressione non è correlata con la variabile biologica d'interesse. Ovvero, questi geni non sono differenzialmente espressi (Risso, Ngai et al. 2014).
Un esempio di controlli negativi sono i geni *spike-ins*, ovvero geni sintetici che vengono aggiunti in proporzione costante a tutti i campioni. Se l'esperimento non prevede ciò, si possono utilizzare i geni *housekeeping*, ovvero geni che hanno funzioni base per la sopravvivenza della cellula e che sono espressi ad alti livelli in tutte le cellule.

Considerando il seguente modello

$$\log E[Y|W, X, O] = W\alpha + X\beta + O, \tag{1.1}$$

dove Y è la matrice dei conteggi dei J geni negli n campioni, X è la matrice $n \times p$ del disegno associata alla matrice $p \times n$ dei coefficienti β, W è una matrice $n \times k$ che contiene i fattori di variazione indesiderata associata alla matrice $k \times n$ dei parametri di disturbo α, e O è la matrice $n \times J$ degli offset, l'algoritmo *RUV* è strutturato nel modo seguente:

1. Si identifica un insieme di J_c controlli negativi. Per definizione $\beta_c = 0$, di conseguenza il modello può essere scritto come
   $\log E[Y_c|W, X, O_c] = W\alpha_c + O_c$ (il pedice c indica che stiamo considerando solo i controlli negativi).

2. Si definisce $Z = \log Y - O$ e Z^* la versione di Z con media delle colonne pari a zero.

3. Si esegue la decomposizione ai valori singolari (SVD) di Z_c^*: $Z_c^* = U\Lambda V^T$, dove U è una matrice ortogonale $n \times n$, le cui colonne sono

i vettori singolari di sinistra, Λ è una matrice diagonale rettangolare $n \times J_c$ che contiene i valori singolari e V è una matrice ortogonale $J_c \times J_c$, le cui colonne sono i vettori singolari di destra. Dato un valore di k, si stima con i primi k valori singolari (e ponendo i restanti a zero) $\widehat{W\alpha_c} = U\Lambda_k V^T$. Di conseguenza, $\hat{W} = U\Lambda_k$.

4. Sostituendo $\hat{W}$ nel modello (1.1), si possono stimare con OLS α e β.

É opportuno tenere in considerazione che non esiste una procedura per stimare il numero di fattori latenti, k. La scelta di k dovrebbe essere guidata da considerazioni che includono la dimensione del campione, l'entità degli effetti tecnici acquisiti dai primi k fattori e dall'osservazione delle analisi esplorative effettuate con diversi valori di k.

1.6 scRNA-seq

ScRNA-seq è una nuova tecnologia di sequenziamento dell'RNA che viene ampiamente utilizzata dal 2014 ed è sempre più applicata nella ricerca.
Ma come si differenzia dalla precedente tecnica, RNA-seq?
La differenza non è affatto banale. Infatti, RNA-seq misura il livello di espressione medio di un gene tra tutte le cellule di un dato tessuto (o organismo o linea cellulare): si preleva un insieme di cellule da un campione, si estrae l'RNA e si procede con il sequenziamento, assumendo che le cellule siano omogenee. Di conseguenza, alla fine del processo non si ha il livello di espressione di una singola cellula, ma di un gruppo di esse, per questo motivo questa tecnologia viene definita "bulk" (letteralmente RNA-seq all'ingrosso). In altre parole, RNA-seq è paragonabile ad un frullato di frutta in cui non si riesce a distinguere perfettamente il gusto di ogni ingrediente, mentre scRNA-seq è

simile ad una macedonia, in cui si riesce a gustare e a distinguere ogni singolo pezzo di frutta (Figura 1.3).

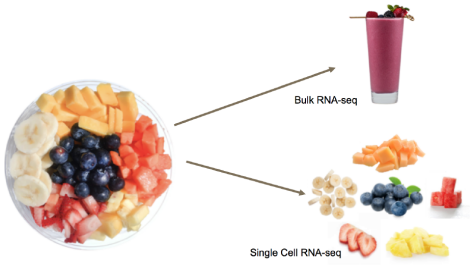


Figura 1.3: Differenza naif tra RNA-seq e scRNA-seq

É noto, tuttavia, che l'espressione genica è eterogenea anche in tipi cellulari simili: un esempio di ciò è la grande diversità cellulare in un tumore o durante lo sviluppo embrionale.
Risulta quindi necessario l'utilizzo di tecniche di sequenziamento che estraggano l'RNA da ogni singola cellula (Figura 1.4). Questa nuova tecnologia è nota con il nome di Single-Cell RNA-seq.
Esistono diversi metodi e protocolli per l'acquisizione delle cellule.
Nei metodi a piastre, le cellule sono isolate manualmente o attraverso uno smistamento automatico e poste in micro-pozzetti. Ciò permette di controllare le cellule al microscopio prima del sequenziamento, così da evitare di sequenziare cellule che stanno morendo o due cellule che sono riamaste attaccate (doublets). Tuttavia, possono essere processate solo poche cellule per volta, poiché si tratta di un protocollo laborioso e manuale.
I metodi basati sui micro-fluidi automatizzano molti dei processi che nel metodo precedente vengono svolti manualmente ed è sempre possibile controllare le cellule prima del sequenziamento. Rimane comunque limitato il numero di

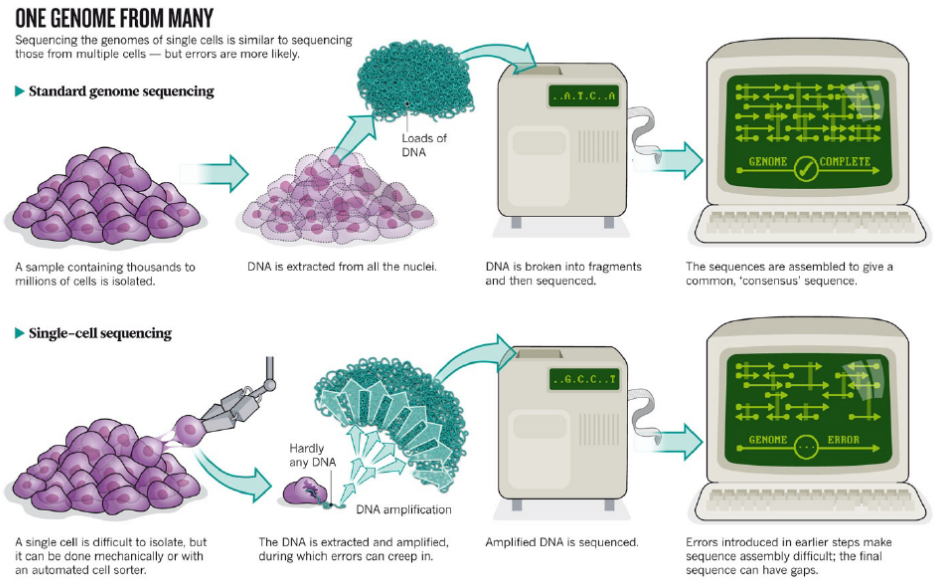


Figura 1.4: Differenza tra RNA-seq e scRNA-seq

cellule che si possono sequenziare (96) e questa tecnologia è spesso soggetta a effetti di batch e ai doublets.

La tecnologia basata su gocce (droplets) consiste nell'incapsulare in una goccia di gel la cellula con una perlina (bead) che contiene gli enzimi per costruire la libreria (insieme delle reads) e un codice a barre univoco che identifica la cellula (Figura 1.5). Questo codice serve per riconoscere a quale cellula appartiene una certa read che viene sequenziata. A differenza dei due protocolli precedenti, questo metodo permette di ottenere migliaia di cellule da ogni campione, ma a causa dei costi elevati per il sequenziamento, vengono sequenziate poche reads per cellula (Macosko et al. 2015).

Anche se il sequenziamento a singola cellula rappresenta un grande passo in avanti per la ricerca, questa tecnologia non è priva di problematiche.

La quantità di RNA estratta da una cellula è molto poca, di conseguenza è opportuno amplificare l'RNA prima di procedere con il sequenziamento.

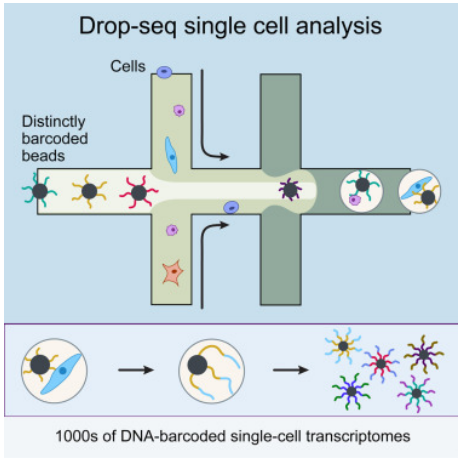


Figura 1.5: Descrizione del protocollo basato sui droplets

L'amplificazione porta a delle distorsioni nei dati (amplification bias), infatti i geni che vengono amplificati, possono essere amplificati a dismisura. Invece, alcuni geni, in particolare i geni che sono poco espressi, non vengono amplificati, pertanto non verranno sequenziati, portando ad avere un eccesso di zeri nella matrice dei conteggi (Hicks, Townes et al. 2018).

Anche i doublets (cellule doppie nello stesso pozzetto/goccia) rappresentano un problema, perché si pensa di misurare l'espressione genica di una cellula, quando in realtà è la somma delle espressioni di due.

Inoltre, nel caso del protocollo basato sui droplets può succedere di sequenziare gocce vuote, cioè che contengono solo l'RNA di cellule che si sono rotte durante il processo e che è immerso nel gel.

Un'altra considerazione importante a livello statistico è l'elevata dimensionalità dei dati che si vanno ad analizzare. Mentre con RNA-seq in alcuni casi si osserva una bassa numerosità campionaria, con scRNA-seq si possono analizzare migliaia di cellule.

Questi dati, inoltre, presentano una variabilità maggiore, infatti, con RNA-

seq si misura il livello di espressione media di un gene in un campione e non l'espressione in una singola cellula.
L'eccesso di zeri e l'elevata variabilità comportano che la distribuzione binomiale negativa non si adatti bene ai dati. Dunque, le tecniche utilizzate per analizzare i dati RNA-seq non sono più idonee per scRNA-seq ed è necessario sviluppare nuovi metodi per poter modellare i dati in maniera corretta.
Nel Paragrafo 1.3 si è detto quanto sia importante la normalizzazione dei dati per bulk RNA-seq, soprattutto per eliminare le distorsioni dovute ai molteplici passaggi che vengono effettuati durante il sequenziamento. Sembra sensato pensare che anche i dati di scRNA-seq debbano essere normalizzati. Tuttavia, Vallejos et al. (2017) hanno dimostrato che, applicando le normalizzazioni utilizzate per bulk RNA-seq, identificare quale tecnica di normalizzazione sia la migliore per un determinato set di dati è molto difficile per scRNA-seq. Infatti, diverse tipologie di normalizzazione influenzano molto i risultati del clustering o dell'identificazione dei geni più variabili o di altre analisi. Nella Figura 1.6 si nota quanto siano distorte le stime del fattore di normalizzazione (il metodo *Scran* è stato sviluppato appositamente per i dati di scRNA-seq, infatti ha una prestazione migliore degli altri metodi).

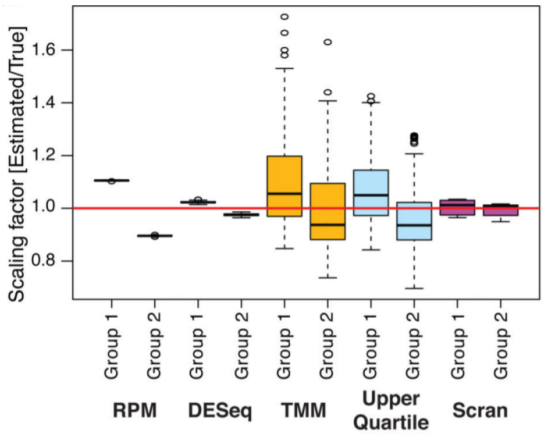


Figura 1.6: Rapporto tra il fattore di normalizzazione stimato e reale per diverse tecniche di normalizzazione in un dataset simulato costituito da due gruppi di cellule

1.6.1 Unique Molecular Identifiers

La distorsione dell'amplificazione può essere ridotta applicando un'etichetta univoca ad ogni read prima di procedere con l'amplificazione. Come si vede in Figura 1.7, ogni read viene etichettata con un colore diverso in modo che si possa distinguere dalle altre; dopo aver effettuato l'amplificazione, si procede con il conteggio: ad esempio, per la prima cellula il gene azzurro non ha 15 reads allineate, ma 4 che corrispondono ai colori delle etichette. In questo modo, non si contano quante reads sono state allineate sul gene, ma quante etichette distinte sono state allineate, così si può risalire con una certa accuratezza al numero originale di molecole prima dell'amplificazione (Islam et al. 2014). Le etichette sono composte da 4-10 nucleotidi e prendono il nome di UMI (Unique Molecular Identifiers).

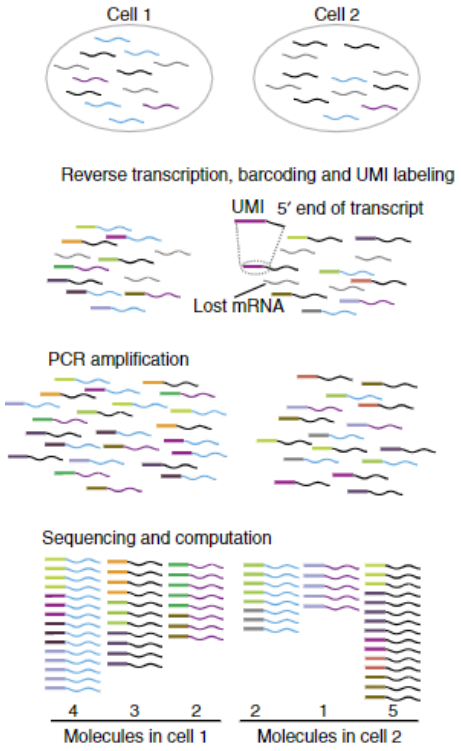


Figura 1.7: Conteggio delle reads con UMI

Capitolo 2

Alcuni modelli per dati di scRNA-seq

In questo capitolo vengono presentati tre modelli studiati appositamente per i dati di scRNA-seq. Vengono illustrati un modello binomiale negativo con inflazione di zeri (ZINB-WaVE) e una generalizzazione dell'analisi delle componenti principali (GLM-PCA) che è stata studiata per la tecnologia UMI. Infine si presenta un modello binomiale negativo proposto per la prima volta in questa tesi.

2.1 Modello Binomiale Negativo con Inflazione di Zeri

Nel Paragrafo 1.6 si è parlato della selection bias. È una conseguenza dei ripetuti cicli per l'amplificazione dell'RNA che avvengono prima del sequenziamento, necessari a causa della scarsissima quantità di RNA estratto da una cellula. Un risultato della distorsione dell'amplificazione sono i così detti *dropouts*, ovvero geni che non vengono rilevati anche se sono espressi nel campione (probabilmente perché molto poco espressi). Questo si traduce in

un eccesso di zeri nella matrice dei conteggi.

Un altro svantaggio dal punto di vista statistico, ma un grande vantaggio dal punto di vista biologico, è l'elevata dimensionalità dei dati. Infatti, se dal punto di vista biologico si hanno più informazioni riguardati l'espressione delle cellule prelevate da un organismo, dal punto di vista statistico si rischia di incorrere nella maledizione della dimensionalità. Risulta, quindi, necessario l'uso di tecniche specifiche per la riduzione della dimensionalità, affinchè i dati siano più trattabili da un punto di vista statistico e computazionale, per cercare di diminuire il rumore di fondo, preservando il segnale di interesse. La scelta della tecnica per la riduzione della dimensionalità diventa, perciò, un punto cruciale nell'analisi dei dati.

L'analisi delle componenti principali (PCA) è la tecnica più conosciuta e più usata per ridurre la dimensionalità dei dati. Tuttavia, si è notato che la prima e la seconda componente spesso dipendono più dalla percentuale di geni con almeno una read per cellula (*detection rate*) che da un reale segnale biologico, come si può vedere in Figura 2.1 (Hicks, Teng e Irizarry 2015).

Risso, Perraudeau et al. (2018) hanno proposto un modello binomiale negativo con inflazione di zeri, Zero-Inflated Negative Binomial Wanted Variation Extraction (ZINB-WaVE), per estrarre il segnale dai dati, tenendo conto dell'eccesso di zeri, della sovra-dispersione e della natura di conteggio dei dati. Dati n campioni (solitamente n singole cellule) e J geni, si indica con Y_{ij} il conteggio del gene j $(j = 1, ..., J)$ nel campione i $(i = 1, ..., n)$ e con Z_{ij} una variabile indicatrice che è posta uguale a uno se il gene j è un dropout nella cellula i e zero altrimenti. Inoltre, si pongono

$$\mu_{ij} = E[Y_{ij}|Z_{ij} = 0, X, V, W]$$

e

$$\pi_{ij} = Pr(Z_{ij} = 1|X, V, W)\,.$$

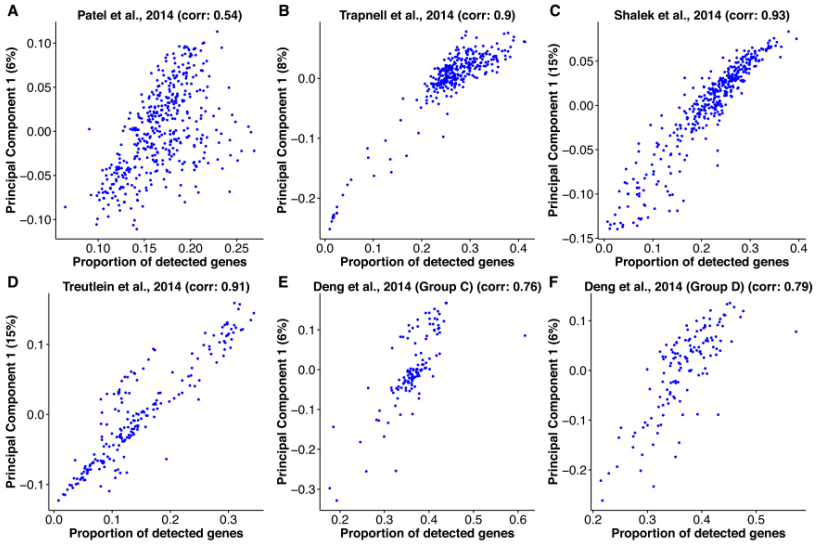


Figura 2.1: Relazione tra la prima componente principale e il detection rate in diversi studi

Y_{ij} si distribuisce come una binomiale negativa con inflazione di zeri (ZINB) che ha funzione di probabilità

$$f_{ZINB}(y;\mu,\theta,\pi) = \pi\delta_0(y) + (1-\pi)f_{NB}(y;\mu,\theta)\,,$$

dove

$$f_{NB}(y;\mu,\theta) = \frac{\Gamma(y+\theta)}{\Gamma(y+1)\Gamma(\theta)}\left(\frac{\theta}{\theta+\mu}\right)^{\theta}\left(\frac{\mu}{\mu+\theta}\right)^{y} \tag{2.1}$$

è la funzione di probabilità di una binomiale negativa con media $\mu \geq 0$ e inverso del parametro di dispersione $\theta > 0$. $\delta_0(\cdot)$ è la funzione di Dirac e $\pi \in [0,1]$ può essere interpretato come la probabilità che si osservi uno zero anziché il conteggio effettivo.

I parametri vengono modellati nel modo seguente:

$$\ln(\mu_{ij}) = (X\beta_\mu + (V\gamma_\mu)^T + W\alpha_\mu + O_\mu)_{ij} \tag{2.2}$$

$$\text{logit}(\pi_{ij}) = (X\beta_\pi + (V\gamma_\pi)^T + W\alpha_\pi + O_\pi)_{ij} \tag{2.3}$$

$$\ln(\theta_{ij}) = \zeta_j \tag{2.4}$$

dove:

- μ è una matrice $n \times J$ che indica il livello di espressione medio del gene j nel campione i;
- π è una matrice $n \times J$ che indica la probabilità di dropouts del gene j nel campione i;
- X è una matrice $n \times M$ di covariate osservate a livello di campione, associata alla matrice $M \times J$ dei coefficienti β;
- V è una matrice $J \times L$ di covariate osservate a livello di gene, associata alla matrice $L \times n$ dei coefficienti γ;
- W è una matrice $n \times K$ di covariate non osservate a livello di campione, associata alla matrice $K \times J$ dei coefficienti α;
- O è una matrice $n \times J$ di offsets;
- ζ è un vettore di dimensione J di parametri di dispersione gene-specifici su scala logaritmica.

Si può osservare che il modello proposto estende il metodo *RUV* (Paragrafo 1.5.2) alla distribuzione ZINB, infatti *RUV* è stato implementato solo per regressioni lineari e log-lineari. Differisce nell'interpretazione di *RUV* nel termine $W\alpha$, che non è necessariamente considerato variabilità indesiderata.

Questo termine si riferisce generalmente a variabilità sconosciuta, che potrebbe essere dovuta a effetti tecnici indesiderati (come in RUV), come ad esempio gli effetti di batch, o ad effetti biologici di interesse, come il ciclo cellulare o la differenziazione cellulare. Di conseguenza, W è una matrice a bassa dimensionalità che può essere usata per il clustering e l'analisi pseudo-temporale [1].

Si è detto quanto sia complicato normalizzare i dati scRNA-seq. Gli autori affermano che l'intercetta gene-specifica γ (che corrisponde ad una colonna di uno nella matrice delle covariate gene-specifiche V) agisce da fattore di normalizzazione globale, rendendo non necessaria la normalizzazione preliminare dei dati.

L'algoritmo ZINB-WaVE è composto da una fase di inizializzazione dei parametri e da una fase di ottimizzazione. L'inizializzazione è necessaria in quanto la stima dei parametri avviene per via numerica, a causa delle non convessità della funzione di verosimiglianza. Si cerca, quindi, di ottimizzare la funzione obiettivo partendo da valori iniziali dei parametri che siano plausibili.
La funzione di log-verosimiglianza è

$$\ell(\beta, \gamma, W, \alpha, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{J} \ln f_{ZINB}(Y_{ij}; \mu_{ij}, \theta_{ij}, \pi_{ij}) \,,$$

dove $\beta = (\beta_\mu, \beta_\pi)$, $\gamma = (\gamma_\mu, \gamma_\pi)$, $\alpha = (\alpha_\mu, \alpha_\pi)$; μ_{ij}, θ_{ij} e π_{ij} dipendono da $(\beta, \gamma, W, \alpha, \zeta)$ tramite le Equazioni (2.2)-(2.4).
Per ridurre l'overfitting e per aumentare la stabilità del problema di ottimizzazione in un contesto di molti parametri, si utilizza un approccio basato

[1] L'analisi pseudo-temporale cerca di ordinare le cellule secondo un ordine cronologico, dalla più immatura alla più matura

sulla massima verosimiglianza penalizzata:

$$\max_{(\beta,\gamma,W,\alpha,\zeta)} \{\ell(\beta,\gamma,W,\alpha,\zeta) - \text{Pen}(\beta,\gamma,W,\alpha,\zeta)\}\,.$$

La penalizzazione schiaccia i parametri verso lo zero, eccetto per le intercette, che non vengono penalizzate, e i parametri di dispersione, che vengono schiacciati verso un valore costante tra i geni.

Durante l'inizializzazione si separano i conteggi pari a zero dagli altri.

Approssimando i conteggi non-zero ad una distribuzione log-normale, si risolve la regressione ridge alternando la stima dei parametri β_μ e γ_μ fino a convergenza.

Successivamente, W e α_μ vengono stimati decomponendo a valori singolari i residui $D = \ln(Y) - O_\mu - X\hat{\beta}_\mu - (V\hat{\gamma}_\mu)^T$. La matrice dei residui presenta dei valori mancanti, che coincidono con i conteggi nulli che sono stati separati dai conteggi non nulli nella fase precedente. Di conseguenza è stato utilizzato un algoritmo basato sulla regolarizzazione della norma per imputare i valori mancanti (Mazumder, Hastie e Tibshirani 2010).

β_π, γ_π e α_π vengono stimati attraverso una regressione ridge logistica (dove la variabile risposta è pari a zero se l'osservazione è un conteggio diverso da zero e pari a uno altrimenti), alternando il processo di stima fino a convergenza. I parametri di dispersione sono inizializzati a zero.

Dopo l'inizializzazione si massimizza localmente la massima verosimiglianza penalizzata alternando la stima dei parametri fino a convergenza. Per prima cosa si stima un parametro di dispersione comune a tutti i geni, ovvero si ottimizza la funzione obiettivo sotto il vincolo $Var(\zeta) = 0$, utilizzando per gli altri parametri i valori stimati durante l'inizializzazione. Usando questa stima comune come valore iniziale, si massimizza la verosimiglianza con uno schema di ottimizzazione quasi-Newton [2]. Successivamente si stimano prima

[2]Nel metodo di ottimizzazione quasi-Newton non serve calcolare la matrice delle

(W, γ), poi (β, α) con il metodo quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS).
Infine, si esegue l'ortogonalizzazione di $(\hat{W}, \hat{\alpha})$ partendo dalla decomposizione a valori singolari del loro prodotto.

2.2 GLM-PCA

Nel Paragrafo 1.6.1 si è parlato dei codici univoci (UMI) che servono per risalire al numero originario di reads dopo l'amplificazione.
Townes et al. (2019) hanno notato che i conteggi UMI di scRNA-seq non presentano eccesso di zeri. Infatti, in Figura 2.2 (Townes et al. 2019) si può osservare che la distribuzione delle reads (a sinistra) presenta un alto picco a zero, evidenziando l'inflazione di zeri, al contrario della distribuzione dei conteggi UMI (a destra). Inoltre, una grande differenza che si può notare è il range di valori che assumono le due distribuzioni (0-300 per le reads, 0-20 per UMI), sottolineando il fatto che alcuni geni vengono amplificati a dismisura.

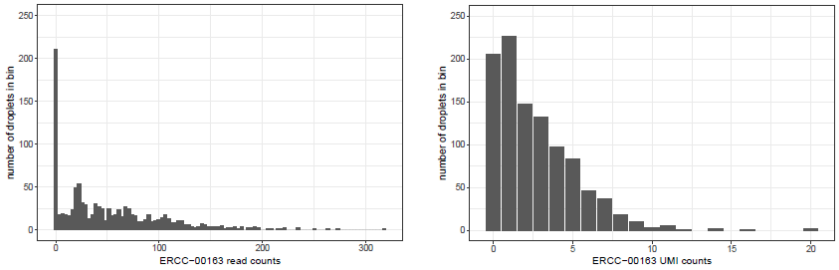
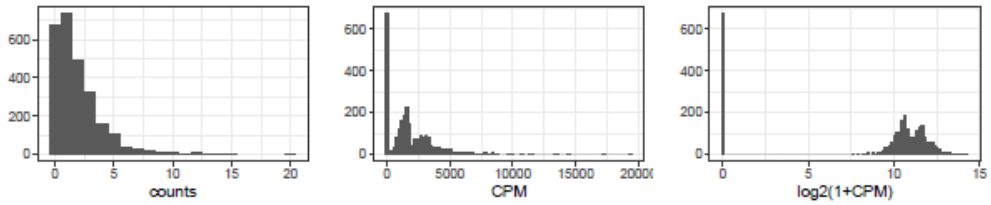


Figura 2.2: Distribuzione delle reads (sinitra) e dei conteggi UMI (destra) per il gene ERCC-00103

derivate seconde per trovare i punti di ottimo

Inoltre, hanno dimostrato che la normalizzazione dei conteggi UMI distorce in maniera evidente le analisi. Infatti, in Figura 2.3 (Townes et al. 2019) si può vedere come le trasformazioni counts per million (CPM), definita $(y_{ij}/n_i)10^6$, e log-CPM, $\log_2(1 + CPM)$, portano la distribuzione dei conteggi UMI ad avere un eccesso di zeri.



(a) UMI counts (b) counts per million (CPM) (c) log of CPM

Figura 2.3: a) Conteggi UMI per il gene ENSG00000114391. b) Distribuzione dei Counts per million per lo stesso gene. c) Distribuzione dei valori di log-CPM per lo stesso gene.

Gli autori hanno, quindi, implementato una generalizzazione dell'analisi delle componenti principali per ridurre la dimensionalità dei dati di scRNA-seq, estendendo tale approccio alle distribuzioni della famiglia esponenziale.
Hanno proposto un modello basato sulla distribuzione multinomiale, senza la necessità di modellare l'inflazione di zeri. Tale scelta è giustificata dal fatto che durante le molteplici fasi del sequenziamento, una frazione di molecole viene persa e il numero di UMI è di molto inferiore al numero originale di reads.
Si indicano con x_{ij} il vero numero di reads e con y_{ij} il numero di UMI per il gene j nella cellula i. Si definisce $\pi_{ij} = x_{ij}/\sum_j x_{ij}$ l'abbondanza relativa, ovvero il vero numero di reads per il gene j nella cellula i diviso il numero totale di reads nella cellula i. È opportuno tenere in considerazione che ge-

ni con una grande abbondanza relativa hanno maggiore probabilità di avere conteggi UMI diversi da zero, mentre geni con una piccola abbondanza relativa hanno maggiore probabilità di avere conteggi UMI pari a zero.
Il vettore aleatorio $\mathbf{y_i} = (y_{i1}, \ldots, y_{iJ})^T$ con il vincolo $\sum_j y_{ij} = n_i$ segue una distribuzione multinomiale con funzione di probabilità

$$f(y_i) = \binom{n_i}{y_{i1}, \ldots, y_{iJ}} \prod_j y_{ij}^{\pi_{ij}} .$$

Considerando il gene j, la distribuzione marginale di y_{ij} è una binomiale con parametri n_i e π_{ij}.
La distribuzione multinomiale diventa difficile da interpretare e da trattare computazionalmente se siamo in un contesto ad elevata dimensionalità, come nel caso di scRNA-seq. Se si ignora la correlazione tra due geni, che è dovuta al vincolo $\sum_j y_{ij} = n_i$, la distribuzione multinomiale potrebbe essere approssimata a J distribuzioni binomiali indipendenti. Questa approssimazione è ragionevole se tutti i π_{ij} sono molto piccoli, visto che la correlazione fra due geni j e k è

$$\mathrm{Cor}[y_{ij}, y_{ik}] = \frac{\sqrt{\pi_{ij}\pi_{ik}}}{\sqrt{(1-\pi_{ij})(1-\pi_{ik})}} .$$

Tale requisito probabilmente è soddisfatto per scRNA-Seq se il numero di geni J è grande e nessun gene costituisce la maggior parte delle copie di RNA nella cellula.
Inoltre, se n_i è grande e π_{ij} è piccolo, ogni distribuzione binomiale può essere ulteriormente approssimata ad una Poisson con media $n_i\pi_{ij}$. Queste caratteristiche sono tipiche dei dati di scRNA-seq, perciò l'approssimazione può essere ragionevole. In conclusione, la distribuzione dei conteggi UMI può essere approssimata ad una Poisson.

Tuttavia, la distribuzione multinomiale non tiene in considerazione l'alta variabilità presente nei dati di scRNA-seq, perciò è necessaria una versione

sovra-dispersa del modello multinomiale: la distribuzione Dirichlet-multinomiale. Infatti, se $\mathbf{y_i}$ si distribuisce come una multinomiale condizionatamente al vettore dei parametri $\pi_\mathbf{i} = (\pi_{i1}, \ldots, \pi_{iJ})$ e $\pi_\mathbf{i}$ è una variabile casuale con distribuzione di Dirichlet [3] con parametro di forma α, allora la distribuzione marginale di $\mathbf{y_i}$ è Dirichlet-multinomiale. Inoltre, si può ottenere un vettore casuale Dirichlet da J variabili Gamma con parametro α divise per la loro somma.

La distribuzione Dirichlet-multinomiale può essere approssimata dalla distribuzione binomiale negativa. Di seguito vengono riportati i passaggi di una semplice dimostrazione.

Si approssima y_{ij} con una Poisson di media $n_i\pi_{ij}$ (come sopra).

Sia $\pi_{ij} = \lambda_{ij} / \sum_j \lambda_{ij}$, allora $\pi_{ij} \sim Dirichlet(\alpha)$ se λ_{ij} è una collezione di variabili casuali che seguono una distribuzione Gamma con parametro di forma α e media n_i/J.

Questo implica che $\sum_j \lambda_{ij}$ è distribuito come una Gamma con parametro di forma $J\alpha$ e media n_i. Se $J \to \infty$, la distribuzione converge verso un punto di massa a n_i, di conseguenza se J è molto grande (come nel caso di scRNA-seq) $\sum_j \lambda_{ij} \approx n_i$.

Segue che y_{ij} è approssimativamente distribuito come una Poisson di media λ_{ij}. Se si integra rispetto a λ_{ij} si ottiene la distribuzione marginale di y_{ij}: binomiale negativa con parametro di forma α e media n_i/J.

In conclusione, un modello binomiale negativo per dati di conteggio può essere considerato come un'approssimazione ad un modello Dirichlet-multinomiale.

[3] La distribuzione Dirichlet generalizza al caso multivariato la distribuzione Beta.

La funzione di densità è

$$f(x_1, \ldots, x_k; \alpha_1, \ldots, \alpha_{k+1}) = \frac{\Gamma(\alpha_1 + \cdots + \alpha_{k+1})}{\Gamma(\alpha_1) \ldots \Gamma(\alpha_{k+1})} x_1^{\alpha_1 - 1} \ldots x_k^{\alpha_k - 1} (1 - x_1 - \cdots - x_k)^{\alpha_{k+1} - 1}$$

dove $x_1, \ldots, x_k \geq 0$ tali che $\sum_{i=1}^k x_i \leq 1$.

L'idea di estendere l'analisi delle componenti principali alle distribuzioni della famiglia esponenziale nasce dalla considerazione seguente. PCA minimizza la distanza euclidea tra i dati e una loro rappresentazione in uno spazio a bassa dimensionalità. Ciò è equivalente a minimizzare l'errore quadratico medio (MSE):

$$\min_{u,v} \sum_{i,j} (z_{ij} - \mathbf{u_i'v_j})^2 \,,$$

dove z_{ij} sono i dati normalizzati e $\mathbf{u_i}, \mathbf{v_j} \in \mathbb{R}^K$ indicano rispettivamente le componenti principali e i loadings. K è il numero delle dimensioni latenti e controlla la complessità del modello.
Inoltre, minimizzare l'MSE equivale a massimizzare la verosimiglianza di un modello lineare normale $z_{ij} \sim N(\mathbf{u_i'v_j}, \sigma^2)$. La distribuzione normale può essere sostituita con le altre distribuzioni della famiglia esponenziale.
Il modello di Poisson, che approssima la distribuzione multinomiale per dati di conteggio, è

$$y_{ij} \sim Poi(n_i \exp\{\mathbf{u_i'v_j}\})$$

e il modello binimiale negativo, che approssima il modello Dirichlet-multinomiale per dati di conteggio sovra-dispersi, è

$$y_{ij} \sim NB(n_i \exp\{\mathbf{u_i'v_j}\}; \phi_j) \,.$$

Si nota che il termine n_i è usato come un offset, e svolge la funzione di fattore di normalizzazione. Inoltre, se il primo elemento di ogni $\mathbf{u_i}$ è vincolato ad essere pari a 1, si crea una intercetta gene-specifica, che equivale a centrare i dati.
Tuttavia, le stime di massima verosimiglianza di $\mathbf{u_i}$ e $\mathbf{v_j}$ non possono essere espresse in forma chiusa, perciò è necessario utilizzare il metodo scoring di

Fischer [4]. Questo algoritmo iterativo è almeno dieci volte più lento rispetto alla PCA. Gli autori hanno proposto una versione più veloce basata sulla devianza. Così come PCA minimizza l'MSE, GLM-PCA minimizza una generalizzazione dell'MSE, la devianza. Per il modello di Poisson la funzione obiettivo da minimizzare, ovvero la devianza totale, risulta

$$D = \sum_{ij} y_{ij} \log\left(\frac{y_{ij}}{\mu_{ij}}\right) - (y_{ij} - \mu_{ij}),$$

mentre per il modello binomiale negativo risulta

$$D = \sum_{ij} y_{ij} \log\left(\frac{y_{ij}}{\mu_{ij}}\right) - (y_{ij} + \phi) \log\left(\frac{y_{ij} + \phi}{\mu_{ij} + \phi}\right),$$

dove $\log \mu_{ij} = \eta_{ij} = \log n_i + \mathbf{u_i' v_j}$ e ϕ è la stima comune a tutti i geni del parametro di dispersione.
La fase di ottimizzazione procede calcolando le derivate rispetto ai parametri ignoti, ovvero i fattori latenti.

2.3 Modello Binomiale Negativo

Nel paragrafo precedente (2.2) si è detto che con la tecnologia UMI non si verifica un eccesso di eventi dropout e di conseguenza non è presente un'inflazione di zeri nei dati.
In questa tesi si vuole proporre una variazione del modello ZINB-WaVE, descritto nel Paragrafo 2.1, in cui viene eliminata la componente che modella l'eccesso di zeri. In questo modo si considera un modello binomiale negativo che tiene in considerazione la natura di conteggio dei dati, la sovra-dispersione

[4] Il metodo scoring di Fischer è una variazione del metodo di Newton-Raphson, dove al posto della matrice di informazione osservata è presente la matrice di informazione attesa: $\hat{\beta}^{(m+1)} = \hat{\beta}^{(m)} + \left(i(\hat{\beta}^{(m)})\right)^{-1} \ell_*(\hat{\beta}^{(m)})$, dove m indica l'm-esima iterazione, i l'informazione attesa e ℓ_* è la funzione score

e la necessità ridurre la dimensionalità dei dati estraendo il segnale di interesse.

Dati n campioni e J geni, si indica con Y_{ij} il conteggio del gene j ($j = 1, ..., J$) nel campione i ($i = 1, ..., n$); Y_{ij} si distribuisce come una binomiale negativa (NB) che ha funzione di densità di probabilità (2.1). La media della distribuzione NB è μ e la varianza è $\sigma^2 = \mu + \mu^2/\theta = \mu + \phi\mu^2$. In particolare, quando $\phi = 0 \Leftrightarrow \theta = +\infty$ la distribuzione NB si riduce ad una distribuzione di Poisson.

La media e l'inverso del parametro di dispersione (θ) vengono modellati nel modo seguente:

$$\ln(\mu_{ij}) = (X\beta + (V\gamma)^T + W\alpha + O)_{ij} \tag{2.5}$$

$$\ln(\theta_{ij}) = \zeta_j \tag{2.6}$$

dove:

- μ è una matrice $n \times J$ che indica il livello di espressione medio del gene j nel campione i;
- X è una matrice $n \times M$ di covariate osservate a livello di campione, associata alla matrice $M \times J$ dei coefficienti β;
- V è una matrice $J \times L$ di covariate osservate a livello di gene, associata alla matrice $L \times n$ dei coefficienti γ;
- W è una matrice $n \times K$ di covariate non osservate a livello di campione, associata alla matrice $K \times J$ dei coefficienti α;
- O è una matrice $n \times J$ di offsets;
- ζ è un vettore di dimensione J di parametri di dispersione gene-specifici su scala logaritmica.

Si tratta di un modello di analisi fattoriale, infatti estende il metodo *RUV* (Paragrafo 1.5.2) alla distribuzione binomiale negativa. Come già descritto nel Paragrafo 2.1, questo permette di ridurre la dimensionalità dei dati, estraendo l'informazione di interesse.
I parametri da stimare sono β, γ, W, α e ζ, mentre i parametri noti (o da fornire) sono X, V, O e K, che indica il numero di fattori latenti.
La procedura di stima è composta da una fase di inizializzazione dei parametri e da una fase di ottimizzazione. L'inizializzazione è necessaria in quanto la stima dei parametri avviene per via numerica, poiché non è possibile rendere esplicita la soluzione. Si cerca, quindi, di ottimizzare la funzione obiettivo partendo da valori iniziali plausibili.
La funzione di log-verosimiglianza è

$$\ell(\beta, \gamma, W, \alpha, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{J} \ln f_{NB}(Y_{ij}; \mu_{ij}, \theta_{ij}) \,,$$

dove μ_{ij} e θ_{ij} dipendono da $(\beta, \gamma, W, \alpha, \zeta)$ tramite le Equazioni (2.5) e (2.6).
Per ridurre l'overfitting e per aumentare la stabilità del problema di ottimizzazione in un contesto di molti parametri, si utilizza un approccio basato sulla massima verosimiglianza penalizzata:

$$\max_{(\beta,\gamma,W,\alpha,\zeta)} \{\ell(\beta, \gamma, W, \alpha, \zeta) - \mathrm{Pen}(\beta, \gamma, W, \alpha, \zeta)\} \,.$$

La penalizzazione schiaccia i parametri verso lo zero, eccetto per le intercette, che non vengono penalizzate, e i parametri di dispersione, che vengono schiacciati verso un valore costante tra i geni. Per valori positivi dei parametri di regolazione $(\epsilon_\beta, \epsilon_\gamma, \epsilon_W, \epsilon_\alpha, \epsilon_\zeta)$ si imposta

$$\mathrm{Pen}(\beta, \gamma, W, \alpha, \zeta) = \frac{\epsilon_\beta}{2} \|\beta^0\|^2 + \frac{\epsilon_\gamma}{2} \|\gamma^0\|^2 + \frac{\epsilon_W}{2} \|W\|^2 + \frac{\epsilon_\alpha}{2} \|\alpha\|^2 + \frac{\epsilon_\zeta}{2} \mathrm{Var}(\zeta) \,,$$

dove β^0 e γ^0 indicano le matrici β e γ senza le righe che corrispondono alle intercette, $\|\cdot\|$ è la norma matriciale di Frobenius [5] e $\text{Var}(\zeta) = 1/(J-1)\sum_{i=1}^{J}\left(\zeta_i - \sum_{j=1}^{J}\zeta_j/J\right)^2$ è la varianza campionaria corretta degli elementi di ζ.

Per bilanciare le penalità applicate alle diverse matrici nonostante le loro diverse dimensioni, una scelta naturale è quella di fissare $\epsilon > 0$ e impostare

$$\epsilon_\beta = \frac{\epsilon}{J}, \quad \epsilon_\gamma = \frac{\epsilon}{n}, \quad \epsilon_W = \frac{\epsilon}{n}, \quad \epsilon_\alpha = \frac{\epsilon}{J}, \quad \epsilon_\zeta = \epsilon\,.$$

Durante l'inizializzazione si approssima la distribuzione dei conteggi ad una log-normale e si stimano i parametri in 3 fasi.

1. Si stimano β e γ risolvendo il problema di regressione ridge che risulta convesso:

$$\min_{\beta\gamma} \sum_{i,j}(L_{ij} - (X\beta)_{ij} - (V\gamma)_{ij})^2 + \frac{\epsilon_\beta}{2}\left\|\beta^0\right\|^2 + \frac{\epsilon_\gamma}{2}\left\|\gamma^0\right\|^2\,,$$

   dove $L_{ij} = \ln(Y_{ij}) - (O)_{ij}$.

   A causa dell'elevata dimensionalità dei dati, β e γ vengono stimati in modo alternato, ripetendo i punti seguenti fino a convergenza:

   (a) L'ottimizzazione di γ può essere eseguita indipendentemente e in parallelo per ogni cellula

$$\hat{\gamma} \in \arg\min_{\gamma} \sum_{i,j}(L_{ij} - (X\hat{\beta})_{ij} - (V\gamma)_{ij})^2 + \frac{\epsilon_\gamma}{2}\left\|\gamma^0\right\|^2\,,$$

   dove $\hat{\beta}$ iniziale è pari a zero.

[5] La norma matriciale di Frobenius della matrice A è $\|A\| = \sqrt{\text{tr}(A^*A)}$, dove A^* indica la matrice trasposta coniugata di A

(b) L'ottimizzazione di β può essere eseguita indipendentemente e in parallelo per ogni cellula

$$\hat{\beta} \in \arg\min_{\beta} \sum_{i,j} (L_{ij} - (X\beta)_{ij} - (V\hat{\gamma})_{ij})^2 + \frac{\epsilon_\beta}{2} \left\| \beta^0 \right\|^2 .$$

2. Si stimano W e α risolvendo

$$(\hat{W}, \hat{\alpha}) \in \arg\min_{W,\alpha} \sum_{i,j} (L_{ij} - (X\hat{\beta})_{ij} - (V\hat{\gamma})_{ij} - (W\alpha)_{ij})^2 + \frac{\epsilon_W}{2} \|W\|^2 + \frac{\epsilon_\alpha}{2} \|\alpha\|^2 .$$

Si denota con $D = L - X\hat{\beta} - (V\hat{\gamma})^T$, il problema può essere riscritto come:

$$\arg\min_{W,\alpha} \|D - W\alpha\|_p^2 + \frac{1}{2}(\epsilon_W \|W\|^2 + \epsilon_\alpha \|\alpha\|^2) ,$$

dove $\|A\|_p^2 = \sum_{i,j} A_{ij}^2$. Dal Lemma 1, se si pone $R = W\alpha$ si può risolvere il problema di ottimizzazione convesso:

$$\hat{R} \in \arg\min_{R:\mathrm{rank}(R) \leq K} \|D - R\|_p^2 + \sqrt{\epsilon_W \epsilon_\alpha} \|R\|_* , \tag{2.7}$$

da cui

$$W = \left(\frac{\epsilon_\alpha}{\epsilon_W} \right)^{\frac{1}{4}} R_L R_\Sigma^{\frac{1}{2}} , \quad \alpha = \left(\frac{\epsilon_W}{\epsilon_\alpha} \right)^{\frac{1}{4}} R_\Sigma^{\frac{1}{2}} R_R ,$$

dove $\hat{R} = R_L R_\Sigma R_R$ è la decomposizione a valori singolari di $\hat{R}$. Questa soluzione è esatta quando K è almeno uguale al rango della soluzione del problema 2.7, altrimenti 2.7 diventa un problema di ottimo non convesso, il cui punto di ottimo globale potrebbe essere difficile da trovare.

3. Si inizializza $\hat{\zeta} = 0$.

Dopo l'inizializzazione, si massimizza localmente la log-verosimiglianza penalizzata alternando la stima del parametro di dispersione e dei fattori di destra e di sinistra, iterando le seguenti fasi fino a convergenza.

1. Ottimizzazione del parametro di dispersione:

$$\hat{\zeta} \in \arg\max_{\zeta} \left\{ \ell(\hat{\beta}, \hat{\gamma}, \hat{W}, \hat{\alpha}, \zeta) - \frac{\epsilon_\zeta}{2} \operatorname{Var}(\zeta) \right\}.$$

Per prima cosa si stima un parametro di dispersione comune a tutti i geni massimizzando la funzione obiettivo con il vincolo $\operatorname{Var}(\zeta) = 0$. Utilizzando questa stima preliminare, si ottimizza la funzione di log-verosimiglianza con una procedura quasi-Newton. Per questo algoritmo è necessario fornire solo la derivata prima della funzione da ottimizzare. La derivata della funzione di log-verosimiglianza secondo θ è:

$$\frac{\partial}{\partial \theta} \ln f_{NB}(y; \mu, \theta) = \Psi(y + \theta) - \Psi(\theta) + \ln \theta + 1 - \ln(\mu + \theta) - \frac{y + \theta}{\mu + \theta},$$

dove $\Psi(x) = \Gamma'(x)/\Gamma(x)$ è la funzione digamma. Di conseguenza, la derivata secondo ζ_j per $j = 1, \ldots, J$ della funzione da ottimizzare è:

$$\sum_{i=1}^{n} \theta_j \frac{\partial}{\partial \theta_j} \ln f_{NB}(y_{ij}; \mu_{ij}, \theta_j) - \frac{\epsilon_\zeta}{J - 1} \left(\zeta_j - \frac{1}{J} \sum_{k=1}^{J} \zeta_k \right).$$

2. Ottimizzazione dei fattori di sinistra:

$$(\hat{\gamma}, \hat{W}) \in \arg\max_{\gamma, W} \left\{ \ell(\hat{\beta}, \gamma, W, \hat{\alpha}, \hat{\zeta}) - \frac{\epsilon_\gamma}{2} \left\| \gamma^0 \right\|^2 - \frac{\epsilon_W}{2} \|W\|^2 \right\}.$$

L'ottimizzazione può essere eseguita indipendentemente e in parallelo per ogni cellula i e a questo proposito viene applicata la funzione `solveZinbRegression` (si veda Appendice B). Si individua un insieme di parametri (a, b) che massimizza localmente la funzione di log-verosimiglianza di un modello binomiale negativo che ha i parametri modellati nel modo seguente:

$$\ln(\mu) = Aa + Bb + C_\mu$$
$$\ln(\theta) = C_\theta \,.$$

Per ogni cellula i si richiama la funzione `solveZinbRegression` con i seguenti parametri:

$$\begin{cases} a = \gamma[., i] \\ b = W[i, .]^T \\ y = Y[i, .]^T \\ A = V \\ B = \alpha^T \\ C_\mu = (X[i, .]\beta + O[i, .])^T \\ C_\theta = \zeta \end{cases}$$

3. Ottimizzazione dei fattori di destra:

$$(\hat{\beta}, \hat{\alpha}) \in \arg\max_{\beta,\alpha} \left\{ \ell(\beta, \hat{\gamma}, \hat{W}, \alpha, \hat{\zeta}) - \frac{\epsilon_\beta}{2} \left\|\beta^0\right\|^2 - \frac{\epsilon_\alpha}{2} \|\alpha\|^2 \right\}.$$

L'ottimizzazione può essere eseguita indipendentemente e in parallelo per ogni gene j e a questo proposito viene applicata la funzione `solveZinbRegression` con i seguenti parametri:

$$\begin{cases} a = (\beta[., j]; \alpha[., j]) \\ b = \emptyset \\ y = Y[., j] \\ A = [X, W] \\ B = \emptyset \\ C_\mu = (V[j, .]\gamma)^T + O[., j]) \\ C_\theta = \zeta_j \mathbb{1}_n \end{cases}$$

4. Ortogonalizzazione:

$$(\hat{W}, \hat{\alpha}) \in \arg \min_{W,\alpha: W\alpha = \hat{W}\hat{\alpha}} \frac{1}{2}(\epsilon_W \|W\|^2 + \epsilon_\alpha \|\alpha\|^2) \,.$$

   Si ottiene applicando il Lemma 1, partendo da una decomposizione a valori singolari delle stime correnti $\hat{W}\hat{\alpha}$.

L'algoritmo che è appena stato decritto è stato implementato nel pacchetto open-source di R `zinbwave` disponibile nell'ambito del progetto Bioconductor (`https://bioconductor.org/packages/release/bioc/html/zinbwave.html`). Il codice R è disponibile in Appendice C.

Capitolo 3

Confronto tra modelli

In questo capitolo si confrontano le prestazioni dei modelli binomiale negativo, binomiale negativo con inflazione di zeri e GLM-PCA descritti nel Capitolo 2. A tale scopo sono state effettuate delle simulazioni a partire da un dataset reale.
Per prima cosa si vuole valutare l'accuratezza della stima dei fattori latenti per i modelli NB e ZINB in diversi contesti.
Successivamente si verifica l'adeguatezza dei modelli nell'ambito della riduzione della dimensionalità dei dati in un'applicazione di clustering.
Infine, si confrontano i modelli dal punto di vista del tempo computazionale.

3.1 Accuratezza dei modelli NB e ZINB

Lo scopo di questa analisi è confrontare l'accuratezza, misurata tramite l'errore quadratico medio (MSE)[1], della stima dei fattori latenti $W\alpha$ per i modelli NB e ZINB in diversi contesti.

[1] $\text{MSE}(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] = \text{Var}(\hat{\theta}) + Bias(\hat{\theta}, \theta)^2$

Le simulazioni sono state effettuate a partire dal dataset reale con tecnologia UMI `pbmc3k`, reso disponibile dal pacchetto `TENxPBMCData` di Bioconductor. Questo pacchetto mette a disposizione nove dataset di RNA-seq a singola cellula di cellule mononucleate del sangue periferico (PBMC) sequenziate da 10x Genomics.
A causa dei lunghi tempi computazionali, si è deciso di utilizzare un sotto-campione casuale di 500 cellule e di filtrare i geni poco espressi, rimuovendo quei geni che non presentavano almeno 5 reads in almeno 5 campioni. Il dataset finale è composto da 500 cellule e 297 geni.
Poiché non esiste un criterio oggettivo per stabilire se un dataset presenta inflazione di zeri, è stata confrontata la probabilità di dropout (π, Paragrafo 2.1) tra `pbmc3k` e `fluidigm`, un dataset senza tecnologia UMI disponibile nel pacchetto `scRNAseq` di Bioconductor.
`fluidigm` è stato utilizzato per verificare la presenza di inflazione di zeri e non verrà considerato nelle successive analisi di simulazione. Per questo dataset è stato utilizzato lo stesso criterio di `pbmc3k` per filtrare i geni e, tra i geni filtrati, sono stati selezionati i 1000 più variabili. Quindi il dataset finale è costituito da 65 cellule e 1000 geni.
Per entrambe è stato stimato un modello ZINB con $K = 2$, parametro di dispersione comune a tutti i geni e solo le intercette. Come si può osservare in Figura 3.1, le distribuzioni della probabilità media di dropout per ogni gene sono molto diverse. Infatti, per `pbmc3k` la distribuzione è concentrata su valori molto bassi, con un picco a zero, mentre per `fluidigm` è concentrata nei valori centrali. Di conseguenza, possiamo affermare con una certa sicurezza che `pbmc3k` non presenta inflazione di zeri.

Per le simulazioni è stata utilizzata la funzione `zinbSim` del pacchetto `zinbwave`, con cui sono stati generati da un modello ZINB 100 dataset con inflazione

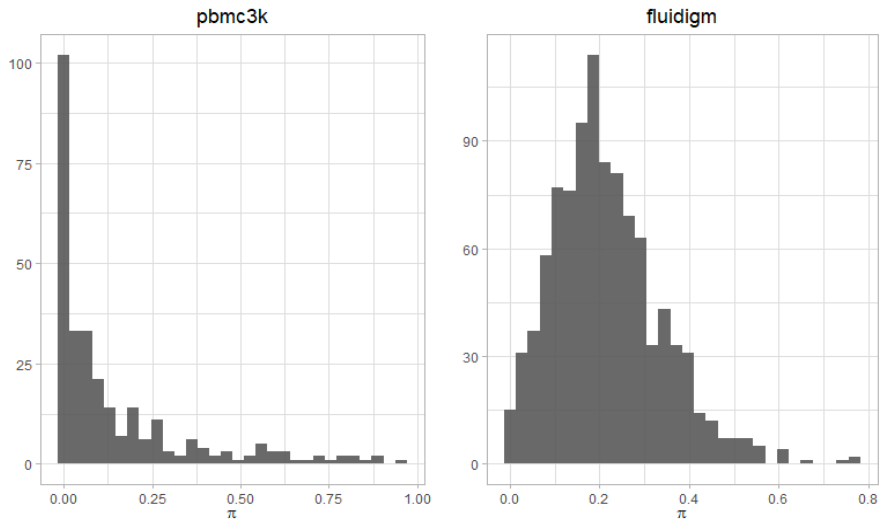


Figura 3.1: Distribuzione della probabilità di dropout dei dataset `pbmc3k` (sinistra) e `fluidigm` (destra)

di zeri e 100 dataset senza inflazione di zeri. Alla funzione sono state fornite le stime dei parametri di `pbmc3k` ottenute con il modello binomiale negativo (Paragrafo 2.3) con $K = 2$, parametro di dispersione gene-specifico e solo le intercette.

La stima di $W\alpha$ ottenuta da questo modello rappresenta il vero valore che assumono i fattori latenti.

Successivamente per i due insiemi di dataset sono stati stimati i modelli NB e ZINB con solo le intercette, con parametro di dispersione comune a tutti i geni e dispersione gene-specifica, con corretta ed errata specificazione del numero di fattori latenti ($K = 2, 5, 10$) ed è stato calcolato l'MSE.

Come si può notare in Figura 3.2, il modello NB ha un'ottima prestazione nel caso in cui non vi sia dropout. L'errata specificazione di K aumenta di poco l'errore, che rimane comunque accettabile: risulta di poco inferiore nel caso in cui la dispersione sia gene-specifica rispetto al caso in cui vi sia una

dispersione comune.

Invece, quando i dati presentano un eccesso di zeri vi è una differenza marcata tra il modello con dispersione comune e il modello con dispersione gene-specifica. Sembra che quest'ultimo riesca a gestire il dropout "incorporandolo" nella stima dei parametri di dispersione. Nel momento in cui si passa al modello NB con parametro di dispersione comune, questo non è più in grado di catturare l'alta variabilità indotta dal dropout.

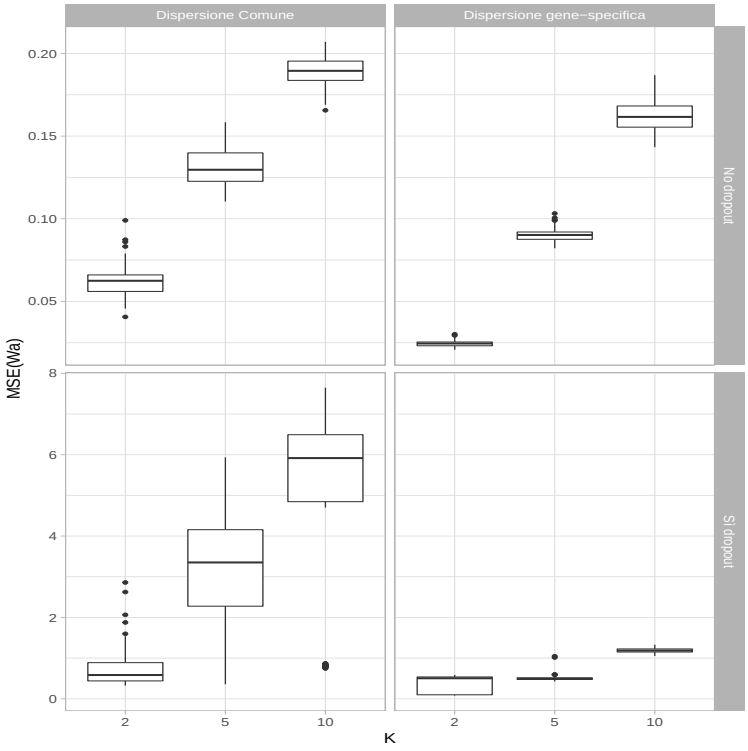


Figura 3.2: MSE per il modello NB

Dalla Figura 3.3 si osserva che le prestazioni del modello ZINB rimangono pressoché invariate nel caso di dispersione comune o gene-specifica. Rispetto al modello NB, l'errore è raddoppiato in caso di assenza di dropout. Inoltre, è evidente che il modello ZINB presenti un MSE maggiore nel caso in cui i da-

taset non presentino dropout rispetto ai dataset con dropout, infatti l'errore è raddoppiato. Tuttavia, è importante sottolineare che, in caso di dropout, il modello NB con dispersione gene-specifica ha una prestazione paragonabile al modello ZINB. Questo evidenzia l'elevata flessibilità del modello NB anche in presenza di inflazione di zeri.

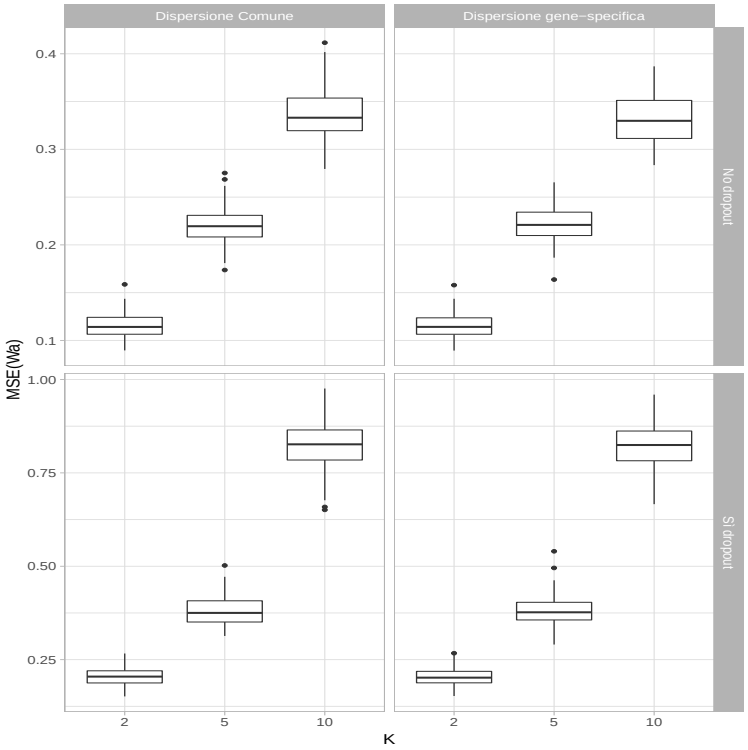


Figura 3.3: MSE per il modello ZINB

3.2 Riduzione della dimensionalità

Lo scopo delle analisi che vengono illustrare di seguito è testare le prestazioni dei modelli presentati nel Capitolo 2 nell'ambito della riduzione della dimensionalità in un'applicazione di clustering.

Anche in questo caso le simulazioni sono state effettuate a partire dal dataset reale `pbmc3k`, da cui sono state selezionate 500 cellule in modo casuale e sono stati filtrati i geni poco espressi come è stato descritto nel paragrafo precedente.

Per le simulazioni è stato utilizzato il pacchetto `splatter` (Zappia, Phipson e Oshlack 2017). Questo pacchetto unifica i diversi metodi esistenti implementati in altri pacchetti per generare dati di scRNA-seq e propone un nuovo metodo, Splat.

Il nucleo di questo nuovo algoritmo di simulazione è un modello gerarchico gamma-Poison, dove il livello di espressione medio di ogni gene è generato da una distribuzione gamma e i conteggi per ogni cellula sono generati successivamente da una distribuzione di Poisson.

In Figura 3.4 si può osservare uno schema del modello, in cui i cerchi colorati in blu indicano i parametri di input che si possono stimare a partire da un dataset reale e il cerchio in rosso indica il risultato finale, ovvero la matrice dei conteggi. È evidente che il modello è molto flessibile e si può personalizzare in base alle proprie esigenze.

Di seguito si illustrano i vari parametri di input e la loro funzione.

È possibile specificare la probabilità che un gene sia un outlier con espressione genica elevata (π^O): si aggiungono, quindi, questi valori anomali alla simulazione sostituendo la media precedentemente simulata con la mediana delle medie dei geni simulati moltiplicata per un fattore di inflazione. Il fattore di inflazione viene campionato da una distribuzione log-normale con parametri μ^O e σ^O.

Molto spesso la dimensione della libreria (cioè il numero totale di reads) varia all'interno di un esperimento scRNA-seq e può essere molto diversa tra

esperimenti a seconda della profondità di sequenziamento [2]. La dimensione della libreria (L_j) è modellata con una distribuzione log-normale (con parametri μ^L e σ^L) e viene utilizzata per aggiustare proporzionalmente la media dei geni per ogni cellula. Questo consente di modificare il numero di conteggi per cellula indipendentemente dai livelli di espressione genica sottostanti.
È possibile includere anche un trend media-varianza: si simula il coefficiente di variazione biologico (BCV) per ciascun gene da una distribuzione chi-quadro inversa riscalata, in cui il fattore di scala è una funzione della media del gene. In seguito, si generano delle nuove medie ($\lambda_{i,j}$) da una distribuzione gamma con parametri di forma e di scala dipendenti dai BCV simulati e dalle medie precedenti.
A questo punto si genera una matrice dei conteggi campionando da una distribuzione di Poisson con media pari a $\lambda_{i,j}$.
Inoltre, Splat permette di aggiungere degli eventi di dropout alla matrice dei conteggi: si ricorre alla relazione tra l'espressione media di un gene e la proporzione dei conteggi pari a zero in quel gene per modellare questo processo e si utilizza una funzione logistica per ottenere la probabilità che un conteggio sia zero (π^D). La funzione logistica è definita da un parametro x_0, che indica il livello di espressione in cui il 50 % delle cellule è zero, e un parametro di forma k che controlla la velocità con cui le probabilità cambiano da x_0. Tramite la distribuzione di Bernoulli con probabilità π^D si determina quali conteggi debbano essere sostituiti con degli zeri.

Il modello appena descritto è sufficiente per simulare un'unica popolazione omogenea, ma non per riprodurre le situazioni più complesse osservate in alcuni campioni biologici reali.

[2] La profondità di sequenziamento è il numero di reads uniche che includono un dato nucleotide nella sequenza ricostruita

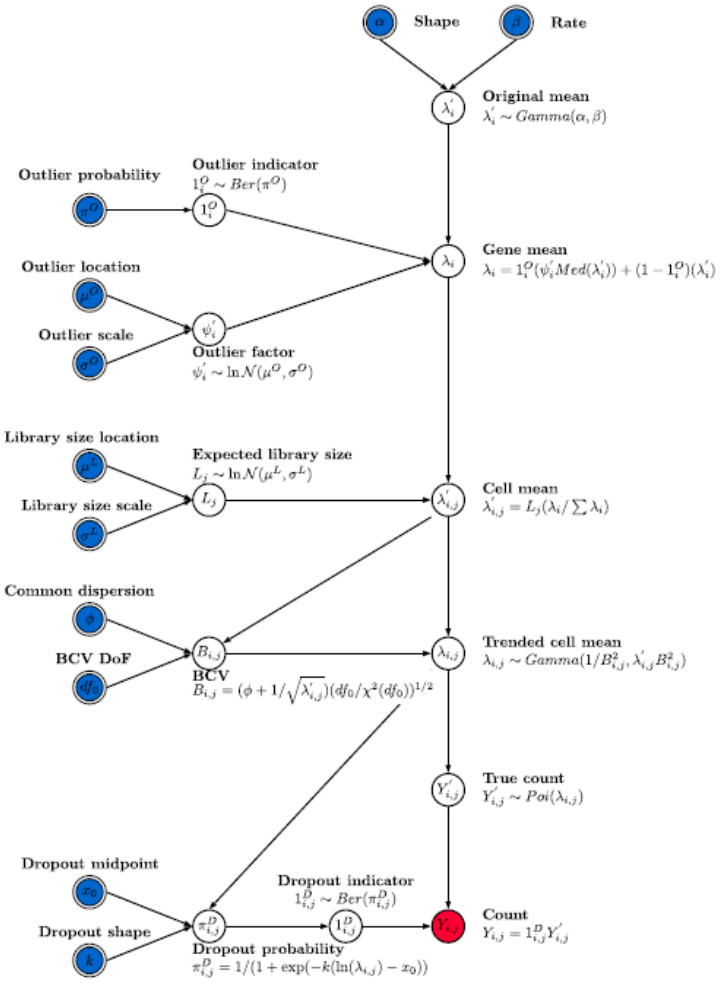


Figura 3.4: Struttura del modello di simulazione Splat

Splat può simulare campioni con più tipi di cellule creando gruppi distinti di cellule in cui diversi geni sono espressi in modo differenziato (DE) tra i diversi gruppi. L'espressione differenziale è modellata utilizzando un processo simile a quello per la creazione degli outlier. In particolare, un fattore di espressione differenziale moltiplicativo è assegnato a ciascun gene e applicato alla media. Per i geni DE, questi fattori sono generati da una distribuzione log-normale, mentre per gli altri sono pari ad uno. La possibilità di decidere il numero di gruppi e la probabilità che una cellula provenga da ciascun gruppo consente che i diversi gruppi siano definiti con flessibilità. Inoltre, i parametri che controllano la probabilità che i geni siano espressi in modo differenziato, nonché l'entità e la direzione dei fattori DE, possono essere impostati individualmente per ciascun gruppo. Il dataset di output contiene le informazioni sul gruppo da cui proviene ciascuna cellula e sui fattori applicati a ciascun gene in ciascun gruppo.
In aggiunta, Splat può includere degli effetti di batch utilizzando dei fattori moltiplicativi applicati a tutti i geni per gruppi di cellule. Questo consente ai ricercatori di valutare le prestazioni dei metodi in presenza di variabilità indesiderata.
Infine, Splat permette di simulare il lignaggio delle cellule, ovvero il percorso che intraprendono per diventare cellule mature e differenziate. Si utilizza il processo di espressione differenziale sopra descritto per definire i livelli di espressione di una cellula iniziale e finale per ciascun percorso.
Vengono quindi definiti degli stati cellulari intermedi tra questi due tipi di cellule e le cellule simulate vengono assegnate in modo casuale a uno di questi stati, ricevendo il livello di espressione medio di quello stato.
Pertanto, la simulazione dei lignaggi è definita dai parametri di espressione differenziale, ma anche dai parametri che definiscono il percorso stesso, co-

me la lunghezza (numero di stati) e l'inclinazione (se è più probabile che le cellule provengano dall'inizio o dalla fine del percorso).
Tuttavia, nei dati reali è stato osservato che l'espressione dei geni può cambiare in modi più complessi e non lineari attraverso una traiettoria di differenziazione. Ad esempio, un gene può essere debolmente espresso all'inizio di un processo, fortemente espresso nel mezzo e debolmente espresso alla fine. Splat modella questo tipo di cambiamenti generando un ponte browniano (un *random walk* con punti finali fissi) tra le due cellule terminali di un percorso, che viene interpolato usando una spline di Akima [3].

L'algoritmo Splat è stato utilizzato in queste analisi per simulare dei dataset con 300 geni e 500 cellule. In tutti i dataset le cellule sono state divise in tre gruppi distinti generati con uguale probabilità e con probabilità che un dato gene sia DE pari a 0.3.
I dati sono stati simulati con diverse caratteristiche: con solo i tre gruppi di cellule, con dropout globale (ovvero si è usato lo stesso set di parametri per ogni cellula), con due batch, in cui le cellule sono state divise equamente, e sia con dropout globale sia con i due batch.
Sono stati simulati 100 dataset per ogni situazione.
Successivamente per tutti i dataset sono stati stimati i modelli binomiale negativo, binomiale negativo con inflazione di zeri e GLM-PCA con distribuzioni Poisson, multinomiale e binomiale negativa. Per quest'ultima il parametro di dispersione comune a tutti i geni è stato stimato con il pacchetto `edgeR`, perché va ricordato che la binomiale negativa è una famiglia esponenziale solo se il parametro di dispersione è noto. Inoltre, per tutti i modelli si è posto il

[3] Una spline di Akima è un tipo di spline non-liscia che si adatta bene alle curve in cui la derivata seconda varia rapidamente (Akima 1970)

numero di fattori latenti pari a 2,5 e 10.
Per i dataset con gli effetti di batch è stata aggiunta alla matrice del disegno la covariata che indica a quale batch appartengono le cellule.
Per i dataset con dropout il modello NB è stato stimato con dispersione gene-specifica, visti i risultati ottenuti nel Paragrafo 3.1.
Inoltre, in alcuni dataset Splat ha generato dei geni con solo conteggi pari a zero; questi geni sono stati eliminati dall'analisi, di conseguenza non tutti i dataset hanno la stessa dimensione, ma alcuni presentano fino ad un massimo di tre geni in meno.

Per il clustering si è utilizzato l'algoritmo k-means con un numero di gruppi pari a 3, ovvero il vero numero di gruppi simulati con Splat. È importante sottolineare che questa funzione è stata applicata alla matrice dei fattori latenti (W per NB e ZINB, matrice dei fattori per GLM-PCA) e non ai dati originali, per testare le capacità dei modelli riguardo la riduzione della dimensionalità.
Dopodiché si è calcolato l'Adjusted Rand Index (ARI): è una misura di accordo tra due partizioni, una data dal processo di clustering e l'altra definita da criteri esterni, nel nostro caso i gruppi simulati con Splat. Questo indice è definito

$$ARI = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right]/\binom{n}{2}},$$

dove, prese due partizioni U e V, n_{ij} è il numero di oggetti che sono sia in u_i sia in v_j (le coppie concordanti), $n_{i.}$ e $n_{.j}$ sono il numero di oggetti in u_i e v_j rispettivamente e n è il numero totale degli oggetti (Hubert e Arabie 1985). ARI assume un valore pari a 1 se vi è perfetta coincidenza tra le due partizioni e valore pari a 0 se le due partizioni non concordano più di quanto

atteso tra due partizioni casuali.
Inoltre, è stato calcolato anche l'indice di Silhouette che indica quanto un gruppo di osservazioni è ben raggruppato. Per una osservazione i è definito

$$sil(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

dove $a(i)$ è la distanza media tra l'osservazione i e tutte le altre osservazioni del cluster a cui appartiene i, $b(i)$ è la minima distanza media tra l'osservazione i e le osservazioni dei cluster a cui non appartiene i (Rousseeuw 1987).

Come si può osservare nelle Figure 3.5-3.7, la distribuzione dell'ARI è concentrata maggiormente su valori prossimi a 1, indicando che le prestazioni dei modelli sono buone.
Confrontando le figure, sembra che l'ARI non cambi molto fra i modelli con diversa specificazione del numero di fattori latenti.
Tuttavia, se ci si concentra ad esempio sulla Figura 3.5, si nota un netto peggioramento generale passando dai dati senza dropout ai dati con dropout: in particolare, il modello ZINB sembra avere risultati migliori degli altri in caso di dropout, seguito dal modello GLM-PCA con distribuzione multinomiale.
Lo stesso peggioramento generale è evidente anche passando dai dati senza effetti di batch ai dati con effetti di batch: la cosa che salta subito all'occhio è il peggioramento dei modelli GLM-PCA con l'aggiunta della covariata che indica il batch. Questo sembra andare contro ciò che ci si sarebbe aspettato, ovvero che, introducendo la variabile che indica a quale batch appartengono le cellule, le prestazioni sarebbero state migliori, come accade in maniera molto evidente per il modello ZINB.
Inoltre, un'altra cosa insolita è l'ARI del modello GLM-PCA binomiale negativo che ha una distribuzione meno variabile in presenza dei batch effect.
Sarebbe opportuno indagare sulle cause di questi risultati in ulteriori studi

approfonditi che vanno oltre gli scopi di questa tesi.

Invece, per quanto riguarda i dati senza dropout la distribuzione dell'indice non subisce variazioni evidenti tra i dataset con o senza batch per i modelli NB e GLM-PCA multinomiale.

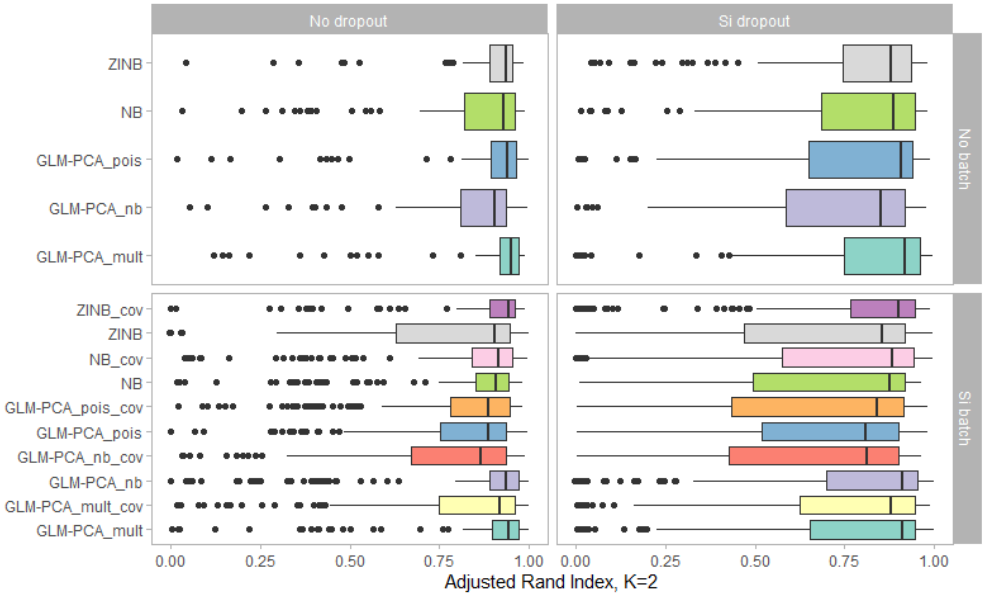


Figura 3.5: Adjusted Rand Index, K=2

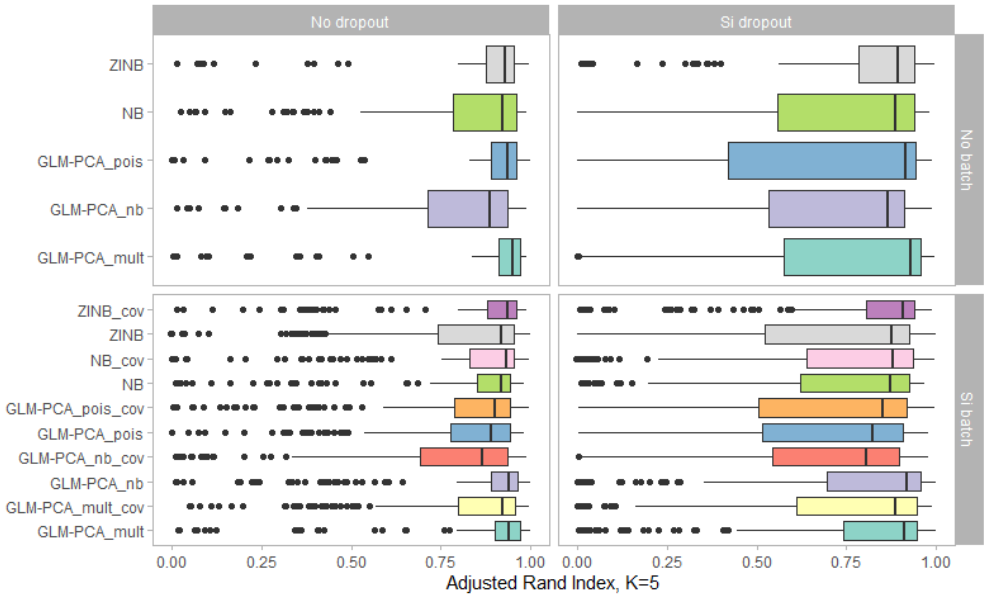


Figura 3.6: Adjusted Rand Index, K=5

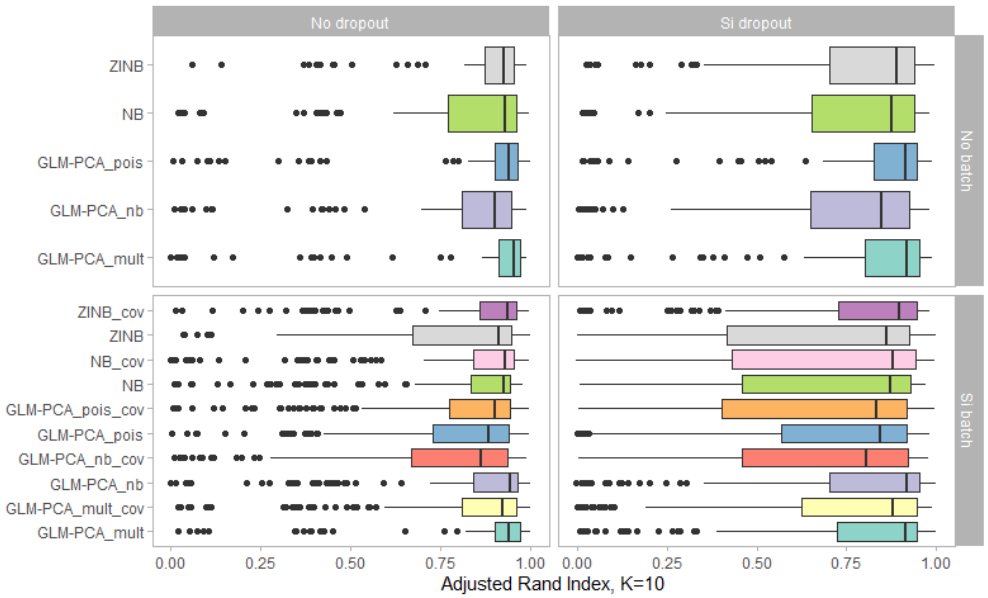


Figura 3.7: Adjusted Rand Index, K=10

La distribuzione dell'indice di Silhouette medio è riportata nelle Figure 3.8-3.10. L'indice è stato calcolato prendendo in considerazione sia i gruppi stimati dall'algoritmo k-menas (etichette stimate), sia i gruppi simulati con Splat (etichette vere) per avere un indice di paragone.
Si nota che la distribuzione non cambia tra l'indice calcolato con le etichette vere e quello con le etichette stimate.
Si vede chiaramente che questo indice presenta un'alta variabilità e che la distribuzione è concentrata su valori principalmente bassi, ma, poiché questo fenomeno è riscontrato anche per le etichette vere, non sembra essere una cosa allarmante.
Inoltre, anche per le diverse specificazioni del numero di fattori latenti non vi sono differenze marcate.
Quindi, soffermandosi ad esempio sulla Figura 3.8, non si osservano evidenti differenze tra i dataset con e senza dropout, con e senza batch e tra i diversi modelli.

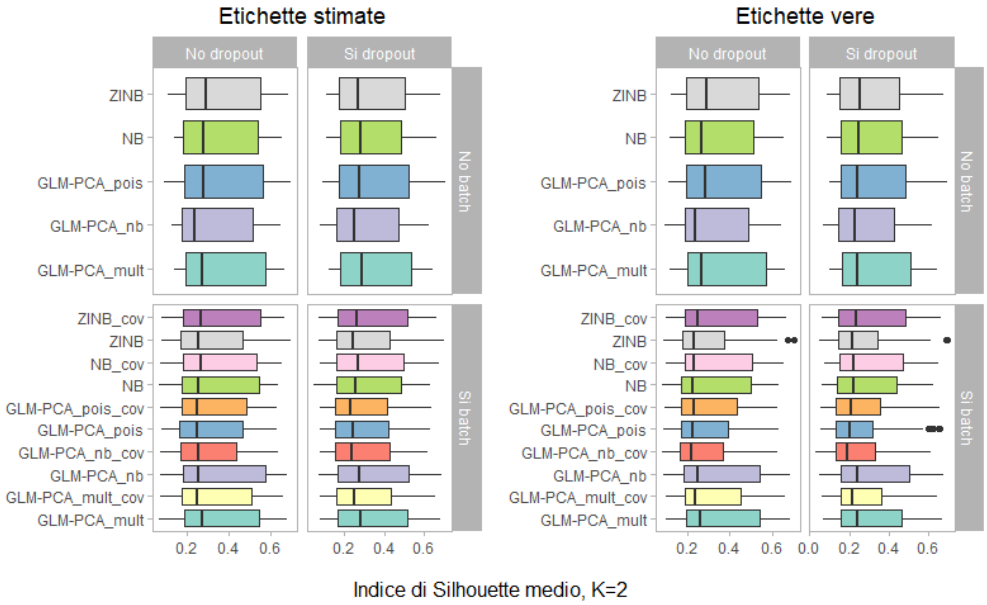


Figura 3.8: Indice di Silhouette medio, K=2

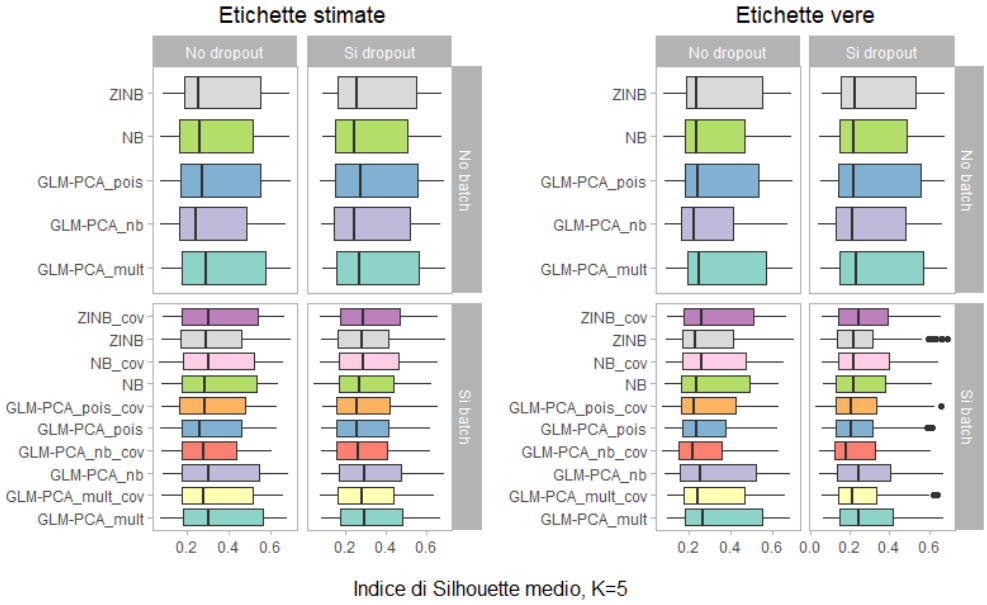


Figura 3.9: Indice di Silhouette medio, K=5

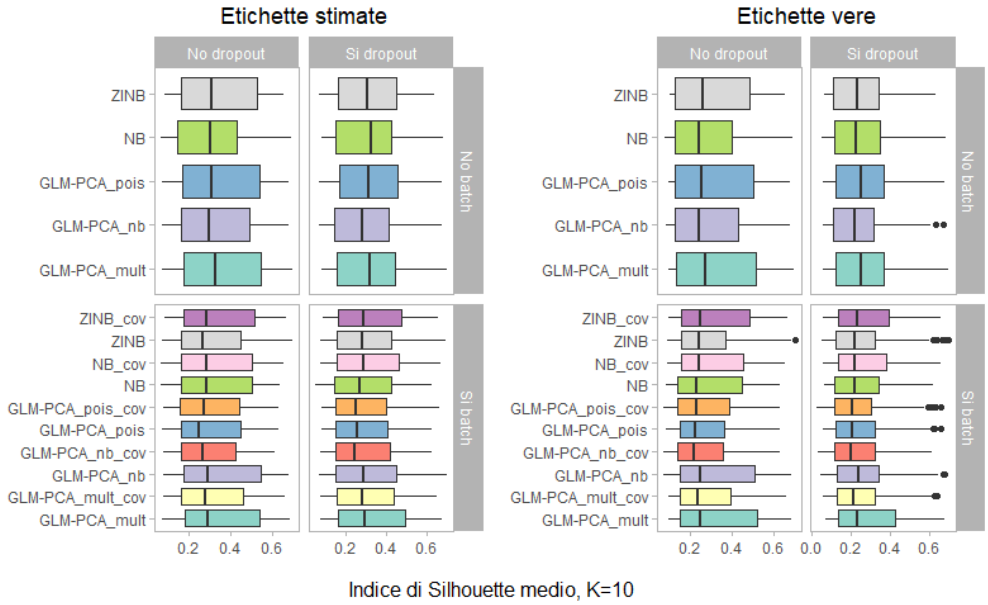


Figura 3.10: Indice di Silhouette medio, K=10

3.3 Tempo computazionale

Dopo aver confrontato i modelli dal punto di vista delle loro capacità, risulta opportuno confrontarli anche dal punto di vista del tempo computazionale [4] impiegato per stimare i parametri.

Nella Figura 3.11 si può osservare come cambiano i tempi computazionali in base al numero di cellule presenti nel dataset. Gli altri parametri sono rimasti invariati tra i modelli stimati: si sono impostati il numero di fattori latenti pari a due, il numero di geni pari a 1000 e la dispersione comune a tutti i geni.

Si nota che il modello più veloce è GLM-PCA, mentre quello più lento è ZINB. In particolare, la differenza tra tutti e tre i modelli diventa molto

[4]Per determinare il tempo computazionale è stato usato un PC con Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz con 64 GB di RAM

marcata quando il numero di cellule passa da 1000 a 10000: NB ci impiega il quadruplo del tempo, ZINB sette volte di più, mentre GLM-PCA presenta solo un leggero incremento.
In conclusione, quando il numero di cellule è molto elevato, NB impiega nove volte il tempo di GLM-PCA, mentre ZINB impiega 29 volte il tempo di GLM-PCA.

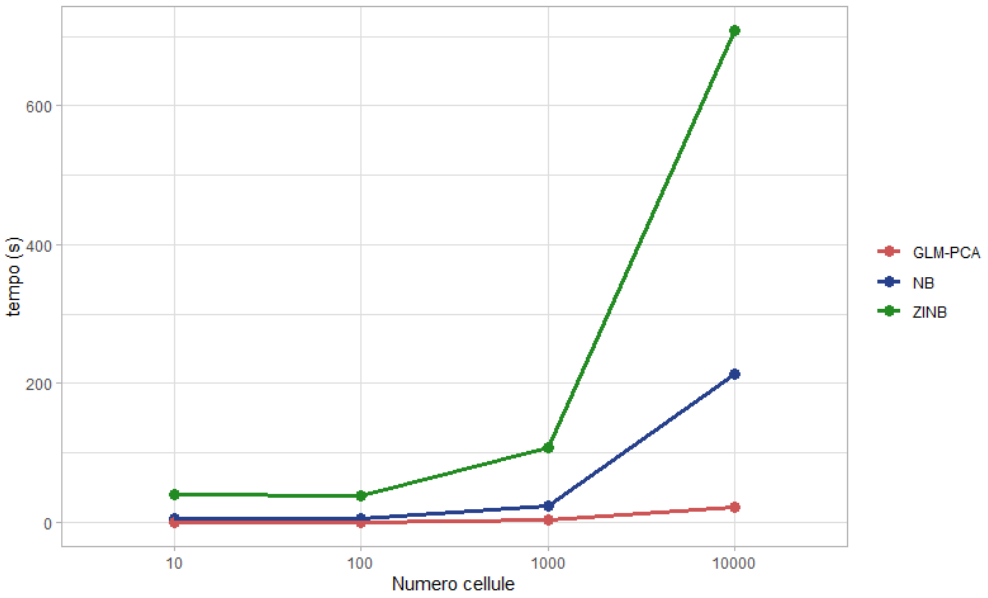


Figura 3.11: Tempo computazionale di NB e GLM-PCA in base al numero di cellule

In Figura 3.12 si può vedere come cambiano i tempi computazionali in base al numero di fattori latenti (va ricordato che il valore di questo parametro è scelto dall'utente). Gli altri parametri sono rimasti invariati tra i modelli stimati: si sono impostati il numero di cellule e di geni pari a 1000 e la dispersione comune a tutti i geni.
La prima cosa che si nota è la crescita quadratica di ZINB, mentre gli altri due modelli sembrano avere una crescita lineare con una differenza poco

marcata anche in caso di fattori latenti pari a 20. In quest'ultimo caso NB ci mette il doppio del tempo impiegato da GLM-PCA, mentre ZINB ci mette 23 volte tanto.

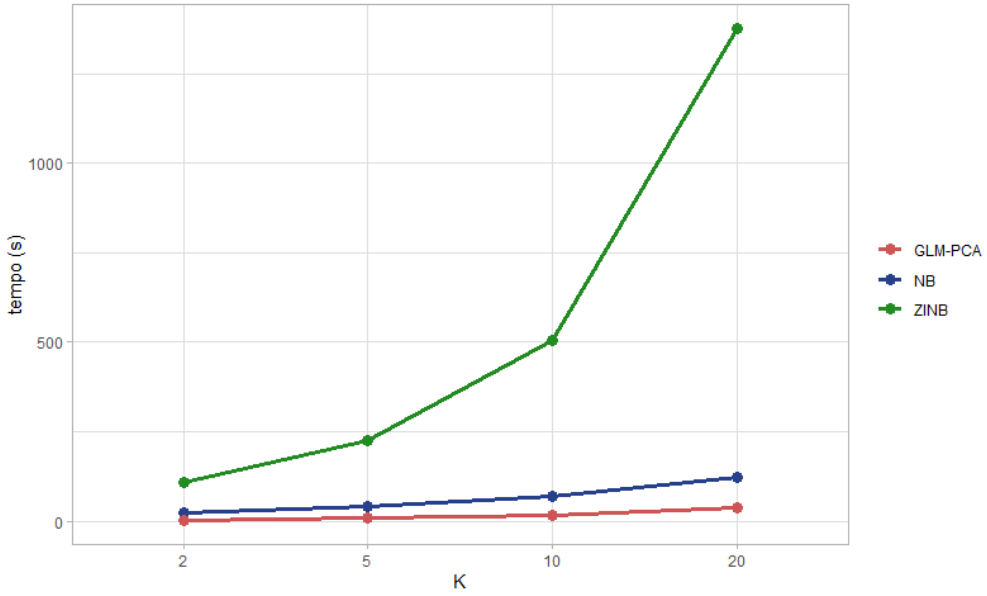


Figura 3.12: Tempo computazionale di NB e GLM-PCA in base al numero di fattori latenti

Capitolo 4

Applicazione su dati reali

Dopo aver verificato le prestazioni dei modelli per dati di scRNA-seq su dataset simulati, si vuole fornire ora un'applicazione del modello binomiale negativo presentato in questa tesi su dati reali.
Lo scopo di queste analisi è verificare se il modello NB è in grado di saper cogliere il segnale di interesse nei dati, evitando di farsi guidare dalla variabilità indesiderata indotta dagli effetti di batch. Questo si traduce nell'identificare le varie tipologie cellulari presenti nei dati e osservare se nella rappresentazione a bassa dimensionalità, le cellule di uno stesso gruppo sono vicine tra loro.
I dati oggetto di studio sono stati resi disponibili dal pacchetto `HCAData` di Bioconductor, che permette un accesso diretto ai dataset generati dal progetto *Human Cell Atlas*.
Il dataset con tecnologia UMI che è stato analizzato è `ica_bone_marrow` ed è composto da 33694 geni e 378000 cellule del midollo osseo di 8 donatori sani. Per ogni donatore sono state create 8 librerie (tranne per il donatore 6 che presenta solo 7 librerie).

A causa delle grandi dimensioni dei dati, si è preferito selezionare 50 cellule per ogni libreria in modo da accorciare i tempi computazionali. Inoltre, si è effettuato il filtraggio dei geni, ovvero si sono eliminati quei geni che non presentavano almeno 5 reads in almeno 5 campioni. Perciò, il dataset finale è costituito da 2151 geni e 3150 cellule.
In questo contesto l'effetto del donatore è considerato un effetto di batch che deve essere eliminato. Proprio per questo motivo la variabile che indica il donatore verrà inserita come covariata nella matrice del disegno.
Come prima cosa sono state osservate le caratteristiche dei dati con delle analisi esplorative.
I classici grafici utilizzati per l'analisi esplorativa dei dati di RNA-seq sono difficili da visualizzare per i dati di scRNA-seq a causa dell'elevata numerosità campionaria. Tuttavia, per comprendere la struttura dei dati, sono state selezionate casualmente 60 cellule ottenendo così il grafico del numero totale di reads mappate per ogni cellula e il grafico RLE (Relative Log Expression). Quest'ultimo è un metodo utile per visualizzare la presenza di variabilità indesiderata: per ogni gene si calcola l'espressione mediana tra i campioni e successivamente si calcola la devianza dalla mediana; poi, per ogni campione si genera un boxplot delle devianze (Gandolfo e Speed 2018).
Dalla Figura 4.1 si nota che alcune cellule presentano un numero totale di reads molto maggiore delle altre, in particolare questo è più evidente per i donatori 2 e 6.

Inoltre, dalla Figura 4.2 è evidente che è necessaria una normalizzazione dei dati. Tuttavia, va ricordato che l'intercetta gene specifica del modello NB agisce da fattore di normalizzazione globale, rendendo non necessaria la normalizzazione preliminare dei dati.

Per concludere le analisi esplorative è stato utilizzato il metodo t-distributed

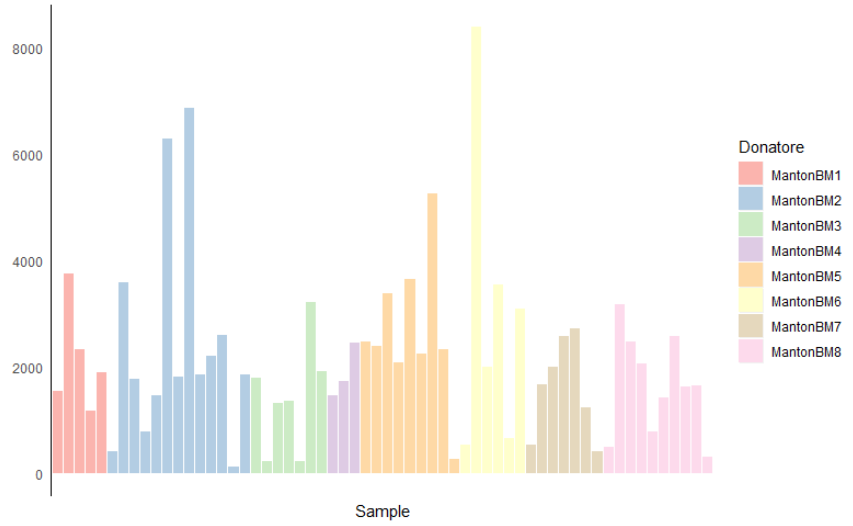


Figura 4.1: Grafico a barre del numero totale di reads mappate per cellula

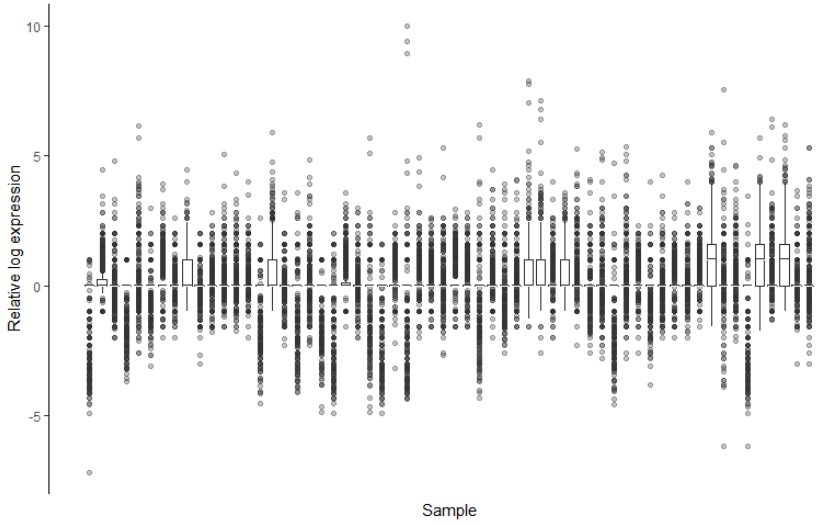


Figura 4.2: Grafico RLE

Stochastic Neighbor Embedding (t-SNE) per visualizzare i dati in due dimensioni. Si tratta di un metodo non lineare utile per la rappresentazione a bassa dimensionalità dei dati (*embedding*) (Van Der Maaten e Hinton 2008). Come si può osservare in Figura 4.3 è evidente che vi siano dei gruppi di cellule simili. Tuttavia, sembra non esserci un effetto evidente dovuto al donatore.

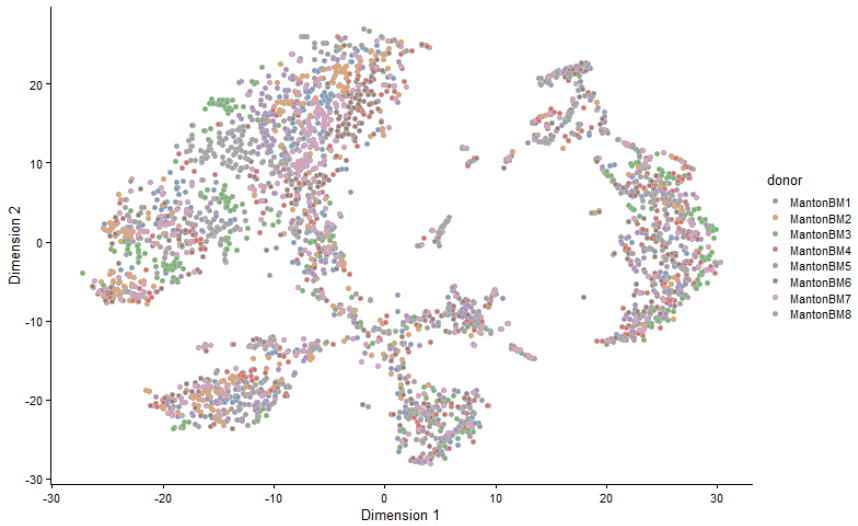


Figura 4.3: Grafico della rappresentazione t-SNE colorato per i diversi donatori

Per quanto riguarda la parte inferenziale, si è stimato il modello binomiale negativo (Paragrafo 2.3) con dispersione comune, con intercetta e covariata che indica il donatore e numero di fattori latenti pari a 2.
Successivamente, sono stati identificati i tipi cellulari con il pacchetto `SingleR` di Bioconductor. Questo pacchetto esegue il riconoscimento del tipo cellulare in maniera indipendente per ogni cellula del dataset, sfruttando alcuni set di dati transcriptomici di riferimento di tipi di cellule pure.
In Figura 4.4 sono illustrate le fasi dell'algoritmo di `SingleR` (Aran et al. 2019): nella prima fase, si calcola il coefficiente di Spearman tra l'espressione

genica di ogni cellula del dataset da etichettare e ciascuno dei campioni nel set di dati di riferimento, utilizzando solo i geni più variabili di quest'ultimo [1]. Questo processo si esegue in modo iterativo selezionando ogni volta solo i tipi di cellule con correlazione maggiore e i geni più variabili. L'algoritmo termina quando rimane un solo tipo di cellula.

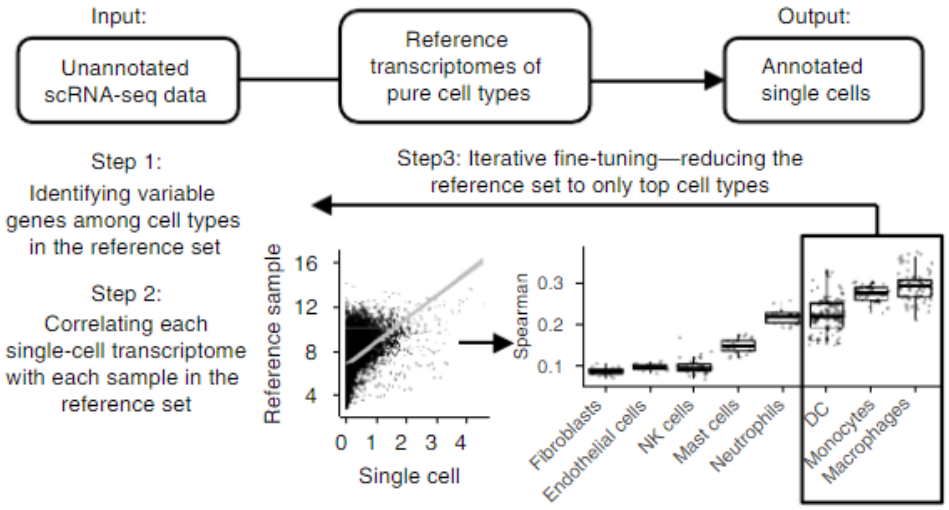


Figura 4.4: Descrizione dell'algoritmo `SingleR`

Il dataset `ica_bone_marrow` è stato etichettato prendendo come riferimento il dataset `HumanPrimaryCellAtlasData`, che presenta 713 campioni, 37 etichette generali e 157 etichette specifiche.

Per questa analisi sono stati considerati solo i geni comuni fra due dataset (1882).

Sono stati identificati 31 tipi cellulari che sono elencati in Tabella 4.1. È evidente che alcune tipologie cellulari sono state erroneamente identificate,

[1] L'uso di soli geni molto variabili tra i tipi di cellule aumentata la capacità di distinguere tipi di cellule strettamente correlate

come i neuroni o i gametociti, ma questo non è importante ai fini di queste analisi.

Tabella 4.1: Tipologia di cellule del dataset `ica_bone_marrow`

Tipo cellulare	**N. cellule**	**Tipo cellulare**	**N. cellule**
Astrociti	4	Macrofago	2
B	415	MEP	59
BM	20	Monociti	531
BM & Prog.	10	Mielociti	25
Condrociti	9	Cellule neuroepiteliali	2
CMP	37	Neuroni	5
DC	2	Neutrofili	60
Cellule endoteliali	2	NK	291
Eritroblasti	150	Piastrine	37
Fibroblasti	1	Cellule pre-B CD34-	24
Gametociti	1	Cellule pro-B CD34+	72
GMP	25	Pro-Mielociti	13
Epatociti	4	Cellule muscolare lisce	8
HSC-G-CSF	24	T	1293
HSC_CD34+	20	Cellule staminali tissutali	2
Cheratinociti	2		

In Figura 4.5 si può osservare che le cellule appartenenti allo stesso gruppo sono molto vicine tra loro, sottolineando il fatto che il segnale raccolto dal modello NB nella matrice a dimensionalità ridotta W non è influenzato dalla variabilità indotta dal donatore.

Inoltre, si è deciso di confrontare questa rappresentazione dei dati in due dimensioni con quella ottenuta dal modello binomiale negativo con inflazione

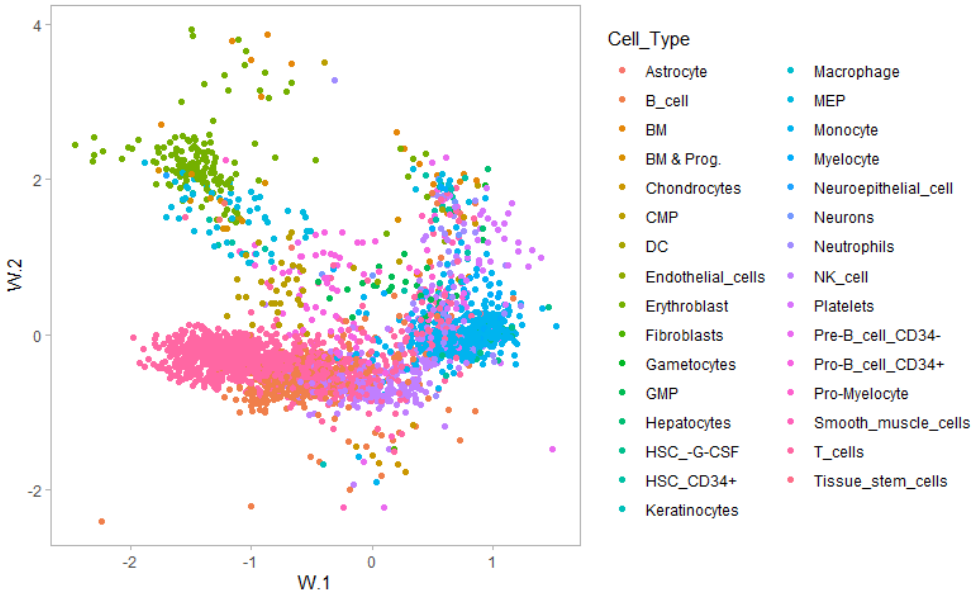


Figura 4.5: Rappresentazione grafica dei dati in due dimensioni colorata per tipo cellulare, modello NB

di zeri (Paragrafo 2.1). Il modello stimato presenta dispersione comune, con intercetta e covariata che indica il donatore e numero di fattori latenti pari a 2.

Come si nota in Figura 4.6, le due rappresentazioni differiscono nella vicinanza delle cellule appartenenti allo stesso gruppo. Ad esempio, nel modello ZINB il gruppo delle cellule T (rosa) contiene al suo interno anche alcune cellule B (arancione); anche il gruppo degli eritroblasti (verde) sembra essere poco coeso. Tuttavia, questo non è riscontrato nel modello NB.

Questi risultati evidenziano la capacità del modello NB nell'eliminare la variabilità indesiderata indotta dall'effetto donatore, riducendo la dimensionalità dei dati senza perdere il segnale d'interesse.

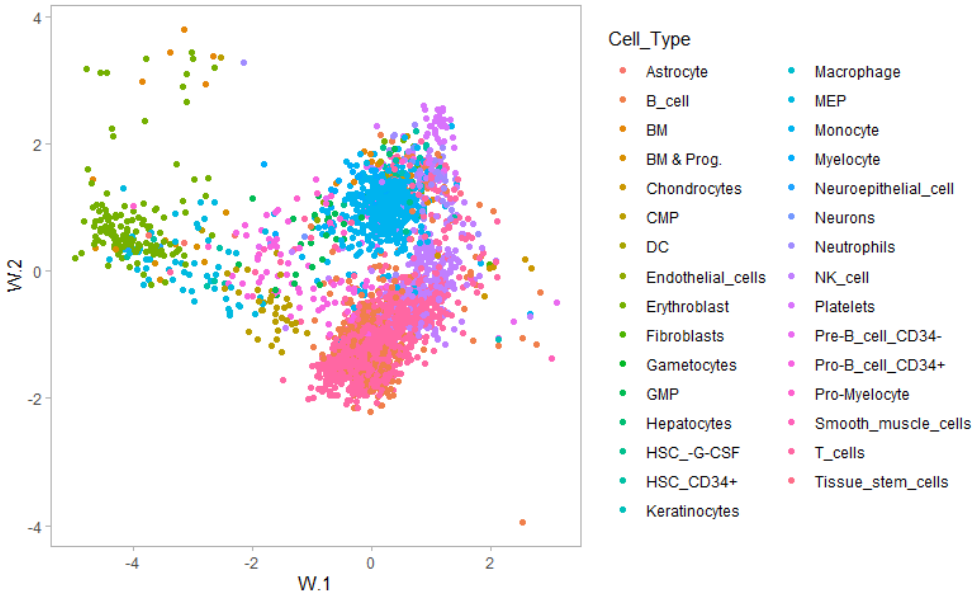


Figura 4.6: Rappresentazione grafica dei dati in due dimensioni colorata per tipo cellulare, modello ZINB

Conclusioni

Giunti alla fine di questa tesi è bene riassumere quanto è stato fatto per non perdere di vista l'obiettivo.
L'argomento principale su cui si è snodata l'intera tesi sono i dati RNA-seq a singola cellula, in particolare i dati con una nuova tecnologia, detta UMI, che riduce la distorsione dovuta all'amplificazione.
Infatti, molto spesso scRNA-seq non riesce a cogliere l'espressione di alcuni geni, in particolare dei geni poco espressi, che non vengono amplificati prima della fase di sequenziamento, portando ad un eccesso di zeri nella matrice dei conteggi.
Di conseguenza, gli usuali metodi statitici per analizzare i dati di RNA-seq non sono più applicabili ai dati di RNA-seq a singola cellula. Proprio per questo motivo Risso, Perraudeau et al. (2018) hanno proposto un modello di analisi fattoriale binomiale negativo con inflazione di zeri (ZINB).
Tuttavia, con i conteggi UMI non si verifica l'inflazione di zeri, perciò Townes et al. (2019) hanno suggerito un'estensione alla famiglia esponenziale dell'analisi delle componenti principali (GLM-PCA).
In questa tesi si è presentato il modello di analisi fattoriale binomiale negativo (NB) per dati UMI, in cui è stata eliminata la parte che modella l'inflazione di zeri del modello ZINB.

Il modello NB presenta un'elevata accuratezza nella stima della matrice dei fattori latenti e si è visto che le sue capacità sono molto buone anche in presenza di dropout, quando il parametro di dispersione è gene-specifico, evidenziando la notevole flessibilità di questo modello.
Soffermandosi sulle prestazioni nella riduzione della dimensionalità dei dati, i modelli NB e GLM-PCA hanno risultati migliori in assenza di dropout. In particolare, hanno prestazioni paragonabili modellando dati senza dropout e con effetti di batch. Tuttavia, GLM-PCA presenta risultati insoliti aggiungendo alla matrice del disegno informazioni riguardo i batch effect.
Nell'applicazione ai dati reali è stato possibile riscontrare e confermare le capacità di NB nel ridurre la dimensionalità dei dati, preservando il segnale d'interesse, eliminando la variabilità indesiderata.

Riassumendo, le novità apportate in questa tesi sono molteplici.
È stato presentato un nuovo modello per dati di scRNA-seq con tecnologia UMI che incorpora la stima dei fattori latenti.
Per la prima volta sono stati confrontati tre modelli molto recenti per i dati di scRNA-seq dal punto di vista della riduzione della dimensionalità, un tema molto delicato e importante nell'ambito del sequenziamento a singola cellula.
Inoltre, sono state verificate le prestazioni di GLM-PCA in un contesto di dati senza tecnologia UMI, in quanto gli autori di GLM-PCA avevano esplicitamente dichiarato che il modello non era stato testato su dati senza tecnologia UMI.
Tuttavia, va sottolineato che sono presenti alcune limitazioni.
Ad esempio, si potevano indagare altre situazioni biologiche, come un numero maggiore di gruppi o di batch, oppure una divisione non equa delle cellule tra i gruppi o tra i batch.

Sarebbe stato opportuno effettuare anche degli studi di simulazione con il lignaggio delle cellule e verificare le capacità dei modelli in questo contesto. Certamente si è consapevoli del fatto che le sfaccettature della realtà sono infinite ed infinite sono le loro combinazioni, da qui nasce l'impossibilità di poterle testare tutte.
Si è cercato comunque di analizzare le principali situazioni che si possono riscontrare nell'analisi dei dati di scRNA-seq.

Una limitazione di tutti e tre i modelli riguarda l'impossibilità di effettuare un'analisi sui geni differenzialmente espressi. Per NB e ZINB questo sarebbe possibile calcolando la derivata seconda della funzione di verosimiglianza e costruendo opportuni test di Wald che tengano in considerazione la molteplicità. Si lascia pertanto una spunto per possibili sviluppi futuri.

In conclusione, i risultati confermano che è opportuno utilizzare ZINB in presenza di dropout, anche se anche il modello NB con dispersione gene-specifica risulta essere molto flessibile per questi dati.
Per i conteggi UMI i modelli NB e GLM-PCA hanno prestazioni più o meno simili, la differenza maggiore è il tempo computazionale. Si potrebbe pensare di rendere l'algoritmo di NB più veloce con una procedura iterativa stimando il modello su diversi sottoinsiemi di cellule.
Si consiglia di inserire sempre nella matrice del disegno le informazioni sulle cellule, come ad esempio gli effetti di batch se sono noti, per i modelli NB e ZINB. Per GLM-PCA valutare questa scelta con attenzione, confrontando i modelli con e senza covariate con opportuni indici.
Infine, si consiglia sempre di stimare i modelli con almeno due fattori latenti.

Appendice A

Lemma 1

Lemma 1. *Per qualsiasi matrice R e scalare positivo s e t, vale quanto segue:*

$$\min_{S,T:R=ST} \frac{1}{2}(s\|S\|^2 + t\|T\|^2) = \sqrt{st}\|R\|_* \,,$$

dove $\|A\|_* = \mathrm{tr}(\sqrt{A^*A})$. *Se* $R = R_L R_\Sigma R_R$ *è una decomposizione a valori singolari di* R, *allora una soluzione al problema di ottimo è:*

$$S = \left(\frac{t}{s}\right)^{\frac{1}{4}} R_L R_\Sigma^{\frac{1}{2}}\,, \quad T = \left(\frac{s}{t}\right)^{\frac{1}{4}} R_\Sigma^{\frac{1}{2}} R_R\,.$$

Dimostrazione. Siano $\tilde{S} = \sqrt{s}S$, $\tilde{T} = \sqrt{t}T$ e $\tilde{R} = \sqrt{st}R$. Allora, $\|\tilde{S}\|^2 = s\|S\|^2$, $\|\tilde{T}\|^2 = t\|T\|^2$ e $\tilde{S}\tilde{T}^2 = \sqrt{st}ST$, di conseguenza il problema di ottimizzazione è equivalente a:

$$\min_{\tilde{S},\tilde{T}:\tilde{R}=\tilde{S}\tilde{T}} \frac{1}{2}(\|\tilde{S}\|^2 + \|\tilde{T}\|^2)\,,$$

che da Srebro, Rennie e Jaakkola (2005) [Lemma 6] ha valore ottimo $\|\tilde{R}\|_* = \sqrt{st}\|R\|_*$, raggiunto a $S = \tilde{R}_L\tilde{R}_\Sigma^{\frac{1}{2}}$ e $T = \tilde{R}_\Sigma^{\frac{1}{2}}\tilde{R}_R$, dove $\tilde{R}_L\tilde{R}_\Sigma\tilde{R}_R$ è una decomposizione a valori singolari di $\tilde{R}$. Osservando che $\tilde{R}_L = R_L$, $\tilde{R}_R = R_R$ e $\tilde{R}_\Sigma = \sqrt{st}R_\Sigma$, si ha che una soluzione per il problema di ottimo è $S =$

$s^{-1/2}\tilde{S} = s^{-1/2}R_L(st)^{1/4}R_\Sigma^{1/2} = (t/s)^{1/4}R_L R_\Sigma^{1/2}$. Un ragionamento simile per T conclude la dimostrazione. □

Appendice B

solveZinbRegression

Data una matrice di conteggi N-dimensionale $y \in \mathbb{N}^N$, e date le matrici $A \in \mathbb{R}^{N\times p}, B \in \mathbb{R}^{N\times r}, (C_\mu, C_\theta) \in \mathbb{R}^N$, per alcuni interi p, q, r, la funzione `solveZinbRegression`$(y, A, B, C_\mu, C_\theta)$ cerca di trovare i parametri $(a, b) \in \mathbb{R}^p \times \mathbb{R}^q \times \mathbb{R}^r$ che massimizzano la funzione di log-verosimiglianza di y con parametri:

$$F(a, b) = \ln(\mu) = Aa + Bb + C_\mu$$
$$\ln(\theta) = C_\theta \,.$$

Partendo da dei valori iniziali, si esegue una minimizzazione locale della funzione $F(a, b)$ utilizzando il metodo quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS). Di seguito si illustra come viene calcolato il gradiente di F. Osservando che

$$\frac{\partial}{\partial\mu} \ln f_{NB}(y; \mu, \theta) = \frac{y}{\mu} - \frac{y+\theta}{\mu+\theta} \,,$$

e che $(\ln^{-1})'(\ln \mu) = \mu$, si ha che

$$\nabla_a F = A^T G$$
$$\nabla_b F = B^T G \,,$$

dove G è il vettore N-dimensionale dato da

$$\forall i \in [1, N] \quad G_i = \mu_i \frac{\partial}{\partial \mu_i} \ln f_{NB}(y; \mu_i, \theta_i)$$

Appendice C

Codice R

Codice C.1: Inizializzazione

```
nbInitialize <- function(m, Y, nb.repeat=2,  it.max = 100,
                         BPPARAM=BiocParallel::bpparam()) {

  n <- NROW(Y)
  J <- NCOL(Y)

  if(n != nSamples(m)) {
    stop("Y needs to have ", nSamples(m), " rows (genes)")
  }

  if(J != nFeatures(m)) {
    stop("Y needs to have ", nFeatures(m), " columns (samples)")
  }

  ## 1. Define P
```

```
P <- Y > 0

if(any(rowSums(P) == 0)) {
  stop("Sample ", which(rowSums(P) == 0)[1], " has only 0 counts!")
}

if(any(colSums(P) == 0)) {
  stop("Gene ", which(colSums(P) == 0)[1], " has only 0 counts!")
}

## 2. Define L
L <- matrix(NA, nrow=n, ncol=J)
L <- log1p(Y) - m@O_mu




## 3. Estimate gamma_mu and beta_mu
iter <- 0
beta_mu <- getBeta_mu(m)
gamma_mu <- getGamma_mu(m)

while (iter < nb.repeat) {

  # Optimize gamma_mu (in parallel for each sample)
  if (NCOL(getV_mu(m)) == 0) {
    iter <- nb.repeat # no need to estimate gamma_mu nor to iterate
  } else {
    Xbeta_mu <- getX_mu(m) %*% beta_mu
    gamma_mu <- matrix(unlist(bplapply(seq(n), function(i) {
```

```
      solveRidgeRegression(x=getV_mu(m),
                           y=L[i,] - Xbeta_mu[i,],
                           epsilon = getEpsilon_gamma_mu(m),
                           family="gaussian")
    } , BPPARAM=BPPARAM
    )), nrow=NCOL(getV_mu(m)))
  }

  # Optimize beta_mu (in parallel for each gene)
  if (NCOL(getX_mu(m)) == 0) {
    iter <- nb.repeat # no need to estimate gamma_mu nor to iterate
  } else {
    tVgamma_mu <- t(getV_mu(m) %*% gamma_mu)
    beta_mu <- matrix(unlist(bplapply(seq(J), function(j) {
      solveRidgeRegression(x=getX_mu(m),
                           y=L[,j] - tVgamma_mu[, j],
                           epsilon = getEpsilon_beta_mu(m),
                           family="gaussian")
    }, BPPARAM=BPPARAM
    )), nrow=NCOL(getX_mu(m)))
  }

  iter <- iter+1
}

## 4. Estimate W and alpha (only if K>0)
if(nFactors(m) > 0) {

  # Compute the residual D
```

```
  D <- L - getX_mu(m) %*% beta_mu - t(getV_mu(m) %*% gamma_mu)

  # SVD
  R <- svd(D, nu=nFactors(m), nv=nFactors(m))


  # Orthogonalize to get W and alpha
  W <- (getEpsilon_alpha(m) / getEpsilon_W(m))[1]^(1/4) *
    R$u %*% diag(sqrt(R$d[1:nFactors(m)]), nrow = length(R$d[1:nFactors(
        m)]))
  alpha_mu <- (getEpsilon_W(m)/getEpsilon_alpha(m))[1]^(1/4) *
    diag(sqrt(R$d[1:nFactors(m)]),nrow = length(R$d[1:nFactors(m)])) %*%
         t(R$v)
} else {
  W <- getW(m)
  alpha_mu <- getAlpha_mu(m)
}


## 5. Initialize dispersion to 0
zeta <- rep(0, J)

out <-  zinbModel(X = m@X, V = m@V, O_mu = m@O_mu,
                  which_X_mu = m@which_X_mu,
                  which_V_mu = m@which_V_mu,
                  W = W, beta_mu = beta_mu,
                  gamma_mu = gamma_mu,
                  alpha_mu = alpha_mu, zeta = zeta,
                  epsilon_beta_mu = m@epsilon_beta_mu,
```

```
                    epsilon_gamma_mu = m@epsilon_gamma_mu,
                    epsilon_W = m@epsilon_W, epsilon_alpha = m@epsilon_
                        alpha,
                    epsilon_zeta = m@epsilon_zeta)

  return(out)
}
```

Codice C.2: Ottimizzazione

```
nbOptimize <- function(m, Y, commondispersion=TRUE, maxiter=25,
                           stop.epsilon=.0001, verbose=FALSE,
                           BPPARAM=BiocParallel::bpparam()) {

  total.lik=rep(NA,maxiter)
  n <- nSamples(m)
  J <- nFeatures(m)

  epsilonright <- c(getEpsilon_beta_mu(m), getEpsilon_alpha(m))

  nright <- c(length(getEpsilon_beta_mu(m)), length(getEpsilon_alpha(m)))

  optimright = (sum(nright)>0)

  epsilonleft <- c(getEpsilon_gamma_mu(m), getEpsilon_W(m))
  nleft <- c(length(getEpsilon_gamma_mu(m)), length(getEpsilon_W(m)))
  optimleft = (sum(nleft)>0)

  orthog <- (nFactors(m)>0)
```

```
# extract fixed quantities from m
X_mu <- getX_mu(m)
V_mu <- getV_mu(m)
O_mu <- m@O_mu

# exctract paramters from m (remember to update!)
beta_mu <- getBeta_mu(m)
alpha_mu <- getAlpha_mu(m)
gamma_mu <- getGamma_mu(m)
W <- getW(m)
zeta <- getZeta(m)

for (iter in seq_len(maxiter)){
  if (verbose) {message("Iteration ",iter)}

  # Evaluate total penalized likelihood
  mu <- exp(X_mu %*% beta_mu + t(V_mu %*% gamma_mu) +
              W %*% alpha_mu + O_mu)

  theta <- exp(zeta)

  loglik <- nb.loglik(Y, mu, rep(theta, rep(n, J)))

  penalty <- sum(getEpsilon_alpha(m) * (alpha_mu)^2)/2 +
             sum(getEpsilon_beta_mu(m) * (beta_mu)^2)/2 +
             sum(getEpsilon_gamma_mu(m)*(gamma_mu)^2)/2 +
             sum(getEpsilon_W(m)*t(W)^2)/2 +
             getEpsilon_zeta(m)*var(zeta)/2
```

```
total.lik[iter] <- loglik - penalty

if (verbose) {message("penalized log–likelihood = ",
                      total.lik[iter])}

# If the increase in likelihood is smaller than 0.5%, stop
    maximization
if(iter > 1){
  if(abs((total.lik[iter]-total.lik[iter-1]) /
         total.lik[iter-1])<stop.epsilon)
    break
}

# 1. Optimize dispersion
zeta <- nbOptimizeDispersion(J, mu, getEpsilon_zeta(m), Y,
                             commondispersion=commondispersion,
                             BPPARAM=BPPARAM)

# Evaluate total penalized likelihood
if (verbose) {
  pen <- sum(getEpsilon_alpha(m) * (alpha_mu)^2)/2 +
         sum(getEpsilon_beta_mu(m) * (beta_mu)^2)/2 +
         sum(getEpsilon_gamma_mu(m)*(gamma_mu)^2)/2 +
         sum(getEpsilon_W(m)*t(W)^2)/2 +
         getEpsilon_zeta(m)*var(zeta)/2
  message("After dispersion optimization = ",
          nb.loglik(Y, mu, exp(zeta)) - pen)
}
```

```
# 2. Optimize right factors

if (optimright) {
  ptm <- proc.time()
  estimate <- matrix(unlist(
    bplapply(seq(J), function(j) {
      optimright_fun(beta_mu[,j], alpha_mu[,j], Y[,j], X_mu,
                     W, V_mu[j,], gamma_mu, O_mu[,j], zeta[j], n,
                         epsilonright)
    },BPPARAM=BPPARAM)), nrow=sum(nright))

  if (verbose) {print(proc.time()-ptm)}
  ind <- 1
  if (nright[1]>0) {
    beta_mu <- estimate[ind:(ind+nright[1]-1),,drop=FALSE]
    ind <- ind+nright[1]
  }
  if (nright[2]>0) {
    alpha_mu <- estimate[ind:(ind+nright[2]-1),,drop=FALSE]
    ind <- ind+nright[2]
  }

}
# Evaluate total penalized likelihood
if (verbose) {
  itermu <- exp(X_mu %*% beta_mu + t(V_mu %*% gamma_mu) +
                  W %*% alpha_mu + O_mu)
```

```
  pen <- sum(getEpsilon_alpha(m) * (alpha_mu)^2)/2 +
         sum(getEpsilon_beta_mu(m) * (beta_mu)^2)/2 +
         sum(getEpsilon_gamma_mu(m)*(gamma_mu)^2)/2 +
         sum(getEpsilon_W(m)*t(W)^2)/2 +
         getEpsilon_zeta(m)*var(zeta)/2
  message("After right optimization = ",
          nb.loglik(Y, itermu, exp(zeta)) - pen)
}

# 3. Orthogonalize
if (orthog) {
  o <- orthogonalizeTraceNorm(W, alpha_mu, m@epsilon_W, m@epsilon_
      alpha)
  W <- o$U
  alpha_mu <- o$V
}

# Evaluate total penalized likelihood
if (verbose) {
  itermu <- exp(X_mu %*% beta_mu + t(V_mu %*% gamma_mu) +
                  W %*% alpha_mu + O_mu)

  pen <- sum(getEpsilon_alpha(m) * (alpha_mu)^2)/2 +
         sum(getEpsilon_beta_mu(m) * (beta_mu)^2)/2 +
         sum(getEpsilon_gamma_mu(m)*(gamma_mu)^2)/2 +
         sum(getEpsilon_W(m)*t(W)^2)/2 +
         getEpsilon_zeta(m)*var(zeta)/2

  message("After orthogonalization = ",
```

```
          nb.loglik(Y, itermu, exp(zeta)) - pen)
}

# 4. Optimize left factors
if (optimleft) {
  ptm <- proc.time()
  estimate <- matrix(unlist(
    bplapply(seq(n), function(i) {
      optimleft_fun(gamma_mu[,i], W[i,], Y[i,], V_mu, alpha_mu,
                    X_mu[i,], beta_mu, O_mu[i,], zeta, epsilonleft)
    }, BPPARAM=BPPARAM)), nrow=sum(nleft))

  if (verbose) {print(proc.time()-ptm)}
  ind <- 1
  if (nleft[1]>0) {
    gamma_mu <- estimate[ind:(ind+nleft[1]-1),,drop=FALSE]
    ind <- ind+nleft[1]
  }
  if (nleft[2]>0) {
    W <- t(estimate[ind:(ind+nleft[2]-1),,drop=FALSE])
    ind <- ind+nleft[2]
  }
}

# Evaluate total penalized likelihood
if (verbose) {
  itermu <- exp(X_mu %*% beta_mu + t(V_mu %*% gamma_mu) +
                  W %*% alpha_mu + O_mu)
```

```
  pen <- sum(getEpsilon_alpha(m) * (alpha_mu)^2)/2 +
         sum(getEpsilon_beta_mu(m) * (beta_mu)^2)/2 +
         sum(getEpsilon_gamma_mu(m)*(gamma_mu)^2)/2 +
         sum(getEpsilon_W(m)*t(W)^2)/2 +
         getEpsilon_zeta(m)*var(zeta)/2

  message("After left optimization = ",
          nb.loglik(Y, itermu, exp(zeta)) - pen)
}

# 5. Orthogonalize
if (orthog) {
  o <- orthogonalizeTraceNorm(W, alpha_mu,
                              m@epsilon_W, m@epsilon_alpha)
  W <- o$U
  alpha_mu <- o$V[,1:J,drop=FALSE]
}
# Evaluate total penalized likelihood
if (verbose) {
  itermu <- exp(X_mu %*% beta_mu + t(V_mu %*% gamma_mu) +
                  W %*% alpha_mu + O_mu)


  pen <- sum(getEpsilon_alpha(m) * (alpha_mu)^2)/2 +
         sum(getEpsilon_beta_mu(m) * (beta_mu)^2)/2 +
         sum(getEpsilon_gamma_mu(m)*(gamma_mu)^2)/2 +
         sum(getEpsilon_W(m)*t(W)^2)/2 +
         getEpsilon_zeta(m)*var(zeta)/2
```

```
      message("After orthogonalization = ",
              nb.loglik(Y, itermu, exp(zeta)) - pen)
    }

  }

  out <- zinbModel(X = m@X, V = m@V, O_mu = m@O_mu,
                   which_X_mu = m@which_X_mu,
                   which_V_mu = m@which_V_mu,
                   W = W, beta_mu = beta_mu,
                   gamma_mu = gamma_mu,
                   alpha_mu = alpha_mu, zeta = zeta,
                   epsilon_beta_mu = m@epsilon_beta_mu,
                   epsilon_gamma_mu = m@epsilon_gamma_mu,
                   epsilon_W = m@epsilon_W, epsilon_alpha = m@epsilon_
                       alpha,
                   epsilon_zeta = m@epsilon_zeta)

  return(out)
}


################################################################################
nb.loglik <- function(Y, mu, theta) {

  # log-probabilities of counts under the NB model
  logPnb <- suppressWarnings(dnbinom(Y, size = theta, mu = mu, log = TRUE)
      )
```

```
  sum(logPnb)

}

##############################################################################
nb.loglik.dispersion <- function(zeta, Y, mu){

  nb.loglik(Y, mu, exp(zeta))

}

##############################################################################
nb.regression.parseModel <- function(alpha, A.mu, B.mu, C.mu) {

  n <- nrow(A.mu)
  logMu <- C.mu
  dim.alpha <- rep(0,2)
  start.alpha <- rep(NA,2)
  i <- 0

  j <- ncol(A.mu)
  if (j>0) {
    logMu <- logMu + A.mu %*% alpha[(i+1):(i+j)]
    dim.alpha[1] <- j
    start.alpha[1] <- i+1
    i <- i+j
```

```
  }

  j <- ncol(B.mu)
  if (j>0) {
    logMu <- logMu + B.mu %*% alpha[(i+1):(i+j)]
    dim.alpha[2] <- j
    start.alpha[2] <- i+1
  }

  return(list(logMu=logMu, dim.alpha=dim.alpha,
              start.alpha=start.alpha))
}


#############################################################################
nb.loglik.regression <- function(alpha, Y,
                                 A.mu = matrix(nrow=length(Y), ncol=0),
                                 B.mu = matrix(nrow=length(Y), ncol=0),
                                 C.mu = matrix(0, nrow=length(Y), ncol=1),
                                 C.theta = matrix(0, nrow=length(Y), ncol
                                     =1),
                                 epsilon=0) {

  # Parse the model
  r <- nb.regression.parseModel(alpha=alpha,
                                A.mu = A.mu,
                                B.mu = B.mu,
                                C.mu = C.mu)
```

```
  # Call the log likelihood function
  z <- nb.loglik(Y, exp(r$logMu), exp(C.theta))

  # Penalty
  z <- z - sum(epsilon*alpha^2)/2
  z
}

#############################################################################
nb.loglik.regression.gradient <- function(alpha, Y,
                                          A.mu = matrix(nrow=length(Y),
                                              ncol=0),
                                          B.mu = matrix(nrow=length(Y),
                                              ncol=0),
                                          C.mu = matrix(0, nrow=length(Y),
                                               ncol=1),
                                          C.theta = matrix(0, nrow=length(
                                              Y), ncol=1),
                                          epsilon=0) {

  # Parse the model
  r <- nb.regression.parseModel(alpha=alpha,
                                A.mu = A.mu,
                                B.mu = B.mu,
                                C.mu = C.mu)
  Y=as.vector(Y)
  theta <- exp(C.theta)
  mu <- exp(r$logMu)
```

```
n <- length(Y)

# Check what we need to compute,
# depending on the variables over which we optimize
need.wres.mu <- r$dim.alpha[1] >0 || r$dim.alpha[2] >0

# Compute the partial derivatives we need
## w.r.t. mu
if (need.wres.mu) {
  wres_mu <- numeric(length = n)
  wres_mu <- Y - mu *
      (Y + theta)/(mu + theta)
  wres_mu <- as.vector(wres_mu)
}

# Make gradient
grad <- numeric(0)

## w.r.t. a_mu
if (r$dim.alpha[1] >0) {
  istart <- r$start.alpha[1]
  iend <- r$start.alpha[1]+r$dim.alpha[1]-1
  grad <- c(grad , colSums(wres_mu * A.mu) -
              epsilon[istart:iend]*alpha[istart:iend])
}

## w.r.t. b
if (r$dim.alpha[2] >0) {
  istart <- r$start.alpha[2]
```

```
    iend <- r$start.alpha[2]+r$dim.alpha[2]-1
    grad <- c(grad , colSums(wres_mu * B.mu) -
                epsilon[istart:iend]*alpha[istart:iend])
  }

  grad
}


################################################################################
optimright_fun <- function(beta_mu, alpha_mu, Y, X_mu, W,
                           V_mu, gamma_mu, O_mu, zeta, n, epsilonright) {
  optim( fn=nb.loglik.regression,
         gr=nb.loglik.regression.gradient,
         par=c(beta_mu, alpha_mu),
         Y=Y,
         A.mu=cbind(X_mu, W),
         C.mu=t(V_mu %*% gamma_mu) + O_mu,
         C.theta=matrix(zeta, nrow = n, ncol = 1),
         epsilon=epsilonright,
         control=list(fnscale=-1,trace=0),
         method="BFGS")$par
}


################################################################################
optimleft_fun <- function(gamma_mu, W, Y, V_mu, alpha_mu,
                          X_mu, beta_mu, O_mu, zeta, epsilonleft) {
  optim( fn=nb.loglik.regression,
```

```
         gr=nb.loglik.regression.gradient,
         par=c(gamma_mu, t(W)),
         Y=t(Y),
         A.mu=V_mu,
         B.mu=t(alpha_mu),
         C.mu=t(X_mu%*%beta_mu + O_mu),
         C.theta=zeta,
         epsilon=epsilonleft,
         control=list(fnscale=-1,trace=0),
         method="BFGS")$par
}


##############################################################################
.is_wholenumber <- function(x, tol = .Machine$double.eps^0.5) {
  abs(x - round(x)) < tol
}


##############################################################################
nbOptimizeDispersion <- function(J, mu, epsilon,
                                 Y, commondispersion=TRUE,
                                 BPPARAM=BiocParallel::bpparam()) {

  # 1) Find a single dispersion parameter for all counts by 1-dimensional
  # optimization of the likelihood

    g=optimize(f=nb.loglik.dispersion, Y=Y, mu=mu,
               maximum=TRUE,interval=c(-100,100))
```

```
    zeta <- rep(g$maximum,J)

  if (!commondispersion) {

    # 2) Optimize the dispersion parameter of each sample
    locfun <- function(logt) {
      s <- sum(unlist(bplapply(seq(J),function(i) {
        nb.loglik.dispersion(logt[i],Y[,i],mu[,i])
      }, BPPARAM=BPPARAM)))
      if (J>1) {
        s <- s - epsilon*var(logt)/2
      }
      s
    }
    locgrad <- function(logt) {
      s <- unlist(bplapply(seq(J),function(i) {
        nb.loglik.dispersion.gradient(logt[i], Y[,i], mu[,i])
      }, BPPARAM=BPPARAM ))
      if (J>1) {
        s <- s - epsilon*(logt - mean(logt))/(J-1)
      }
      s
    }
    zeta <- optim(par=zeta, fn=locfun , gr=locgrad,
                  control=list(fnscale=-1,trace=0), method="BFGS")$par
  }
  return(zeta)
}
```

```
##############################################################################
nb.loglik.dispersion.gradient <- function(zeta, Y, mu) {
  theta <- exp(zeta)

  grad <- 0
  grad <- grad + sum( theta * (digamma(Y + theta) - digamma(theta) +
                                 zeta - log(mu + theta) + 1 -
                                 (Y + theta)/(mu + theta) ) )

  grad
}


##############################################################################
orthogonalizeTraceNorm <- function(U, V, a=1, b=1) {

  # do QR of U
  U.qr <- qr (U)
  U.Q <- qr.Q (U.qr)
  U.R <- qr.R (U.qr)

  # do QR of t(V)
  V.qr <- qr (t(V))
  V.Q <- qr.Q (V.qr)
  V.R <- qr.R (V.qr)

  # do SVD of the U.R %*% t(V.R) matrix to have orthog %*% diag %*% orthog
```

```
  A <- svd( U.R %*% t(V.R) )

  # Scaling factor
  s <- (a/b)^{1/4}

  # orthogonalized U
  U2 <- 1/s * U.Q %*% A$u %*% sqrt(diag(A$d,length(A$d)))

  # orthogonalized V and W
  V2 <- s * t( V.Q %*% A$v %*% sqrt(diag(A$d,length(A$d))) )

  list(U=U2, V=V2)
}
```

Bibliografia

Akima, Hiroshi (1970). «A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures». In: *Journal of the ACM (JACM)* 17.4, pp. 589–602.

Aran, Dvir et al. (2019). «Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage». In: *Nature Immunology* 20.

Benjamini, Yoav e Yosef Hochberg (1995). «Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing». In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1, pp. 289–300.

Chen, Yen-Chun et al. (2013). «Effects of GC Bias in Next-Generation-Sequencing Data on De Novo Genome Assembly». In: *PLoS ONE* 8.4, e62856.

Gandolfo, Luke C. e Terence P. Speed (2018). «RLE plots: Visualizing unwanted variation in high dimensional data». In: *PLOS ONE* 13.2, e0191629.

Hicks, Stephanie C., Mingxiang Teng e Rafael A. Irizarry (2015). «On the widespread and critical impact of systematic bias and batch effects in single-cell RNA-Seq data». In: *bioRxiv*.

Hicks, Stephanie C., F. William Townes et al. (2018). «Missing data and technical variability in single-cell RNA-sequencing experiments». In: *Biostatistics* 19.4, pp. 562–578.

Hubert, Lawrence e Phipps Arabie (1985). «Comparing partitions». In: *Journal of Classification* 2.1, pp. 193–218.

Islam, Saiful et al. (2014). «Quantitative single-cell RNA-seq with unique molecular identifiers». In: *Nature Methods* 11.2, pp. 163–166.

Johnson, W. Evan, Cheng Li e Ariel Rabinovic (2007). «Adjusting batch effects in microarray expression data using empirical Bayes methods». In: *Biostatistics* 8.1, pp. 118–127.

Law, Charity W. et al. (2014). «Voom: Precision weights unlock linear model analysis tools for RNA-seq read counts». In: *Genome Biology* 15.2.

Li, Jun e Robert Tibshirani (2013). «Finding consistent patterns: A nonparametric approach for identifying differential expression in RNA-Seq data». In: *Statistical Methods in Medical Research* 22.5, pp. 519–536.

Love, Michael I., Wolfgang Huber e Simon Anders (2014). «Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2». In: *Genome Biology* 15.12, pp. 1–21.

Macosko, Evan Z. et al. (2015). «Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets». In: *Cell* 161.5, pp. 1202–1214.

Mazumder, Rahul, Trevor Hastie e Robert Tibshirani (2010). «Spectral regularization algorithms for learning large incomplete matrices». In: *Journal of Machine Learning Research* 11, pp. 2287–2322.

Risso, Davide, John Ngai et al. (2014). «Normalization of RNA-seq data using factor analysis of control genes or samples». In: *Nature Biotechnology* 32.9, pp. 896–902.

Risso, Davide, Fanny Perraudeau et al. (2018). «A general and flexible method for signal extraction from single-cell RNA-seq data». In: *Nature Communications* 9.1.

Robinson, Mark D. e Gordon K. Smyth (2007). «Moderated statistical tests for assessing differences in tag abundance». In: *Bioinformatics* 23.21, pp. 2881–2887.

Rousseeuw, Peter J. (1987). «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis». In: *Journal of Computational and Applied Mathematics* 20.C, pp. 53–65.

Srebro, Nathan, Jason D.M. Rennie e Tommi S. Jaakkola (2005). «Maximum-margin matrix factorization». In: *Advances in Neural Information Processing Systems* 17, pp. 1329–1336.

Townes, F. William et al. (2019). «Feature Selection and Dimension Reduction for Single Cell RNA-Seq based on a Multinomial Model». In: *bioRxiv*.

Vallejos, Catalina A. et al. (2017). «Normalizing single-cell RNA sequencing data: Challenges and opportunities». In: *Nature Methods* 14.6, pp. 565–571.

Van Der Maaten, Laurens e Geoffrey Hinton (2008). «Visualizing Data using t-SNE». In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

Zappia, Luke, Belinda Phipson e Alicia Oshlack (2017). «Splatter: Simulation of single-cell RNA sequencing data». In: *Genome Biology* 18.1, pp. 1–15.

Ringraziamenti

La fine di questa tesi coincide con la fine di cinque anni di università.
Il tempo sembra essere volato, tra momenti difficili, gioie e soddisfazioni.
Ho imparato molto, ma c'è ancora molto da scoprire. Non si finisce mai di imparare e chi pensa il contrario non credo sia così saggio.
Ora, aspetto solo di imbattermi in una nuova avventura.
Spero che tutto ciò continui ad appassionarmi ed interessarmi, senza perdere mai la sana curiosità che dovrebbero avere tutti gli statistici.
Tutto questo non sarebbe stato possibile senza l'aiuto di alcune persone.
Vorrei ringraziare il mio Relatore, il prof. Davide Risso, per la pazienza e la disponibilità dimostrata nei mie confronti in questi mesi.
Un ringraziamento speciale va alla mia famiglia, in particolare ai miei genitori, Angelica e Michele, per avermi sempre sostenuto in ogni mia scelta, senza farmi pesare nulla. Senza di loro non avrei potuto portare avanti gli studi con serenità e per questo mi ritengo molto fortunata.
Sono grata a tutte le mie amiche e a tutti i miei amici per aver condiviso con me gioie, sacrifici e successi. L'affetto e il sostegno che mi hanno dimostrato rendono questo traguardo ancora più prezioso.
Non posso non ringraziare i miei compagni di corso che hanno allietato le intense giornate universitarie tra una risata e l'altra.
Infine, ringrazio con tutto il cuore il mio fidanzato, Alberto, che non ha mai

smesso di supportarmi, sopportarmi, spronarmi, incoraggiarmi e calmarmi. A tutti voi un immenso GRAZIE!