Bayesian Variable Selection nel caso di modelli lineari ad effetti fissi e misti

Aspetti teorici e applicazioni pratiche di alcune tecniche di selezione delle variabili in ambito Bayesiano

Lo scopo di questo progetto è quello di presentare alcuni tecniche di variable selection in ambito Bayesiano sia da un punto di vista teorico che pratico, mediante un'applicazione reale su dati provenienti dal mondo calcistico.

AUTORE/AUTRICE
Luca Riotto, Marius Viorel Parvu

DATA DI PUBBLICAZIONE
2 luglio 2024

DATA DI MODIFICA
2 luglio 2024

Introduzione

La variable selection (anche detta feature selection) corrisponde al processo di identificazione di un sottoinsieme rilevante di predittori da includere nel modello in esame.

Al fine di perseguire tale obiettivo possiamo identificare due macro-classi di metodi; i metodi basati sulla verifica di ipotesi e quelli che effettuano una stima dei parametri penalizzata. In ambito Bayesiano, nel primo caso si fa ricorso al Bayes factor e alle posterior model probabilties, mentre nel secondo caso vengono specificate delle shrinkage priors che inducono sparsità.

Partiamo dalle basi accennando quanto visto fino ad ora:

- Metodo screening: stimare il modello completo –> calcolare le statistiche T per i parametri –> rimuovere i parametri per i quali le statistiche sono troppo piccole –>ristimare il modello con le variabili restanti;
- Stepwise variable regression: le variabili vengono incluse una per volta partendo dal modello nullo (Forward Selection), oppure vengono eliminate una alla volta partendo dal modello completo (Backward Elimination), oppure una combinazione delle due precedenti.

Per una rassegna delle tecniche di base, si veda Miller (2002).

Quello che hanno in comune questi metodi è il fatto che scelgono un modello tra i diversi possibili e poi procedono come se quello fosse l'unico. Questo, però, può portare a dei risultati fuorvianti. Questo perchè, scegliendo un solo modello tra tanti, aumenta la probabilità di trovare variabili significative per puro caso. Per comprendere meglio la problematica si consideri l'esempio proposto da Freedman (1983): si supponga di disporre di una matrice formata da 100 righe (le unità statistiche) e 51 colonne (le v.c. che, per costruzione, sono indipendenti tra loro). Scegliamo, in modo arbitrario, di attribuire alla 51-ma variabile il ruolo di variabile dipendente Y, mentre alle altre il ruolo di variabili dipendenti $X_1, X_2, \dots X_{50}$. Per costruzione, ci aspetteremmo di avere un R^2 molto basso e dei coefficienti non significativi. Cosa succede nella realtà: se regrediamo Y su $X_1, X_2, \dots X_{50}$ otteniamo un $R^2 = 0.60$ e 21 coefficienti significativi a livello 0.25. La regressione è stata rifatta su queste 21 variabili, ottenendo un $R^2 = 0.50$ e 20 coefficienti significativi a livello 0.25 (14 dei quali significativi al livello 0.05). L'esempio porta alla luce dei risultati fuorvianti: sembrerebbe che ci sia una relazione tra la risposta e le esplicative. Un risultato simile si sarebbe ottenuto applicando la stepwise regressione: in questo caso si otterrebbe un $R^2 = 0.18$ e 4 coefficienti significativi al livello 0.05. Una soluzione a questo problema arriva dal mondo Bayesiano: grazie a questo approccio, infatti, si potrebbe tenere in considerazione, in modo esplicito, l'incertezza del modello.

Stima Bayesiana

Come sappiamo, il mondo Bayesiano esprime tutta l'incertezza (inclusa quella relativa ai

parametri ignoti del modello) in termini di probabilità e i parametri ignoti vengono considerati, a loro volta, come delle variabili casuali.

Si consideri di avere a disposizione un insieme di dati D, caratterizzato da un insieme di d parametri $\theta = (\theta_1, \dots, \theta_d)$. Sia:

- $p(\theta)$ la prior, ovvero la quantità che rappresenta la nostra conoscenza di θ prima di aver osservato i dati;
- $p(D \mid \theta)$ la verosimiglianza dei dati, ovvero la probabilità di osservare D dato che θ è il vero valore dei parametri;
- $p(\theta \mid D) = \frac{p(D|\theta)p(\theta)}{p(D)} \propto p(D \mid \theta)p(\theta)$ la posterior, ovvero l'aggiornamento delle nostre conoscenze su θ dopo aver osservato i dati. $p(\theta \mid D)$ contiene tutta l'informazione necessaria per fare inferenza su θ, basta solo decidere come riassumerla e comunicarla (es, utilizzando la moda).

Bayes Factors

Supponiamo di voler utilizzare i dati D per comparare due ipotesi, rappresentate dai modelli M_1, con parametri θ_1, e M_2, con parametri θ_2. Per il teorema di Bayes, la probabilità a posteriori che M_1 sia il modello corretto è

$$p(M_1 \mid D) = \frac{p(D \mid M_1)p(M_1)}{p(D \mid M_1)p(M_1) + p(D \mid M_2)p(M_2)}$$

dove $p(D \mid M_k)$, chiamata verosimiglianza integrata per il modello M_k, è la probabilità dei dati dato il modello M_k, mentre $p(M_k)$ è la probabilità a priori del modello $M_k (k = 1, 2)$. $p(D \mid M_1)$ si ottiene integrando θ_1, ovvero $p(D \mid M_1) = \int p(D \mid \theta_1, M_1)p(\theta_1 \mid M_1)d\theta_1$, dove $p(D \mid \theta_1, M_1)$ è la verosimiglianza per θ_1 sotto il modello M_1. Per misurare quanto i dati supportino M_2 rispetto a M_1 si calcola il rapporto delle probabilità a posteriori:

$$\frac{p(M_2 \mid D)}{p(M_1 \mid D)} = \left[\frac{p(D \mid M_2)}{p(D \mid M_1)}\right] \left[\frac{p(M_2)}{p(M_1)}\right]$$

Il primo fattore sul lato destro corrisponde al rapporto delle verosimiglianze integrate dei due modelli (Bayes factor per M_2 contro M_1, B_{21}); il secondo è il rapporto delle prior e molto spesso corrisponde a 1 per rappresentare l'assenza di preferenza (a priori) per uno dei due modelli. Seguendo le regole empirirche proposte da Jeffreys (1998):

Condizione	Esito
$1 \leq B_{21} < 3$	evidenza molto debole per M_2
$3 \leq B_{21} < 10$	evidenza positiva per M_2
$10 \leq B_{21} < 100$	evidenza forte per M_2
$B_{21} \geq 100$	evidenza molto forte per M_2

Utilizzare il Bayes factor richiede di calcolare integrali, che a volte possono essere intrattabili. Per questa ragione si sfruttano approssimazioni numeriche, come la BIC approximation. Per esempio, nel caso in cui M_1 sia nested dentro M_2 si ha che:

$$2log(B_{21}) \approx \chi^2_{21} - df_{21}log(n)$$

dove χ^2_{21} è la statistica LRT per M_1 contro M_2 e $df_{21} = d_2 - d_1$ sono i gradi di libertà associati al test.

Quando vengono considerati diversi modelli è utile compare ciascuno di loro con un modello baseline, come ad esempio il modello saturo. Nel caso del modello baseline saturo (M_S), il valore del BIC per il modello M_k, chiamato BIC_k è l'approssimazione di $2log(B_{Sk})$. Proseguendo, BIC_S, ovvero il BIC per il modello saturo, sarà zero, quindi il modello saturo sarà da preferire se $BIC_k > 0$. Quando compariamo due modelli, M_j e M_k si nota che:

$$2logB_{jk} = 2logB_{Sk} - 2logB_{Sj} \approx BIC_k - BIC_j.$$

Si preferisce il modello con valore di BIC più piccolo.

Incertezza del modello

Passiamo ora dal cosiderare due soli modelli al considerarne molti: $M_1, M_2, \ldots; M_K$. Sia Δ un parametro di interesse. L'inferenza Bayesiana per Δ si basa sulla sua posterior, ovvero:

$$p(\Delta \mid D) = \sum_{k=1}^{K} p(\Delta \mid D, M_k) p(M_k \mid D)$$

Quindi la distribuzione a posteriori per Δ è una media pesata delle sue posterior distributions sotto ciascun modello, con pesi pari alle posterior model probabilities ($p(M_k \mid D)$). In questo modo consideriamo anche l'incertezza del modello. Per individuare un sottoinsieme di k modelli da considerare, si possono applicare delle regole euristiche (es, Occam's window). Per ottenere le posterior model probabilities si procede nel seguente modo:

$$p(M_k \mid D) = \frac{p(D \mid M_k) p(M_k)}{\sum_{l=1}^{K} p(D \mid M_l) p(M_l)}$$

Spesso tutti i modelli hanno a priori pari probabilità, quindi, $p(M_1) = \ldots = p(M_k) = 1/K$. Nel caso in cui volessimo vedere se un coefficiente è incluso nel modello dovremmo calcolare:

$$P(\beta_1 \neq 0 \mid D) = \sum_{A_1} p(M_k \mid D)$$

dove A_1 è l'insieme dei modelli che includono β_1.

Per concludere, tornando all'esempio di Freedman (1983): applicando la metodologia appena descritta, si identificano 5 modelli (tra cui anche il modello nullo). Poi, calcolando $P(\beta_j \neq 0 \mid D)$ si scopre che 44 variabili vengono poste pari a zero, mentre le restanti 5 presentano una evidenza molto debole.

Per ulteriori approfondimenti su questa metodologia, si veda Raftery (1995).

Abbiamo voluto portare l'esempio di questo articolo per far comprendere le potenzialità dell'adottare l'approccio Bayesiano alla selezione delle variabili. Passiamo, ora, a vedere una metodologia comunemente usata per la Bayesian variable selection: le Spiake & Slab prior.

Spike & Slab Priors: caso del modello di regressione lineare

Si supponga di avere a disposizione una risposta continua, y_i, modellata mediante una combinazione lineare delle p covariate, $x_i = (x_1, \ldots, x_p) \in \mathbb{R}^p$, nel seguente modo:

$$y_i = \alpha + x_i^T \beta + \epsilon_i, \quad i = 1, \ldots, n \epsilon_i \sim N(0, \sigma^2), \quad \beta = (\beta_1, \ldots, \beta_p)^T$$

Come sappiamo, il problema della variable selection nasce quando si presume che non tutte le p covariate siano importanti per spiegare la risposta. Chiaramente, porre pari a zero alcuni parametri equivale a eslcludere le corrispondenti variabili dal modello. Nel paradigma Bayesiano, questo può essere raggiunto utilizzando misture di priori (come, per esempio, le spike & slab) per i coefficienti β. Tale formulazione introduce un vettore latente $\gamma = (\gamma_1, \ldots, \gamma_p)$ di indicatrici, dove

$$\gamma_j = \begin{cases} 1 & \text{se la variabile j viene inclusa nel modello} \\ 0 & \text{altrimenti} \end{cases}$$

La letteratura ha sviluppato, in parallelo, due costruzioni delle spike-and-slab: una costruzione discreta e una continua. La costruzione discreta utilizza una mistura tra una funzione di Dirac e una distribuzione (ad es una normale):

$$\beta_j \mid \sigma^2, \gamma_j \sim (1-\gamma_j)\delta_0(\beta_j) + \gamma_j N(0, h_j\sigma^2) \quad j = 1, \ldots, p$$

dove $\delta_0()$ rappresenta la funzione di Dirac in $\beta_j = 0$ e h_j un insieme di iper-parametri. Qui, $\gamma_j = 0$ esclude la j-esima variabile dal modello, mentre $\gamma_j = 1$ include il predittore nel modello, portando a una prior normale per β_j. Concludiamo la formulazione assumendo che:

$$\alpha \mid \sigma^2 \sim N(\alpha_0, h_0\sigma^2), \quad \sigma^2 \sim IG(\nu/2, \lambda/2),$$

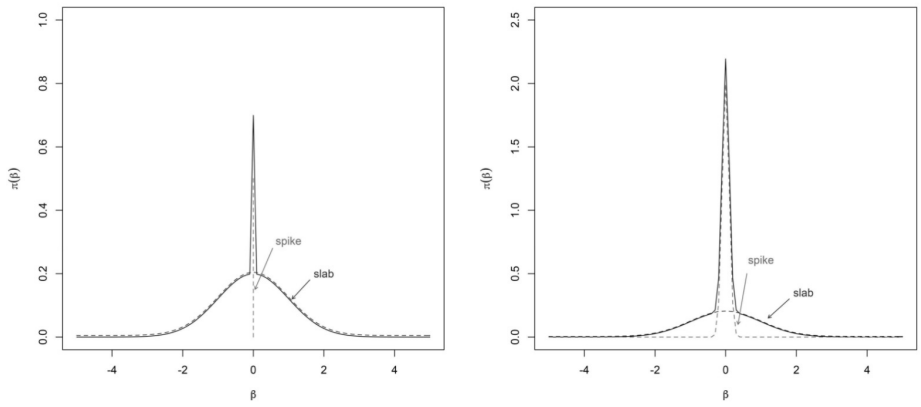
dove $\alpha_0, h_0, \nu, \lambda$ sono iperparametri da scegliere. Per esempio, ponendo $\alpha_0 = 0$ e prendendo un h_0 molto grande, si ottiene una a priori non informativa per l'intercetta. Proseguendo, è uso comune porre $h_j = c \quad \forall j$, assumendo così che i β_j siano a priori indipendenti dato γ.

La costruzione presentata richiede la scelta di una prior anche per γ. La scelta più semplice è considerare il prodotto di bernoulli indipendenti con parametro comune w:

$$p(\gamma \mid w) = \prod_{j=1}^{p} w^{\gamma_j}(1-w)^{1-\gamma_j} w \sim Beta(a, b)$$

Per una a priori poco informativa si imposta a=b=1, quindi il valore atteso a priori risulta essere $m = a/(a+b) = 0.5$ N.B.: pw corrisponde al numero di variabili attese che, a priori, si pensa vengano incluse nel modello.

La costruzione discreta presentata poc'anzi differisce poco da quella continua, la quale utilizza una mistura di due componenti continue (solitamente due gaussiane), una centrata in zero e con poca varianza, l'altra invece molto diffusa. Di seguito viene riportato un esempio di Spike & Slab continua e discreta:



Stochastic Search MCMC

Consideriamo il modello lineare, di cui abbiamo parlato precedentemente, assieme alla prior Spike & Slab. La scelta di a priori coniugate rende possibile integrare via i parametri del modello e ottenere la relative posterior distribution per γ:

$$p(\gamma \mid y, X) \propto f(y \mid \gamma, X)p(\gamma)$$

Questa distribuzione permette di individuare i migliori modelli, ovvero quelli con le più alte probabilità a posteriori. Quando il numero di predittori non permette di esplorare completamente lo spazio dei modelli, i metodi MCMC possono essere usati come stochastic searches per esplorare le distribuzioni a posteriori e identificare i modelli con le più alte probabilità a posteriori. Il metodo consiste nel visitare una serie di modelli dove, ad ogni step, il nuovo modello visitato differisce dal precedente per l'inclusione e/o l'esclusione di una o due variabili. Più nello specifico, dato un valore casuale di input, γ_0, a una generica iterazione il nuovo modello viene generato dal precedente scegliendo casualmente tra una delle seguenti:

1. Adding or deleting: scegliere casualmente un elemento di γ^{old} e cambiare il suo valore. Questo porta a includere una nuova variabile nel modello, oppure a eliminarla;

2. Swapping: estrarre in modo indipendente e casuale uno 0 e un 1 da γ^{old} e scambiare i loro valori. Questo porta all'inclusione di una nuova variabile nel modello e all'esclusione di una già presente. Indicando, ora, come γ^{new} il modello candidato, la probabilità di accettazione si calcola come:

$$min\left[\frac{p(\gamma^{\text{new}} \mid y, X)}{p(\gamma^{\text{old}} \mid y, X)}, 1\right]$$

Quindi, se il nuovo modello candidato ha una probabilità più alta del modello corrente, allora la catena si muove verso la nuova configurazione. Se così non è, allora il movimento è ancora possibile, ma solo con una certa probabilità. La stochastic search consiste, quindi, in una lista di modelli ispezionati, $\gamma^{(0)}, \ldots, \gamma^{(T)}$, e nelle loro relative posterior probabilities. Possiamo, poi, fare selezione delle variabili o cercando i vettori γ con le più grandi joint posterior probabilities tra i modelli visitati, oppure calcolando le frequenze di inclusione per ogni γ_j e poi scegliendo quei γ_j con le frequenze che eccedono un certo cut-off (solitamente pari a 0.5).

Per ulteriori approfondimenti, si veda Tadesse e Vannucci (2021).

Spike & Slab Priors for Linear Models: full conditional per γ

Modello di partenza:

$$Y|\beta_\gamma, \sigma^2_\epsilon, \gamma \quad \sim \; N(X_\gamma \beta_\gamma, \; \sigma^2_\epsilon I_n)$$

$$\text{dove}$$

$$\beta_\gamma|\sigma^2_\epsilon, \gamma \quad \sim \; N_{p_\gamma}(0, \; \sigma^2_\epsilon D_{0\gamma})$$

$$\gamma = [\gamma_1, \; \ldots, \; \gamma_p]^T \;\; \text{con} \;\; \gamma_i = \begin{cases} 1 & \text{se } i\text{-esima variabile è nel modello} \\ 0 & \text{altrimenti} \end{cases}$$

$$\gamma_i \quad \sim \; Be(\theta) \;\; \text{e} \;\; p_\gamma = \sum_{i=1}^{p} \gamma_i \quad \sim \; Binom(p, \theta)$$

La distribuzione congiunta a posteriori è:

$$\begin{aligned} p(\beta_\gamma, \sigma^2_\epsilon, \gamma, y|X) \quad &\propto \; (\sigma^2_\epsilon)^{-n/2} exp\left\{-\frac{1}{2\sigma^2_\epsilon}(Y - X_\gamma \beta_\gamma)^T (Y - X_\gamma \beta_\gamma)\right\} \\ &\times \; |\sigma^2_\epsilon \, D_{0\gamma}|^{-1/2} \, exp\left\{-\frac{1}{2\sigma^2_\epsilon} \beta^T_\gamma D^{-1}_{0\gamma} \beta_\gamma\right\} \\ &\times \; (\sigma^2_\epsilon)^{-\nu_\epsilon - 1} \, exp\left\{-\frac{\lambda_\epsilon}{\sigma^2_\epsilon}\right\} \times \theta^{p_\gamma} (1-\theta)^{(p - p_\gamma)} \end{aligned}$$

Otteniamo la marginale per γ integrando β_γ e σ^2_ϵ:

$$
\begin{aligned}
&\int\int p(\beta_\gamma, \sigma^2_\epsilon, \gamma, y|X)\, d\beta_\gamma\, d\sigma^2_\epsilon \\
&= \int\int (\sigma^2_\epsilon)^{-\frac{n}{2}-\frac{p_\gamma}{2}} |D_{0\gamma}|^{-\frac{1}{2}} exp\left\{-\frac{y^Ty}{2\sigma^2_\epsilon} + \frac{y^TX_\gamma\beta_\gamma}{\sigma^2_\epsilon} - \frac{\beta^T_\gamma X^T_\gamma X_\gamma \beta_\gamma}{2\sigma^2_\epsilon} - \frac{\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma}{2\sigma^2_\epsilon}\right\} \\
&\quad \times \left(\frac{1}{\sigma^2_\epsilon}\right)^{\nu_\epsilon+1} exp\left\{-\frac{\lambda_\epsilon}{\sigma^2_\epsilon}\right\} d\beta_\gamma\, d\sigma^2_\epsilon \\
&= \int\int (\sigma^2_\epsilon)^{-\frac{n}{2}-\frac{p_\gamma}{2}-\nu_\epsilon-1} |D_{0\gamma}|^{-\frac{1}{2}} exp\left\{-\frac{y^Ty}{2\sigma^2_\epsilon}\right\} exp\left\{\frac{y^TX_\gamma\beta_\gamma}{\sigma^2_\epsilon} - \frac{\beta^T_\gamma(X^T_\gamma X_\gamma + D^{-1}_{0\gamma})\beta_\gamma}{2\sigma^2_\epsilon}\right\} \\
&\quad \times exp\left\{-\frac{\lambda_\epsilon}{\sigma^2_\epsilon}\right\} d\beta_\gamma\, d\sigma^2_\epsilon \\
&\propto \int (\sigma^2_\epsilon)^{-\frac{n}{2}-\frac{p_\gamma}{2}-\nu_\epsilon-1} |D_{0\gamma}|^{-\frac{1}{2}} exp\left\{-\frac{y^Ty}{2\sigma^2_\epsilon}\right\} |\sigma^2_\epsilon(X^T_\gamma X_\gamma + D_{0\gamma})^{-1}|^{\frac{1}{2}} \\
&\quad \times exp\left\{-\frac{1}{2\sigma^2_\epsilon}(X^T_\gamma X_\gamma + D^{-1}_{0_\gamma})y^TX_\gamma(X^T_\gamma X_\gamma + D^{-1}_{0_\gamma})^{-2}X^T_\gamma y\right\} d\sigma^2_\epsilon \\
&\propto \int (\sigma^2_\epsilon)^{-\frac{n}{2}-\frac{p_\gamma}{2}-\nu_\epsilon-1} (\sigma^2_\epsilon)^{\frac{p_\gamma}{2}} |(X^T_\gamma X_\gamma + D_{0\gamma})^{-1}|^{\frac{1}{2}} |D_{0\gamma}|^{-\frac{1}{2}} \\
&\quad \times exp\left\{-\frac{1}{\sigma^2_\epsilon}\left[\frac{y^Ty}{2} + \lambda_\epsilon - \frac{y^TX_\gamma(X^T_\gamma X_\gamma + D^{-1}_{0\gamma})^{-1}X^T_\gamma y}{2}\right]\right\} d\sigma^2_\epsilon \\
&\propto |(X^T_\gamma X_\gamma + D^{-1}_{0\gamma})^{-1}|^{\frac{1}{2}} |D_{0\gamma}|^{-\frac{1}{2}} \frac{\Gamma(n/2+\nu_\epsilon)}{\left(\lambda_\epsilon + \frac{y^Ty}{2} - \frac{y^TX_\gamma(X^T_\gamma X_\gamma + D^{-1}_{0\gamma})^{-1}X_\gamma y}{2}\right)^{n/2+\nu_\epsilon}}
\end{aligned}
$$

Quindi:

$$
p(\gamma|y,X) \propto |(X^T_\gamma X_\gamma + D^{-1}_{0\gamma})^{-1}|^{\frac{1}{2}} |D_{0\gamma}|^{-\frac{1}{2}} \left(\lambda_\epsilon + \frac{y^Ty}{2} - \frac{y^TX_\gamma(X^T_\gamma X_\gamma + D^{-1}_{0\gamma})^{-1}X_\gamma y}{2}\right)^{n/2+\nu_\epsilon} \times \theta^{p_\gamma}(1-\theta)^{(p-p_\gamma)}
$$

Spike & Slab Priors for Linear Models: codifica

Per poter ispezionare la posterior di γ si è fatto uso di un algoritmo di tipo Reversible-Jump (segue lo stesso schema di un Metropolis). Di seguito viene riportato il codice:

```
# compute the marginal likelihood of gamma given y and X
m_like <- function(gamma, y, modelMAT, theta, lambda, ni, v, pos_list, r
  # v parameter of the prior variance (in D0)
  # n and p are sample size and number of parameter
  # assume prior beta0 = 0 e D0 = diag(p)
  # the intercept is always included
  if(colnames(modelMAT)[1] == "(Intercept)" )
  {
    pos_gamma <- c(1, which(gamma == 1)+1)
  } else pos_gamma <- which(gamma == 1)
  pos_in_data <- do.call(c, lapply(pos_gamma, function(i) pos_list[[i]])
  X <- Matrix(modelMAT[, pos_in_data])
  D0 <- diag(ncol(X)) * v
  yty <- crossprod(y)
  XtX <- crossprod(X) + diag(rep(1/v, ncol(X)))
  XtX_inv <-chol2inv(chol(XtX))
  S <- yty -  crossprod(y, X) %*% XtX_inv %*% t(crossprod(y, X))

  gamma_post <- 0.5* determinant(XtX_inv, logarithm = T)$modulus -
    (n/2 + ni) * log(lambda + S/2) -
    (1/2)*log(v**sum(gamma)) + dbinom(sum(gamma), size = p, prob = theta

  return(gamma_post)
```

```
}


# RJ MCMC function
RJ_MCMC <- function(R, y, data, lambda, ni, theta = 0.5, gamma_init, v =
  # lambda, ni iperparameter for sigma
  # zeta, phi iperparameter for theta
  # theta prob for gamma distribution
  n <- nrow(data)
  p <- length(gamma_init)
  gamma_sim <- matrix(NA, R+1, length(gamma_init)) # store simulated val
  m_sim <- numeric(R+1) # store m(gamma| X, y)
  # init marginal likelihood
  m_0 <- m_like(gamma_init, y, data, theta, lambda, ni, v, pos_list, n,
  # store the inizialisation parameter
  gamma_sim[1, ] <- gamma_init
  m_sim[1] <- m_0
  gamma0 <- gamma_init
  for(i in 1:R){
    gamma1 <- gamma0
    pos <- sample(1:p, size = 1)
    ifelse(gamma1[pos]==0, gamma1[pos] <- 1, gamma1[pos] <- 0)
    m_1 <- m_like(gamma1, y, data, theta, lambda, ni, v, pos_list, n, p)
    alpha <- exp(m_1 - m_0)
    if(as.numeric(runif(1) < alpha))
    {
      gamma0 <- gamma1
      m_0 <- m_1
    }
    gamma_sim[i+1, ] <- gamma0
    m_sim[i+1] <- m_0
    if(verbose == T) {
      if(i %% 100 == 0) print(i)
    }
  }
  return(list(gamma = gamma_sim, lm = m_sim))
}
```

Spike & Slab Priors per Linear Mixed Models

Fino ad ora ci si è concentrati sulla selezione delle variabili per i modelli lineari. Nel caso di modelli ad effetti misti si fa ricorso a uno schema gibbs, dove, ad ogni iterazione, i parametri vengono generati dalle distribuzioni condizionate che verranno presentate nel seguito.

Modello di partenza:

$$Y|\beta_\gamma, u, \sigma^2_\epsilon, \sigma^2_u, \gamma \quad \sim \; N(X_\gamma \beta_\gamma + Zu,\; \sigma^2_\epsilon I_n)$$

dove

$$\beta_\gamma|\sigma^2_\epsilon, \gamma \quad \sim \; N_{p_\gamma}(0,\; \sigma^2_\epsilon D_{0\gamma})$$

$$\sigma^2_\epsilon \quad \sim \; IG(\nu_\epsilon,\; \lambda_\epsilon)$$

$$u \quad \sim \; N_k(0,\; \sigma^2_u I_k)$$

$$\sigma^2_u \quad \sim \; IG(\nu_u,\; \lambda_u)$$

$$\gamma = [\gamma_1, \ldots, \gamma_p]^T \;\; \text{con} \;\; \gamma_i = \begin{cases} 1 & \text{se } i\text{-esima variabile è nel modello} \\ 0 & \text{altrimenti} \end{cases}$$

$$\gamma_i \quad \sim \; Be(\theta) \;\; \text{e} \;\; p_\gamma = \sum_{i=1}^{p} \gamma_i \quad \sim \; Binom(p, \theta)$$

La distribuzione congiunta a posteriori risulta:

$$
\begin{aligned}
p(\beta_\gamma, u, \sigma^2_\epsilon, \sigma^2_u, \gamma, y|X) \quad &\propto \ (\sigma^2_\epsilon)^{-n/2} exp\left\{-\frac{1}{2\sigma^2_\epsilon}(Y - X_\gamma\beta_\gamma - Zu)^T(Y - X_\gamma\beta_\gamma - Zu)\right\} \\
&\times \ (\sigma^2_\epsilon)^{-p_\gamma/2} \left|D_{0\gamma}\right|^{-1/2} exp\left\{-\frac{1}{2\sigma^2_\epsilon}\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma\right\} \\
&\times \ (\sigma^2_\epsilon)^{-\nu_\epsilon - 1} exp\left\{-\frac{\lambda_\epsilon}{\sigma^2_\epsilon}\right\} \\
&\times \ (\sigma^2_u)^{k/2} exp\left\{-\frac{1}{2\sigma^2_u}u^T u\right\} \\
&\times \ (\sigma^2_u)^{-\nu_u - 1} exp\left\{-\frac{\lambda_u}{\sigma^2_u}\right\} \\
&\times \ \theta^{\sum_{i=1}^p \gamma_i}(1-\theta)^{p - \sum_{i=1}^p \gamma_i}
\end{aligned}
$$

Full-conditional di β

Sia $\epsilon_u = Y - Zu$:

$$
\begin{aligned}
p(\beta|-) \quad &\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(\epsilon_u - X_\gamma\beta_\gamma\right)^T\left(\epsilon_u - X_\gamma\beta_\gamma\right) - \frac{1}{2\sigma^2_\epsilon}\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma\right\} \\
&\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(\epsilon^T_u\epsilon_u - \epsilon^T_u X_\gamma\beta_\gamma - \beta^T_\gamma X^T_\gamma\epsilon_u + \beta^T_\gamma X^T_\gamma X_\gamma\beta_\gamma\right) - \frac{1}{2\sigma^2_\epsilon}\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma\right\} \\
&\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(-2\beta^T_\gamma X^T_\gamma\epsilon_u + \beta^T_\gamma X^T_\gamma X_\gamma\beta_\gamma\right) - \frac{1}{2\sigma^2_\epsilon}\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma\right\} \\
&\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(-2\beta^T_\gamma(X^T_\gamma\epsilon_u) + \beta^T_\gamma(X^T_\gamma X_\gamma D^{-1}_{0\gamma})\beta_\gamma\right)\right\}
\end{aligned}
$$

Quindi

$$
(\beta|-) \quad \sim \ N_{p_\gamma}\left(\left(X^T_\gamma X_\gamma + D^{-1}_{0\gamma}\right)^{-1}X^T_\gamma\epsilon_u \, , \, \sigma^2_\epsilon\left(X^T_\gamma X_\gamma + D^{-1}_{0\gamma}\right)^{-1}\right)
$$

Full-conditional di u
Sia $\epsilon_\beta = Y - X_\gamma\beta_\gamma$:

$$
\begin{aligned}
p(u|-) \quad &\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(\epsilon_\beta - Zu\right)^T\left(\epsilon_\beta - Zu\right) - \frac{1}{2\sigma^2_u}u^T u\right\} \\
&\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(\epsilon^T_\beta\epsilon_\beta - 2u^T Z^T\epsilon_\beta + u^T Z^T Zu\right) - \frac{1}{2\sigma^2_u}u^T u\right\} \\
&\propto \quad exp\left\{-\frac{1}{2\sigma^2_\epsilon}\left(-2u^T Z^T\epsilon_\beta + u^T\left(Z^T Z + \frac{\sigma^2_\epsilon}{\sigma^2_u}I_k\right)u\right)\right\}
\end{aligned}
$$

Quindi:

$$
(u|-) \quad \sim \ N_k\left(\left(Z^T Z + \frac{\sigma^2_\epsilon}{\sigma^2_u}I_k\right)^{-1}Z^T\epsilon_\beta, \sigma^2_\epsilon\left(Z^T Z + \frac{\sigma^2_\epsilon}{\sigma^2_u}I_k\right)^{-1}\right)
$$

Full-conditional di σ^2_ϵ
Sia $\epsilon_{TOT} = Y - X_\gamma\beta_\gamma - Zu$:

$$
p(\sigma^2_\epsilon|-) \quad \propto \quad \left(\frac{1}{\sigma^2_\epsilon}\right)^{\frac{n}{2} + \frac{p_\gamma}{2} + \nu_\epsilon + 1} exp\left\{-\frac{1}{\sigma^2_\epsilon}\left(\frac{\epsilon^T_{TOT}\epsilon_{TOT}}{2} + \frac{\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma}{2} + \lambda_\epsilon\right)\right\}
$$

Quindi:

$$
(\sigma^2_\epsilon|-) \quad \sim \ IG\left(\frac{n}{2} + \frac{p_\gamma}{2} + \nu_\epsilon, \, \frac{\epsilon^T_{TOT}\epsilon_{TOT}}{2} + \frac{\beta^T_\gamma D^{-1}_{0\gamma}\beta_\gamma}{2} + \lambda_\epsilon\right)
$$

Full-conditional di σ_u^2

$$\begin{aligned} p(\sigma_u^2|-) &\propto \left(\frac{1}{\sigma_u^2}\right)^{\frac{k}{2}+\nu_u+1} exp\left\{-\frac{1}{2\sigma_u^2}u^Tu - \frac{\lambda_u}{\sigma_u^2}\right\} \\ &\propto \left(\frac{1}{\sigma_u^2}\right)^{\frac{k}{2}+\nu_u+1} exp\left\{-\frac{1}{\sigma_u^2}\left(\frac{u^Tu}{2} + \lambda_u\right)\right\} \end{aligned}$$

Quindi:

$$(\sigma_u^2|-) \sim IG\left(\frac{k}{2} + \nu_u,\ \frac{u^Tu}{2} + \lambda_u\right)$$

Distribuzione condizionata di γ:
Sia

$$\epsilon_u = Y - Zu$$

Consideriamo come costante tutto ciò che non dipende da $\sigma_\epsilon^2, \sigma_u^2, u$ ed integriamo via i parametri β

$$\begin{aligned} const =& (\sigma_\epsilon^2)^{-\nu_\epsilon-1-n/2} exp\left\{-\frac{\lambda_\epsilon}{\sigma_\epsilon^2}\right\} \\ &\times (\sigma_u^2)^{k/2} exp\left\{-\frac{1}{2\sigma_u^2}u^Tu\right\} \\ &\times (\sigma_u^2)^{-\nu_u-1} exp\left\{-\frac{\lambda_u}{\sigma_u^2}\right\} \\ &\times \theta^{\sum_{i=1}^p \gamma_i}(1-\theta)^{p-\sum_{i=1}^p\gamma_i} \end{aligned}$$

$$\begin{aligned} p(\gamma|u,\sigma_\epsilon^2,\sigma_u^2) &\propto const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2}\int exp\left\{-\frac{(\epsilon_u - X_\gamma\beta_\gamma)^T(\epsilon_u - X_\gamma\beta_\gamma)}{2\sigma_\epsilon^2} - \frac{\beta_\gamma^T D_{0\gamma}^{-1}\beta_\gamma}{2\sigma_\epsilon^2}\right\} d\beta_\gamma \\ &= const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2}\int exp\left\{-\frac{\left(\epsilon_u^T\epsilon_u - 2\beta_\gamma^T X_\gamma^T\epsilon_u + \beta_\gamma^T X_\gamma^T X_\gamma\beta_\gamma\right)}{2\sigma_\epsilon^2} - \frac{\beta_\gamma^T D_{0\gamma}^{-1}\beta_\gamma}{2\sigma_\epsilon^2}\right\} d\beta_\gamma \\ &= const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2}\int exp\left\{-\frac{\epsilon_u^T\epsilon_u}{2\sigma_\epsilon^2} + \frac{\beta_\gamma^T X_\gamma^T\epsilon_u}{\sigma_\epsilon^2} - \frac{\beta_\gamma^T X_\gamma^T X_\gamma\beta_\gamma}{2\sigma_\epsilon^2} - \frac{\beta_\gamma^T D_{0\gamma}^{-1}\beta_\gamma}{2\sigma_\epsilon^2}\right\} d\beta_\gamma \\ &= const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2} exp\left\{-\frac{\epsilon_u^T\epsilon_u}{2\sigma_\epsilon^2}\right\}\int exp\left\{\frac{\beta_\gamma^T X_\gamma^T\epsilon_u}{\sigma_\epsilon^2} - \frac{\beta_\gamma^T X_\gamma^T X_\gamma\beta_\gamma}{2\sigma_\epsilon^2} - \frac{\beta_\gamma^T D_{0\gamma}^{-1}\beta_\gamma}{2\sigma_\epsilon^2}\right\} d\beta_\gamma \\ &= const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2} exp\left\{-\frac{\epsilon_u^T\epsilon_u}{2\sigma_\epsilon^2}\right\}\int exp\left\{\frac{\beta_\gamma^T X_\gamma^T\epsilon_u}{\sigma_\epsilon^2} - \frac{\beta_\gamma^T\left(X_\gamma^T X_\gamma + D_{0\gamma}^{-1}\right)\beta_\gamma}{2\sigma_\epsilon^2}\right\} d\beta_\gamma \end{aligned}$$

A questo punto riconosciamo una distribuzione normale, definita a meno di costanti di normalizzazione, per i parametri beta nell'integrale.

Moltiplichiamo e dividiamo per le costanti di normalizzazione ed integriamo via la distribuzione normale (che integrerà ora a 1).

$$p(\gamma|u,\sigma_\epsilon^2,\sigma_u^2) \propto const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2} exp\left\{-\frac{\epsilon_u^T\epsilon_u}{2\sigma_\epsilon^2}\right\}\int |(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{-\frac{1}{2}}|(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{\frac{1}{2}}(\sigma_\epsilon^2)^{\frac{1}{2}}(\sigma_\epsilon^2)^{-\frac{1}{2}}$$
$$exp\left\{-\frac{(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\mu^T\mu}{2\sigma_\epsilon^2}+\frac{\epsilon_u^TX_\gamma\beta_\gamma}{\sigma_\epsilon^2}\right.$$
$$\left.-\frac{\beta_\gamma^T(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\beta_\gamma}{2\sigma_\epsilon^2}+\frac{(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\mu^T\mu}{2\sigma_\epsilon^2}\right\}d\beta_\gamma$$

$$= const \times (\sigma_\epsilon^2)^{-p_\gamma/}|D_{0\gamma}|^{-1/2} exp\left\{-\frac{\epsilon_u^T\epsilon_u}{2\sigma_\epsilon^2}\right\}|(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{\frac{1}{2}}(\sigma_\epsilon^2)^{\frac{1}{2}}exp\left\{\frac{(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\mu^T\mu}{2\sigma_\epsilon^2}\right\}$$
$$\times \int |(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{-\frac{1}{2}}(\sigma_\epsilon^2)^{-\frac{1}{2}}exp\left\{-\frac{(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\mu^T\mu}{2\sigma_\epsilon^2}+\frac{\epsilon_u^TX_\gamma\beta_\gamma}{\sigma_\epsilon^2}\right.$$
$$\left.-\frac{\beta_\gamma^T(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\beta_\gamma}{2\sigma_\epsilon^2}\right\}d\beta_\gamma$$

$$= const \times (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2} exp\left\{-\frac{\epsilon_u^T\epsilon_u}{2\sigma_\epsilon^2}\right\}|(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{\frac{1}{2}}(\sigma_\epsilon^2)^{\frac{1}{2}}exp\left\{\frac{(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\mu^T\mu}{2\sigma_\epsilon^2}\right\}$$

$$\propto (\sigma_\epsilon^2)^{-p_\gamma/2}|D_{0\gamma}|^{-1/2}\theta^{\sum_{i=1}^p\gamma_i}(1-\theta)^{p-\sum_{i=1}^p\gamma_i}|(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{\frac{1}{2}}$$
$$\times exp\left\{\frac{1}{2\sigma_\epsilon^2}(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})\epsilon_u^TX_\gamma(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}X^T\epsilon_u\right\}$$

Quindi si ottiene:

$$p(\gamma|u,\sigma_\epsilon^2,\sigma_u^2) \propto |D_{0\gamma}|^{-1/2}\theta^{\sum_{i=1}^p\gamma_i}(1-\theta)^{p-\sum_{i=1}^p\gamma_i}|(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}|^{\frac{1}{2}}$$
$$\times exp\left\{\frac{1}{2\sigma_\epsilon^2}\epsilon_u^TX_\gamma(X_\gamma^TX_\gamma+D_{0\gamma}^{-1})^{-1}X_\gamma^T\epsilon_u\right\}$$

Spike & Slab Priors per Linear Mixed Models: codifica

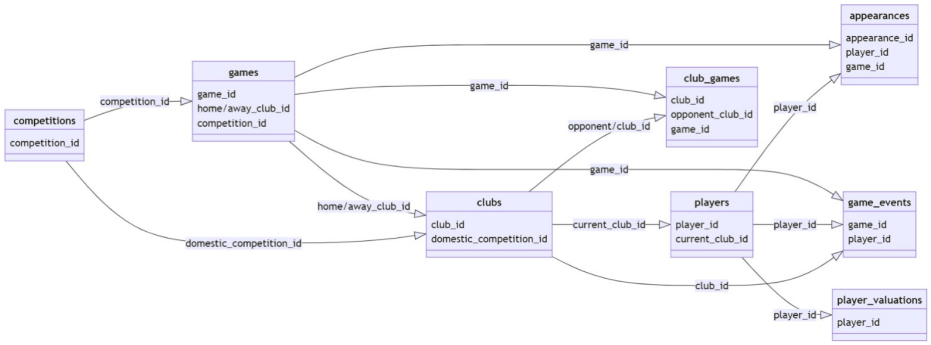
```
# funzione che calcola la marginale di gamma
pi_gamma <- function(gamma, sigma_e, theta, Z, data, response, v, u)
{
  # v parameter of the prior variance (in D0)
  # n and p are sample size and number of parameter
  # assume prior beta0 = 0
  # the intercept is always included
  y <- as.matrix((data[, response]))
  data <- data %>% dplyr::select(-response)
  p <- length(gamma)
  p_u <- ncol(Z)
  n <- nrow(data)
  # calcolo le quantità per inizializzare la componente di metropolis
  pos_gamma <- which(gamma == 1)
  X <- sparse.model.matrix(~., data = data[, pos_gamma])

  # costruisco le quantità da passare a pi_gamma e per generare i parame
  D <- Diagonal(ncol(X))*v
  D_inv <- Diagonal(ncol(X))*(1/v)
  XtX <- crossprod(X) + D_inv
  XtX_inv <- chol2inv(chol(XtX))
  epsilon_u <- y - Z %*% u

  S <- quad.tform(XtX_inv, as.vector(crossprod(epsilon_u, X)))

  m_like <- - (0.5*sum(gamma))*log(sigma_e) -
    0.5*determinant(D, logarithm = T)$modulus +
    dbinom(sum(gamma), p, theta, log = T)  +
    0.5*determinant(XtX_inv, logarithm = T)$modulus + 1/(2*sigma_e) * S
```

```
    1/(2*sigma_e) * crossprod(epsilon_u)

  return(list(mlike = m_like, XtX = XtX, D = D, D_inv = D_inv,
              XtX_inv = XtX_inv, epsilon_u = epsilon_u, X = X, gamma = g
}


# gibbs sampler
gibbs_spikeslab <- function(R, beta_init, sigma_init, u_init, gamma_init
                            ni_e, ni_u, lambda_e, lambda_u, v, verbose =
{
  y <- as.matrix((data[, response]))
  p_b <- length(gamma_init)
  p_u <- ncol(Z)
  p <- length(gamma_init)
  n <- nrow(data)
  par_init <- c(u_init, sigma_init)
  # matrici che conterrà le simulazioni
  parout <- matrix(NA, R+1, length(par_init))
  gammaout <- matrix(NA, R+1, length(gamma_init))
  beta_list <- list()
  colnames(gammaout) <- data %>% dplyr::select(-response) %>% colnames
  # inserisco i parametri inizializzati
  gammaout[1, ] <- gamma_init
  parout[1, ] <- par_init
  beta_list[[1]] <- beta_init
  # fisso i valori di inizializzazione dell'algoritmo
  m_0 <- pi_gamma(gamma_init, sigma_e = par_init[length(par_init)-1],
                  theta = theta, Z, data, response, v, par_init[1:p_u])
  gamma0 <- gamma_init

  # matrice servirà in seguito
  Z <- Matrix(Z)
  ZtZ <- crossprod(Z)

  for(i in 1:R)
  {
    gamma1 <- gamma0
    # proposal per gamma in cui cambia solo un indicatore alla volta
    pos <- sample(1:p, size = 1)
    ifelse(gamma1[pos]==0, gamma1[pos] <- 1, gamma1[pos] <- 0)

    m_1 <- pi_gamma(gamma1, sigma_e = parout[i, length(par_init)-1],
                    theta = theta, Z, data, response, v, parout[i, 1:p_u
    m_0 <-  pi_gamma(gamma0, sigma_e = parout[i, length(par_init)-1],
                     theta = theta, Z, data, response, v, parout[i, 1:p_
    alpha <- exp(m_1$mlike - m_0$mlike) # prob di accettazione della pro
    if(as.numeric(runif(1) < alpha))
    {
      gamma0 <- gamma1
      m_0 <- m_1
    }
    gammaout[i+1, ] <- gamma0

    # genero i beta
    sigma_e <- parout[i, length(par_init)-1]
    epsilon_u <- y - Z %*% parout[i, 1:p_u]
    beta_list[[i+1]] <- rmvn.sparse(1, mu = m_0$XtX_inv %*% crossprod(m_
                                                CH = Cholesky(m_0$XtX_inv
    colnames(beta_list[[i+1]]) <- colnames(m_0$X)

    # genero gli u
    sigma_u <- parout[i, length(par_init)]
    B <- Diagonal(ncol(ZtZ)) * (sigma_e / sigma_u)
    epsilon_b <- y - m_0$X %*% as.vector(beta_list[[i+1]])
    parout[i+1, 1:p_u] <- rmvn.sparse(1, mu = chol2inv(chol(ZtZ + B)) %*
                                      CH = Cholesky(1/sigma_e * (ZtZ + B)))@

    # genero sigma epsilon
```

```
            epsilon_tot <- epsilon_b - Z %*% parout[i+1, 1:p_u]
            beta_cor <- as.vector(beta_list[[i+1]])
            p_gamma <- sum(gammaout[i+1, ] )
            parout[i+1, length(par_init)-1] <- 1/rgamma(1, (n/2 + ni_e + p_gamma
                                                    drop(crossprod(epsilon_t
                                                       quad.form(m_0$D_inv,
                                                       lambda_e))
            # genero sigma_u
            u_cor <- parout[i+1, 1:p_u]
            parout[i+1, length(par_init)] <- 1/rgamma(1, ni_u + p_u/2, (crossprc

            if(verbose==T & (i %% 100) == 0) print(i)
          }
          return(list(beta = beta_list, parameters = parout, gamma = gammaout))
        }
```

Dataset: Football Data from Transfermarkt

I dati a nostra disposizione sono stati reperiti dal sito Kaggle e forniscono informazioni su alcuni giocatori di calcio professionisti, sulle partite da loro sostenute e sui club a cui appartengono (dal 2012 al 2024). Il dataset è stato composto a partire da una serie di CSV strutturati nel seguente modo:



Per esempio, *players* contiene informazioni sui giocatori, quali il loro nome, la loro cittadinanza, etc.; *player_valutations* contiene il valore di mercato (in euro) di ciascun giocatore; *appearances* contiene per ogni record le prestazioni di una partita in cui un giocatore ha preso parte.

L'insieme dei dati sulle valutazioni di mercato contiene naturalmente meno record rispetto all'insieme di dati sulle partite, questo perchè il valore di mercato viene registrato con una frequenza (circa) semestrale, mentre le partite settimanalmente. Abbiamo quindi deciso di aggregare le informazioni sulle prestazioni in campo dei giocatori in semestri, ottenendo così per ogni giocatore delle statistiche di performance semestrali ed un associato valore di mercato (variabile risposta).

Si ha inoltre che per ogni partita (dataframe appearances) abbiamo a disposizione il tipo di competizione in cui è stata giocata (ad es. campionato, coppa internazionale ...), dato che si ritiene che le performance di partita abbiano un'influenza sulla risposta differente a seconda del fatto che la partita sia un'amichevole o una partita di champion league, nell'aggregare i dati semestralmente si è creata l'interazione tra le statistiche di performance e la competizione. Ad esempio per ogni giocatore si sono crete le variabili "numero gi goal in campionato", "numero di goal nelle coppe internazionali" ...

Il dataset costruito presenta circa 110 mila righe e 43 variabili:

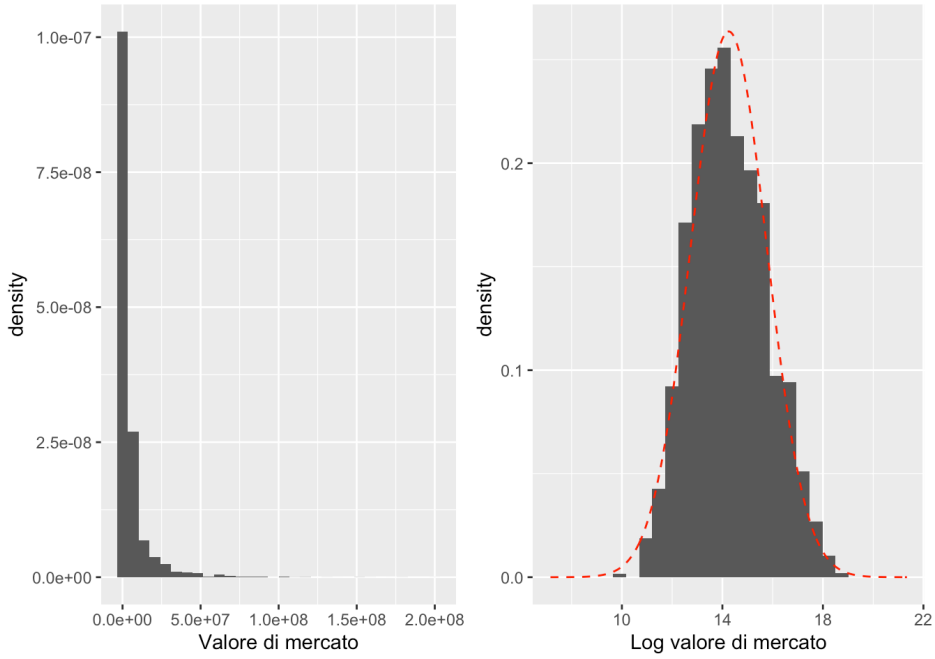
Analisi esplorative

Di seguito vengono riportate alcune analisi esplorative sui dati.

Distribuzione della variabile risposta:

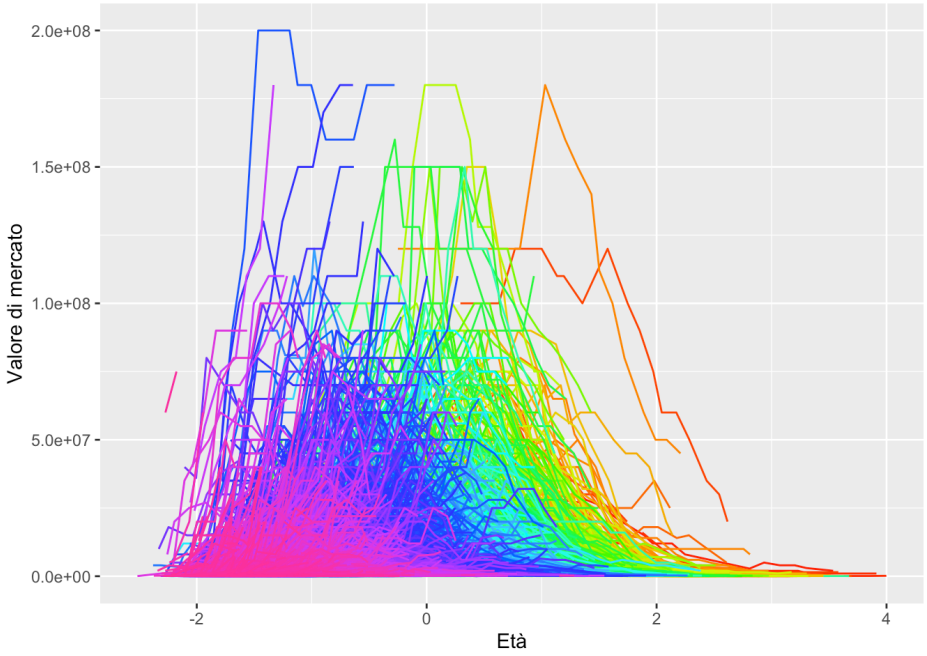
```
p1=newdata %>%
  ggplot(aes(x = mkval)) +
  geom_histogram(aes(y = ..density..)) +
  xlab("Valore di mercato")
newdata$log_mkval <- log(newdata$mkval)
p2=newdata %>%
  ggplot(aes(x = log_mkval)) +
  geom_histogram(aes(y = ..density..), bins = 20) +
  xlab("Log valore di mercato") +
  stat_overlay_normal_density(color = "red", linetype = "dashed")
gridExtra::grid.arrange(p1,p2, ncol=2)
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
ℹ Please use `after_stat(density)` instead.

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Si nota che la variabile risposta presenta un andamento fortemente asimmetrico: la maggior parte dei giocatori presentano valori di mercato bassi (il valore di mercato minimo è pari a 10000 euro). Tuttavia, l'utilizzo della trasformata logaritmica migliora di molto la situazione.

```
newdata %>% group_by(player_id) %>%
  ggplot(aes(x = Age, y = mkval,
             group = as.numeric(player_id), col = as.numeric(player_id))
  geom_line() +
  xlab("Età") + ylab("Valore di mercato") +
  scale_color_gradientn(colours = rainbow(10)) +
  theme(legend.position="none")
```



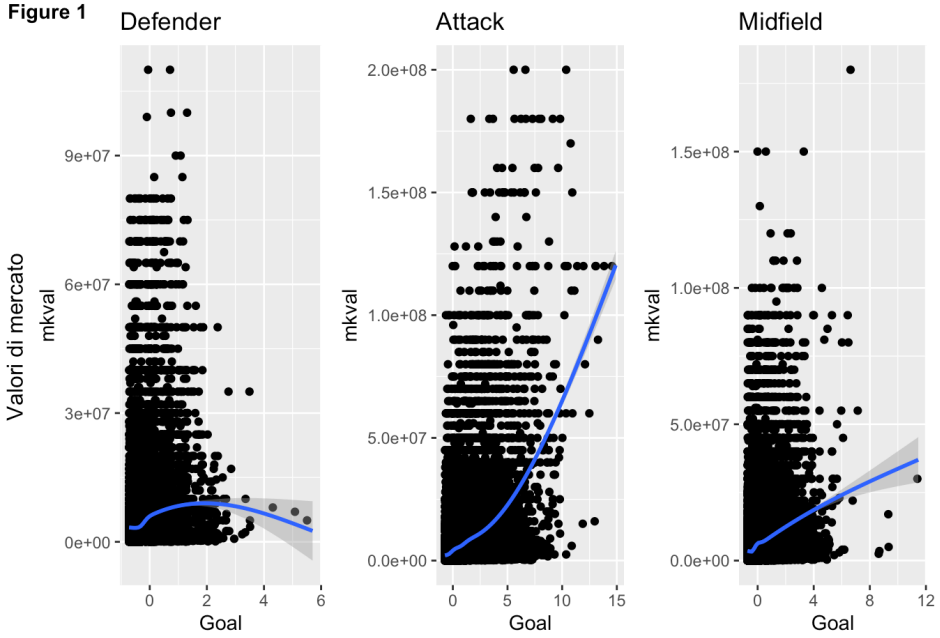
In questo grafico si riporta, per ciascun giocatore, l'andamento nel tempo del proprio valore di mercato: notiamo che la maggior parte dei calciatori ha un'età inferiore ai 35 anni e un valore di mercato non superiore ai 50 mln.

```
# valuto l'interazione con la posizione del giocatore
p1 <- newdata %>% filter(position == "Defender") %>%
  ggplot(aes(y = mkval, x = jitter(typedomestic_league_x_month_goal, fac
  geom_point() +
  geom_smooth() +
  xlab("Goal") +
  labs(title = "Defender")

p2 <- newdata %>% filter(position == "Attack") %>%
  ggplot(aes(y = mkval, x = jitter(typedomestic_league_x_month_goal, fac
  geom_point() +
  geom_smooth() +
  xlab("Goal") +
  labs(title = "Attack")

p3 <- newdata %>% filter(position == "Midfield") %>%
  ggplot(aes(y = mkval, x = jitter(typedomestic_league_x_month_goal, fac
  geom_point() +
  geom_smooth() +
  xlab("Goal") +
  labs(title = "Midfield")

figure <- ggarrange(p1, p2, p3,  nrow = 1, ncol = 3)
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
annotate_figure(figure,
                bottom = text_grob("Valore di mercato in funzione del nu
                                   color = "black",
                                   hjust = 1, x = 1, face = "italic", si
                left = text_grob("Valori di mercato", color = "black", r
                fig.lab = "Figure 1", fig.lab.face = "bold"
)
```

Figure 1

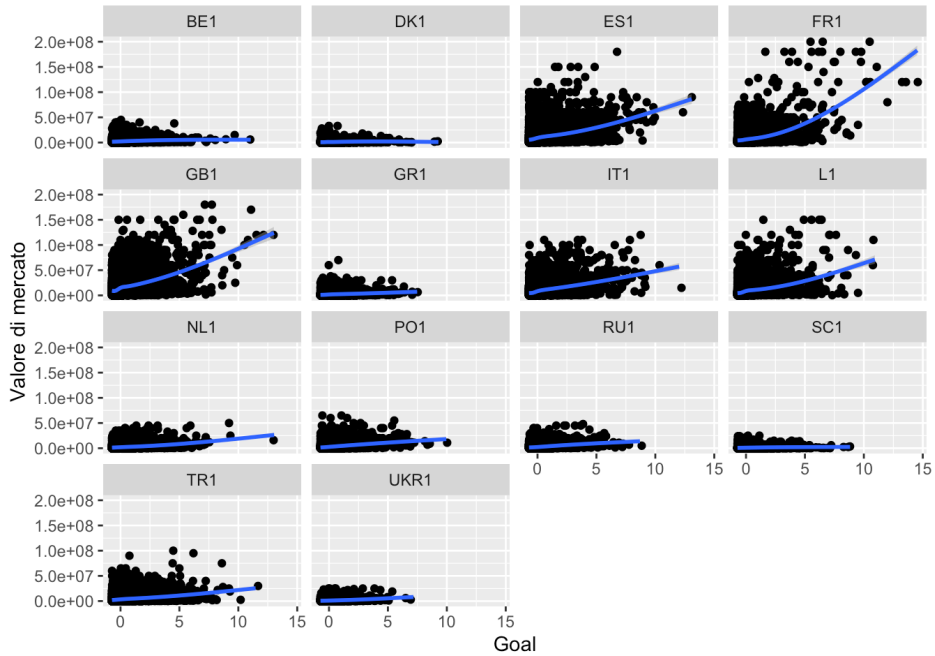


Valore di mercato in funzione del numero di goal in campionato a seconda del ruolo del giocatore

Qui vediamo riportato l'andamento del valore di mercato in funzione del numero di goal a seconda del ruolo del giocatore. Non sorprende che a beneficiare maggiormente di un numero elevato di goal siano gli attaccanti (da notare come, a partire dai 10 goal, ci sia un'impennata del valore di mercato).

```
# e l'interazione con il campionato
newdata %>%
  ggplot(aes(y = mkval, x = jitter(typedomestic_league_x_month_goal, fac
  geom_point() +
  geom_smooth() +
  facet_wrap(~player_club_domestic_competition_id) +
  xlab("Goal") + ylab("Valore di mercato")
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



In questo grafico ogni panel rappresenta una competizione diversa, identificata dalle sigle come BE1, DK1, etc. (indicatori delle diverse leghe/campionati). In alcune leghe, come la Premier League (GB1), si osserva una grande concentrazione di giocatori con valori di

mercato alti e un numero variabile di goal. In altre leghe, come la Super League (GR1) e la Premier League russa (RU1), la concentrazione di valori di mercato piè alti è meno evidente.

```
tmpdata <- newdata %>%
    mutate(across(where(is.numeric), scale))

mu_data <- colMeans(newdata %>% dplyr::select(where(is.numeric)))
sigma_data <- apply(newdata%>% dplyr::select(where(is.numeric)), 2, var)

tmpdata$player_id <- newdata$player_id
tmpdata$mkval <- newdata$mkval
tmpdata$log_mkval <- newdata$log_mkval
newdata <- tmpdata
newdata$Age2 <- newdata$Age**2
```

```
set.seed(989)
plid <- unique(newdata$player_id) %>%
  sample(size = 0.30*length(unique(newdata$player_id)))

# dati in cui verifico la bontÃ  di adattamento dei vari modelli
te_data <- newdata %>% filter(player_id %in% plid) %>%
  group_by(player_id) %>%
  arrange(year_Q) %>%
  filter(row_number() == n()) %>%
  ungroup()


te_data <- te_data %>% filter(country_of_citizenship != "St. Lucia")

# dati di stima dei vari modelli
tr_data <- newdata %>% filter(player_id %in% plid) %>%
  group_by(player_id) %>%
  arrange(year_Q) %>%
  filter(row_number() < n()) %>%
  rbind.data.frame(newdata %>% filter(!(player_id %in% plid))) %>%
  ungroup

te_data <- te_data %>% arrange(player_id)
tr_data <- tr_data %>% arrange(player_id)

te_data <- te_data |> filter(!(country_of_citizenship %in%
      setdiff(unique(te_data$country_of_citizenship),
              unique(tr_data$country_of_citizenship))))
```

Al fine di confrontare modelli differenti si dividono i dati, precedentemente standardizzati, in un insieme di stima e uno di verifica. In particolare, si sono selezionati casualmente il 30% dei giocatori e per ognuno di essi si è preso l'ultimo valore osservato per comporre l'insieme di verifica.

Un semplice modello lineare

Sfruttando solo alcune delle variabili a disposizione, si decide di implementare un semplice modello lineare coniugato:

$$y_i = x_i^T \beta + \epsilon_i, \quad i = 1, \dots, n$$

$$\sigma_\epsilon^2 \sim IG(\nu, \lambda), \quad \beta | \sigma_\epsilon^2 \sim N(\beta_0, \sigma_\epsilon^2 D_0).$$

Il primo passo passo per stimare un modello in stan è costruire una lista contenente i dati necessari per la stima del modello

```
data_list <- list()
data_list$y <- tr_data$log_mkval # variabile risposta per il modello
prednames <- c("Age", "Age2", "sub_position", "year_Q",
               "typedomestic_league_x_month_min_palyed",
```

```
                    "typedomestic_league_x_month_goal",
                    "typedomestic_league_x_month_assists")
# covariate per la stima:
X <- tr_data %>% dplyr::select(all_of(prednames))
#Creiamo le dummy per la variabile sub_position:
Xmat <- dummy_cols(X, select_columns = "sub_position", remove_first_dumm
                   remove_selected_columns = T)
Xmat <- cbind(1, Xmat)
data_list$X <- Xmat
# covariate per il test set:
Xtest <- te_data %>% dplyr::select(all_of(prednames))
Xmat_test <- dummy_cols(Xtest, select_columns = "sub_position", remove_f
                   remove_selected_columns = T)
Xmat_test <- cbind(1, Xmat_test)
data_list$Xtest <- Xmat_test


data_list$D0 <- diag(ncol(Xmat)) * 10**2 # matrice delle covarianze dei
data_list$N <- nrow(Xmat) #n. di osservazioni train set
data_list$Ntest <- nrow(Xmat_test) #n. di osservazioni test set
data_list$J <- ncol(Xmat) #n. di colonne
data_list$B0 <- rep(0, ncol(Xmat)) #vettore delle medie dei Beta a prior
```

Costruiamo ora il modello in STAN:

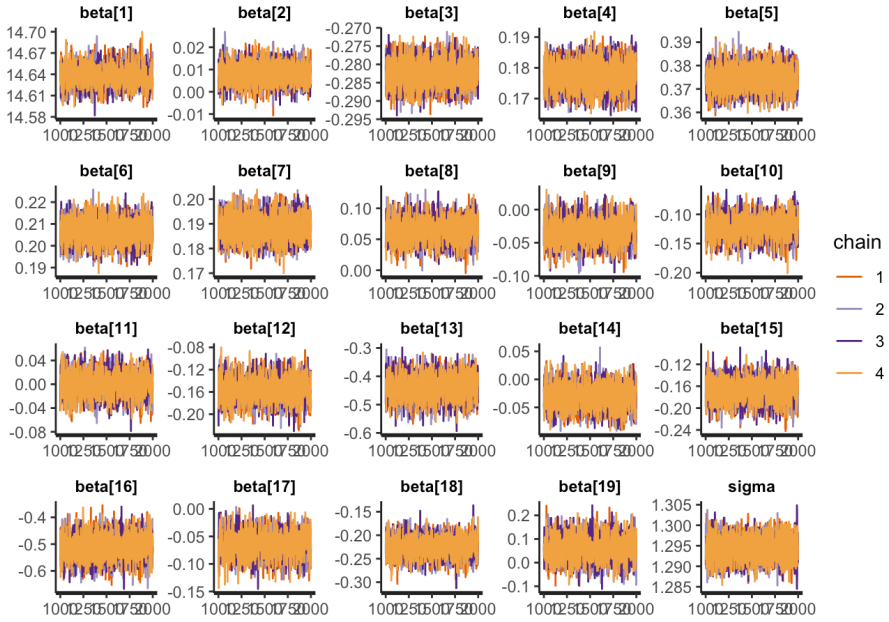
```
data {
  int <lower = 0> N;                    // numerosità campionaria
  int <lower = 0> Ntest;
  int <lower = 0> J;                    // numero di covariate
  vector[N] y;                          // variabile risposta
  matrix[N,J] X;                        // matrice dei predittori
  matrix[J,J] D0;                       // matrice della varianza a pric
  vector[J] B0;
  matrix[Ntest,J] Xtest;
}

parameters {
  vector[J] beta;                       // effetti fissi
  real <lower=0> sigma;                 // varianza
}

transformed parameters {
  real lprior;
  lprior = inv_gamma_lpdf(sigma | 0.01, 0.01);  // prior per sigma
  lprior +=  multi_normal_lpdf(beta | B0, D0*sigma); // prior per beta
}

model {
  target += lprior;
  target += normal_id_glm_lpdf(y | X, 0, beta, sigma); // postirior
}

generated quantities {
  vector[Ntest] prev;
  prev = Xtest * beta;
}
```

Stimiamo quattro catene di 2000 iterazioni ciascuna; per ogni catena fissiamo un burn-in di 1000 iterazioni.

```
lin_mod1 <- sampling(lin_mod, data = data_list, iter = 2000, chains = 4,
```

Valutiamo la convergenza delle catene:

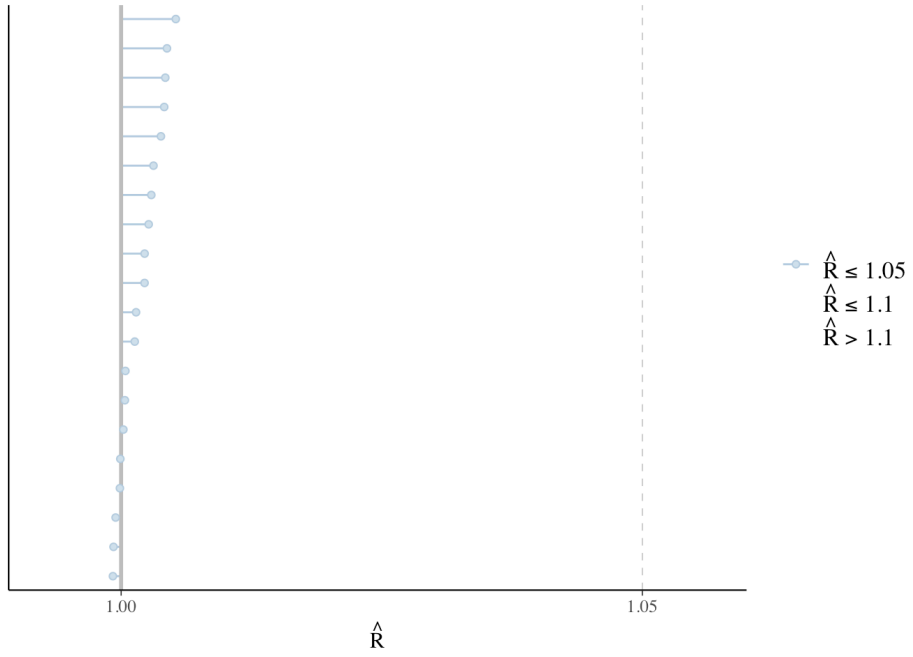
```
rstan::traceplot(lin_mod1, pars = names(lin_mod1)[1:20])
```



Le catene sembrano convergere alla medesima distribuzione.

Vediamo il grafico dell'Rhat (statistica di Gelman-Rubin) per ogni parametro stimato.

```
mcmc_rhat(rhat(lin_mod1, pars = names(lin_mod1)[1:20]))
```

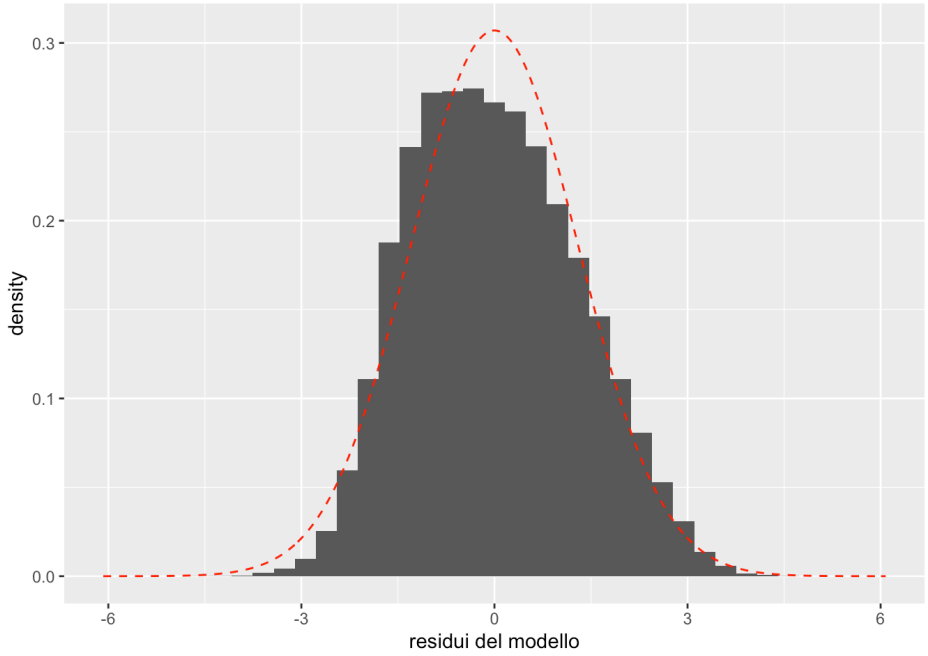


L' R-hat è una statistica che consente di comprendere se le catene convergono alla medesima distribuzione. Nel caso ideale, l'R-hat dovrebbe essere circa pari a 1, indicando quindi che le catene campionano dalla medesima distribuzione sottostante. Se dovessimo avere dei valori più grandi di 1.1/1.2 allora le nostre catene non convergono, quindi è necessario aumentare il numero di iterazioni oppure apportare altre modifiche (ad esempio riparametrizzare il modello).

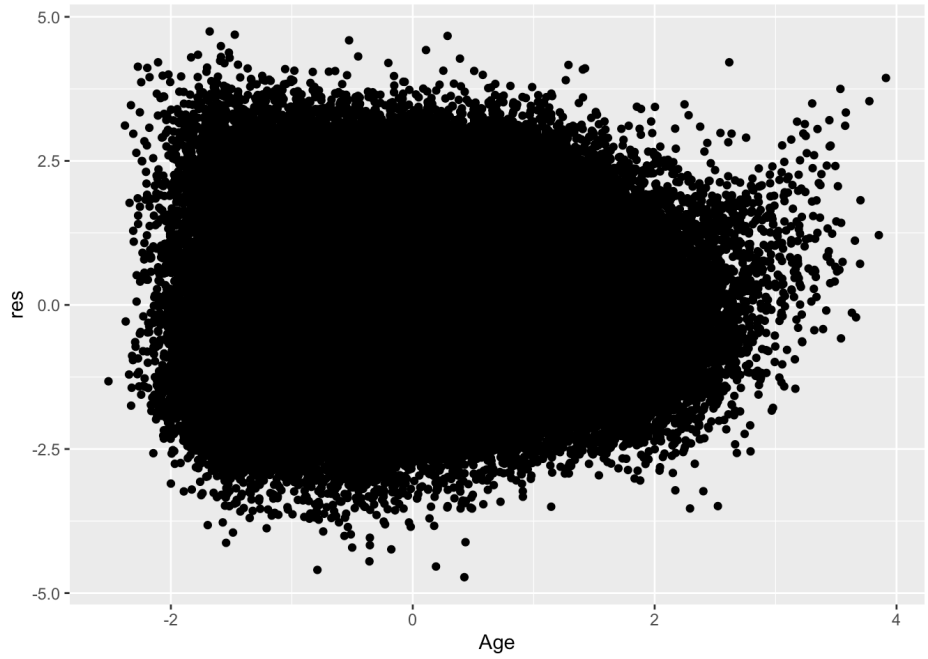
Le diagnostiche di convergenza non mostrano alcuna particolare problematica: sembra che le catene raggiungano la convergenza poichè R-hat è inferiore a 1.1.

Valutiamo la bontà di adattamento del modello ai dati attraverso l'analisi dei residui.

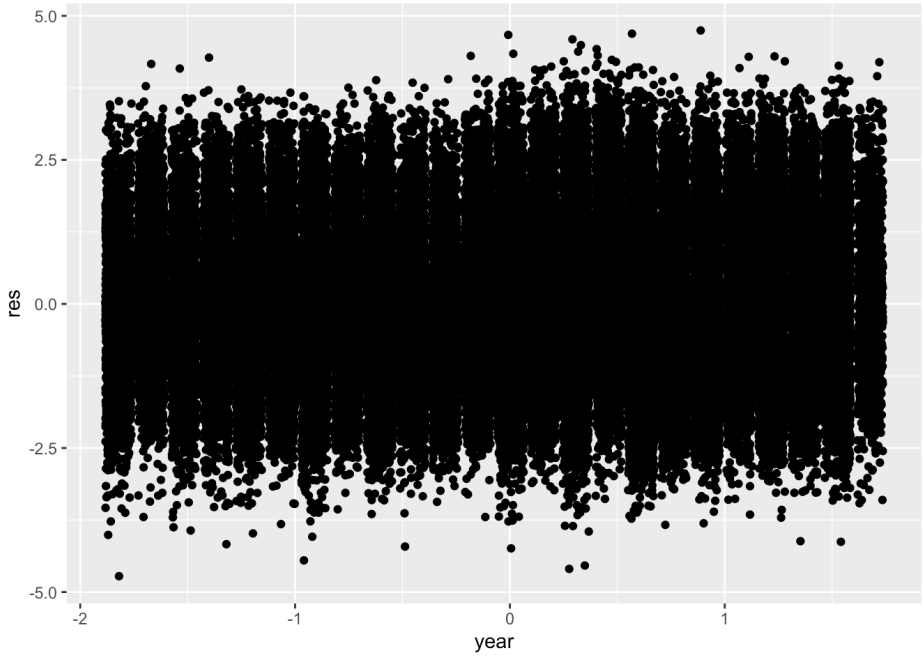
```
betahat_lin1 <- summary(lin_mod1)$summary[1:19, "mean"]
res_lin1 <- tr_data$log_mkval - (as.matrix(data_list$X)%*% betahat_lin1)

data.frame(res = res_lin1) %>%
  ggplot(aes(x = res)) +
  geom_histogram(aes(y=after_stat(density)), bins = 30) +
  xlab("residui del modello") +
  stat_overlay_normal_density(color = "red", linetype = "dashed")
```



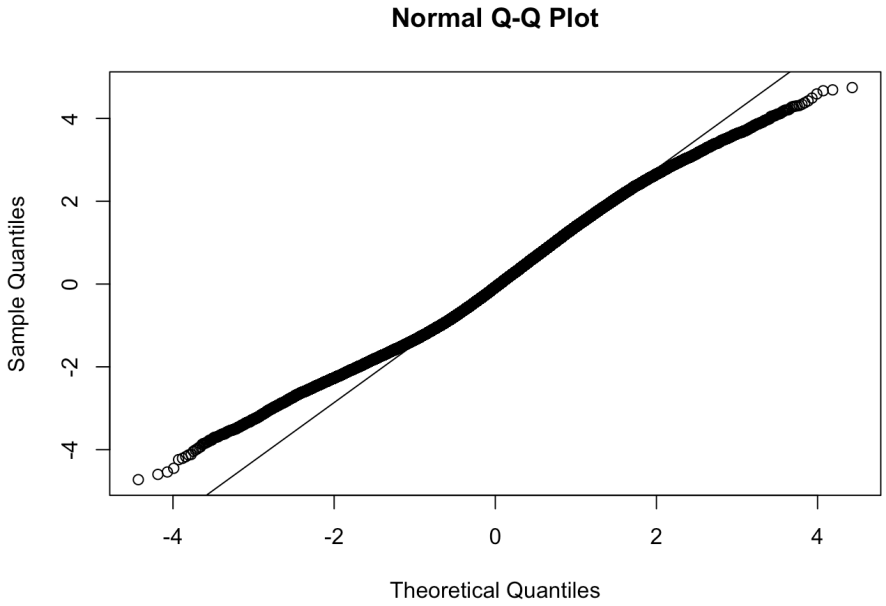

```
data.frame(res = res_lin1, Age = tr_data$Age) %>%
  ggplot(aes(y = res, x = Age)) +
  geom_point()
```




```
data.frame(res = res_lin1, year = jitter(tr_data$year_Q, 2)) %>%
  ggplot(aes(y = res, x = year)) +
  geom_point()
```




```
qqnorm(res_lin1)
qqline(res_lin1)
```



Notiamo che i residui hanno un comportamento simile alla normale, vi è una leggera deviazione sulle code della distribuzione, che potrebbe essere modellata attraverso l'utilizzo di una distribuzione a code pesanti, come ad esempio la t di student. Si osserva inoltre che non sembra esserci evidenti pattern tra i residui e i predittori.

Variable selection: caso modello lineare coniugato

Sfruttando l'algoritmo mostrato in precedenza, procediamo a fare selezione delle variabili per il modello lineare coniugato con tutte le variabili del dataset.

```
name <- colnames(tr_data %>% dplyr::select(-all_of(c("mkval", "player_ic
formula <- paste("log_mkval ~", paste(name, collapse = "+"))
simgamma <- rs_spikeslab(formula = formula, R = 5000, burnin = 1000,
```

```
                                  data = tr_data, verbose = F, v = 100)
        gamma_means <- colMeans(simgamma$gamma)
```

Nel vettore (gamma_means) si possono osservare le probabilità di inclusione delle variabili all'interno del modello.

Le 10 variabili con maggiore probabilità di inclusione sono:

```
        gamma_means[order(gamma_means, decreasing = T)][1:10]
```

```
                                       year_Q
                                            1
            typedomestic_league_x_month_min_palyed
                                            1
              typedomestic_cup_x_month_min_palyed
                                            1
                    typeother_x_month_min_palyed
                                            1
           typedomestic_league_x_month_yellow_card
                                            1
                  typedomestic_league_x_month_goal
                                            1
              typedomestic_league_x_month_assists
                                            1
      typedomestic_league_x_month_club_goal_scored
                                            1
         typedomestic_cup_x_month_club_goal_scored
                                            1
    typedomestic_league_x_month_club_goal_conceded
                                            1
```

Mentre le 10 variabili con minor probabilità di inclusione sono:

```
        gamma_means[order(gamma_means)][1:10]
```

```
          typeinternational_cup_x_month_min_palyed
                                           0.00000
              typedomestic_cup_x_month_yellow_card
                                           0.00000
                    typeother_x_month_yellow_card
                                           0.00000
                      typeother_x_month_red_cards
                                           0.00000
               typeinternational_cup_x_month_goal
                                           0.00000
    typeinternational_cup_x_month_club_goal_scored
                                           0.00000
                                              foot
                                           0.00000
                         typeother_x_month_assists
                                           0.00075
               typedomestic_cup_x_month_red_cards
                                           0.00200
         typeinternational_cup_x_month_yellow_card
                                           0.00825
```

Un'alternativa alla funzione da noi implementata consiste nell'utilizzo di una funzione della libreria BoomSpikeSlab, Scott e Scott (2023), che stima i parametri attraverso un gibbs sampling, in cui ogni indicatore di inclusione (di ogni variabile) viene campionato ad ogni passo condizionatamente a tutti gli altri indicatori. Un'importante differenza tra l'algoritmo che abbiamo scritto e la funzione $lm.spike$ del pacchetto è che $lm.spike$ specifica un indicatore γ per ogni parametro, quindi per ogni modalità delle variabili qualitative, mentre il nostro l'algoritmo specifica un parametro γ per ogni variabile. Di conseguenza, noi forniamo la probabilità di inclusione a livello di variabile, mentre il pacchetto la fornisce a livello di modalità delle variabili.

```
        model_boom <-  lm.spike(formula, niter = 5000, data = tr_data, ping = 0)
```

NOTA: niter= n° di iterazioni; ping= ogni quante iterazioni stampare a video lo stato di avanzamento

Confrontiamo le probabilità di inclusione stimate mediante il nostro algoritmo e quelle ottenute mediante il pacchetto BoomSpikeSlab per alcune variabili.

```
data.frame("BoomSpikeSlab" = apply(model_boom$beta, 2, function(x) 1-mea
           "RJMCMC" = round(gamma_means[1:34], 4))
```

	BoomSpikeSlab	RJMCMC
year_Q	0.9996	1.0000
typedomestic_league_x_month_min_palyed	0.9988	1.0000
typeinternational_cup_x_month_min_palyed	0.0022	0.0000
typedomestic_cup_x_month_min_palyed	0.8498	1.0000
typeother_x_month_min_palyed	0.9992	1.0000
typedomestic_league_x_month_yellow_card	0.9956	1.0000
typeinternational_cup_x_month_yellow_card	0.0000	0.0082
typedomestic_cup_x_month_yellow_card	0.0000	0.0000
typeother_x_month_yellow_card	0.0000	0.0000
typedomestic_league_x_month_red_cards	0.0238	0.9782
typeinternational_cup_x_month_red_cards	0.0016	0.0810
typedomestic_cup_x_month_red_cards	0.0000	0.0020
typeother_x_month_red_cards	0.0000	0.0000
typedomestic_league_x_month_goal	0.9994	1.0000
typeinternational_cup_x_month_goal	0.0000	0.0000
typedomestic_cup_x_month_goal	0.0002	0.1100
typeother_x_month_goal	0.0000	0.0095
typedomestic_league_x_month_assists	0.9994	1.0000
typeinternational_cup_x_month_assists	0.0000	0.0650
typedomestic_cup_x_month_assists	0.0000	0.0887
typeother_x_month_assists	0.0000	0.0008
typedomestic_league_x_month_club_goal_scored	0.9992	1.0000
typeinternational_cup_x_month_club_goal_scored	0.0012	0.0000
typedomestic_cup_x_month_club_goal_scored	0.5134	1.0000
typeother_x_month_club_goal_scored	0.3216	0.0565
typedomestic_league_x_month_club_goal_conceded	0.9974	1.0000
typeinternational_cup_x_month_club_goal_conceded	0.9986	1.0000
typedomestic_cup_x_month_club_goal_conceded	0.8516	1.0000
typeother_x_month_club_goal_conceded	0.5710	1.0000
typedomestic_league_x_month_club_points	0.7628	1.0000
typeinternational_cup_x_month_club_points	0.9982	1.0000
typedomestic_cup_x_month_club_points	0.4828	0.0145
typeother_x_month_club_points	0.6702	1.0000
typedomestic_league_x_best_club_position	0.9992	1.0000

Si osserva che, ad eccezione di poche variabili, i due algoritmi producono risultati simili.

Passiamo, ora, a:
- stimare un modello lineare conigato contenente i predittori con una probabilità di inclusione superiore a 0.5;
- stimare un modello con tutte le varaibili;
-confrontare i due modelli in termini di capacità predittiva.

A causa dell'elevato numero di predittori non possiamo fare ricorso a STAN per la stima dei modelli in quanto vi si presentano problemi di convergenza. Utilizziamo, invece, il pacchetto Martin, Quinn, e Park (2011) che permette di effettuare un gibbs sampling per il modello lineare con errori gaussiani:

$$y_i = x_i^T\beta + \epsilon_i, \quad \epsilon_i \sim N(0,\sigma^2)\beta \sim N(b_0, B_0^{-1}), \quad \sigma^{-2} \sim Ga(c_0/2, d_0/2), \qquad \beta \perp \sigma^{-2} \quad \text{a priori}$$

```
top_prob_variable <- names(gamma_means[gamma_means>0.5])
Xmat_selected <- model.matrix(~., data = rbind.data.frame(tr_data, te_da
                      dplyr::select(all_of(top_prob_variable)))
Xmat_train_sel <- Xmat_selected[1:nrow(tr_data), ]
lapply(1:4, function(l) {
  MCMCregress(tr_data$log_mkval ~ Xmat_train_sel - 1, mcmc = 2000, burni
```

```
}) %>% mcmc.list() -> model_selected
```

NOTA: verbose=0 impone di non stampare a video i progressi.

Per brevità non riportiamo tutte le analisi di convergenza delle catene.

Stimiamo anche il modello completo:

```
name_disc <- c("mkval", "player_id", "name", "log_mkval", "position")
Xmat_all <- model.matrix(~., data = rbind.data.frame(tr_data, te_data) %
                         dplyr::select(-all_of(name_disc)))
Xmat_train_all  <- Xmat_all[1:nrow(tr_data), ]
lapply(1:4, function(l) {
  MCMCregress(tr_data$log_mkval ~ Xmat_train_all - 1, mcmc = 2000, burni
}) %>% mcmc.list() -> model_all
```

Valutimo i diversi modelli in termini di capacità predittiva:

Le previsioni che si ottengono con il pacchetto BoomSpikeSlab mediano le previsioni che si ottengono con ogni generazione dei parametri, ad ogni generazione vengono stimati alcuni parametri a zero mentre altri diversi da zero, quindi la densità previsiva in corrispondenza di un insieme di predittori $\mathbf{x}_i$ viene costruita utilizzando diverse dimensionalità del vettore di parametri β.

Confrontiamo i diversi errori

```
err_dat <- data.frame("MSE" = c(err_lin_stan, err_boom_spsl, err_all, er
rownames(err_dat) <- c("Variabili di interesse", "BoomSpikeSlab",
                       "Tutte le variabili", "RJMCMC")
err_dat
```

	MSE
Variabili di interesse	5.084115e+06
BoomSpikeSlab	8.605812e-01
Tutte le variabili	8.618438e-01
RJMCMC	8.621516e-01

Come ci aspettavamo l'errore ottenuto mediante il modello in cui abbiamo utilizzato un numero ristretto di variabili (le più interessanti) è il più elevato. L'errore ottenuto mediante gli altri tre modelli risulta essere molto simile. Questo ci porta ad affermare che la selezione delle variabili non apporta benefici alla previsione, ma sicuramente migliora l'aspetto interpretativo, perchè ci permette di selezionare le variabili che influenzano maggiormente la risposta.

L' R^2 del modello stimato con variabili selezionate mediante sthocastic search risulta essere pari a

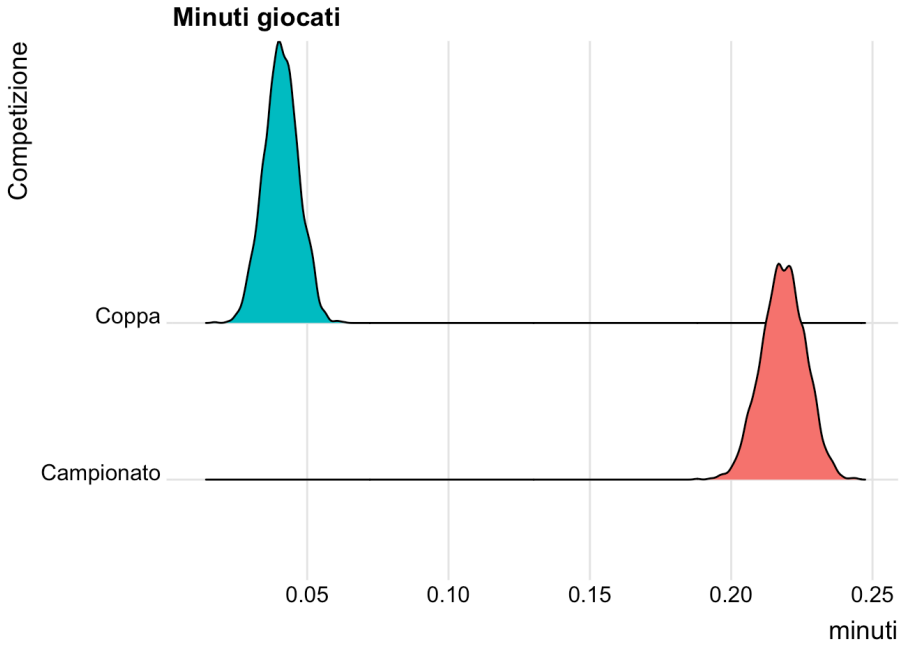
```
res_sel_lin <- tr_data$log_mkval - rowMeans(Xmat_train_sel %*% t(model_s
R2 <- 1 - var(res_sel_lin) / var(tr_data$log_mkval)
R2
```

```
[1] 0.6146645
```

Densità del parametro dei minuti giocati in campionato e coppe

```
data.frame("minuti" = c(model_sel_sim[, 3], model_sel_sim[, 5]),
           "Competizione" = c(rep("Campionato", 8000), rep("Coppa", 8000
  ggplot(aes(x = minuti, y = Competizione, fill = Competizione)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none") +
  labs(title = "Minuti giocati")
```

```
Picking joint bandwidth of 0.00102
```



Vediamo che vi è un ordinamento stocastico tra le due variabili casuali, il parametro associato alla variabile minuti giocati in campionato risulta essere stocasticamente più grande.

L'effetto stimato dell'età sul valore del giocatore risulta essere

Come abbiamo visto precedentemente i dati a nostra disposizione sono longitudinali, infatti per molti giocatori disponiamo di più osservazioni nel tempo. Possiamo quindi inserire un'intercetta casuale per giocatore e stimare un modello ad effetti misti.

Modello ad effetti misti

Adattiamo un modello lineare ad effetti misti del tipo

$$
\begin{aligned}
Y|\beta_\gamma, u, \sigma^2_\epsilon, \sigma^2_u &\sim N(X_\beta + Zu,\, \sigma^2_\epsilon I_n) \\
&\text{dove} \\
\beta|\sigma^2_\epsilon &\sim N(0,\, \sigma^2_\epsilon D_0) \\
\sigma^2_\epsilon &\sim IG(\nu_\epsilon,\, \lambda_\epsilon) \\
u &\sim N_k(0,\, \sigma^2_u I_k) \\
\sigma^2_u &\sim IG(\nu_u,\, \lambda_u)
\end{aligned}
$$

Per stimare il modello utilizziamo uno schema gibbs sampling. Le full conditional sono analoghe a quelle viste per il gibbs spike and slab, ad eccezione per il fatto che fissiamo tutti i parametri gamma a 1.

```
# gibbs sampler
gibbs_random_eff <- function(R, beta_init, sigma_init, u_init, Z, y, X,
                             ni_e, ni_u, lambda_e, lambda_u, D0, verbose
{ # in sigma_init prima posizione sigma_e ed in seconda posizione sigma_
  p_b <- ncol(X)
  p_u <- ncol(Z)
  n <- nrow(X)
  par_init <- c(beta_init, u_init, sigma_init)
  # matrici che conterrà le simulazioni
  parout <- matrix(NA, R+1, length(par_init))
  colnames(parout) <- c(colnames(X), colnames(Z), "Sigma_e^2", "Sigma_u^

  # inserisco i parametri inizializzati
  parout[1, ] <- par_init
```

```
  # matrici che serviranno in tutto l'algoritmo
  X <- Matrix(X) # trasformo X in matrice sparsa
  Z <- Matrix(Z) # trasformo Z in matrice sparsa
  XtX <- crossprod(X)         # t(X) %*% X
  D_inv <- Matrix(solve(D0))
  XtX_inv <- chol2inv(chol(XtX + D_inv))  # (t(X) %*% X + D0^-1)^-1
  ZtZ <- crossprod(Z)

  for(i in 1:R)
  {
    # genero i beta
    sigma_e <- parout[i, length(par_init)-1]
    epsilon_u <- y - Z %*% parout[i, (p_b+1):(p_b + p_u)]
    parout[i+1, 1:p_b] <- mvnfast::rmvn(1, mu = XtX_inv %*% crossprod(X,
                                            sigma = XtX_inv * sigma_e)

    # genero gli u
    sigma_u <- parout[i, length(par_init)]
    B <- Diagonal(p_u) *  (sigma_e / sigma_u)
    epsilon_b <- y - X %*% as.vector(parout[i+1, 1:p_b])
    ZTZ_inv <- chol2inv(chol(ZtZ + B))
    parout[i+1, (p_b+1):(p_b+p_u)] <- rmvn.sparse(1, mu = ZTZ_inv %*% cr
                                  CH = Cholesky(1/sigma_e * (ZtZ + B)),

    # genero sigma epsilon
    epsilon_tot <- epsilon_b - Z %*% parout[i+1, (p_b+1):(p_b+p_u)]
    beta_cor <- as.vector(parout[i+1, 1:p_b])
    parout[i+1, length(par_init)-1] <- 1/rgamma(1, (n/2 + ni_e + p_b/2),
                                            drop(crossprod(epsilon_t
                                               quad.form(D_inv, beta
                                               lambda_e))
    # genero sigma_u
    u_cor <- parout[i+1, (p_b+1):(p_b+p_u)]
    parout[i+1, length(par_init)] <- 1/rgamma(1, ni_u + p_u/2, (crosspro

    if(verbose == T) print(i)

  }
  return(coda::mcmc(parout[-(1:burnin), ]))
}
```

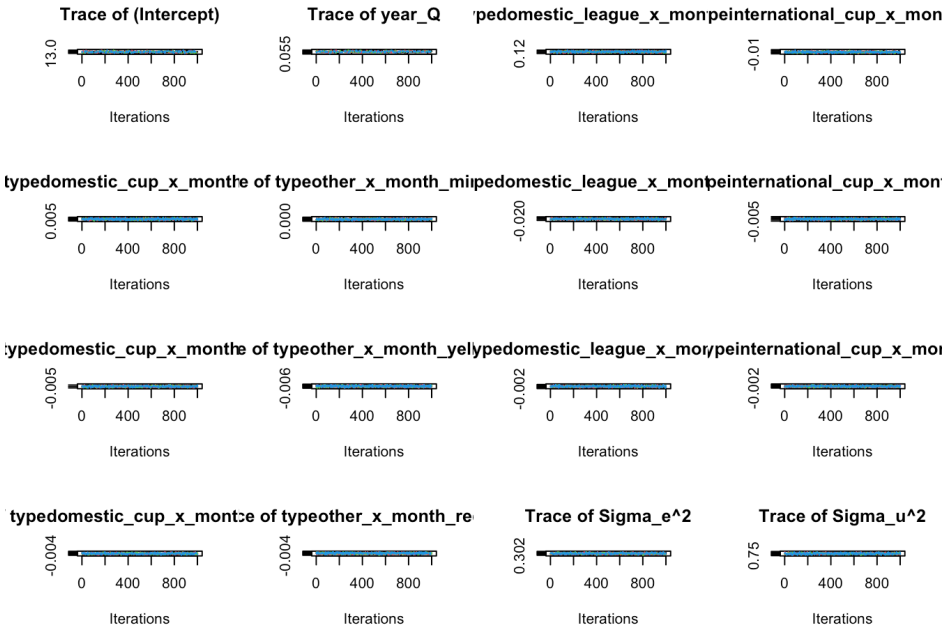
Stimiamo un modello ad effetti casuali inizialmente considerando tutte le variabili a disposizione. Utiliziamo 4 catene per poter poi valutare la convergenza alla distribuzione target.

```
X_all <- model.matrix(as.formula(formula), data = rbind.data.frame(tr_da
X_train <- X_all[1:nrow(tr_data), ]
N <- tr_data %>% count(player_id)

I <- sapply(N$n, function(x) rep(1, x))
Z <- bdiag(I)
colnames(Z) <- N$player_id


# inizializzazione dei parametri
lmodel <- lm(tr_data$log_mkval ~ -1 + X_train)

set.seed(1234)
beta_init <- mvnfast::rmvn(4, coef(lmodel), 2*diag(summary(lmodel)$coeff
set.seed(4321)
sigma_init <- mvnfast::rmvn(4, c(summary(lmodel)$sigma**2, 1), diag(2)*0
u_init <- matrix(0, 4, ncol(Z))

mix_lin_mod <- mcmc.list(lapply(1:4, function(i)
  gibbs_random_eff(2 * 10**3, beta_init[i, ], sigma_init[i, ], u_init[i,
                                y = tr_data$log_mkval, X = X_train, ni_e
                                lambda_e=0.1, lambda_u=0.1, D0 = 100*dia
```

```
                                        verbose = F)))
```

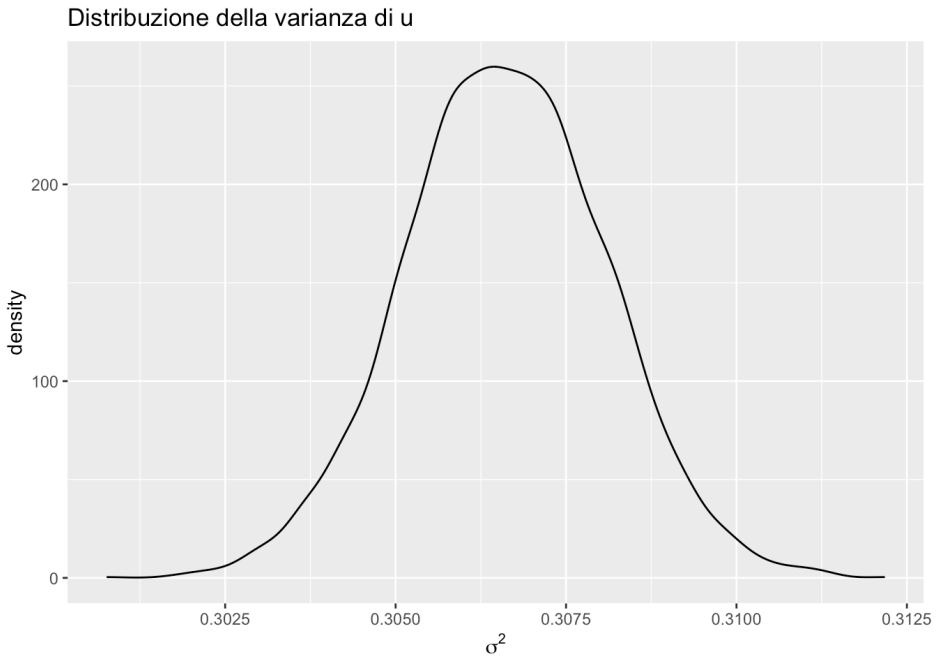
Per brevità non riportiamo tutte le diagnostiche di convergenza. Vediamo il traceplot di alcuni parametri (tra cui le varianze σ_ϵ e σ_u)

```
par(mfrow = c(4, 4))
coda::traceplot(mix_lin_mod[, c(1:14, ncol(mix_lin_mod[[1]])-1, ncol(mix
```



Possiamo guardare la stima della distribuzione della varianza dell'effetto casuale

```
mix_lin_mod_sim <- do.call(rbind, lapply(mix_lin_mod, function(l) l))
data.frame("sigma2_u" = mix_lin_mod_sim[, ncol(mix_lin_mod_sim)-1]) %>%
  ggplot(aes(x = sigma2_u)) +
  geom_density()+
  labs(title = "Distribuzione della varianza di u") +
  xlab(expression(sigma^2))
```



Osservando una stima della distribuzione possiamo affermare che il parametro di σ_u^2 sia

diverso da zero, risulta quindi ragionevole inserire l'intercetta casuale nel modello.

L'R^2 del modello risulta essere pri a:

```
        1 - var(res_mixed_all)/var(tr_data$log_mkval)
```

```
[1] 0.8854975
```

Con un solo parametro in più, rispetto al modello ad effetti fissi, riusciamo a migliorare notevolmente l'indice R^2.

Variable selection: modello ad effetti misti

Fino ad ora abbiamo affrontato il problema della selezione del modello nell'ambito del modello lineare con effetti fissi. Potremmo essere interessati a fare selezione del modello per modelli ad effetti misti. Consideriamo il modello lineare con un'intercetta casuale, in cui ci condizioniamo alla presenza dell'effetto casuale e facciamo selezione delle variabili mediante distribuzione a priori dirac spike and slab per i parametri fissi del modello.

Per stimare il modello specificato (precedentemente) abbiamo implementato uno schema ibrido gibbs-metropolis, in cui utilizziamo le distribuzioni condizionate note per generare tutti i parametri ad eccezione dei parametri γ che richiedono un passo di metropolis.

```
        # inizializzazione dei parametri
        p_beta <- tr_data %>%
          dplyr::select(-all_of(c("player_id", "position","name", "log_mkval", '
          ncol
        gamma_init <- rbinom(p_beta, size = 1, p = 0.5)
        beta_init <- beta_init[1, ]
        sigma_init <- sigma_init[1, ]
        u_init <- rep(0, ncol(Z))

        gibbs_spike_samp <- gibbs_spikeslab(4 * 10**3, beta_init, sigma_init, u_
                              tr_data %>% dplyr::select(-all_of(c("player_id", '
                              Z=Z, response = "log_mkval", theta=0.5, ni_e=0.01,
                              lambda_e=0.01, lambda_u=0.01, v=100, verbose = F)

        mix_mod_sel <- colMeans(gibbs_spike_samp$gamma[-(1:1000), ]) # tolgo il
```

Un breve confronto delle probabilità stimate da quest'ultimo algoritmo ed i due precedenti

```
        data.frame("BoomSpikeSlab" = apply(model_boom$beta, 2, function(x) 1-mea
                   "RJMCMC" = round(gamma_means[1:34], 4),
                   "mixedmodel" =mix_mod_sel[1:34])
```

	BoomSpikeSlab	RJMCMC
year_Q	0.9996	1.0000
typedomestic_league_x_month_min_palyed	0.9988	1.0000
typeinternational_cup_x_month_min_palyed	0.0022	0.0000
typedomestic_cup_x_month_min_palyed	0.8498	1.0000
typeother_x_month_min_palyed	0.9992	1.0000
typedomestic_league_x_month_yellow_card	0.9956	1.0000
typeinternational_cup_x_month_yellow_card	0.0000	0.0082
typedomestic_cup_x_month_yellow_card	0.0000	0.0000
typeother_x_month_yellow_card	0.0000	0.0000
typedomestic_league_x_month_red_cards	0.0238	0.9782
typeinternational_cup_x_month_red_cards	0.0016	0.0810
typedomestic_cup_x_month_red_cards	0.0000	0.0020
typeother_x_month_red_cards	0.0000	0.0000
typedomestic_league_x_month_goal	0.9994	1.0000
typeinternational_cup_x_month_goal	0.0000	0.0000
typedomestic_cup_x_month_goal	0.0002	0.1100
typeother_x_month_goal	0.0000	0.0095
typedomestic_league_x_month_assists	0.9994	1.0000
typeinternational_cup_x_month_assists	0.0000	0.0650

```
typedomestic_cup_x_month_assists                         0.0000 0.0887
typeother_x_month_assists                                0.0000 0.0008
typedomestic_league_x_month_club_goal_scored             0.9992 1.0000
typeinternational_cup_x_month_club_goal_scored           0.0012 0.0000
typedomestic_cup_x_month_club_goal_scored                0.5134 1.0000
typeother_x_month_club_goal_scored                       0.3216 0.0565
typedomestic_league_x_month_club_goal_conceded           0.9974 1.0000
typeinternational_cup_x_month_club_goal_conceded         0.9986 1.0000
typedomestic_cup_x_month_club_goal_conceded              0.8516 1.0000
typeother_x_month_club_goal_conceded                     0.5710 1.0000
typedomestic_league_x_month_club_points                  0.7628 1.0000
typeinternational_cup_x_month_club_points                0.9982 1.0000
typedomestic_cup_x_month_club_points                     0.4828 0.0145
typeother_x_month_club_points                            0.6702 1.0000
typedomestic_league_x_best_club_position                 0.9992 1.0000
                                                      mixedmodel
year_Q                                                0.89103632
typedomestic_league_x_month_min_palyed                0.72542486
typeinternational_cup_x_month_min_palyed              0.21026325
typedomestic_cup_x_month_min_palyed                   0.35654782
typeother_x_month_min_palyed                          0.44451849
typedomestic_league_x_month_yellow_card               0.29590137
typeinternational_cup_x_month_yellow_card             0.15361546
typedomestic_cup_x_month_yellow_card                  0.02699100
typeother_x_month_yellow_card                         0.00000000
typedomestic_league_x_month_red_cards                 0.08197268
typeinternational_cup_x_month_red_cards               0.00000000
typedomestic_cup_x_month_red_cards                    0.00000000
typeother_x_month_red_cards                           0.00000000
typedomestic_league_x_month_goal                      0.99166944
typeinternational_cup_x_month_goal                    0.76207931
typedomestic_cup_x_month_goal                         0.69910030
typeother_x_month_goal                                0.50916361
typedomestic_league_x_month_assists                   0.77774075
typeinternational_cup_x_month_assists                 0.36421193
typedomestic_cup_x_month_assists                      0.10463179
typeother_x_month_assists                             0.26957681
typedomestic_league_x_month_club_goal_scored          0.80973009
typeinternational_cup_x_month_club_goal_scored        0.37754082
typedomestic_cup_x_month_club_goal_scored             0.16061313
typeother_x_month_club_goal_scored                    0.17127624
typedomestic_league_x_month_club_goal_conceded        0.80073309
typeinternational_cup_x_month_club_goal_conceded      0.52715761
typedomestic_cup_x_month_club_goal_conceded           0.09430190
typeother_x_month_club_goal_conceded                  0.12462512
typedomestic_league_x_month_club_points               0.46251250
typeinternational_cup_x_month_club_points             0.50749750
typedomestic_cup_x_month_club_points                  0.37754082
typeother_x_month_club_points                         0.27157614
typedomestic_league_x_best_club_position              0.88303899
```

Utilizzando il gibbs scritto precedentemente adattiamo un modello con tutte le covariate con probabilità di inclusione maggiore di 0.5.

Le variabili che includiamo nel modello sono

```
sel_var_mix <- names(mix_mod_sel)[mix_mod_sel>0.5]
sel_var_mix
```

```
[1] "year_Q"
[2] "typedomestic_league_x_month_min_palyed"
[3] "typedomestic_league_x_month_goal"
[4] "typeinternational_cup_x_month_goal"
[5] "typedomestic_cup_x_month_goal"
[6] "typeother_x_month_goal"
[7] "typedomestic_league_x_month_assists"
[8] "typedomestic_league_x_month_club_goal_scored"
[9] "typedomestic_league_x_month_club_goal_conceded"
```

```
 [10] "typeinternational_cup_x_month_club_goal_conceded"
 [11] "typeinternational_cup_x_month_club_points"
 [12] "typedomestic_league_x_best_club_position"
 [13] "Age"
 [14] "sub_position"
 [15] "foot"
 [16] "player_club_domestic_competition_id"
 [17] "Age2"
```

Adattiamo il modello e valutiamo l'errore che commettiamo nel test-set

```
form_mix <- paste("log_mkval ~ ", paste(sel_var_mix, collapse = "+"))
X_sel <- model.matrix(as.formula(form_mix), data = rbind.data.frame(tr_d
X_train_sel <- X_sel[1:nrow(tr_data), ]

# inizializzazione dei parametri
lmodel <- lm(form_mix, data = tr_data)

set.seed(1234)
beta_init <- mvnfast::rmvn(4, coef(lmodel), 2*diag(summary(lmodel)$coeff
set.seed(4321)
sigma_init <- mvnfast::rmvn(4, c(summary(lmodel)$sigma**2, 1), diag(2)*0
u_init <- matrix(0, 4, ncol(Z))

mix_lin_sel_mod <- gibbs_random_eff(5 * 10**3, beta_init[1, ], sigma_ini
                                    u_init[1, ], Z=Z, y = tr_data$log_mk
                                    X = X_train_sel, ni_e=0.1, ni_u=0.1,
                                    lambda_u=0.1, D0 = 100*diag(ncol(X_t
```

L'errore di previsione su tale modello risulta essere pari a

```
# matrice del modello per il test set
X_test_all <- X_all[(nrow(tr_data)+1):nrow(X_all), ]
X_test_sel <- X_sel[(nrow(tr_data)+1):nrow(X_all), ]

# giocatori nel test presenti anche nel train
player_tt <- te_data$player_id %in% colnames(Z)
# costruzione delle stime
yhat_mixed_sel_test <- matrix(NA, nrow(X_test_sel), nrow(mix_lin_sel_moc
u_est <- mix_lin_sel_mod[, (ncol(X_test_sel)+1):(ncol(X_test_sel)+ncol(Z

for(i in 1:nrow(te_data))
{
  if(player_tt[i])
  {
    u_pos <- which(colnames(Z) %in% te_data$player_id[i])
    yhat <- X_test_sel[i, ] %*% t(mix_lin_sel_mod[, 1:ncol(X_test_sel)])
  } else yhat <-  X_test_sel[i, ] %*% t(mix_lin_sel_mod[, 1:ncol(X_test_

  yhat_mixed_sel_test[i, ] <- yhat
}

# errore di previsione
y_hat <- rowMeans(yhat_mixed_sel_test)
err_mix_sel <- mean((te_data$log_mkval - y_hat)**2)
```

L'errore che otteniamo utilizzando tutte le covariate risulta essere pari a

```
# matrice del modello per il test set
X_test_all <- X_all[(nrow(tr_data)+1):nrow(X_all), ]

# costruzione delle stime
yhat_mixed_test <- matrix(NA, nrow(X_test_sel), nrow(mix_lin_mod_sim))
u_est <- mix_lin_mod_sim[, (ncol(X_test_all)+1):(ncol(X_test_all)+ncol(Z

for(i in 1:nrow(te_data))
{
  if(player_tt[i])
```

```
    {
      u_pos <- which(colnames(Z) %in% te_data$player_id[i])
      yhat <- X_test_all[i, ] %*% t(mix_lin_mod_sim[, 1:ncol(X_test_all)])
    } else yhat <-  X_test_all[i, ] %*% t(mix_lin_mod_sim[, 1:ncol(X_test_

    yhat_mixed_test[i, ] <- yhat
  }

  # errore di previsione
  y_hat_all <- rowMeans(yhat_mixed_test)
  err_mix_all <- mean((te_data$log_mkval - y_hat_all)**2)
```

Osserviamo che l'errore che si commette sul test set mediante il modello con tutte le covariate a disposizione risulta essere leggermente inferiore. Tuttavia il modello selezionato dalla procedura spike and slab riduce notevolmente il numero di covariate utilizzate e migliora fortemente l'interpretabilità del modello. La differenza di errore commesso dai due modelli risulta essere trascurabile. Osserviamo infine che il modello ad effetti misti porta ad un errore di previsione notevolmente inferiore rispetto al modello ad effetti fissi.

Confrontiamo alcune probabilità di inclusione delle covariate stimate dai tre metodi visti

```
  data.frame("BoomSpikeSlab" = apply(model_boom$beta, 2, function(x) 1-mea
             "RJMCMC" = round(gamma_means[1:34], 4),
             "Gibbs_LMM" = round(mix_mod_sel[1:34], 4))
```

	BoomSpikeSlab	RJMCMC	Gibbs_LMM
year_Q	0.9996	1.0000	0.8910
typedomestic_league_x_month_min_palyed	0.9988	1.0000	0.7254
typeinternational_cup_x_month_min_palyed	0.0022	0.0000	0.2103
typedomestic_cup_x_month_min_palyed	0.8498	1.0000	0.3565
typeother_x_month_min_palyed	0.9992	1.0000	0.4445
typedomestic_league_x_month_yellow_card	0.9956	1.0000	0.2959
typeinternational_cup_x_month_yellow_card	0.0000	0.0082	0.1536
typedomestic_cup_x_month_yellow_card	0.0000	0.0000	0.0270
typeother_x_month_yellow_card	0.0000	0.0000	0.0000
typedomestic_league_x_month_red_cards	0.0238	0.9782	0.0820
typeinternational_cup_x_month_red_cards	0.0016	0.0810	0.0000
typedomestic_cup_x_month_red_cards	0.0000	0.0020	0.0000
typeother_x_month_red_cards	0.0000	0.0000	0.0000
typedomestic_league_x_month_goal	0.9994	1.0000	0.9917
typeinternational_cup_x_month_goal	0.0000	0.0000	0.7621
typedomestic_cup_x_month_goal	0.0002	0.1100	0.6991
typeother_x_month_goal	0.0000	0.0095	0.5092
typedomestic_league_x_month_assists	0.9994	1.0000	0.7777
typeinternational_cup_x_month_assists	0.0000	0.0650	0.3642
typedomestic_cup_x_month_assists	0.0000	0.0887	0.1046
typeother_x_month_assists	0.0000	0.0008	0.2696
typedomestic_league_x_month_club_goal_scored	0.9992	1.0000	0.8097
typeinternational_cup_x_month_club_goal_scored	0.0012	0.0000	0.3775
typedomestic_cup_x_month_club_goal_scored	0.5134	1.0000	0.1606
typeother_x_month_club_goal_scored	0.3216	0.0565	0.1713
typedomestic_league_x_month_club_goal_conceded	0.9974	1.0000	0.8007
typeinternational_cup_x_month_club_goal_conceded	0.9986	1.0000	0.5272
typedomestic_cup_x_month_club_goal_conceded	0.8516	1.0000	0.0943
typeother_x_month_club_goal_conceded	0.5710	1.0000	0.1246
typedomestic_league_x_month_club_points	0.7628	1.0000	0.4625
typeinternational_cup_x_month_club_points	0.9982	1.0000	0.5075
typedomestic_cup_x_month_club_points	0.4828	0.0145	0.3775
typeother_x_month_club_points	0.6702	1.0000	0.2716
typedomestic_league_x_best_club_position	0.9992	1.0000	0.8830

Possiamo osservare che la selezione nel modello LMM produce probabilità di inclusione leggermente differenti rispetto agli altri due metodi visti. In particolare si osserva un risultato molto coerente con le conoscenze che abbiamo sulle valutazioni dei giocatori, nella selezione nel modello LMM viene stimata una probabilità di inclusione per la variabile "minuti giocati nelle coppe internazionali" più alta rispetto agli altri due metodi. Ancora più sorprendente è il risultato legato alla probabilità di inclusione della variabile "goal segnati nelle coppe

internazionali", in cui i due metodi per modelli ad effetti fissi stimano una probabilità di inclusione pari a 0.0004 e 0.002, mentre l'algoritmo per la selezione nel modello LMM stima una probabilità di inclusione pari a 0.6231. Tale covariata non presenta molta variabilità tra i giocatori, in quanto la maggior parte di essi non segna nelle coppe internazionali, non è quindi sorprendente che i due algoritmi di selezione per il modello ad effetti fissi stimino una bassa probabilità di inclusione di tale variabile, è piuttosto sorprendente che l'algoritmo di selezione per il modello LMM riesca a stimare una probabilità di inclusione parecchio alta.

Bayesian model selection for generalized linear mixed models

Terminiamo questa nostra presentazione mostrando un nuovo approccio alla Bayesian model selection, introdotto da Xu et al. (2023).

I Generalized linear mixed models (GLMM) sono ampiamente utilizzati per modellare dati di natura non Gaussiana con osservazioni dipendenti. Nonostante le procedure di stima Bayesiane per i GLMM siano utlizzate, ci sono pochi paper che parlano della Bayesian model selection per i GLMM. Attualmente, molti articoli utilizzano il deviance information criterion (DIC) (Spiegelhalter et al. (2002)) per effettuare la Bayesian model selection per i GLMM:

$$\text{DIC} = D(\bar{\theta}) + 2p_D$$

dove
- $D(\theta) = -2logp(y|\theta)$ è la devianza, che misura la qualità dell'adattamento del modello ai dati: minore è la devianza, migliore è l'adattamento;
- $\overline{D(\theta)} = \mathbb{E}_{\theta|y}[D(\theta)]$ è la media della devianza rispetto alla distribuzione a posteriori di θ;
- p_D è la complessità effettiva del modello (n° di parametri). Un modello con un DIC più basso è preferibile poichè indica un miglior equilibrio tra adattamento ai dati e complessità del modello.

Il problama del DIC è che non fa bene la selezione delle variabili nel caso in cui gli effetti casuali abbiano varianze grandi. Quindi, per ottenere dei risultati migliori, è stato sviluppato un nuovo Bayesian model selection approach per la selezione simultanea delle covariate e degli effetti casuali per i GLMM. Come sappiamo, fare inferenza sui GLMM è difficile perchè la funzione di verosimiglianza integrata non è disponibile in forma chiusa. Per risolvere il problema dell'integrazione degli effetti casuali, approssimiamo la funzione di verosimiglianza integrata utilizzando un approccio di pseudo-verosimiglianza che porta a una Gaussian likelihood approximation. Poi, assegniamo una flat prior al vettore dei coef di regressione e una aproximate reference prior per le componenti di varianza. In più, si considera anche il caso di una half-Cauchy prior per le radici quadrate delle componenti di varianza. Poichè la prior del vettore dei coef di regressione è impropria, è stato sviluppato un fractional Bayesian factor (FBF) approach (O'Hagan, 1995). L'approccio per la model selection è completamente automatico, quindi si evita l'inconveniente della specificazione soggettiva degli iperparametri e rende il metodo più accessibile. I due metodi per la selezione proposti prendono il nome di:
- the approximate reference method (ARM); - the half-Cauchy method (HCM).

NOTA: l'idea alla base delle reference prior è quella di formalizzare esattamente quello he si intende per a priori non informative: è una funzione che massimizza una qualche misura di distanza o divergenza (es. Kullback-Leibler) tra la posterior e la prior. Massimizzando la divergenza, si consente ai dati di avere il massimo effetto sulle stime a posteriori. Il problema delle reference prior esatte è che possono essere difficili da ottenere, quindi, spesso, si ricorre a loro approssimazioni che mantengono le proprietà desiderate di minima informatività.

NOTA: la half-Cauchy prior è utile nel caso di parametri non negativi che possono assumere valori anche molto grandi.

Modello di partenza

Si consideri: - il vettore della risposta $y = (y_1, y_2, \dots, y_n)^T$; - la matrice X, di dimensioni nxp, dei predittori e β il corrispettivo vettore p-dimensionale di effetti fissi; - una matrice Z_j,

di dimensioni nxq_j e il corrispettivo vettore α_j, di dimensione q_j, di effetti casuali, per $j = 1, \dots, Q$; - le osservazioni $y_1, \dots, Y_n$ sono indipendenti con densità appartenente alla famiglia esponenziale. Ciascuna osservazione y_i ha media mu_i e varianza ν_i; - ciascun α_j ha una distribuzione normale multivariata con vettore delle medie 0 e matrice di covarianza $\tau_j \Sigma_j$, con τ_j parametro ignoto e Σ_j matrice nota simmetrica e semi-definita positiva;

Pseudo Likelihood function

Un passo fondamentale nella Bayesian model selection è integrare via gli effetti casuali dalla funzione di verosimiglianza. Tuttavia, mentre per gli LMM gli effetti casuali possono essere integrati analiticamente , per i GLMM questo non è possibile.
Per risolvere questo problema, qui utilizziamo un approccio alla pseudo-verosimiglianza che approssima un GLMM per dati non-Gaussiani calcolando osservazioni aggiustate che vengono modellate utilizzando un LMM gaussiano approssimato. Consideriamo che α rappresenti tutti gli effetti casuali e che τ rappresenti tutte le componenti di varianza. Allora, la funzione di verosimiglianza con l'integrale (non risolvibile) sugli effetti casuali è:

$$L(\beta, \tau | y) = \int p(y|\alpha, \beta) p(\alpha|\tau) d\alpha$$

Il metodo qui proposto approssima questo integrale con un Gaussian LMM attraverso un approccio ala pseudo-verosimiglianza. Poi, per un modello LMM il corrispondente integrale può essere risolto analiticamente.

##Cenni al pseudo-likelihood approach L'approccio alla pseudo-verosimiglianza è una procedura iterativa che inizia scrivendo il modello come $y = \mu + \epsilon$, con $\mu = (\mu_1, \dots, \mu_n)^T$ e ϵ il vettore degli errori con $Cov(\epsilon) = V = diag(\nu_1, \dots, \nu_n)$. Sia $\hat{\alpha}, \hat{\beta}, \hat{\mu}, \hat{V}$ le stime correnti di α, β, μ, V. La procedura, a grandi linee, è la seguente: - $\hat{\beta}$ si inizializza sfruttando le stime di un GLM; - si approssima μ_i con uno sviluppo di Taylor del primo ordine intorno $\alpha = \hat{\alpha}$ e $\beta = \hat{\beta}$; - si riordinano tutti i termini in $y = \mu + \epsilon$ in modo tale che i termini che dipendono da y, $\hat{\alpha}$, $\hat{\beta}$, $\hat{\mu}$, appaiano sul lato sinistro dell'equazione, mentre tutto il resto va a destra; - si moltiplicano entrambi i lati per $\hat{V}^{-1}$. Come risultato, la parte di sinistra dell'equazione avrà la seguente forma: $y^* = \hat{V}^{-1}(y - \hat{\mu}) + X\hat{\beta} + \sum_j Z_j \hat{\alpha_j}$. y^* prenderà il nome di vettore di pseudo-osservazioni o vettore delle osservazioni aggiustate. - eguagliando y^* con la parte destra dell'equazione, otteniamo il seguente modello per le osservazioni aggiustate:

$$y^* \approx X\beta + \sum_j Z_j \alpha_j + \hat{V}^{-1}\epsilon \alpha_j \sim N(0, \tau_j \Sigma_j) \quad \epsilon \sim N(0, V)$$

Sostituendo a V la sua stima $\hat{V}$ allora $y*$ può essere modellato (approssimativamente) tramite un LMM: $ y^* N(X, _{j}_jZ_j_jZ_j^T + ^{-1})$. Quindi, adesso abbiamo a disposizione una funzione di pseudo-verosimiglianza in forma chiusa $p(y^*|\beta, \tau)$. Per fare selezione, si utilizzarà la pseudo-verosimiglianza in modo iterativo per ottenere delle stime dei parametri, per poi calcolare osservazioni aggiustate $y*$; poi, queste $y*$ verranno utilizzate al posto delle osservazioni originarie per fare selezione del modello. Nella pratica, il vettore delle osservazioni aggiustate verà calcolato usando il modello completo, ovvero quello con tutti i regressori e tutti gli effetti casuali.

Model selection

Si consideri lo spazio dei modelli $M = \{M_c, c = 1, .., C\}$, con C il numero di possibili modelli. Assumiamo che il modello M_c abbia K_c regressori; che X_c sia la matrice delle variabili esplicative; che β_c sia il corrispondente vettore di coefficienti; e che Q_c siano gli effetti casuali. Sia $\tau_c = (\tau_{c,1}, \dots, \tau_{c,Q_c})$ il vettore delle componenti di varianza dei Q_c effetti casuali nel modello M_c. La verosimiglianza integrata basata sulle y^* è:

$$p(y^*|M_c) = \int \int p(y^*|\beta_c, \tau_c) \pi(\beta_c, \tau_c|M_c) d\beta_c d\tau_c$$

dove $\pi(\beta_c, \tau_c|M_c)$ è la distribuzione a priori per (β_c, τ_c) condizionatamente al modello M_c. Assumendo che $\pi(M_c)$ sia la probabilità a priori del modello M_c, allora, tramite l'applicazione del modello di Bayes si ottiene che la probabilità a posteriori del modello è

$$P(M_c|y^*) = \frac{p(y^*|M_c)\pi(M_c)}{\sum_{r=1}^{C} p(y^*|M_c)\pi(M_c)} \; \alpha \; p(y^*|M_c)\pi(M_c)$$

Prior for model parameters

In questo contesto si usa una flat prior per β: $\pi(\beta|M) \; \alpha \; 1$. Nel caso della componente di varianza τ si usano una delle seguenti: - una approximate reference prior $ _1(|M)=, $ - $\pi_2(\tau) \; \alpha \; \tau^{\frac{-1}{2}}(\tau+1)^{-1}$ (deriva dall'assumere una half-Cauchy prior per $\sqrt{\tau}$).

Da qui emergono due varianti dell'approccio: ARM, che usa l'approximate reference prior per τ; HCM, che usa una half-Cauchy prior per $\sqrt{\tau}$.

Priors on the model space

Sia K il numero di covariate candidate e sia Q il numero di effetti casuali candidati. Siano K_c e Q_c, rispettivamente, il numero di covariate e il numero di effetti casuali nel modello M_c. Assumendo a priori indipendenza tra l'inclusione di effetti fissi e casuali, la probabilità a priori per il modello M_c è $P(M_c) = \frac{1}{2^Q(K+1)\binom{K}{K_c}}$. Questa formula deriva dal fatto che, nel caso di effetti fissi la probabilità del modello M_c con K_c covariate sarebbe $\frac{1}{(K+1)\binom{K}{K_c}}$; mentre, per gli efetti casuali avremmo 2^Q possibilità per l'inclusione e l'esclusione degli effetti casuali. Assumendo che ciascun effetto casuale abbia una probabilità a priori di inclusione pari a 0.5, allora la probabilità per il molldeo M_c con Q_c effetti casuli è pari a $1/2^Q$.

Dopo aver definito le priors per i parametri, possiamo ottenere la verosimiglianza integrata: i β possono essere integrati analiticamente; mentre, per l'integrazione dei τ non si procede in maniera analitica (non c'è modo), ma si sfrutta un'approssimazione di Laplace.

Fractional Bayes factors

Per ottenere le posterior model probabilities si usano i Fractional Bayes factors (FBF), che non sono altro che una modifica dei Bayes factors per il caso di a priori improrie. Non possiamo utilizzare il Bayes factor "classico" perchè i regressori hanno una a priori impropria e questo porta a un Bayes factor definito a meno di una costante di prorporzionalità non definita e non può essere usato per confrontare modelli direttamente. Per risolvere questo problema si utilizza l'FBF: in questo modo si fa un "train" della a priori impropria in modo tale da poter calcolare un BF sensato. Per "training" della a priori improria si intende l'uso del teorema di Bayes per combinare la a priori impropria con una frazione della verosimiglianza, in modo tale da ottenere una distribuzione propria. Quest'ultima verrà poi usata per calcolare il Bayes factor. La trained prior per il modello M_c si ottiene come:

$$\pi^b(\beta_c, \pi_c) = \frac{p^b(y^*|\beta_c, \tau_c)\pi(\beta_c, \tau_c|M_c)}{\int p^b(y^*|\beta_c, \tau_c)\pi(\beta_c, \tau_c|M_c)d\beta_c},$$

dove b è la frazione della funzione di verosimiglianza usata per fare il training della prior. La verosimiglianza integrata del modello M_c viene poi calcolata come l'integrale del prodotto tra la funzione di verosimiglianza elevata alla 1-b e la trained prior:

$$\begin{aligned} q_c(b, y^*) &= \int p^{1-b}(y^* \mid \beta_c, \tau_c)\pi^b(\beta_c, \tau_c)\, d\beta_c\, d\tau_c \\ &= \int p^{1-b}(y^* \mid \beta_c, \tau_c)\frac{p^b(y^* \mid \beta_c, \tau_c)\pi(\beta_c, \tau_c \mid M_c)}{\int p^b(y^* \mid \beta_c, \tau_c)\pi(\beta_c, \tau_c \mid M_c)\, d\beta_c\, d\tau_c}\, d\beta_c\, d\tau_c \\ &= \frac{\int p(y^* \mid \beta_c, \tau_c)\pi(\beta_c, \tau_c \mid M_c)\, d\beta_c\, d\tau_c}{\int p^b(y^* \mid \beta_c, \tau_c)\pi(\beta_c, \tau_c \mid M_c)\, d\beta_c\, d\tau_c}. \end{aligned}$$

La frazione b deve essere scelta con cautela: se è troppo piccolo allora il denominatore della

formula precedente potrebbe divergere; se è troppo grande allora usiamo troppa verosimiglianza per fare training delle prior e, quindi, avremo poca informazione nella verosimiglianza integrata per aggiornare le probabilità a priori dei modelli (avremo delle probabilità a posteriori per i modelli meno distinte). Considerazioni empiriche hanno portato alla scelta di un $b = (p+1)/n$. Quindi l'FBF del modello M_c versus M_l è definito come $BF^b_{cl} = \frac{q_c(b,y^*)}{q_l(b,y^*)}$. Infine, calcoliamo la posterior probability del modello M_c come:

$$P^b(M_c|y) = \frac{BF^b_{cl} \times P(M_c)}{\sum_{k=1}^{C} BF^b_{cl} \times P(M_k)}.$$

Referenze

Freedman, David A. 1983. «A note on screening regression equations». *the american statistician* 37 (2): 152–55.

Jeffreys, Harold. 1998. *The theory of probability*. OuP Oxford.

Martin, Andrew D, Kevin M Quinn, e Jong Hee Park. 2011. «MCMCpack: Markov chain monte carlo in R».

Miller, Alan. 2002. *Subset selection in regression*. chapman; hall/CRC.

Raftery, Adrian E. 1995. «Bayesian model selection in social research». *Sociological methodology*, 111–63.

Scott, Steven L, e Maintainer Steven L Scott. 2023. «Package "BoomSpikeSlab"».

Spiegelhalter, David J, Nicola G Best, Bradley P Carlin, e Angelika Van Der Linde. 2002. «Bayesian measures of model complexity and fit». *Journal of the royal statistical society: Series b (statistical methodology)* 64 (4): 583–639.

Tadesse, Mahlet G, e Marina Vannucci. 2021. «Handbook of bayesian variable selection».

Xu, Shuangshuang, Marco AR Ferreira, Erica M Porter, e Christopher T Franck. 2023. «Bayesian model selection for generalized linear mixed models». *Biometrics* 79 (4): 3266–78.