```
library(tidyverse); theme_set(theme_bw())
library(nimble)
```

IRT model network data

I modelli IRT (Item Response Theory) hanno come scopo primario lo studio delle abilità (latenti) di un gruppo di n individui che risponde a p domande, delle quali si vuole stabilire la difficoltà.

Un modello di base è il modello di **Rasch**. La variabile risposta $Y_{ij} \in \{0, 1\}$ per l'individuo i e la domanda j è uguale ad 1 se la risposta alla domanda è corretta, e 0 altrimenti. La probabilità di rispondere correttamente ad una domanda dipende da due componenti, l'abilità dell'individuo η_i e la difficoltà della domanda β_j, entrambi considerati parametri reali.

Il modello può essere scritto come

$$Y_{ij} \sim \text{Be}(\pi_{ij}), g(\pi_{ij}) = \eta_i - \beta_j,$$

con $g : (0, 1) \to \mathbb{R}$, e.g. logit. Si assume in genere che sia l'abilità che la difficoltà abbiano una distribuzione normale e, per rendere il modello identificabile, che $\sum_{j=1}^{p} \beta_j = 0$, o un qualche altro vincolo.

Se il nostro gruppo di individui $i = 1, \dots, n$ rappresenta una (o più) coorti di studenti di un dipartimento (come Statistica a Padova) può essere di interesse capire se l'abilità degli studenti sia in qualche modo legata alle loro amicizie, frequentazioni, o compagnie di studio. Un *proxy* di queste variabili può essere la rete di amicizie di Facebook. Quindi, interessa capire se e come l'abilità η_i dello studente i dipende dalla rete di amicizie dello studente, ed eventualmente da altre caratteristiche.

Indicando con $\mathbf{W}$ la matrice di adiacenza di dimensioni $n \times n$ contenente le connessioni di ogni studente, i.e. l'elemento $w_{uv} = 1$ se lo studente u e lo studente v sono amici su Facebook e 0 se non lo sono ($w_{uu} = 0$), possiamo tenere conto di questa informazione con un modello **CAR** (Conditionally Autoregressive). L'idea è che l'abilità di un individuo dipende dai suoi vicini, in termini di rete. Indicando con $\mathbf{C}$ la matrice con elementi $c_{uv} = w_{uv} / \sum_{h=1}^{n} w_{uh}$ e con $\mathbf{M} = \text{diag}(1/\sum_{h=1}^{n} w_{1h}, \dots, 1/\sum_{h=1}^{n} w_{nh})$, questa dipendenza può essere specificata con la distribuzione

$$\boldsymbol{\eta} \mid \boldsymbol{\mu}, \mathbf{C}, \mathbf{M}\tau, \gamma \sim N_n(\boldsymbol{\mu}, \tau^{-1}(I - \gamma\mathbf{C})^{-1}\mathbf{M}),$$

da cui

$$\eta_i \mid \boldsymbol{\eta}_{-i}, \dots \sim N(\mu_i + \gamma \sum_{k=1}^{n} c_{ik}(\eta_k - \mu_i), M_{ii}\tau^{-1})$$

dove $\boldsymbol{\eta}_{-i} = (\eta_1, \dots, \eta_{i-1}, \eta_{i+1}, \dots, \eta_n)$.

Note: ci sono varie specificazioni del modello CAR, in questo caso lo usiamo direttamente come effetto casuale, quindi deve essere una distribuzione propria (possiamo usare diverse specificazioni per ovviare al problema).

Simulazione in NIMBLE

Consideriamo una semplice simulazione e stimiamo il modello utilizzando un approccio bayesiano.

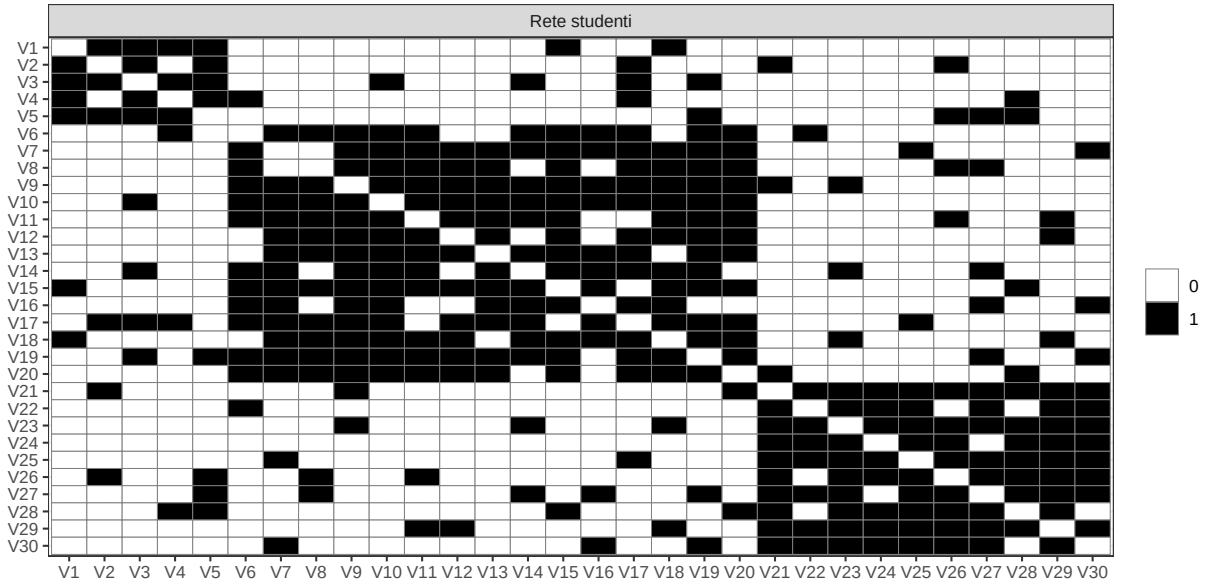
Generazione dei dati di rete: Per una semplice simulazione consideriamo $n = 30$ divisi in 3 macro gruppi di studio, il primo da 5 il secondo da 15 ed il terzo da 10 studenti. La probabilità di connessione all'interno di ogni gruppo è 0.9 mentre tra gruppi è 0.1

```
n = 30
set.seed(1234)
groups = c(rep(1,5),rep(2,15),rep(3,10))
```

```
W = matrix(0,n,n)
for(i in 2:n)
{
 for(j in 1:(i-1))
 {
  if(groups[i] == groups[j]) W[i,j] = rbinom(1,1, prob = 0.8)
  else W[i,j] = rbinom(1,1, prob = 0.1)

  W[j,i] = W[i,j]
 }
}
```

```
as_tibble(W) %>%
        mutate( row = paste0('V',1:n)) %>%
        gather(col,value,-row) %>%
    mutate(row = factor(row, levels =  paste0('V',1:n)),
     col = factor(col, levels =  paste0('V',n:1))) %>%
    ggplot +
    geom_tile(aes(x = row, y = col, fill = factor(value)),col = 'grey50')+
    scale_fill_manual(values= c('white','black'))+
    xlab('')+
    ylab('')+
    facet_wrap(~I("Rete studenti"))+
    theme(legend.title = element_blank())
```



Parametri della simulazione: Assumiamo che l'abilità nei gruppi sia normale con varianza 2 e che la media sia diversa per ogni gruppo, con valori $-1, 0, 2$. Consideriamo $p = 5$ domande con difficoltà generate da una normale con media 0 e varianza 2.

```
p=5
## true simulation parameters
dataAndParameters                          = lst()
dataAndParameters$trueAbilityMeans         = c(-1,0,2)
dataAndParameters$trueAbilityVars          = c(2,2,2)
dataAndParameters$trueDifficultiesMeans = 0
```

```
dataAndParameters$trueDifficultiesVars  = 2
dataAndParameters$groups                = groups

## place holders
dataAndParameters$data                  = matrix(NA,n,p)
dataAndParameters$ability               = numeric(n)
dataAndParameters$difficulty            = numeric(p)


set.seed(123)
dataAndParameters$difficulty  = with(dataAndParameters,
                     rnorm(p, trueDifficultiesMeans,
                       sqrt(trueDifficultiesVars )))


for(i in 1:n)
{
  dataAndParameters$ability[i] = with(dataAndParameters,
                       rnorm(1, trueAbilityMeans[groups[i]],
                         sqrt(trueAbilityVars[groups[i]])))
}

## data
for(i in 1:n)
{
 for(j in 1:p)
 {
  dataAndParameters$data[i,j] =  with(dataAndParameters,
                       rbinom(1,1,
                          plogis(ability[i] - difficulty[j])))
 }
}
```

Codice per il modello NIMBLE

Il codice implementa il modello descritto, e comprende il vincolo che le difficoltà sommino a zero.

```
codiceModello = nimbleCode({
for(i in 1:n)
{

  for(j in 1:p)
  {
    Y[i,j] ~ dbin(pi_ij[i,j],1)
    pi_ij[i,j] <- expit(eta[i] - beta[j])
  }
  mu[i] ~ dnorm(0,1)
}

for(j in 1:p)
{
 beta_tmp[j] ~ dnorm(0,1)
}
```

```
beta[1:p] <- beta_tmp[1:p] - mean(beta_tmp[1:p])

 eta[1:n] ~ dcar_proper(mu = mu[1:n], adj = adj[1:L], num = num[1:n], tau= tau, gamma= gamma)
 tau~dgamma(0.001,0.001)
 gamma~dunif(-1,1)
})
```

```
W_nimble = as.carAdjacency(W)


carModel = nimbleModel(code = codiceModello,
                           name = 'irtCar',
                           constants = list(d =4,
                                            n = n,
                                            p = p,
                          L = length(W_nimble$adj),
                          adj = W_nimble$adj,
                          num = W_nimble$num
                          ),
                    inits = list( eta = rnorm(n,0,1),
                   beta = rnorm(p,0,1),
                   mu = rep(0,n),
                   tau =1,
                   gamma = 1),
                               data = list(Y = dataAndParameters$data))


carConf    = configureMCMC(carModel, print = TRUE)
carConf$addMonitors('eta')
carMCMC    = buildMCMC(carConf)

carMCMCc   = compileNimble(carModel)
carMCMCc   = compileNimble(carMCMC, project = carModel)

system.time({carSamples = runMCMC(carMCMCc, niter = 300000, nburnin = 20000)})
##user  system elapsed
## 86.886   0.313  87.920

saveRDS(carSamples, file = 'carSamples.rds')
```

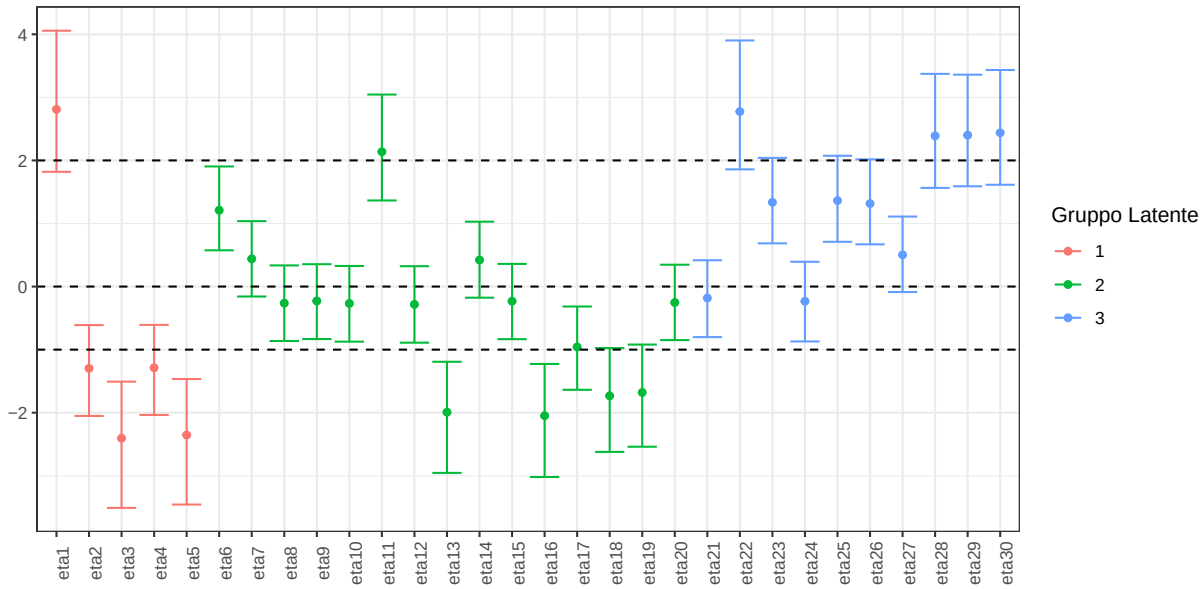
```
carSamples = readRDS('carSamples.rds')
etaSamples = carSamples[, grep("^eta",colnames(carSamples))]


plotData = tibble(id    = factor(paste0('eta',1:30), levels = paste0('eta',1:30)),
                  gr    = dataAndParameters$groups,
                  value = apply(etaSamples,2, median),
                  low   = apply(etaSamples,2, quantile, prob = 0.25),
                  up    = apply(etaSamples,2, quantile, prob = 0.75))
```

Abilità

Il seguente grafico mostra le mediane e i quantili 0.25 e 0.75 per le a posteriori dei parametri di abilità degli $n = 30$ studenti. Le linee tratteggiate corrispondo alle 'vere' medie dei gruppi 1,2,3 partendo dal basso.

```
plotData %>%
    ggplot+
    geom_point(aes(x = id, y = value, col = factor(gr)))+
    geom_errorbar(aes(x = id, ymin = low, ymax = up,col = factor(gr) )) +
    geom_hline(yintercept = dataAndParameters$trueAbilityMean,lty ='dashed')+
    labs(x = '', y = '',col = "Gruppo Latente") +
    theme(axis.text.x = element_text(angle =90))
```



Possibili Direzioni Estensioni

- Abilità non normali (mistura DP etc)
- Abilità dipendenti da più covariate (modello latente sulle abilita con rete + regressione)
- Considerate modelli con più parametri e/o diverso modello per la risposta (per esempio non solo giusto sbagliato ma punteggio)
- 'Invertire' il problema: Considerare un modello che fa clustering per i nodi di una rete e definire un modello di Rasch condizionatamente al cluster.