**IMECC-UNICAMP**
**Atividade 1 - Métodos computacionais**
Professor: Dr. Guilherme Ludwig
Vitor Macedo Rocha RA: 216394

# 1 Distribuições condicionais completas e implementação Gibbs

Considere $\mathbf{p} = [p_1, ..., p_{14}]^\top$, $\mathbf{x} = [n_1, ..., n_{14}, m_1, ..., m_{14}]^\top$ e $\mathbb{B} = \{0, 1, 2, ..., N\}$, temos que a verosimilhança e as distribuições a priori são dadas por

$$\mathcal{L}(N, \mathbf{p}|\mathbf{x}) \propto \frac{N!}{(N-r)!} \prod_{i=1}^{14} p_i^{n_i} (1-p_i)^{N-n_i} \mathbb{I}(N \in \mathbb{N}) \mathbb{I}(p_i \in [0,1]) \mathbb{I}(r \in \mathbb{B}) \tag{1}$$

$$\pi(N) = \frac{e^{-\lambda}\lambda^N}{N!} \mathbb{I}(N \in \mathbb{N}) \tag{2}$$

$$\pi(p_i) = \mathbb{I}(p_i \in [0,1]), i = 1, 2, ..., 14. \tag{3}$$

A seguir vamos derivar a distribuição condicional completa de $N|\mathbf{p}, \mathbf{x}$. Note que $\mathbb{I}(N \in \mathbb{N})\mathbb{I}(p_i \in [0,1])\mathbb{I}(r \in \mathbb{B}) = \mathbb{I}(N \in r, r+1, ...)\mathbb{I}(p_i \in [0,1])$, pois $\mathbb{I}(r \in \mathbb{B}) = \mathbb{I}(N \in \{r, r+1, ...\})$.

$$\begin{aligned}
\pi(N|\mathbf{p}, \mathbf{x}) &\propto \mathcal{L}(N, \mathbf{p}|\mathbf{x})\pi(N) \\
&= \frac{N!}{(N-r)!} \prod_{i=1}^{14} p_i^{n_i}(1-p_i)^{N-n_i} \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \frac{e^{-\lambda}\lambda^N}{N!} \mathbb{I}(N \in \mathbb{N}) \\
&= \frac{e^{-\lambda}\lambda^N}{(N-r)!} \prod_{i=1}^{14} p_i^{n_i}(1-p_i)^{N}(1-p_i)^{-n_i} \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \\
&= \frac{e^{-\lambda}\lambda^N}{(N-r)!} \prod_{i=1}^{14} [p_i^{n_i}] \prod_{i=1}^{14} \left[(1-p_i)^N\right] \prod_{i=1}^{14} \left[(1-p_i)^{-n_i}\right] \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \\
&\propto \frac{e^{-\lambda}\lambda^N}{(N-r)!} \left[\prod_{i=1}^{14}(1-p_i)\right]^N \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \\
&= \frac{e^{-\lambda}\left[\lambda \prod_{i=1}^{14}(1-p_i)\right]^N}{(N-r)!} \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \frac{e^{-\lambda\prod_{i=1}^{14}(1-p_i)}}{e^{-\lambda\prod_{i=1}^{14}(1-p_i)}} \\
&\propto \frac{\exp\{-\lambda \prod_{i=1}^{14}(1-p_i)\}\left[\lambda \prod_{i=1}^{14}(1-p_i)\right]^N}{(N-r)!} \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \\
&\propto \frac{\exp\{-\lambda \prod_{i=1}^{14}(1-p_i)\}\left[\lambda \prod_{i=1}^{14}(1-p_i)\right]^{N-r}}{(N-r)!} \mathbb{I}(N \in \{r, r+1, ...\}) \mathbb{I}(p_i \in [0,1]) \\
&= \frac{\exp\{-\lambda \prod_{i=1}^{14}(1-p_i)\}\left[\lambda \prod_{i=1}^{14}(1-p_i)\right]^{N-r}}{(N-r)!} \mathbb{I}(N - r \in \mathbb{N}) \mathbb{I}(p_i \in [0,1])
\end{aligned}$$

Portanto

$$\pi(N|\mathbf{p}, \mathbf{x}) \propto \frac{\exp\{-\lambda \prod_{i=1}^{14}(1-p_i)\}\left[\lambda \prod_{i=1}^{14}(1-p_i)\right]^{N-r}}{(N-r)!} \mathbb{I}(N - r \in \mathbb{N}) \mathbb{I}(p_i \in [0,1]) \tag{4}$$

Temos que $(N-r)|(\mathbf{p}, \mathbf{x}) \sim Poisson(\lambda \prod_{i=1}^{14}(1-p_i))$, identificado pelo *kernel* apresentado em 4.

Agora vamos derivar a distribuição condicional completa de $p_i|N, \mathbf{x}, i = 1, 2, ..., 14$. Considere $\mathbf{p_{(i)}} = (p_1, p_2, ..., p_{i-1}, p_{i+1}, ..., p_{14}$, isto é, o vetor de parâmetros $\mathbf{p}$ sem o i-ésimo elemento.

$$
\begin{aligned}
\pi(p_i|N, \mathbf{p_{(i)}}, \mathbf{x}) &\propto \mathcal{L}(N, \mathbf{p}|\mathbf{x})\pi(\mathbf{p_{(i)}}) \\
&= \frac{N!}{(N-r)!}\prod_{l=1}^{14} p_i^{n_l}(1-p_l)^{N-n_l}\mathbb{I}(N \in \{r, r+1, ...\})\mathbb{I}(p_l \in [0,1]) \left[\prod_{j\neq i}\mathbb{I}(p_j \in [0,1])\right] \\
&= \frac{N!}{(N-r)!}\prod_{l=1}^{14} p_l^{n_l}(1-p_l)^{N-n_l}\mathbb{I}(N \in \{r, r+1, ...\})\mathbb{I}(p_l \in [0,1]) \\
&\propto p_i^{n_i}(1-p_i)^{N-n_i}\mathbb{I}(p_i \in [0,1]) \\
&= p_i^{(n_i-1)+1}(1-p_i)^{(N-n_i-1)+1}\mathbb{I}(p_i \in [0,1])
\end{aligned}
$$

Podemos então identificar que $p_i|N, \mathbf{x} \sim \text{Beta}(n_i + 1, N - n_i + 1), i = 1, 2, ..., 14$ pelo kernel que apresentado acima. Chegamos a conclusão que

$$
(N-r)|(\mathbf{p}, \mathbf{x}) \sim Poisson(\lambda\prod_{i=1}^{14}(1-p_i)) \tag{5}
$$

$$
p_i|N, \mathbf{x} \sim Beta(n_i + 1, N - n_i + 1), i = 1, 2, ..., 14 \tag{6}
$$

Para avaliar se as distribuições a posteriori, vamos gerar as 4 diferentes cadeias de Markov.

# 2 Implementação ***Hamiltonian Monte Carlo***

Como bem sabemos, o HMC *sampler* não comporta suportes discretos pela fato de que a função que descreve a energia cinética há de ser derivável em relação a "trajetória"da cadeia. Por esse fato, optei por procurar uma solução na própria documentação do stan e acabei essbarrando com alguns modelos de captura-marcação-recaptura como: *one stage Marked-recapture* e *Cormack-Jolly-Seber*.

Minha ideia inicial foi utilizar a relação entre a distribuição exponencial e poisson que vi a partir do proceso de poisson.

# 3 Implementação *Hamiltonian Monte Carlo* no STAN

Vi a possibilidade da marginalização de parâmetros discretos, a seguir são apresentados os cálculos da marginalização de N e a implementação da minha ideia no STAN.

# Apêndice A

## A.1 Implementação Gibbs Sampler

```
library(readr)
library(tidyverse)
dados <- read_table("./Atividade1/resolucao/dados.txt",
                    col_names = FALSE)
colnames(dados) <- c('onca',paste0('armadilha',1:14))
nj <- dados%>%
  pivot_longer(-onca,values_to = "capturada",
               names_to = "armadilha",names_prefix = 'armadilha',
               names_transform = list(armadilha = as.numeric))%>%
  group_by(armadilha)%>%
  summarise(nj=sum(capturada))
mj <- dados%>%
  pivot_longer(-onca,values_to = "capturada",names_to = "armadilha",
               names_prefix = 'armadilha',
               names_transform = list(armadilha = as.numeric))%>%
  group_by(onca)%>%
  mutate(mj=case_when(cumsum(capturada)*capturada>1 ~ 1 , TRUE ~ 0))%>%
  group_by(armadilha)%>%
  summarise(mj=sum(mj))
Ngibbs=10e4
iter0 <- c(rpois(1,50),runif(14))
chain <- matrix(0,ncol = Ngibbs,nrow=length(iter0))
chain[,1] <- iter0
rownames(chain) <- c("N",paste0("p",1:14))
lambda=50
r <- sum(nj$nj)-sum(mj$mj)
nj <- nj$nj
for (i in 2:Ngibbs) {
  lambda_pos <- lambda * prod(1-chain[-1,i-1])
  N_i_plus_1 <- rpois(n=1,lambda = lambda_pos) + r
  chain[1,i] <- N_i_plus_1
  for (j in 1:length(nj)) {
    aux <- rbeta(n = 1,shape1 = nj[j]+1,shape2 = N_i_plus_1-nj[j]+1)
    chain[j+1,i] <- aux
  }
}
```