SME0809 - Inferência Bayesiana - Cadeias de Markov

Francisco Miranda - 4402962

Outubro 2021

```r
library(tidyverse)
library(matrixcalc)
library(knitr)
set.seed(42)
P <- mat.trans <- matrix(c(
                          0.65, 0.15, 0.12,
                          0.28, 0.67, 0.36,
                          0.07, 0.18, 0.52
                           ), 3, 3)
```

Seja a cadeia de Markov com estados $\{1, 2, 3\}$ definida pela seguinte matriz de transição:

$$\mathbb{P} = \begin{bmatrix} 0.65 & 0.28 & 0.07 \\ 0.15 & 0.67 & 0.18 \\ 0.12 & 0.36 & 0.52 \end{bmatrix}$$

Escolhida a condição inicial $X_0 = [0,0102 \quad 0,1009 \quad 0,8889]$, vamos calcular X_t para $t = 1, ..., 10$ utilizando a fórmula de recorrência $X_t = X_{t-1} \cdot \mathbb{P}$.

É possivel ver na tabela abaixo que as distribuições se estabilizam em aproximadamente duas casas decimais para o X_0 escolhido para um burn-in a partir de $t = 6$.

```r
p0 <- c(0.0102, 0.1009, 0.8889)
pi <- p0
p_t <- p0

for (xi in 1:10) {
  pi <- pi %*% P
  p_t <- rbind(p_t, pi)
}
p_t <- cbind(t = 0:10, p_t)

kable(round(p_t, 3),
  col.names = c("t", "Classe Baixa", "Classe Média", "Classe Alta"),
  caption = "Probabilidade de pertencer a classe após uma geração t"
)
```

Table 1: Probabilidade de pertencer a classe após uma geração t

	t	Classe Baixa	Classe Média	Classe Alta
p_t	0	0.010	0.101	0.889
	1	0.128	0.390	0.481
	2	0.200	0.471	0.329
	3	0.240	0.490	0.270
	4	0.262	0.493	0.245
	5	0.274	0.492	0.235
	6	0.280	0.491	0.230
	7	0.283	0.490	0.227
	8	0.285	0.489	0.226
	9	0.286	0.489	0.226
	10	0.286	0.489	0.225

Nota: Há uma outra forma de calcular X_t que não requer que se conheça X_{t-1}, elevando a matriz P a potencia de N, ou simplesmente:

$$X_N = X_0 \cdot \mathbb{P}^N$$

Exemplo: Calculando X_{10}:

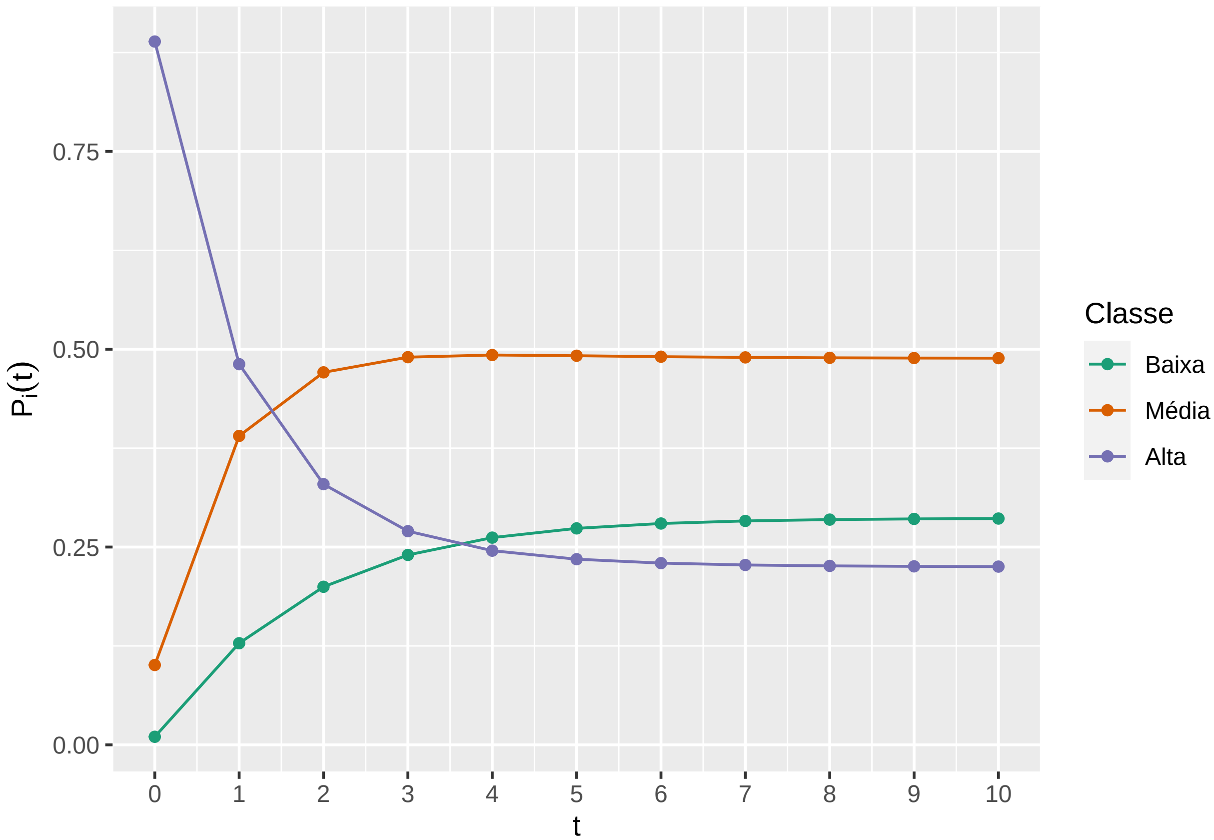
```
N <- 10
round(p0 %*% matrix.power(P, N), 3)
```

```
##          1     2     3
## [1,] 0.286 0.489 0.225
```

Gráfico de t em função de $P_i(t)$

Para $X_0 = [0,0102 \;\; 0,1009 \;\; 0,8889]$ temos o seguinte gráfico:

```
as_tibble(p_t) %>%
  pivot_longer(cols = !"t") %>%
  ggplot(aes(x = t)) +
  geom_line(aes(y = value, color = name)) +
  geom_point(aes(y = value, color = name)) +
  labs(
    y = expression(P[i](t)),
    color = "Classe"
  ) +
  scale_x_continuous(breaks = 0:10) +
  scale_color_brewer(palette = "Dark2", labels = c("Baixa", "Média", "Alta"))
```



Simulação para diferentes valores de X_0

A fim de verificar computacionamente a convergência da distribuição de equilíbrio para diferentes valores iniciais, geramos aleatoriamente triplas (x, y, z) a partir de uma Uniforme(0,1) e procedemos da seguinte forma para gerar os valores $X_0 = (p_1, p_2, p_3)$:

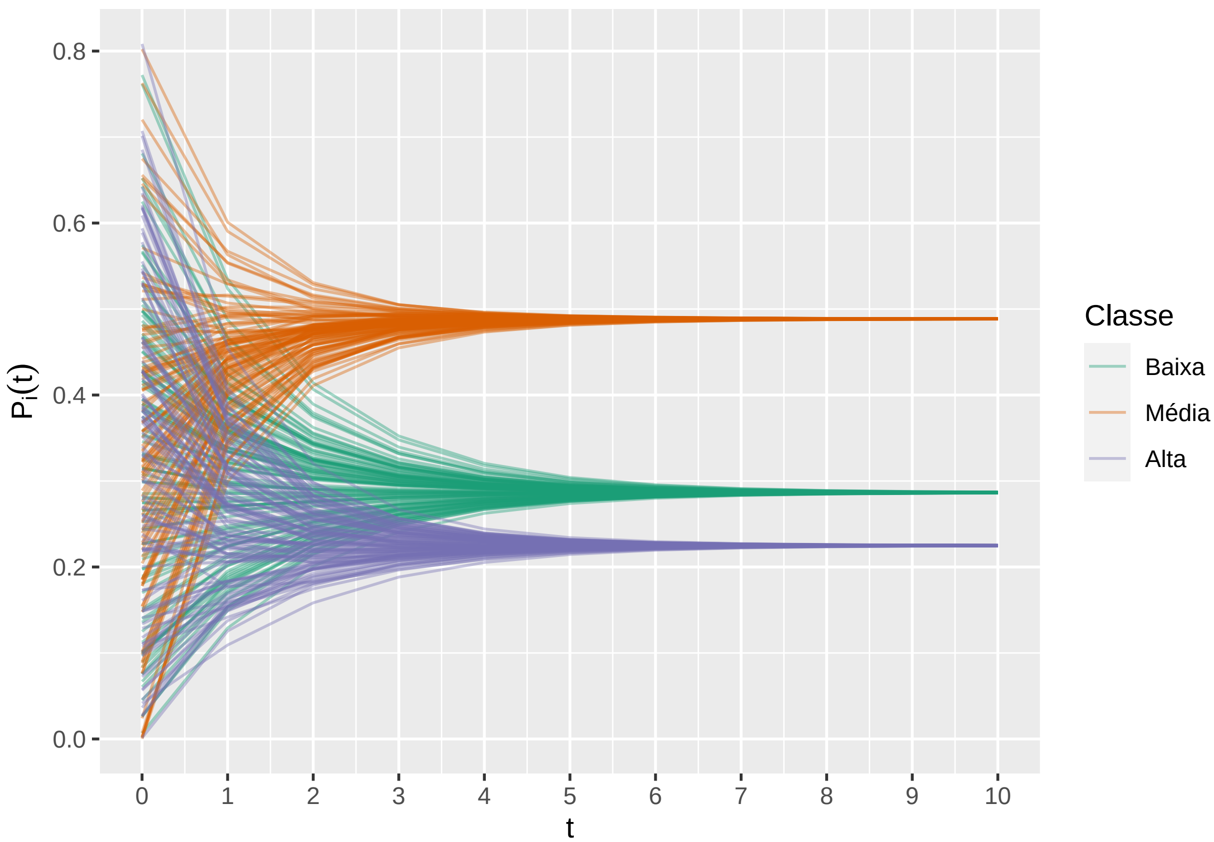
$$c(x + y + z) = 1 \Rightarrow c = \frac{1}{x + y + z}, x + y + z > 0.$$

$$X_0 = (p_1, p_2, p_3) = (cx, cy, cz)$$

```
# gera n amostras de triplas
gera_amostras <- function(n) {
  # gera uma tripla com soma 1 a partir de uma uniforme
  smp <- function() {
    r <- tibble(x = t(runif(3)))
    1 / sum(r) * r
  }
  1:n %>% map_dfr(~ tibble(i = .x, t = 0, smp()))
}

X_t <- gera_amostras(100)
# calculando X_t para i de 1 a 10 nos diferentes valores de x0
for (i in 1:nrow(X_t)) {
  pi <- as.numeric(X_t[i, 3:5])
  p_t <- pi
```

```r
  for (j in 1:10) {
    pi <- pi %*% P
    X_t <- rbind(X_t, c(i, j, pi))
  }
}
# agrupa as variáveis por t,i e plota o gráfico
X_t %>%
  pivot_longer(cols = !c("t", "i")) %>%
  ggplot(aes(x = t)) +
  geom_line(aes(y = value, color = name, colour = as.factor(i)), alpha = 0.4) +
  labs(
    y = expression(P[i](t)),
    color = "Classe"
  ) +
  scale_x_continuous(breaks = 0:10) +
  scale_color_brewer(palette = "Dark2", labels = c("Baixa", "Média", "Alta"))
```



A teoria nos garante que podemos aproximar-nos da distribuição de equilíbrio o tanto quanto quisermos, conforme aumentamos o valor de t. É possivel observar no gráfico acima que os 100 diferentes valores de X_0 gerados da forma descrita anteriormente aparentemente convergem em $t = 1, ..., 10$.