Regressão Linear Bayesiana

Rener Oliveira

Sumário

Regressão Linear Bayesiana

Rener Oliveira
19 de junho de 2021

1 Introdução

A regressão linear é um modelo muito utilizado em diversas áreas de aplicações. Meu intuito com este trabalho era estudar a regressão linear sobre o ponto de vista bayesiano. Inicio com uma reapresentação dos métodos de estimação de coeficientes frequentista.

A inspiração surgiu na leitura de um texto do blog do PyMC3[3], biblioteca Python de análise bayesiana. O texto era um simples tutorial de como usar as ferramentas da biblioteca para fazer um regressão linear bayesiana simples (1 preditor) posterioris dos coeficientes via MCMC. O texto entretanto não justifica com clareza a escolha das formas funcionais das prioris. Decidi então buscar materiais mais teóricos sobre o assunto, e encontrei um capítulo de um livro[1], que estebejecia resultados de uma análise conjugada dos parâmetros.

Na seção seguinte após a revisão do método frequentista, inicio a discussão sobre regressão bayesiana, estabalecendo de forma simplificada os resultados de [1]. O produto final dessa seção teórica são fórmulas analíticas para parâmetros da distribuição à posteriori, partindo de um priori conjugada.

Depois disso faço uma simulação computacional similar à de [3], porém em maior dimensão, já que os resultados teóricos formalizados permitem esse tipo de extensão. São gerados dados sintéticos com parâmetros fixos para avaliar a qualidade das estimativa.

Com a análise conjudada realizada, fiz uma implementação que gerava distribuições à posteriori amostrando das distribuições conjugadas obtidas analiticamente. E no final utilizei o MCMC para amostrar posterioris via Cadeia de Markov, utilizando as ferramentas da biblioteca PyMC3.

Um trabalho futuro seria aplicar esses algoritmos em dados reais, porém neste texto me contentei com dados sintéticos, pois o intuito era apenas estudar a teoria bayesiana de regressões, e melhorar as implementações do blog[3].

Durante o corpo do texto, deixo explícito alguns trechos de códigos python utilizado, porém todo o desenvolvimento computacional deste trabalho pode ser consultado neste link do Google Colab, uma plataforma online de notebooks Jupyter.

2 O Modelo Frequentista

A regressão linear é um modelo que visa explicar a relação entre uma variável reposta real $\boldsymbol{y}$ e outras variáveis explicativas $X_1, X_2, \ldots, X_p$ através de algumas premissas. No caso cada X_j é um vetor coluna de n observações $X_j = [x_{1j}, x_{2j}, \ldots, x_{nj}]^T$, assim como a variável resposta: $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$.

A principal é a linearidade da esperança de $\boldsymbol{y}$ em relação aos predidores, isto é, existem $\beta_0, \beta_1, \ldots, \beta_p$ tais que

$$E[\boldsymbol{y}|\boldsymbol{X}] = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

Onde $\boldsymbol{X}$ é uma matrix $n \times (p+1)$, com n sendo a quantidade de observações e p a quantidade de preditores. O número de colunas p+1, se dá por conta da inclusão de uma coluna constante igual a 1, para que possamos representar a expressão acima de forma matricial (*design matrix*).

$$\boldsymbol{X} = \begin{bmatrix} 1 & \vdots & \vdots & \ldots & \vdots \\ \vdots & X_1 & X_2 & \ldots & X_n \\ 1 & \vdots & \vdots & \ldots & \vdots \end{bmatrix}$$

Sendo $\beta = [\beta_0, \beta_1, \ldots, \beta_p]^T$ vetor coluna $(p+1) \times 1$, temos que:

$$E[\boldsymbol{y}|\boldsymbol{X}] = \boldsymbol{X}\beta$$

Um outra hipótese importante é a de normalidade, ou seja,

$$\boldsymbol{y} = \boldsymbol{X}\beta + \varepsilon,$$

onde o vetor de ruídos $\varepsilon = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n]^T$ com cada $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$.

Disso segue a hipótese de homocedasticidade, ou seja, variância comum entre as observações.

No modelo linear frequentista, se assume que os coeficientes β são valores fixos. Assim, a estimação ocorre por minimização dos erros quadráticos, ou seja, o estimador $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p]^T$ é tal que minimiza a forma quadrática de perda:

$$RSS = (\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta),$$

onde RSS é uma sigla em inglês para soma dos resíduos ao quadrado (*residual sum of squares*).

Segundo [2] o vetor que minimiza o erro acima é dados por:

$$\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{1}$$

3 O Modelo Bayesiano

No paradigma bayesiano, os coeficientes β_i, com $i = 0, 1, \ldots, p$ deixam de ser considerados como valores fixos e passam a ser variáveis aleatórias. A hipótese de normalidade (iid) se mantém, ou seja, se $\boldsymbol{x}_i$ é a i-ésima linha da matrix $\boldsymbol{X}$.

$$y_i|\boldsymbol{x}_i, \beta, \sigma^2 \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{x}_i\beta, \sigma^2) \tag{2}$$

Nesta seção vamos mostrar os passos da derivação de uma família de distribuições à priori conjugadas para σ^2 e para $\beta|\sigma^2$. O objetivo será obter uma forma funcional para distribuição à posteriori conjunta dos parâmetros desconhecidos $p(\beta, \sigma^2|\boldsymbol{X}, \boldsymbol{y})$. Desta forma é possível gerar distribuições para novos dados y não previamente observados, ao invés de um valor fixo, como no caso frequentista.

A verossimilhança condicional $p(\boldsymbol{y}|\boldsymbol{X}, \beta, \sigma^2)$, pela distribuição por (2) será o produtório:

$$\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{X}, \beta, \sigma^2) &= \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \beta, \sigma^2) \\
&\propto \prod_{i=1}^{n} (\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i\beta)^2\right) \\
&= (\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2}\sum_{i}^{n}(y_i - \boldsymbol{x}_i\beta)^2\right) \\
&= (\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta)\right)
\end{aligned}$$

Podemos expandir a forma quadrática da exponencial, considerando $\hat{\beta}$ como o estimador frequentista dos coeficientes dado pela expressão (1).

$$\begin{aligned}
(\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta) &= (\boldsymbol{y}^T - \beta^T\boldsymbol{X}^T)(\boldsymbol{y} - \boldsymbol{X}\beta) \\
&= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}\beta - \beta^T\boldsymbol{X}^T\boldsymbol{y} + \beta^T\boldsymbol{X}^T\boldsymbol{X}\beta \\
&\quad \text{Completando o quadrado } \beta^T\boldsymbol{X}^T\boldsymbol{X}\beta\text{:} \\
&= (\beta - \hat{\beta})^T(\boldsymbol{X}^T\boldsymbol{X})(\beta - \hat{\beta}) + \beta^T\boldsymbol{X}^T\boldsymbol{X}\hat{\beta} + \hat{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\beta - \hat{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\beta} \\
&\quad + \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}\beta - \beta^T\boldsymbol{X}\boldsymbol{y} \\
&\quad \text{Com alguns passos de Álgebra Linear usando a fórmula 1, chegamos em:} \\
&\quad \hat{\beta}^T = \boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}
\end{aligned}$$

Substituindo na expressão acima, e fazendo mais algumas simplificações, chegamos em:

$$(\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta) = (\boldsymbol{y} - \boldsymbol{X}\hat{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\beta}) + (\beta - \hat{\beta})^T(\boldsymbol{X}^T\boldsymbol{X})(\beta - \hat{\beta}),$$

Sendo assim, podemos fatorizar a verossimilhança de uma forma conveniente como abaixo:

$$p(\boldsymbol{y}|\boldsymbol{X},\beta,\sigma^2) \propto (\sigma^2)^{-v/2}\exp\left(-\frac{vs^2}{2\sigma^2}\right)(\sigma^2)^{-\frac{n-v}{2}}\exp\left(-\frac{1}{2\sigma^2}(\beta-\hat{\beta})^T(\boldsymbol{X}^T\boldsymbol{X})(\beta-\hat{\beta})\right), \quad (3)$$

com $vs^2 = (\boldsymbol{y}-\boldsymbol{X}\hat{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\hat{\beta})$ e $v = n-(p+1)$.

A forma funcional acima é a multiplicação do núcleo da densidade de uma Gama Inversa com o núcleo de uma Normal. Isso sugere a elicitação de prioris dessas distribuições para $p(\beta,\sigma^2) = p(\sigma^2)p(\beta|\sigma^2)$.

As prioris conjugadas serão então:

$$\sigma^2 \sim \text{Inv-Gamma}(a_0, b_0) \quad (4)$$
$$\beta|\sigma^2 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma^2\boldsymbol{\Lambda}_0^{-1}), \quad (5)$$

onde $a_0 = \frac{v_0}{2}$, $b_0 = \frac{v_0 s_0^2}{2}$, $\boldsymbol{\mu}_0$ é a média à priori para β e $\boldsymbol{\Lambda}_0$ é uma matriz diagonal de precisão à priori. Nesse caso v_0 e s_0^2 podem ser interpretados como valores à priori para os graus de liberdade e para a variancia amostral, dado que a média da Gama Inversa é $\frac{b_0}{a_0-1} = \frac{v_0 s_0^2}{v_0-2}$.

A matriz de precisão é interpretada como o quão certo o modelador está em relação à sua priori $\boldsymbol{\mu}_0$.

As densidades serão:

$$p(\sigma^2) \propto (\sigma^2)^{-\frac{v_0}{2}-1}\exp\left(-\frac{v_0 s_0^2}{2\sigma^2}\right)$$
$$p(\beta|\sigma^2) \propto (\sigma^2)^{-\frac{p+1}{2}}\exp\left(-\frac{1}{2\sigma^2}(\beta-\boldsymbol{\mu}_0)^T\boldsymbol{\Lambda}_0(\beta-\boldsymbol{\mu}_0)\right)$$

Unindo as prioris acima com a verossimilhança em (3) e fazendo uma reorganização dos termos que será omitida aqui, é possível chegar em:

$$\begin{aligned} p(\beta,\sigma^2|\boldsymbol{y},\boldsymbol{X}) &\propto p(\boldsymbol{y}|\boldsymbol{X},\beta,\sigma^2)p(\beta|\sigma^2)p(\sigma^2) \\ &\propto (\sigma^2)^{-\frac{p+1}{2}}\exp\left(-\frac{1}{2\sigma^2}(\beta-\boldsymbol{\mu}_n)^T(\boldsymbol{X}^T\boldsymbol{X}+\boldsymbol{\Lambda}_0)(\beta-\boldsymbol{\mu}_n)\right) \\ &\times (\sigma^2)^{-\frac{n+2a_0}{2}-1}\exp\left(-\frac{2b_0\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{\mu}_n^T(\boldsymbol{X}^T\boldsymbol{X}+\boldsymbol{\Lambda}_0)\boldsymbol{\mu}_n + \boldsymbol{\mu}_0^T\boldsymbol{\lambda}_0\boldsymbol{\mu}_0}{2\sigma^2}\right), \end{aligned}$$

com $\boldsymbol{\mu}_n = (\boldsymbol{X}^T\boldsymbol{X}+\boldsymbol{\Lambda}_0)^{-1}(\boldsymbol{X}^T\boldsymbol{X}\hat{\beta}+\boldsymbol{\Lambda}\boldsymbol{\mu}_0)$.

A posteriori pode ser escrita então como o produto das posterioris de $\beta|\sigma^2$ e de σ^2, onde

$$\sigma^2|\boldsymbol{y},\boldsymbol{X} \sim \text{Inv-Gamma}(a_n, b_n)$$
$$\beta|\sigma^2,\boldsymbol{y},\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}_n, \sigma^2\boldsymbol{\Lambda}_n^{-1}),$$

na qual os novos parâmetros são:

$$\begin{aligned}
\boldsymbol{\mu}_n &= (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Lambda}_0)^{-1}(\boldsymbol{X}^T\boldsymbol{X}\hat{\beta} + \boldsymbol{\Lambda}_0\boldsymbol{\mu}_0) \\
\boldsymbol{\Lambda}_n &= \boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Lambda}_0 \\
a_n &= a_0 + \frac{n}{2} \\
b_n &= b_0 + \frac{\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{\mu}_n^T\boldsymbol{\Lambda}_n\boldsymbol{\mu}_n + \boldsymbol{\mu}_0^T\boldsymbol{\Lambda}_0\boldsymbol{\mu}_0}{2}
\end{aligned} \tag{6}$$

De fato, tal distribuição conjunta de (β, σ^2) recebe o nome de Normal-Gamma-Inversa, e é uma família conjugada, como demonstrado com mais detalhes em [1].

4 Simulações Computacionais

4.1 Geração de dados

Nesta seção vamos gerar distribuições à posteriori para (β, σ^2) de forma numérica utilizando MCMC e o amostrador NUTS, implementado no pacote PyMC3 da liguagem Python.

A referência [3], do próprio site da biblioteca PyMC3, para exemplificar suas funcionalidades de modelos lineares bayesianos, faz uma simulação utilizando dados sintéticos normalmente distribuídos, porém é utilizada uma formulação com apenas um preditor, e uma distribuição diferente da que derivamos anteriormente para $p(\sigma^2)$. Vamos reproduzir o exemplo do site utilizando dados gerados sinteticamente a partir da seguinte lei:

$$\begin{aligned}
Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon \\
\varepsilon &\sim \mathcal{N}\left(0, \sigma^2 \boldsymbol{I}\right)
\end{aligned}$$

Com variância $\sigma^2 = 4$, vetor de coeficientes escolhido como $\beta = [-1, 3, 6]^T$. Além disso, usamos $n = 350$ pontos de dados.

No caso, X_1 é um vetor de n pontos igualmente espaçados em $[0, 1]$ e X_2 é gerado à de uma distribuição uniforme em $[0, 1]$.

Na Figura 1 foi plotado um plano laranja que é a equação puramente linear, e 350 pontos azuis baseando-se na adição do erro ε aos pontos $\boldsymbol{X}\beta$

Abaixo segue o código de geração dos dados.

```
n = 350

true_beta = [-1,3,6]
true_sigma2 = 2**2
X1 = np.linspace(0,1,n)
X2 = np.random.uniform(size = n)

# y = beta0+beta1*x1+beta2*x2
```

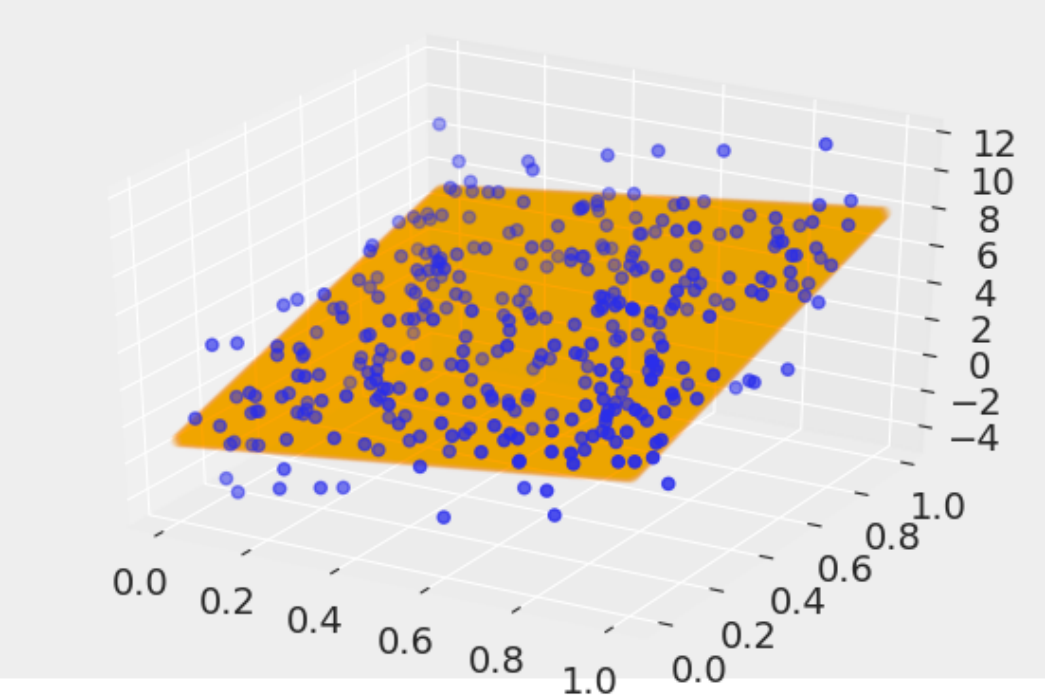

Figura 1: Laranja: Plano base de geração dos pontos; Azul: Plano + ε

```
true_regression_plane = true_beta[0] + true_beta[1] * X1 + true_beta[2]*X2

# Adding Noise with variance true_sigma2
y = true_regression_plane + np.random.normal(scale=true_sigma2**(1/2), size=n)

```

4.2 Estimação Frequentista

Utilizando o pacote statsmodels, podemos estimar uma regressão linear frequentista, nos dando estimadores pontuais para os coeficientes obtidos por minimização quadrática, ou equivalentemente neste caso, maximização da merossimilhança.

Segue abaixo o relatório da regressão:

Dep. Variable:	y	**R-squared:**	0.496
Model:	OLS	**Adj. R-squared:**	0.493
Method:	Least Squares	**F-statistic:**	170.7
Date:	Tue, 13 Apr 2021	**Prob (F-statistic):**	2.38e-52
Time:	21:08:53	**Log-Likelihood:**	-739.93
No. Observations:	350	**AIC:**	1486.
Df Residuals:	347	**BIC:**	1497.
Df Model:	2		

	coef	std err	t	P> |t|	[0.025	0.975]
const	-1.2781	0.288	-4.434	0.000	-1.845	-0.711
x1	2.5910	0.372	6.963	0.000	1.859	3.323
x2	6.4928	0.372	17.453	0.000	5.761	7.225

Como os dados foram gerados de forma sintética obedecendo as principais premissas de Regressão Linear, obtivemos um resultado satisfatório com a estimação via OLS[1]. A estimação foi $\hat{\beta} = [-1.2781, 2.5910, 6.4928]^T$ razoavelmente próximo do verdadeiro β. Perceba que os testes de hipótese Student-t apontam estimadores estatisticamente significantes, O teste F também deu um bom resultado. Além disso os intervalos de confiança 95% cobrem os valores reais dos parâmetros; O comprimento de tais intervalos depende da variância intrínseca dos dados, que foi 4 no caso.

4.3 Estimação Bayesiana - Implementação Analítica

Como derivamos formas funcionais explícitas para a distribuição à posteriori, podemos fazer uma implementação de amostragem de posterioris simplesmente simulando pontos dessa distribuição.

Utilizaremos os resultados conjugados obtidos nas seções anteriores, com prioris não informativas.

Para $p(\sigma^2)$, pela distribuição em 4, escolhemos $s_0^2 = 20$ e $v_0 = 3$ um valores que geram $a_0 = 3/2$ e $b_0 = 30$, e consequentemente $E[\sigma^2] = \frac{b_0}{a_0 - 1} = 60$ um valor bem alto pra situação, de forma proposital.

Segue na Figura (2) a densidade de probabilidade à priori de σ^2 com os parâmetros escolhidos.

Veja abaixo a codificação dos hyperparâmetros à priori:

```
# Prior parameters
v0 = 3
s02 = 20
a0 = v0/2
b0 = v0*s02/2
mu0 = np.zeros(3)
Lambda0 = 0.01*np.eye(3)

```

A atualização dos parâmetros conjugados é regida pelas fórmulas em 6. Abaixo segue o código de um função em python que recebe os parâmetros à priori $(a_0, b_0, \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0)$. O operador @ representa multiplicação matricial na sintaxe NumPy. Perceba que as fórmulas de $\hat{\beta}$ em 1, e a fórmula de $\boldsymbol{\mu}_n$ envolvem o cálcul ode matriz inversa, porém por questões de eficiência, não calculamos essa inversa diretamente. Ao invés disso, é feita a resolução de um sistema linear equivalente.

[1] Ordinary Least Squares

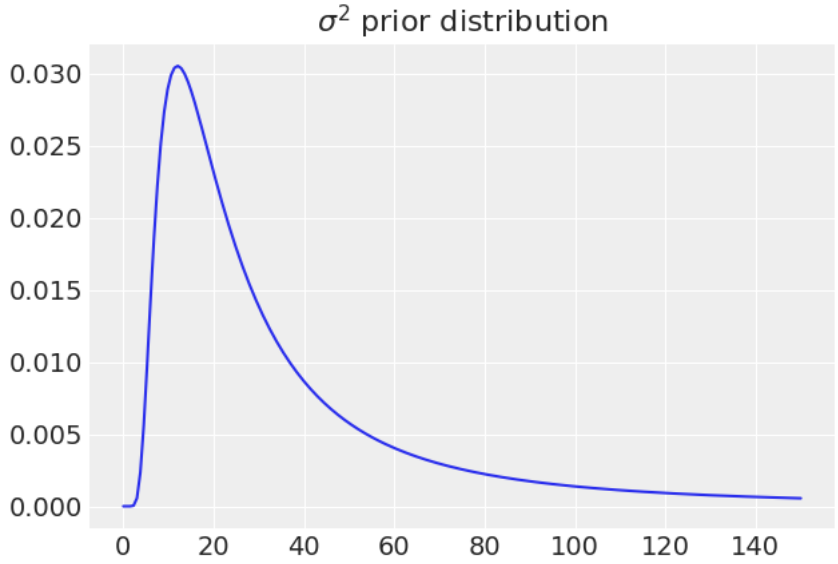

Figura 2: Função de Densidade à priori de $p(\sigma^2)$

```
def update_parameters(X,y,a0,b0,mu0,Lambda0):
        beta_hat = np.linalg.solve(X.T@X,X.T@y)
        mu_n = np.linalg.solve(X.T@X+Lambda0,X.T@X@beta_hat+Lambda0@mu0)
        Lambda_n = X.T@X+Lambda0
        a_n = a0+n/2
        b_n = b0 + (y.T@y-mu_n.T@Lambda_n@mu_n+mu0.T@Lambda0@mu0)/2
        return dict(a_n=a_n,
                    b_n=b_n,
                    mu_n=mu_n,
                    Lambda_n=Lambda_n)

```

Executando o código acima com as prioris definidas anteriormente, obtemos os seguintes parâmetros à posteriori:

$$
\begin{aligned}
a_n &= 176.5 \\
b_n &\approx 733.08 \\
\boldsymbol{\mu}_n &\approx [-1.2762, 2.5898, 6.4903]^T \\
\boldsymbol{\Lambda}_n &\approx \begin{bmatrix} 350.01 & 175 & 171.882 \\ 175 & 116.8438 & 84.4265 \\ 171.882 & 84.4265 & 113.7668 \end{bmatrix}
\end{aligned}
$$

Perceba que a média à posteriori $\boldsymbol{\mu}_n$ de β é muito próxima dos estimadores de máxima verossimilhança encontrados anteriormente. Veja também que os valores da matriz de precisão estão bem alto em contraste com a precisão imprópria $\boldsymbol{\Lambda}_0$.

Para interpretar os parâmetros de (a_n, b_n) de forma mais clara, vejamos na Figura (3) um plot da densidade da distribuição $p(\sigma^2|\boldsymbol{y}, \boldsymbol{X}) \sim \text{Inv-Gamma}(a_n, b_n)$.

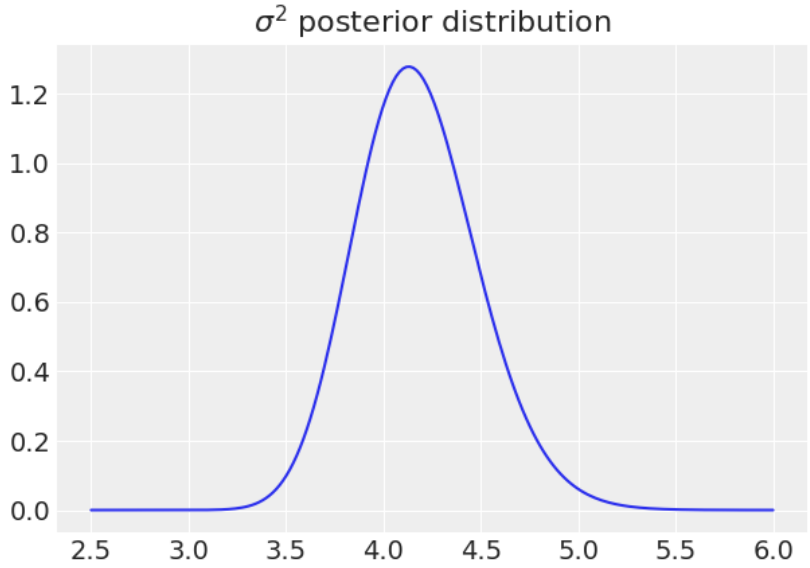

Figura 3: Função de Densidade à posteriori de $p(\sigma^2|\boldsymbol{y}, \boldsymbol{X})$

Perceba que mesmo com uma priori bem dispersa, a distribuição posterior está muito melhor definida, e sua média aproxima 4 que é o seu valor real.

O processo de geração de amostras à posteriori é muito simples. No caso de σ^2, geramos variáveis aleatórias Inv-Gamma(a_n, b_n), utilizando o pacote SciPy. Para amostrar da distribuição condicional de β, primeiro geramos um σ^2 pela Gama-Inversa, e depois utilizando o σ^2 gerado para amostrar da distribuição $\mathcal{N}(\boldsymbol{\mu}_n, \sigma^2\boldsymbol{\Lambda}_n^{-1})$.

Segue abaixo o código:

```
def posterior_invgamma(a_n,b_n,size=200):
  return invgamma.rvs(a=a_n, scale=b_n,size=size)
def posterior_normal(mu_n,Lambda_n,sigma2):
  sample = []
  cov = np.linalg.inv(Lambda_n)
  for s2 in sigma2:
  sample.append(multivariate_normal.rvs(mean=mu_n,cov=s2*cov))
  return np.array(sample)
posterior = update_parameters(X,y,a0,b0,mu0,Lambda0)
a_n,b_n,mu_n,Lambda_n = posterior.values()
sigma2 = posterior_invgamma(a_n,b_n,size=5000)
beta = posterior_normal(mu_n,Lambda_n,sigma2)

```

Como podemos ver no código geramos uma amostra de 5mil pontos. Pelos valores dos parâmetros $\boldsymbol{\mu}_n$ e a densidade de σ^2 à posteriri, vimos que o processo aproximou bem os valores reais dos parâmetros.

Na Figura 4 vemos a distribuição desses parâmetros.

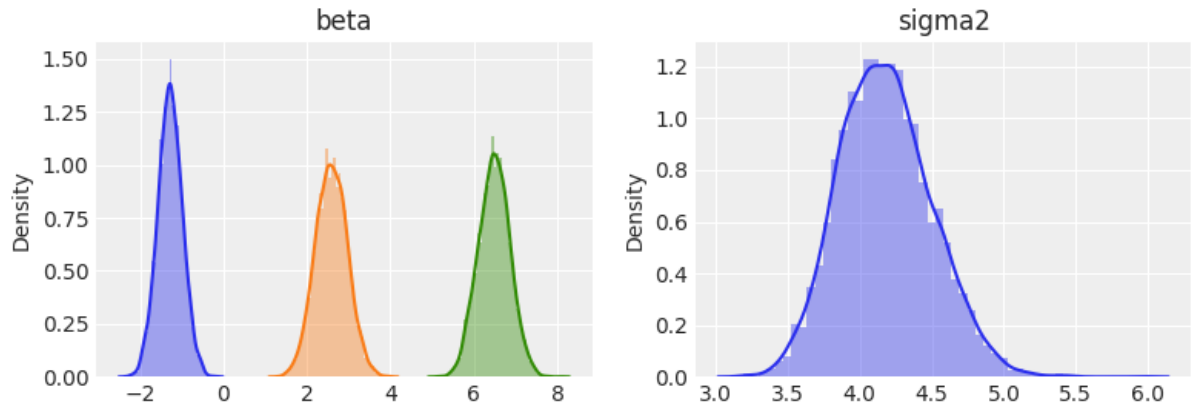

Figura 4: Legenda da figura esquerda: β_0 - Azul; β_1-Lanranja; β_2-Verde

4.4 Estimação Bayesiana - MCMC

Utilizando MCMC podemos amostrar numericamente uma distribuição à posteriori para os parâmetros.

Para $\boldsymbol{\mu}_0$ e $\boldsymbol{\Lambda}_0$ da densidade (5) de $\beta|\sigma^2$, utilizamos $\boldsymbol{\mu}_0 = [0,0,0]^T$ e $\boldsymbol{\Lambda}_0 = 0.01\boldsymbol{I}$ indicando baixa precisão sobre a média nula dada à priori.

Segue a codificação no PyMC3 utilizando amostrador NUTS, que é o padrão do programa. utilizamos 4 cadeias, com 5000 iterações e 1000 iterações *tuning*.

```
# MCMC estimation
with Model() as model:
        # Priors
        sigma2 = InverseGamma("sigma2",alpha=a0,beta=b0)
        beta = MvNormal("beta",mu=mu0,cov=sigma2*pm.math.matrix_inverse(Lambda0),shape=(3))
        # Likelihood
        likelihood = Normal("y", mu=beta[0] + beta[1] * X1 + beta[2]*X2, sigma=sigma2**(1/2)
    , observed=y)
        trace = sample(5000, cores=4,tune=1000)  # NUTS sampler

```

Segue na Figura (5) o gráfico do traço das cadeias e na Figura (6) a distribuição à posteriori dos parâmetros gerada pela biblioteca visualição ArviZ[4].

Visualmente podemos ver que a moda de σ^2 está próxima de 4, β_0 (verde) próxima de -1, β_1 (laranja) perto de 3, e a moda de β_2 (verde) próxima de 6. Vemos então que as distribuições aproximaram bem os parâmetros reais. Segue abaixo as estatísticas das cadias de Markov geradas:

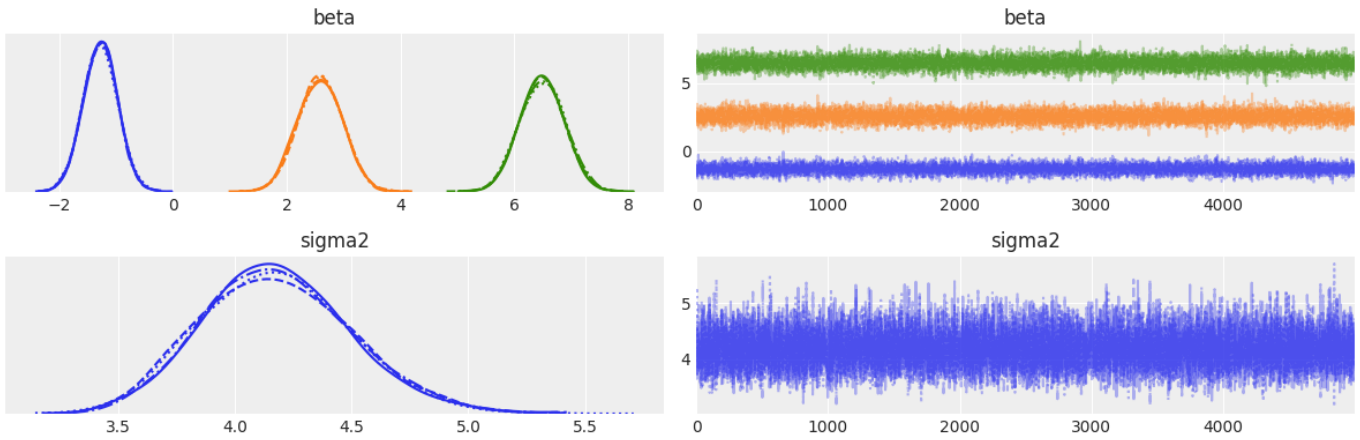

Figura 5: Plot das Cadeias de Markov

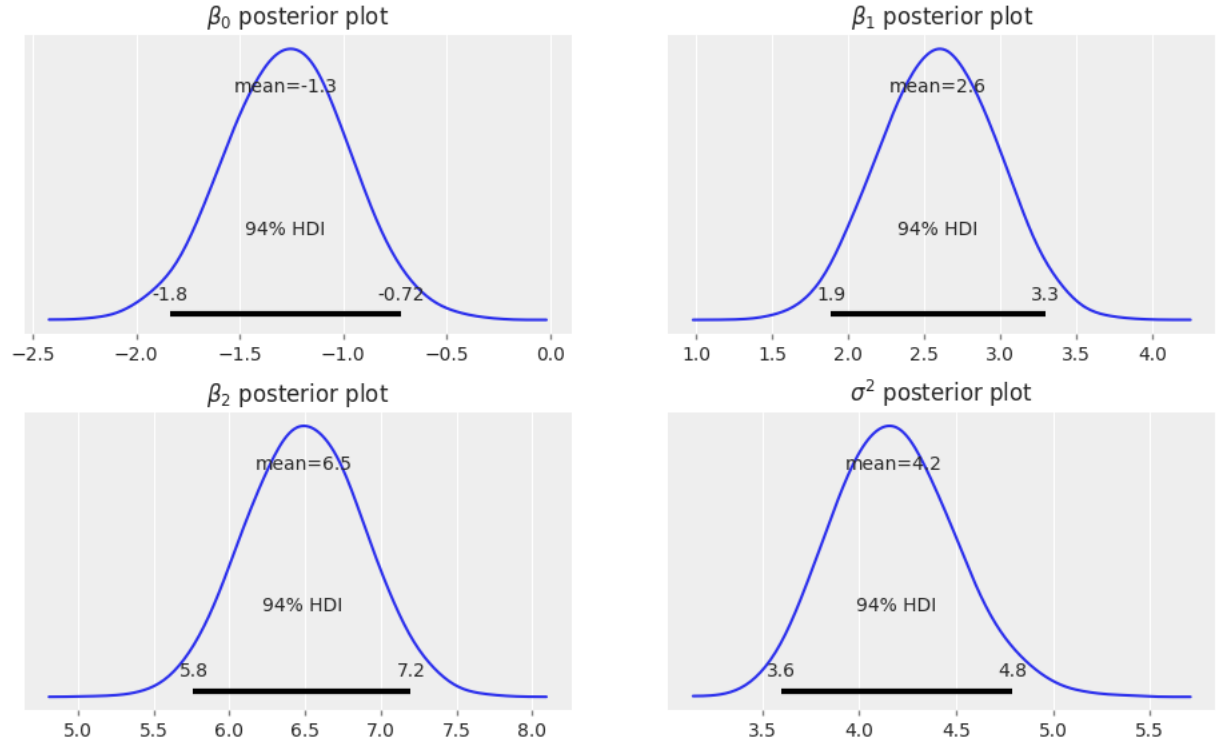

Figura 6: Distribuição à posteriori dos parâmetros

	mean	sd	hdi_3%	hdi_97%	ess_sd	ess_bulk	ess_tail	r_hat
beta[0]	-1.28	0.30	-1.84	-0.72	8569.0	8590.0	10503.0	1.0
beta[1]	2.59	0.38	1.88	3.30	9832.0	9841.0	11369.0	1.0
beta[2]	6.49	0.38	5.76	7.20	10736.0	10729.0	11958.0	1.0
sigma2	4.18	0.32	3.60	4.79	14823.0	15045.0	12363.0	1.0

Podemos ver que as médias à posteriori se aproximam bastante dos os estimadores frequentistas de β e também dos parâmetros encontrados analiticamente. A primeira vantagem do modelo bayesiano é que agora temos informação sobre σ^2, que não era modelado anteriormente. Acima vemos que sua média fica próxima de 4 com um HDI 95% de $[3.48, 4.66]$. Um segunda vantagem dessa modelagem, é que com uma distribuição à posteriori podemos responder perguntas como $\mathbb{P}(\beta_i \in I | \boldsymbol{y}, \boldsymbol{X})$ para algum intervalo qualquer I.

Uma outra análise diagnóstica interessante a ser feita, é através dos plots de autocorrelação, na Figura (7).

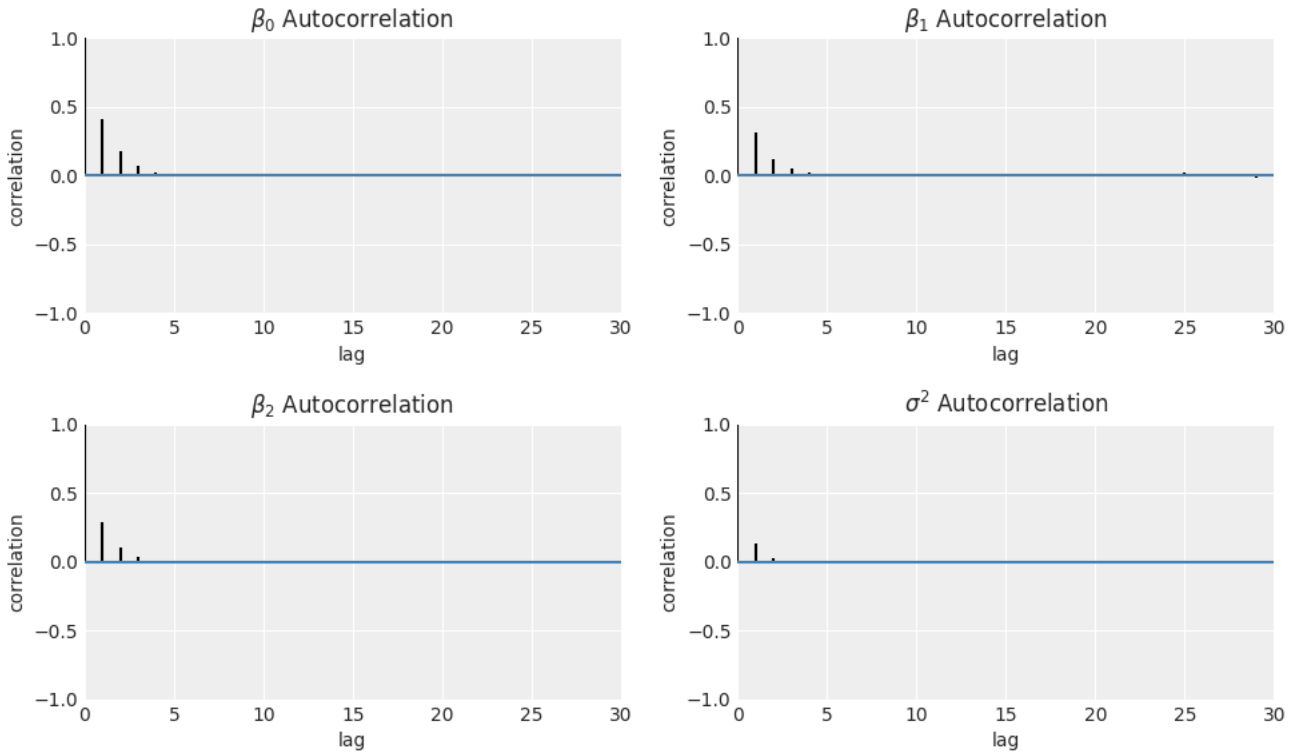

Figura 7: Plot de Autocorrelação

Vemos que após poucos lags, a autocorrelação de todos os parâmetros vai pra zero, o que indica que as cadeias conseguiram convergir relativamente rápido, pois se tivéssesmos auta correlação por muitos lags, isso indicaria que o amostrador não fez uma visita satisfatória ao espaço de parâmetros, ficando "preso" em valores autocorrelatos.

Referências

[1] A. O'Hagan e M.G. Kendall. "Kendall's Advanced Theory of Statistics: Bayesian Inference. Volume 2B". Em: v. 2,pt. 2. Edward Arnold, 1994, pp. 244–250. ISBN: 9780340529225.

[2] M.H. DeGroot e M.J. Schervish. *Probability and Statistics, 4th ed.* Addison-Wesley, 2012. ISBN: 9780321500465.

[3] Thomas Wiecki. *GLM: Linear regression, Bayesian GLMs in PyMC3*. https://docs.pymc.io/notebooks/GLM-linear.html. PyMC Development Team. 2018.

[4] Ravin Kumar et al. "ArviZ a unified library for exploratory analysis of Bayesian models in Python". Em: *Journal of Open Source Software* 4.33 (2019), p. 1143. DOI: 10.21105/joss.01143. URL: https://doi.org/10.21105/joss.01143.