SME0809 - Inferência Bayesiana

Grupo 16:

Alvaro Valentim P M Bandeira - 10392150
Lua Nardi Quito - 11371270

October 19, 2021

Primeira Avaliação

Nosso objetivo nessa atividade é realizar uma curta análise Bayesiana de dados de uma distribuição Normal com Variância conhecida, simulando amostras com tamanhos diferentes,plotando os gráficos e exibindo os resumos para facilitar a análise, fazendo estudo de uma *priori* informativa e uma não informativa e por fim analisar esses conceitos aplicados a dados reais.

Índice - Primeira Avaliação

Página

1 Análise Bayesiana **3**

1.1 Posteriori com variância conhecida e priori informativa 3

1.2 *Priori* não informativa de Jeffreys com variância conhecida 4

1.3 Simulações 4

1.3.1 Para amostra de tamanho $n = 15$ 6

1.3.2 Para amostra de tamanho $n = 30$ 8

1.3.3 Para amostra de tamanho $n = 1000$ 9

1.3.4 Conclusão 11

1.4 Dados reais 12

1.4.1 Apresentação dos dados 12

1.4.2 Teste de Normalidade dos dados 14

1.4.3 Simulação Bayesiana 14

1 Análise Bayesiana

1.1 Posteriori com variância conhecida e priori informativa

Supondo que nossos dados seguem uma distribuição normal com variância conhecida e média θ, $X|\theta \sim N(\theta, \sigma^2)$, e nossa *priori* também segue uma distribuição normal $\pi(\theta) \sim N(\mu_0, \sigma_0^2)$, queremos descobrir a *posteriori* $\pi(\theta|X))$. Para uma única observação, temos que:

$$
\begin{aligned}
\pi(\theta|X) &\propto f(X|\theta)\pi(\theta) \\
&\propto \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{\frac{-(x-\theta)^2}{2\sigma^2}\right\} \frac{1}{\sqrt{2\pi\sigma_0^2}} exp\left\{\frac{-(\theta-\mu_0)^2}{2\sigma_0^2}\right\} \\
&\propto \frac{1}{\sqrt{2\pi\sigma^2\sigma_0^2}} exp\left\{\frac{-\theta^2 + 2\theta\mu_0\sigma^2 - \mu_0^2}{2\sigma_0^2} - \frac{x^2 - 2\theta x + \theta^2}{\sigma^2}\right\}
\end{aligned}
$$

como $\frac{1}{\sqrt{2\pi\sigma^2\sigma_0^2}}$ não apresenta o parâmetro θ, será uma constante, então:

$$
\begin{aligned}
\pi(\theta|X) &\propto exp\left\{\frac{-\theta^2\sigma^2 + 2\theta\mu_0\sigma^2 - \mu_0^2\sigma^2 - \sigma_0^2 x^2 + 2\theta\sigma_0^2 x - \theta^2\sigma_0^2}{2\sigma^2\sigma_0^2}\right\} \\
&\propto exp\left\{\frac{-\theta^2(\sigma^2+\sigma_0^2) + 2\theta(\mu_0\sigma^2 + \sigma_0^2 x) - (\mu_0^2\sigma^2 + \sigma_0^2 x^2)}{2\sigma_0^2\sigma^2}\right\}
\end{aligned}
$$

$$
\propto exp\left\{\frac{-\theta^2 + 2\theta\frac{\mu_0\sigma^2+\sigma_0^2 x}{\sigma^2+\sigma_0^2} - \left(\frac{\mu_0\sigma^2+\sigma_0^2 x}{\sigma^2+\sigma_0^2}\right)^2}{\frac{2\sigma_0^2\sigma^2}{\sigma^2+\sigma_0^2}}\right\} exp\left\{-\frac{\mu_0^2\sigma^2+\sigma_0^2 x^2}{\sigma^2+\sigma_0^2}\right\}
$$

como $\frac{\mu_0^2\sigma^2+\sigma_0^2x^2}{\sigma^2+\sigma_0^2}$ independe do parâmetro θ, é constante, então, temos:

$$
\pi(\theta|X) \propto exp\left\{\frac{-\left(\theta - \frac{\mu_0\sigma^2+\sigma_0^2 x}{\sigma^2+\sigma_0^2}\right)^2}{2\frac{\sigma^2\sigma_0^2}{\sigma^2+\sigma_0^2}}\right\}
$$

A partir do resultado acima, podemos ver que a *posteriori* segue uma distribuição normal com média μ_1 e variância σ_1^2, onde:

$$
\mu_1 = \frac{\mu_0\sigma^2 + \sigma_0^2 x}{\sigma^2 + \sigma_0^2} = \frac{\mu_0\sigma_0^{-2} + x\sigma^{-2}}{\sigma^{-2} + \sigma_0^{-2}},
$$

$$\sigma_1^2 = \frac{\sigma^2\sigma_0^2}{\sigma^2+\sigma_0^2} = \frac{1}{\sigma^{-2}+\sigma_0^{-2}}.$$

Logo $\theta|X \sim N(\mu_1, \sigma_1^2)$.

Com base nos resultados anteriores fica fácil encontrarmos a *posteriori* no caso de uma amostra de tamanho n, precisando apenas trocar a $f(X|\theta)$ pela verossimilhança $L(X|\theta)$. Com isso, obtemos que $\theta|X \sim N(\mu_1, \sigma_1^2)$, onde:

$$\sigma_1^2 = \left(\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2/n}\right)^{-1},$$

$$\mu_1 = \sigma_1^2\left(\frac{\mu_0}{\sigma_0^2} + \frac{\bar{x}}{\sigma^2/n}\right).$$

1.2 *Priori* não informativa de Jeffreys com variância conhecida

Dado uma Normal com variância conhecida, queremos calcular a *priori* não informativa de Jeffreys para a média.

Como sabemos, ela é dada por:

$$\pi(\theta) \propto [I(\theta)]^{1/2}$$

Então vamos começar calculando a informação de Fisher para uma única observação, onde obtemos:

$$I_1(\theta) = -E\left[\frac{\partial^2}{\partial\theta^2}\frac{(X-\theta)^2}{2\sigma^2}\right] = 1/\sigma^2$$

Porém, como σ é conhecido, nossa informação de fisher é constante, o que implica que nossa *priori* também será uma constante. Ao tentarmos integra-la no espaço paramétrico, vemos que seu valor vai para infinito, o que é o mesmo que dizer que no caso de uma normal com variância conhecida a *priori* não informativa de Jeffreys é imprópria. Mas, mesmo assim a *posteriori* associada a ela é própria e segue uma distribuição $N \sim (\bar{x}, \frac{\sigma^2}{n})$.

1.3 Simulações

[ ]:
```python
#importando blibliotecas que serão utilizadas
import numpy as np
import seaborn as sns
import scipy.stats as stats
import pandas as pd
import matplotlib.pyplot as plt
from tabulate import tabulate
```

Vamos criar uma função que recebe os parâmetros μ_0 e σ_0 da *priori*, θ e σ dos dados (likelihood) e o tamanho da amostra n e nos retorna um dataset contendo os valores da *posteriori*.

```
[ ]: def func(mu_0, sigma_0, theta, sigma, n):

        df = pd.DataFrame({'Valor':[], 'Tipo':[]})#inicializando um dataset␣
     ↪vazio para armazenar os valores

        #priori:
        for i in range(n):#amostra n vezes
            dados1 = np.random.normal(mu_0, sigma_0, size=1)#gera uma␣
     ↪observação da priori
            dados1 = dados1.astype(float)
            x = pd.DataFrame({'Valor' : dados1, 'Tipo' : "Priori"})#coloca em␣
     ↪formato de DataFrame
            df = df.append(x,ignore_index=True)#anexa ao dataset criado para␣
     ↪armazenar os valores

        #posteriori:
        mu_post = (((sigma_0 ** -2) * mu_0) + ((sigma ** -2) * theta)) /␣
     ↪((sigma_0 ** -2) + (sigma ** -2))#média posteriori
        sigma_post = np.sqrt(((sigma_0 ** -2) + (sigma ** -2)) ** -1)#desvio␣
     ↪padrão posteriori

        for i in range(n):#amostra n vezes
            dados2 = np.random.normal(mu_post, sigma_post, size=1)#gera uma␣
     ↪observação a posteriori
            dados2 = dados2.astype(float)
            y = pd.DataFrame({'Valor' : dados2, 'Tipo' :␣
     ↪"Posteriori"})#coloca em formata de DataFrame
            df = df.append(y,ignore_index=True)#anexa ao dataset criado para␣
     ↪armazenar os valores

        return df #retorna o dataset criado
```

```
[ ]: def mean_confidence_interval(data, confidence=0.95): #funcao para␣
     ↪calcular o intervalo de confianca da media da priori
        a = 1.0 * np.array(data)
        n = len(a)
        m, se = np.mean(a), stats.sem(a)
        h = se * stats.t.ppf((1 + confidence) / 2., n-1)
        return m, m-h, m+h #retorna o valor da media na amostra, o limite␣
     ↪inferior e o superior
```

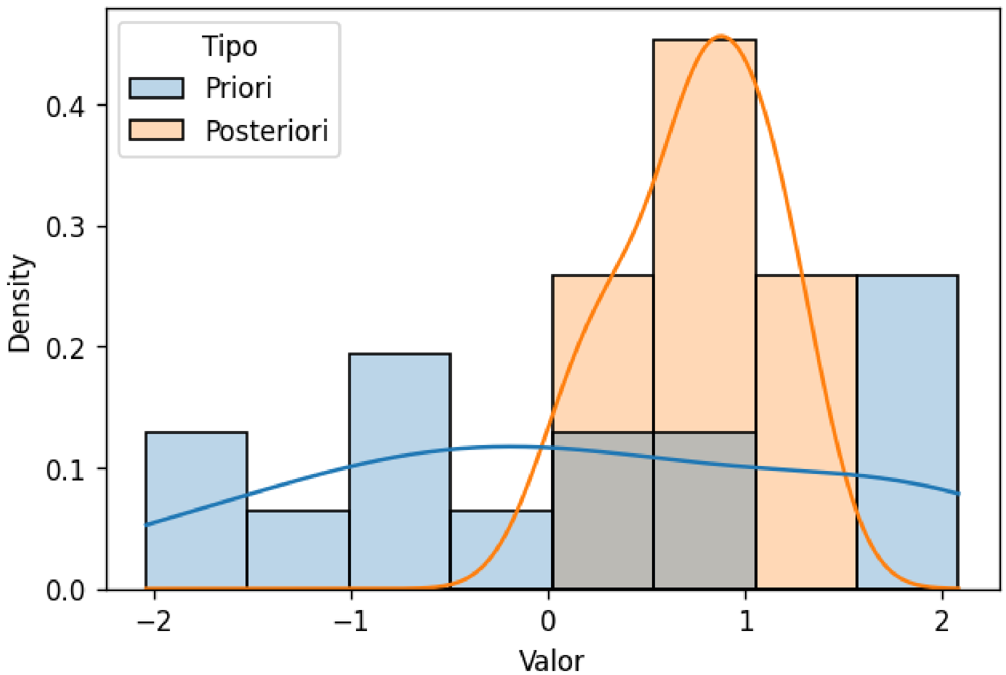
Executaremos 3 simulações como mesmas *prioris* e verossimilhanças, alterando apenas o número n das amostras.

Vamos considerar um problema hipotético onde a *priori* é dada por uma $N(0,1)$ e a verossimilhança é dada por uma $N(1, 0.5)$.

1.3.1 Para amostra de tamanho $n = 15$

[ ]:
```
a = func(0, 1, 1, 0.5, 15)
```

[ ]:
```
plt.figure(figsize=(6,4), dpi=120)
sns.histplot(data = a, x='Valor', hue='Tipo', bins= 8,stat=␣
↪'density',alpha=0.3, kde=True)
plt.show()
```



[ ]:
```
po = a[a["Tipo"] == "Posteriori"]
pr = a[a["Tipo"] == "Priori"]
mean, var, std = stats.bayes_mvs(po['Valor'], alpha=0.95)#funcao do scipy␣
↪stats que calcula o intervalo de credibilidade
print("Priori", "\n", pr.describe())
print()
print("Posteriori", "\n", po.describe())
print()
print(mean_confidence_interval(pr['Valor'], 0.95))
print()
print(mean)
```

```
Priori
            Valor
count  15.000000
mean    0.198176
std     1.350548
min    -2.038825
25%    -0.677810
50%     0.200480
75%     1.413673
max     2.082292

Posteriori
            Valor
count  15.000000
mean    0.753052
std     0.377920
min     0.046033
25%     0.492113
50%     0.815064
75%     1.052518
max     1.318108

(0.1981761955250901, -0.549732185624779, 0.9460845766749593)

Mean(statistic=0.7530517740286661, minmax=(0.5437666068678999,
0.9623369411894322))
```

[ ]:
```python
print("Resumo")
tabel = [['Tipo', 'Média', 'Desvio Padrão ', 'Mediana', 'IC 95%'],
 ['Priori', '0.198', '1.350', '0.200', '-0.549 a 0.946'], ['Posteriori',
 '0.753', '0.377', '0.815', '0.543 a 0.962']]
print(tabulate(tabel, headers='firstrow', tablefmt='fancy_grid'))
```

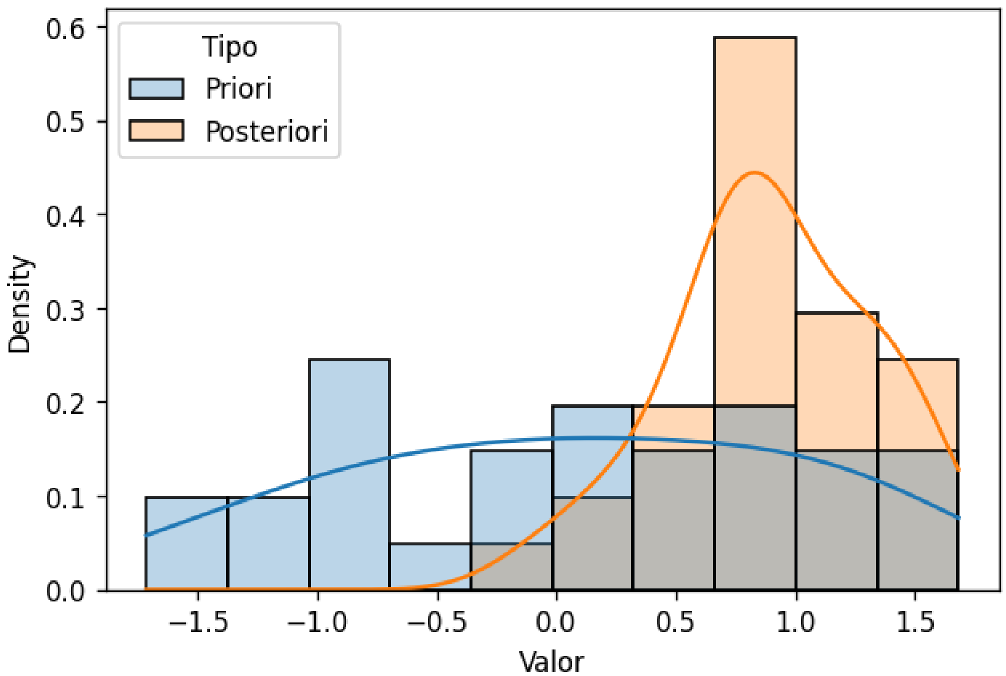
Resumo

Tipo	Média	Desvio Padrão	Mediana	IC 95%
Priori	0.198	1.35	0.2	-0.549 a 0.946
Posteriori	0.753	0.377	0.815	0.543 a 0.962

1.3.2 Para amostra de tamanho $n = 30$

[ ]:
```
b = func(0,1,1,0.5,30)
```

[ ]:
```
plt.figure(figsize=(6,4), dpi=120)
sns.histplot(data = b, x='Valor', hue='Tipo', bins= 10,stat=
↪'density',alpha=0.3, kde=True)
plt.show()
```



[ ]:
```
pos = b[b["Tipo"] == "Posteriori"]
pri = b[b["Tipo"] == "Priori"]
mean, var, std = stats.bayes_mvs(pos['Valor'], alpha=0.95)#funcao do
↪scipy stats que calcula o intervalo de credibilidade
print("Priori", "\n", pri.describe())
print()
print("Posteriori", "\n", pos.describe())
print()
print(mean_confidence_interval(pri['Valor'], 0.95))
print()
print(mean)
```

```
Priori
             Valor
count   30.000000
```

```
mean     0.070094
std      0.961682
min     -1.714906
25%     -0.733274
50%      0.062975
75%      0.797371
max      1.680643

Posteriori
             Valor
count  30.000000
mean    0.899039
std     0.422875
min    -0.086146
25%     0.674313
50%     0.876795
75%     1.270635
max     1.634779

(0.07009435991096051, -0.2890034969062174, 0.42919221672813845)

Mean(statistic=0.8990392309100589, minmax=(0.7411351025974554,
1.0569433592226622))
```

[ ]:
```
print("Resumo")
tabel = [['Tipo', 'Média', 'Desvio Padrão ', 'Mediana', 'IC 95%'],␣
 ↪['Priori', '0.070', '0.961', '0.062', '-0.289 a 0.429'], ['Posteriori',␣
 ↪'0.899', '0.422', '0.876', '0.741 a 1.056']]
print(tabulate(tabel, headers='firstrow', tablefmt='fancy_grid'))
```

Resumo

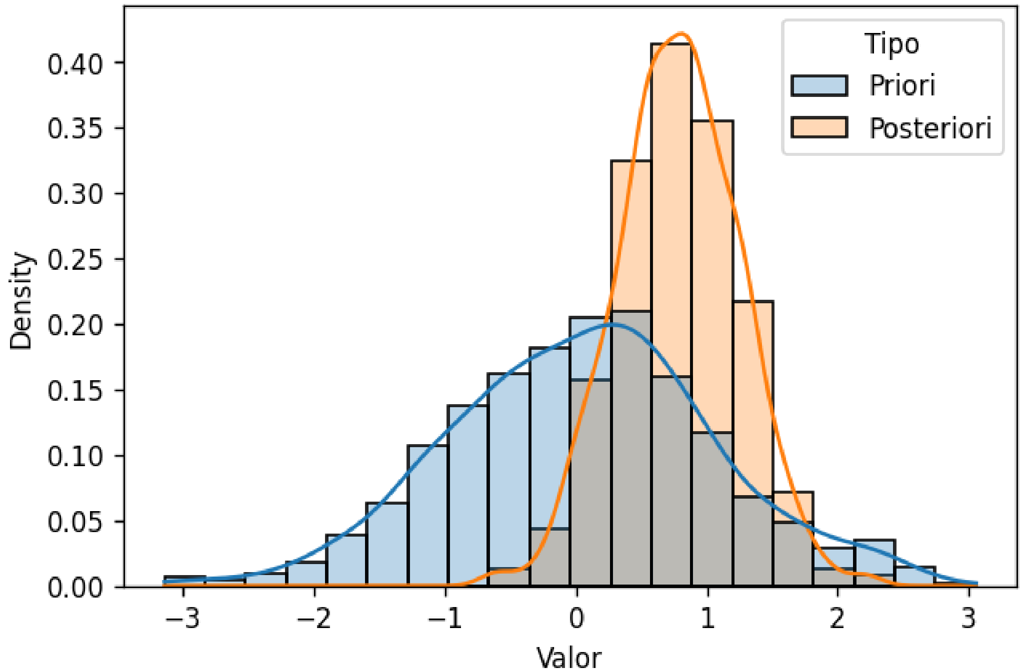
Tipo	Média	Desvio Padrão	Mediana	IC 95%
Priori	0.07	0.961	0.062	-0.289 a 0.429
Posteriori	0.899	0.422	0.876	0.741 a 1.056

1.3.3 Para amostra de tamanho $n = 1000$

[ ]:
```
c = func(0, 1, 1, 0.5, 1000)
```

[ ]:
```
plt.figure(figsize=(6,4), dpi=120)
```

```
sns.histplot(data = c, x='Valor', hue='Tipo', bins= 20,stat=␣
↪'density',alpha=0.3, kde=True)
plt.show()
```



[ ]:
```
post = c[c["Tipo"] == "Posteriori"]
prior = c[c["Tipo"] == "Priori"]
mean, var, std = stats.bayes_mvs(post['Valor'], alpha=0.95)#funcao do␣
↪scipy stats que calcula o intervalo de credibilidade
print("Priori", "\n", prior.describe())
print()
print("Posteriori", "\n", post.describe())
print(mean_confidence_interval(prior['Valor'], 0.95))
print()
print(mean)
```

```
Priori
              Valor
count  1000.000000
mean      0.060139
std       1.009295
min      -3.139391
25%      -0.619421
50%       0.084837
```

```
75%        0.689098
max        3.054776

Posteriori
              Valor
count  1000.000000
mean      0.784180
std       0.461777
min      -0.654458
25%       0.483323
50%       0.790596
75%       1.095487
max       2.275100
(0.06013929552193297, -0.002492215960197823, 0.12277080700406376)

Mean(statistic=0.784180488217815, minmax=(0.7555250502518885,
0.8128359261837416))
```

[ ]:
```python
print("Resumo")
tabel = [['Tipo', 'Média', 'Desvio Padrão ', 'Mediana', 'IC 95%'],
 ['Priori', '0.060', '1.009', '0.084', '-0.002 a 0.122'], ['Posteriori',
 '0.784', ' 0.461', '0.790', '0.755 a 0.812']]
print(tabulate(tabel, headers='firstrow', tablefmt='fancy_grid'))
```

```
Resumo

 Tipo          Média     Desvio Padrão      Mediana   IC 95%

 Priori        0.06              1.009        0.084   -0.002 a 0.122

 Posteriori    0.784             0.461        0.79    0.755 a 0.812
```

1.3.4 Conclusão

A partir dos gráficos e resumos das simulações com amostras de diferentes tamanhos, podemos observar que a medida que a nossa amostra aumenta, nosso Intervalo de Credibilidade diminui sua amplitude, o que implica numa maior "certeza" da nossa *Posteriori*.

1.4 Dados reais

1.4.1 Apresentação dos dados

Consiste em uma conjunto de dados de licença CC0: Domínio Público, encontrado em

https://www.kaggle.com/atmcfarland/historical-us-president-physical-data-more.

Nome: **Historical US President Physical Data (+More)**

Conteúdo: *Contém dados de todos os 45 presidentes dos Estados Unidos da America, como Altura, Peso, Índice de Massa Corporal, Aniversário, Idade quando assumiu a presidência, etc.*

[ ]:
```
data = pd.read_csv('/content/Historical Presidents Physical Data (More).
    ↪csv')
data.head()
```

[ ]:
```
   order               name  ...         political_party  corrected_iq
0      1  George Washington  ...             Unaffiliated         140.0
1      2         John Adams  ...               Federalist         155.0
2      3   Thomas Jefferson  ...   Democratic-Republican         160.0
3      4      James Madison  ...   Democratic-Republican         160.0
4      5       James Monroe  ...   Democratic-Republican         139.0

[5 rows x 32 columns]
```

Para o nosso estudo, utilizaremos as Idades quando assumiram a presidência, pois como veremos a seguir, elas muito provavelmente seguem uma distribuição normal.

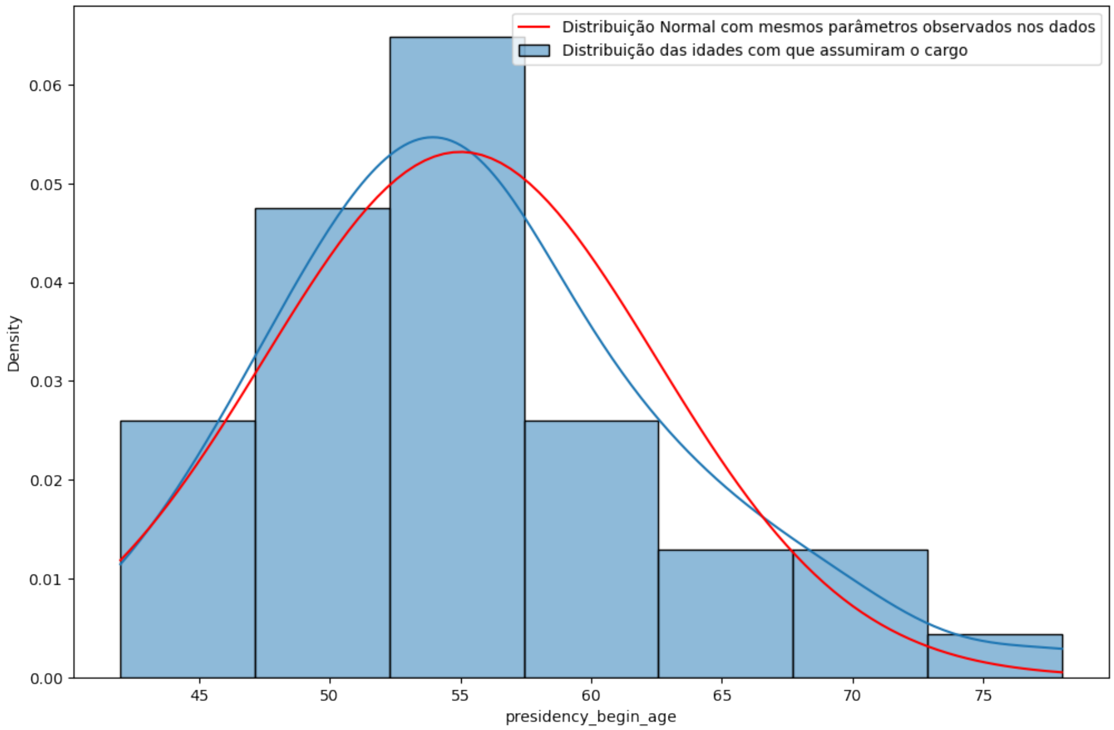
[ ]:
```
#breve estatística descritiva dos dados:
data['presidency_begin_age'].describe()
```

[ ]:
```
count    45.000000
mean     55.511111
std       7.433408
min      42.000000
25%      51.000000
50%      55.000000
75%      60.000000
max      78.000000
Name: presidency_begin_age, dtype: float64
```

[ ]:
```
#plotando os gráficos:
plt.figure(figsize=(12,8), dpi=100)
x = np.linspace(42,78, 100)
```

```
sns.histplot(data['presidency_begin_age'],bins=7, kde=True,␣
 ↪stat='density', label='Distribuição das idades com que assumiram o␣
 ↪cargo')
plt.plot(x,stats.norm.pdf(x, 55, 7.5), color='r', label='Distribuição␣
 ↪Normal com mesmos parâmetros observados nos dados')

plt.legend()
plt.show()
```



A partir de uma breve análise gráfica podemos perceber uma grande semelhança entre as duas curvas, porém isso ainda não suficiente para assumir a normalidade dos dados.

1.4.2 Teste de Normalidade dos dados

Realizaremos o teste de Shapiro-Wilk para testarmos a normalidade dos dados.

$$H_0 : \text{Os dados apresentam uma distribuição Normal}$$

$$vs$$

$$H_1 : \text{Os dados não apresentam uma distribuição Normal}$$

```
[ ]: stat, p_value = stats.shapiro(data['presidency_begin_age'])
     print('Valor da estatísta do teste = ' + str(stat) )
     print('p-valor do teste = ' + str(p_value) )
```

```
Valor da estatísta do teste = 0.9641824960708618
p-valor do teste = 0.17612870037555695
```

Como o p-valor do teste é maior que 0.05, não temos evidência estatistica suficiente para rejeitar a Hipótese Nula de que os dados possuem distribuicao normal, ao nivel de significancia de 95%.

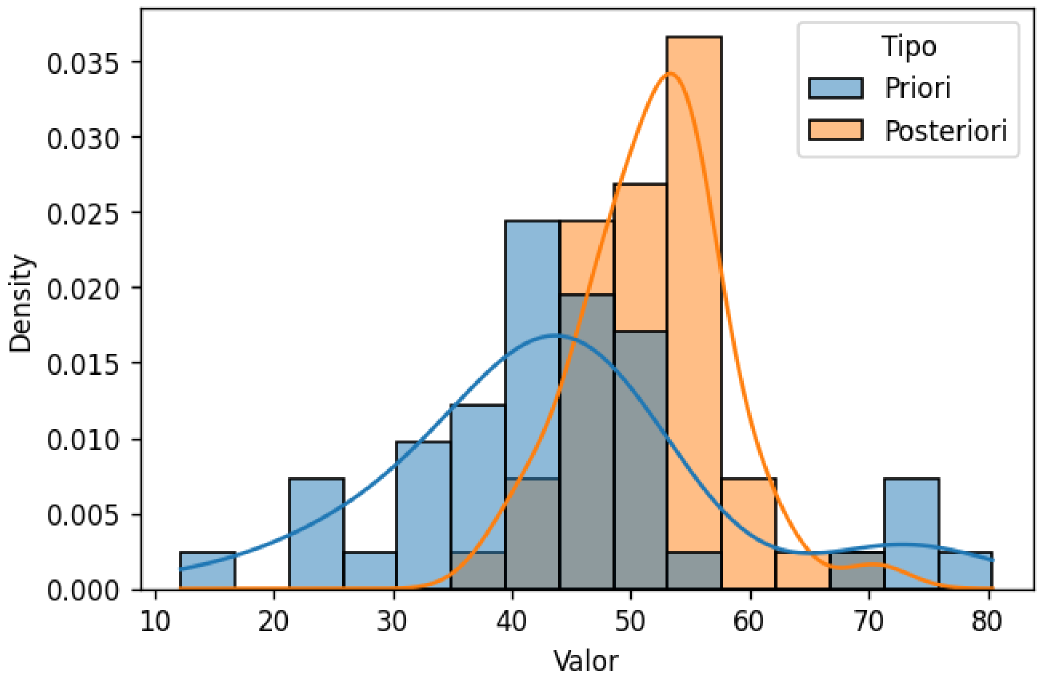
1.4.3 Simulação Bayesiana

Para a nossa *priori* consideraremos a distribuição de idades retirada de uma amostra hipotética nos Estados Unidos, que seguem uma distribuição Normal com média 46 e desvio padrão 13. $\pi(\theta) \sim N(46, 13^2)$.

Como observados nos dados, temos uma amostra de tamanho $n = 45$ seguindo uma distribuição $X|\theta \sim N(55, 7.5^2)$.

```
[ ]: #simulando:
     d = func(46, 13, 55, 7.5, 45)

     plt.figure(figsize=(6,4), dpi=120)
     sns.histplot(data = d, x='Valor', hue='Tipo', bins= 15,stat= 'density',␣
      ↪kde=True)
     plt.show()
```



[ ]:
```
p = d[d["Tipo"] == "Posteriori"]
pr = d[d["Tipo"] == "Priori"]
mean, var, std = stats.bayes_mvs(p['Valor'], alpha=0.95)#funcao do scipy␣
↪stats que calcula o intervalo de credibilidade
print("Priori", "\n", b.describe())
print()
print("Posteriori", "\n", c.describe())
print(mean_confidence_interval(pr['Valor'], 0.95))
print()
print(mean)
```

```
Priori
             Valor
count  45.000000
mean   44.065040
std    11.192310
min    19.817425
25%    35.235984
50%    45.752468
75%    51.565167
max    70.094163

Posteriori
```

```
              Valor
count  45.000000
mean   52.883178
std     8.729304
min    30.938260
25%    46.488378
50%    54.275061
75%    57.769006
max    71.378574
(44.15035055202995, 39.98177019855833, 48.31893090550157)

Mean(statistic=51.95928673803789, minmax=(50.15464490621801, 53.
 ↪76392856985777))
```

[ ]:
```
#vamos criar uma tabela do resumo

tabela = [['Tipo', 'Média', 'Desvio Padrão ', 'Mediana', 'IC 95%'],␣
 ↪['Priori', '44.065', '11.192', '45.752', '39.981 a 48.318'],␣
 ↪['Posteriori', '52.883','8.729', '54.275', '50.154 a 53.763']]
print(tabulate(tabela, headers='firstrow', tablefmt='fancy_grid'))
```

Tipo	Média	Desvio Padrão	Mediana	IC 95%
Priori	44.065	11.192	45.752	39.981 a 48.318
Posteriori	52.883	8.729	54.275	50.154 a 53.763