

Distribuição Normal e Teorema Central do Limite

1. Introdução

Foi falado anteriormente sobre as distribuições mais comuns, mas quando se diz a respeito da **distribuição normal**, deve-se ter um capítulo à parte por ser a distribuição **mais conhecida e utilizada** na Estatística/Probabilidade. Esta distribuição é representada por uma **curva simétrica em torno da média**, apresentando uma forma de **sino** (*bell shape*).

A curva da distribuição normal representa o comportamento de **diversos processos e fenômenos comuns**, como por exemplo: altura ou peso de uma população, a pressão sanguínea de um grupo de pessoas, o tempo que um grupo de estudantes gasta para realizar uma prova, entre outras aplicações.

A distribuição normal serve também como base para a **inferência estatística clássica**, sendo a premissa em diversos modelos e métodos.

2. Definição Matemática

A variável aleatória contínua X que tome todos os valores na reta real $-\infty < X < \infty$ segue uma distribuição normal (também nomeada como Gaussiana), se sua função de densidade de probabilidade é dada por:

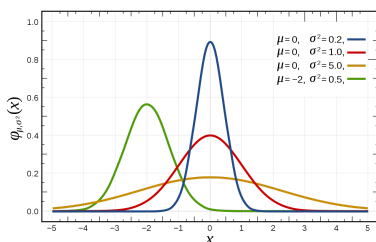
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Onde respectivamente temos como **valor esperado** e **variância** da distribuição normal:

Valor esperado: $E[X] = \mu$

Variância: $V[X] = \sigma^2$

Usualmente aplica-se a notação $X \sim N(\mu, \sigma^2)$ para representar uma variável aleatória **com distribuição normal** de valor esperado/média μ e variância σ^2 .



Fonte: [Wikimedia](#)

Uma propriedade da distribuição Normal é que pela sua característica **simétrica**, todas as métricas de posição **coincidem** no ponto médio (sendo elas média, mediana e moda).

Para criar a curva da distribuição normal, pode-se utilizar uma implementação em *Python*:

```
# Função densidade de probabilidade para a distribuição normal
def normal_dist(x , mu, sigma):
    prob_density = (1/(sigma*(math.sqrt(2*np.pi))))*np.exp(-0.5*((x - mu)/sigma)**2)
    return prob_density

# Define valores para média e desvio padrão (Obs.: sigma = sqrt(variância))
mu = 75
sigma = 20

# Define o tamanho de uma figura para o gráfico
plt.figure(figsize=(10,6))

# Define os valores de x entre o mínimo e máximo para gerarmos a curva normal
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 500)

# Dados para a curva normal
pdf = normal_dist(x, mu, sigma)

# Cria um título
plt.title('Distribuição Normal')

# Gera a curva normal
plt.plot(x, pdf, color = 'red')

# Label X
plt.xlabel('Data points')

# Label Y
plt.ylabel('Probability Density')

# Mostra o gráfico
plt.show()
```

Podendo também utilizar a biblioteca *SciPy* para implementar a função da distribuição normal com a função *cdf*:

```
# Carrega as funções para a Normal da biblioteca SciPy
from scipy.stats import norm

# Define valores para média e desvio padrão (Obs.: sigma = sqrt(variância))
mu = 75
sigma = 20

# Define o tamanho de uma figura para o gráfico
plt.figure(figsize=(10,6))

# Define os valores de x entre o mínimo e máximo para gerarmos a curva normal
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 500)
```

```

# Dados para a curva normal
y = norm(loc = mu, scale = sigma).pdf(x)

# Cria um título
plt.title('Distribuição Normal')

# Gera a curva normal
plt.plot(x, y, color = 'red')

# Label X
plt.xlabel('Data points')

# Label Y
plt.ylabel('Probability Density')

# Mostra o gráfico
plt.show()

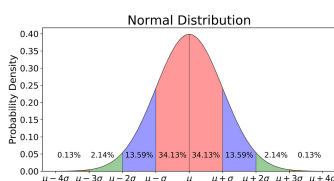
```

3. Z-Score (Normalização dos Dados)

Naturalmente, devido à complexidade da função densidade de probabilidade para a distribuição normal, não é tão usual trabalhar diretamente com a fórmula. Para isso então que existe uma transformação conhecida como normalização ou mesmo **Z-Score**, onde este parâmetro nada mais é que **o número de desvios padrões que a observação está com relação à média**:

$$z = \frac{x - \mu}{\sigma}$$

$$z\sigma = x - \mu$$




Fonte: [Medium](#)

Vale ressaltar que, este método é altamente dependente da hipótese de que os dados observados são normalmente distribuídos, portanto funciona como apenas uma boa aproximação para muitos casos, em geral.

Os valores de *Z-score* também podem ser definidos a partir de uma tabela padronizada para distribuições normais, seguindo uma forma mais tradicional:

Standard Normal Cumulative Probability Table

Cumulative probabilities for POSITIVE z-values are shown in the following table:



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8829
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9725	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9915
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9942	0.9943	0.9944	0.9945	0.9946	0.9947	0.9948
2.6	0.9949	0.9950	0.9951	0.9952	0.9953	0.9954	0.9955	0.9956	0.9957	0.9958
2.7	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964	0.9965	0.9966	0.9967	0.9968
2.8	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974	0.9975	0.9976	0.9977	0.9978
2.9	0.9979	0.9980	0.9981	0.9982	0.9983	0.9984	0.9985	0.9986	0.9987	0.9988
3.0	0.9989	0.9990	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.1	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.2	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.3	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998

Fonte [Chegg](#)

4. Aplicações

Para ilustrar as funcionalidades da Distribuição Normal, seguem alguns exemplos de aplicação prática:

Exemplo 1: Se uma variável aleatória $X \sim \mathcal{N}(\mu = 165, \sigma^2 = 9)$, calcule $P(X < 162)$

Dado os valores de μ e σ^2 , defina a probabilidade a partir do *Z-Score*:

```
# Carrega a função do Scipy para calcular a normal a partir do Z-Score
import scipy.stats as st

# Define o valor de média
mu = 165

# Define o valor de sigma (desvio padrão)
sigma = np.sqrt(9)

# Calcula o Z-Score para 162
z = (162 - mu)/sigma

# Mostra o resultado do Z-Score e da probabilidade pela distribuição normal
print('Valor do Z - Score: ', z)
print('Probabilidade do Evento: ', st.norm.cdf(z))
```

Exemplo 2: Dado a variável aleatória $X \sim \mathcal{N}(\mu = 10, \sigma^2 = 4)$, calcule a probabilidade $P(X > 13)$

Lembrando que para calcular essa probabilidade temos que fazer da seguinte forma:
 $P(X > 13) = 1 - P(X \leq 13)$

```
# Define o valor de média
mu = 10

# Define o valor de sigma (desvio padrão)
sigma = np.sqrt(4)
```

```
# Calcula o Z-Score para 162
z = (13 - mu)/sigma

# Cálculo da probabilidade complementar
prop_comp = 1 - st.norm.cdf(z)

# Mostra o resultado do Z-Score e da probabilidade pela distribuição normal
print('Valor do Z - Score: ', z)
print('Probabilidade do Evento: ', prop_comp)
```

Exemplo 3: O peso médio de 500 estudantes do sexo masculino de uma determinada universidade é 75,5 Kg e o desvio padrão é 7,5 Kg. Admitindo que os pesos são normalmente distribuídos, determine a percentagem de estudantes que pesam:

A) entre 60 e 77,5 Kg:

Desenvolvendo o exercício a partir do *Z-Score*:

$$P(60 \leq X \leq 77,5) = P\left(\frac{60 - \mu}{\sigma} \leq \frac{X - \mu}{\sigma} \leq \frac{77,5 - \mu}{\sigma}\right) = P\left(\frac{60 - \mu}{\sigma} \leq Z \leq \frac{77,5 - \mu}{\sigma}\right) =$$

$$= P\left(Z \leq \frac{77,5 - \mu}{\sigma}\right) - P\left(Z \leq \frac{60 - \mu}{\sigma}\right)$$

Agora fazendo a implementação dos cálculos utilizando a biblioteca *SciPy*:

```
# Valor da média
mu = 75.5

# Valor do desvio padrão
sigma = 7.5

# Calculando Z1 para 60 kg
z1 = (60 - mu)/sigma

# Calculando Z2 para 77.5 kg
z2 = (77.5 - mu)/sigma

# Mostra os valores de Z-Score e o resultado da probabilidade
print('Z - Score para 60 kg: ', z1)
print('Z - Score para 77.5 kg: ', z2)
print('Probabilidade: ', st.norm.cdf(z2) - st.norm.cdf(z1))
```

O mesmo exemplo poder ser resolvido também com a implementação de simulações de uma distribuição normal com os mesmos parâmetros μ e σ e contar quantos elementos da simulação se encontram dentro do intervalo entre 60 e 77,5 Kg:

```
# Define os parâmetros de média e desvio padrão
mu = 75.5
sigma = 7.5

# Define a quantidade de simulações
N = 100000

# Fixa um valor de semente aleatória
np.random.seed(2)

# Gera uma amostra a partir de uma distribuição normal com mu e sigma para os N valores
X = np.random.normal(mu, sigma, N)

# Desenvolve um laço onde serão contados de todos os elementos quantos estão dentro do intervalo
m = 0 # inicia a contagem com valor 0
for x in X: # varrer cada elemento de X
    if x > 60 and x < 77.5: # verifica se o valor encontra-se no intervalo
        m = m + 1 # caso afirmativo, adiciona a contagem anterior

# Define a probabilidade pela divisão da contagem pelo número de simulações
print('A probabilidade a partir das simulações será: ', np.round(m/N, 4))
```

B) pesam mais do que 92,5 Kg

$$P(X \geq 92.5) = P\left(\frac{X - \mu}{\sigma} \geq \frac{92.5 - \mu}{\sigma}\right) = P\left(Z \geq \frac{92.5 - \mu}{\sigma}\right) = 1 - P\left(Z < \frac{92.5 - \mu}{\sigma}\right)$$

```
# Valor da média
mu = 75.5

# Valor do desvio padrão
sigma = 7.5

# Valor do Z-Score para 92.5 kg
z = (92.5 - mu)/sigma

# Cálculo da probabilidade condicional
prop = 1 - st.norm.cdf(z)

# Mostra os valores de Z-Score e o resultado da Probabilidade
print('Z - Score para 92,5 kg: ', z)
print('Probabilidade do evento: ', np.round(prop, 4))
```

Aplicando novamente o mesmo processo mas com simulações:

```
# Define os parâmetros de média e desvio padrão
mu = 75.5
```

```

sigma = 7.5

# Define a quantidade de simulações
N = 1000000

# fixa um valor de semente aleatória
np.random.seed(3)

# Gera uma amostra a partir de uma distribuição normal com mu e sigma para os N valores
X = np.random.normal(mu, sigma, N)

# Desenvolve um laço onde serão contados de todos os elementos quantos estão dentro do intervalo
m = 0                # inicia a contagem com valor 0
for x in X:          # varrer cada elemento de X
    if x > 92.5:      # verifica se o valor encontra-se no intervalo
        m = m + 1    # caso afirmativo, adiciona a contagem anterior

# Define a probabilidade pela divisão da contagem pelo número de simulações
print('A probabilidade a partir das simulações será: ', np.round(m/N, 4))

```

5. Teorema Central do Limite

O Teorema Central do Limite (algumas vez denominado *Teorema do Limite Central*) é uma das ferramentas mais poderosas da estatística, e é esse teorema que dá fundamentação para a distribuição normal ser amplamente utilizada como base para muitos outros resultados. O princípio por trás deste teorema é que muitos dos resultados da inferência estatística são válidos assumindo a hipótese que esses dados sob análise (mais precisamente, os **estimadores pontuais**) **seguem uma distribuição normal**.

No entanto, há muitos casos de interesse em que a **distribuição populacional não é normal**, onde na verdade não é possível afirmar isso sobre a distribuição populacional, a partir apenas das amostras. Dado este contexto que o Teorema Central do Limite atua, pois ele mostra que os **estimadores pontuais de parâmetros populacionais serão normalmente distribuídos**, independente da distribuição populacional destes dados.

Vamos enunciá-lo, como:

Seja uma amostra aleatória (X_1, X_2, \dots, X_n) retiradas de uma população com média μ e variância σ . A distribuição amostral de \bar{X} aproxima-se, para n grande ($n > 30$), de uma distribuição normal com média $E[\bar{X}] = \mu$ e variância σ^2/n .

Exemplo de Aplicação: Seja a variável aleatória com distribuição de probabilidade: $P(X = 3) = 0,4$; $P(X = 6) = 0,3$; $P(X = 8) = 0,3$. Uma amostra com 40 observações é sorteada. Qual é a probabilidade de que a média amostral seja maior do que 5?

Primeiramente, deve-se definir o valor esperado e a esperança desta distribuição para usar como referência para a distribuição normal:

```

# Função para a esperança
def esperanca(X, P):
    E = 0

```

```

    for i in range(0, len(X)):
        E = E + X[i]*P[i]
    return E

# Função para a variância
def variancia(X ,P):
    E = 0; E2 = 0
    for i in range(0, len(X)):
        E = E + X[i]*P[i]
        E2 = E2 + (X[i]**2)*P[i]
    V = E2-E**2
    return V

# Vetor de eventos
X = [3,6,8]

# Vetor de probabilidades
P = [0.4,0.3,0.3]

# Cálculo da esperança
E = esperanca(X,P)

# Cálculo da variância
V = variancia(X,P)

# Mostra o resultado para a esperança e variância
print("Esperança: ", E)
print("Variância: ", V)

```

Calculando o valor teórico da probabilidade, utilizando o *Z-Score*:

```

# Define o valor da média como a esperança
mu = E

# Define o desvio padrão como a raiz da variância
sigma = np.sqrt(V)

# Tamanho da amostra
n = 40

# X a verificar
x = 5

# Z-score
Z = (x - mu)/(sigma/np.sqrt(n))

# Cálculo da probabilidade
prob = 1 - st.norm.cdf(Z)

```



```
# Print da probabilidade
print('A probabilidade será: ', prob)
```

Agora, dado que não se sabe se a distribuição é uma normal ou não, deve-se sortear várias amostras de tamanho $n = 40$ e verificar qual a probabilidade da média dessas amostras ser maior do que 5:

```
# Tamanho de amostras
n = 40

# Número de simulações
ns = 1000

# Vetor vazio para armazenar a média amostral
vx = [] # armazena a média amostral

# Laço para as simulações
for s in range(0, ns):
    A = np.random.choice(X, n, p = P)
    vx.append(np.mean(A))

# Cria um histograma para as simulações
plt.figure(figsize=(8,6))
plt.hist(x=vx, bins='auto', color='#0504aa', alpha=0.7, rwidth=0.85, density = True)
plt.xlabel(r'$\bar{X}$', fontsize=20)
plt.ylabel(r'$P(\bar{X})$', fontsize=20)
plt.show()

# Compara as métricas do teórico e da simulação
print("Média das amostras: ", np.mean(vx))
print("Média da população: ", E)

# Calculando a probabilidade das amostras
prob_s = 0
for i in range(0, len(vx)): # Varredura de cada uma das amostras
    if(vx[i] > 5): # Verifica a condição para a amostra
        prob_s = prob_s + 1 # Soma a probabilidade a anterior

# Calcula a probabilidade média das amostras
prob_s = prob_s/len(vx)

# Compara a probabilidade teórica com a simulação
print("Valor teórico : ", prob)
print("Valor simulado : ", prob_s)
```

Referências

Pedro A. Morettin, Wilton O. Bussab, Estatística Básica, 8ª edição

Peter Bruce, Andrew Bruce & Peter Gedeck, Practical Statistics for Data Scientists, 50+ Essential Concepts Using R and Python, 2ª edition

Ron Larson & Betsy Farber, Estatística Aplicada, 6ª edição.