Aluno: Felipe Trindade Radovanovic

Matrícula: 18100830

Curso: Engenharia Eletrônica

Disciplina: EEL7417-07235 (20202) - Fundamentos de Comunicação Digital

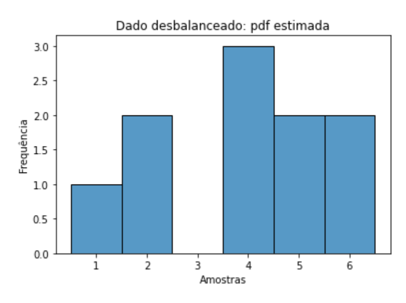
Professor: Leonardo Silva Resende

1) Modelagem de uma variável aleatória: caso de um dado justo

1.1) Gerar amostras aleatórias jogando N vezes um único dado

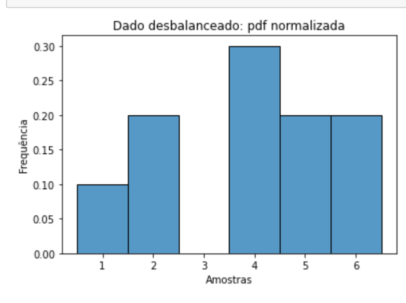
In [1]:
```
import random
import numpy as np
import seaborn as sns
import statistics
import matplotlib.pyplot as plt
import scipy.stats as stats
import math

n = 10
N = np.arange(0,n,1)
Amostras = np.zeros_like(N)
for i in range(N.shape[0]):
  Amostras[i] = random.randrange(6) + 1

print('Amostras geradas: {}'.format(Amostras))
sns.histplot(Amostras, discrete=True).set(title='Dado desbalanceado: pdf estimada', xlabel=
"Amostras", ylabel="Frequência");
```

```
Amostras geradas: [6 1 4 4 2 4 5 6 5 2]
```

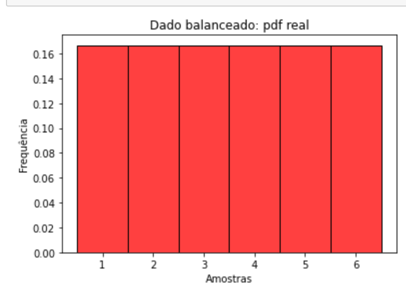


1.2) Pode-se normalizar os valores do histograma com N para que possa ser obtida a função densidade de probabilidade estimada

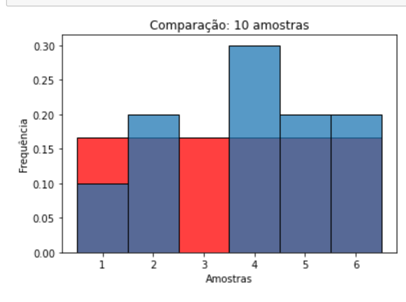
In [2]:
```
sns.histplot(Amostras, stat="density", discrete=True).set(title='Dado desbalanceado: pdf nor
malizada', xlabel="Amostras", ylabel="Frequência");
```



In [3]:
```
Amostras_ideal = [1, 2, 3, 4, 5, 6]

sns.histplot(Amostras_ideal, stat="density", discrete=True, color='red').set(title='Dado bal
anceado: pdf real', xlabel="Amostras", ylabel="Frequência");
```



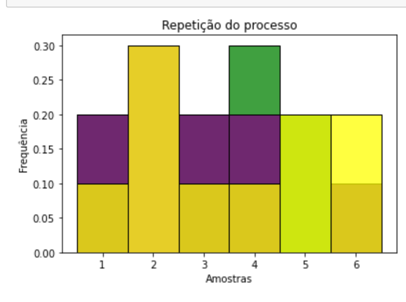
In [4]:
```
sns.histplot(Amostras_ideal, stat="density", discrete=True, color='red').set(title='Comparaç
ão: 10 amostras', xlabel="Amostras", ylabel="Frequência");
sns.histplot(Amostras, stat="density", discrete=True);
```



A pdf estimada é diferente da real, pois a quantidade de amostras é pequena. Quanto maior o número de amostras, mais a pdf estimada se aproximará da real. Se tornará igual, quando o número de amostras tender a infinito.

1.3) Repita o processo mais algumas vezes

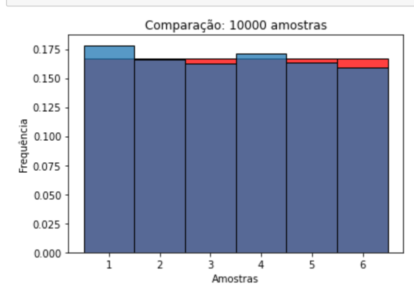
In [5]:
```
n = 10
N = np.arange(0,n,1)
Amostras_1 = np.zeros_like(N)
Amostras_2 = np.zeros_like(N)
Amostras_3 = np.zeros_like(N)
for i in range(N.shape[0]):
  Amostras_1[i] = random.randrange(6) + 1
  Amostras_2[i] = random.randrange(6) + 1
  Amostras_3[i] = random.randrange(6) + 1

sns.histplot(Amostras_1, stat="density", discrete=True, color='green').set(title='Repetição
 do processo', xlabel="Amostras", ylabel="Frequência");
sns.histplot(Amostras_2, stat="density", discrete=True, color='purple');
sns.histplot(Amostras_3, stat="density", discrete=True, color='yellow');
```



É possível notar que a pdf estimada nem sempre é igual. Além disso, como o número de amostras é pequeno, as variações são bem grandes.

1.4) Agora incremente o número de experimentos para N=10.000

In [6]:
```
n = 10000
N = np.arange(0,n,1)
Amostras = np.zeros_like(N)
for i in range(N.shape[0]):
  Amostras[i] = random.randrange(6) + 1

sns.histplot(Amostras_ideal, stat="density", discrete=True, color='red').set(title='Comparaç
ão: 10000 amostras', xlabel="Amostras", ylabel="Frequência");
sns.histplot(Amostras, stat="density", discrete=True);
```



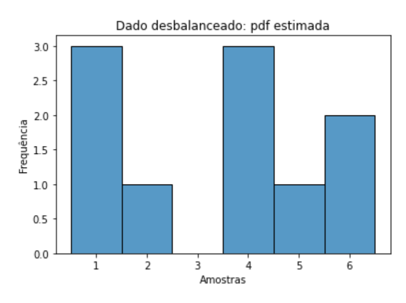
É possível perceber que a pdf estimada agora se aproxima bem mais da real.

2) Geração de uma variável aleatória: jogar um dado desbalanceado

2.1) Gerar amostras aleatórias jogando o dado, onde a probabilidade para o valor 6 é 1/4 e para os demais valores são equiprováveis

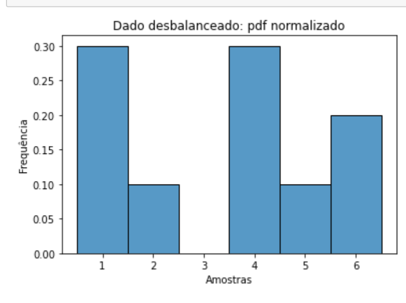
In [7]:
```
n = 10
N = np.arange(0,n,1)
Amostras_des = np.zeros_like(N)
for i in range(N.shape[0]):
  Amostras_des[i] = random.choice([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6,
6])

print('Amostras geradas: {}'.format(Amostras_des))
sns.histplot(Amostras_des, discrete=True).set(title='Dado desbalanceado: pdf estimada', xlab
el="Amostras", ylabel="Frequência");
```

```
Amostras geradas: [6 4 4 1 1 4 2 6 5 1]
```



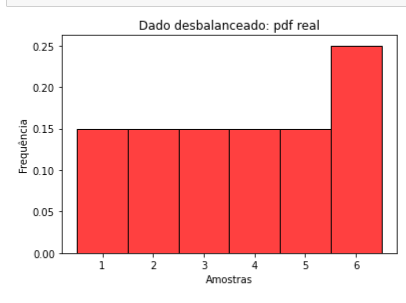
2.2) Plote o histograma normalizado

In [8]:
```
sns.histplot(Amostras_des, stat="density", discrete=True).set(title='Dado desbalanceado: pdf
normalizado', xlabel="Amostras", ylabel="Frequência");
```

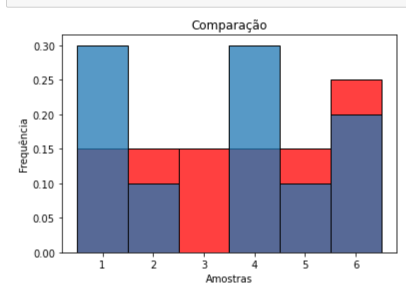


2.3) Definir a pdf verdadeira para o dado desbalanceado e plote junto com a pdf estimada

In [9]:
```
Amostras_des_ideal = [1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 6]
sns.histplot(Amostras_des_ideal, stat="density", discrete=True, color='red').set(title='Dado
desbalanceado: pdf real', xlabel="Amostras", ylabel="Frequência");
```



In [10]:
```
sns.histplot(Amostras_des_ideal, stat="density", discrete=True, color='red').set(title='Comp
aração', xlabel="Amostras", ylabel="Frequência");
sns.histplot(Amostras_des, stat="density", discrete=True);
```



3) Modelagem de uma variável aleatória normalmente distribuída X com média zero e variância unitária

3.1) Gerar 10.000 amostras para a variável X

In [11]:
```
m = 0
var = 1
sigma = math.sqrt(var)

n = 10000
N = np.arange(0,n,0.1)
Amostras_nor = np.zeros_like(N)
for i in range(N.shape[0]):
  Amostras_nor[i] = random.gauss(m, sigma)

print('Amostras geradas: {}'.format(Amostras_nor))
```

```
Amostras geradas: [ 1.81130334 -2.26571358 -0.07105999 ... -0.44124108  1.17235516
 -0.71703883]
```

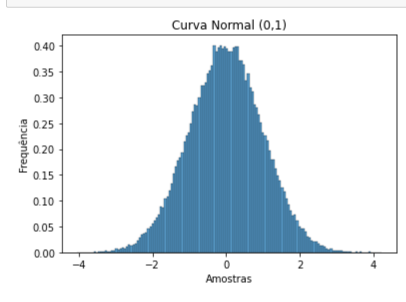
3.2) Calcular as seguintes estatísticas das amostras observadas: Média, Desvio padrão e Variância

In [12]:
```
Media = statistics.mean(Amostras_nor)
Desvio = statistics.stdev(Amostras_nor)
Variancia = statistics.variance(Amostras_nor)
print('A Média é de: {} \nO desvio é de: {} \nA variância é de: {}'.format(Media, Desvio, Va
riancia))
```

```
A Média é de: -0.004576039255284858
O desvio é de: 0.9965632501421562
A variância é de: 0.9931383115338978
```

3.3) Plotar o histograma normalizado

In [13]:
```
sns.histplot(Amostras_nor, stat="density").set(title='Curva Normal (0,1)', xlabel="Amostras"
, ylabel="Frequência");
```



4) Modelagem de uma variável aleatória normalmente distribuída X com média μ e variância σ²

4.1) Repetir o exercício anterior com μ=2 e σ² =4

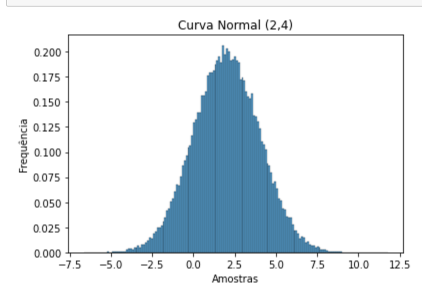
In [14]:
```
m = 2
var = 4
sigma = math.sqrt(var)

n = 10000
N = np.arange(0,n,0.1)
Amostras_nor = np.zeros_like(N)
for i in range(N.shape[0]):
  Amostras_nor[i] = random.gauss(m, sigma)

print('Amostras geradas: {}'.format(Amostras_nor))
```

```
Amostras geradas: [2.30573821 1.49157566 1.99646416 ... 0.10482557 5.40124319 3.62671539]
```

In [15]:
```
Media = statistics.mean(Amostras_nor)
Desvio = statistics.stdev(Amostras_nor)
Variancia = statistics.variance(Amostras_nor)
print('A Média é de: {} \nO desvio é de: {} \nA variância é de: {}'.format(Media, Desvio, Va
riancia))
```

```
A Média é de: 2.00294129546118
O desvio é de: 2.0021368656551592
A variância é de: 4.008552028815465
```

In [16]:
```
sns.histplot(Amostras_nor, stat="density").set(title='Curva Normal (2,4)', xlabel="Amostras"
, ylabel="Frequência");
```



5) Uso da função Q para cálculos de probabilidade P(a<X<b) para variável aleatória normalmente distribuída X~(2,4)

5.1) Definir a probabilidade P(a<X<b), em que a=2 e b=6

In [17]:
```
a = 2
b = 6
n = 10000

Amostras_cont = 0
for i in range(n):
  if a < Amostras_nor[i]:
    if Amostras_nor[i] < b:
      Amostras_cont = Amostras_cont +1

Amostras_nor_prob_ab = Amostras_cont/n

print('A probabilidade P(a<X<b) é de: {}'.format(Amostras_nor_prob_ab))
```

```
A probabilidade P(a<X<b) é de: 0.4813
```

5.2) Calcular a probabilidade P(X<a) e P(X>b)

In [18]:
```
a = 2
b = 6
n = 10000

Amostras_cont_a = 0
Amostras_cont_b = 0
for i in range(n):
  if a > Amostras_nor[i]:
      Amostras_cont_a = Amostras_cont_a +1
  if Amostras_nor[i] > b:
      Amostras_cont_b = Amostras_cont_b +1

Amostras_nor_prob_a = Amostras_cont_a/n
Amostras_nor_prob_b = Amostras_cont_b/n

print('A probabilidade P(X<a) é de: {} \nA probabilidade P(X>b) é de: {}'.format(Amostras_no
r_prob_a, Amostras_nor_prob_b))
```

```
A probabilidade P(X<a) é de: 0.4962
A probabilidade P(X>b) é de: 0.0225
```

5.3) Checar numericamente que a soma das probabilidades deve ser igual a 1

In [19]:
```
Amostras_nor_prob = Amostras_nor_prob_a + Amostras_nor_prob_b + Amostras_nor_prob_ab
print('A soma das probabilidades é de: {}'.format(Amostras_nor_prob))
```

```
A soma das probabilidades é de: 1.0
```

6) Experimento para testar o Teorema do Limite Central

6.1) Pode-se observar este fato com o exemplo do dado justo, visto anteriormente. Em vez de jogar o dado uma vez, faz-se K vezes e observa-se a soma das jogadas. Por exemplo, com 2 (dois) dados, X1 e X2, a variável aleatória seria Z = X1 + X2, em que os valores estariam entre 2 e 12

In [20]:
```
d = 2
n = 100
N = np.arange(0,n,1)
Amostras_soma = np.zeros_like(N)
for i in range(N.shape[0]):
  soma = 0
  for j in range(d):
    soma = soma + random.randrange(6) + 1
  Amostras_soma[i] = soma

print('Amostras geradas: {}'.format(Amostras_soma))
```

```
Amostras geradas: [10 10  7  5  3  4  8  7  8  4  7  7  6  9  7  6  9  5  5  7  7  4  2  5
  8  8  7  8  9 11  4  6  5 11  8  3  8 10  8  5  9  6  6  3  3  9  9 11
  3  5  2  9  6  3  9  9  3  5  5  4  6  6  5  5 11 10 10 10  7 12  9  6
  9  6  4  8  8  4  8  9 10  6  3  6  9  9  2  7  8  4  8  7  3  3  8 10
  4  7  7  7]
```

6.2) Gerar amostras de valores aleatórios para Z (soma dos dados) jogando-os 20 vezes

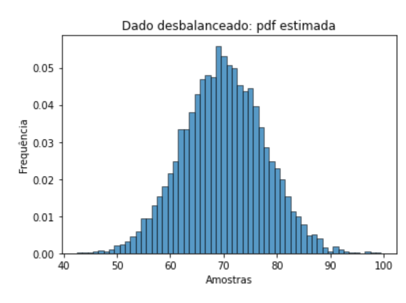
In [21]:
```
d = 20
soma = 0
for j in range(d):
  soma = soma + random.randrange(6) + 1

print('Amostra gerada: {}'.format(soma))
```

```
Amostra gerada: 80
```

6.3) Repita o experimento por 10.000 vezes e plote o histograma de Z

In [22]:
```
d = 20
n = 10000
N = np.arange(0,n,1)
Amostras_soma = np.zeros_like(N)
for i in range(N.shape[0]):
  soma = 0
  for j in range(d):
    soma = soma + random.randrange(6) + 1
  Amostras_soma[i] = soma

print('Amostras geradas: {}'.format(Amostras_soma))
sns.histplot(Amostras_soma, stat="density", discrete=True).set(title='Dado desbalanceado: pd
f estimada', xlabel="Amostras", ylabel="Frequência");
```

```
Amostras geradas: [74 71 77 ... 70 68 55]
```



O valor esperado da distribuição é 70, pois o valor médio entre 1 e 6 é 3,5 e multiplicando 3,5 por 20, obtemos 70. Faz sentido que a destribuição seja uma Gaussiana, pois os valores centrais tem uma maior probabilidade de acontecer. Para deixar isso mais fácil de se entender, podemos pensar no caso da soma de dois dados. Para que a soma seja 7, as possibilidades são (1, 6), (2, 5), (3, 4), (4, 3), (5, 2) e (6, 1), em quanto que para ser 2, a única possibilidade é (1, 1).