Conociendo nuestros datos de pingüinos. 🗺🧭🐧

Instalar librerías necesarias

```
!pip install --upgrade pip
```

```
!pip install palmerpenguins==0.1.4 numpy==1.23.4 pandas==1.5.1 seaborn==0.12.1 matplotlib==3.
statsmodels==0.13.5 scikit-learn==1.1.2 pyjanitor==0.23.1 session-info
```

Importar librerías

```
import empiricaldist
import janitor
import matplotlib.pyplot as plt
import numpy as np
import palmerpenguins
import pandas as pd
import scipy.stats
import seaborn as sns
import sklearn.metrics
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats as ss
import session_info
```

Establecer apariencia general de los gráficos

```
%matplotlib inline
sns.set_style(style='whitegrid')
sns.set_context(context='notebook')
plt.rcParams['figure.figsize'] = (11, 9.4)

penguin_color = {
    'Adelie': '#ff6602ff',
    'Gentoo': '#0f7175ff',
    'Chinstrap': '#c65dc9ff'
}
```

Cargar los datos

Utilizando el paquete `palmerpenguins`

Datos crudos

```
raw_penguins_df = palmerpenguins.load_penguins_raw()
raw_penguins_df
```

	studyName object	Sample Number i...	Species object	Region object	Island object	Stage o
	PAL0910 34.9% PAL0809 33.1% PAL0708 32%	1 - 152	Adelie Pen... 44.2% Gentoo peng... 36% Chinstrap p... 19.8%	Anvers 100%	Biscoe 48.8% Dream 36% Torgersen 15.1%	Adult, 1 l
0	PAL0708	1	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
1	PAL0708	2	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
2	PAL0708	3	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
3	PAL0708	4	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
4	PAL0708	5	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
5	PAL0708	6	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
6	PAL0708	7	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
7	PAL0708	8	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
8	PAL0708	9	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
9	PAL0708	10	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1

Datos previamente procesados

```
preprocessed_penguins_df = palmerpenguins.load_penguins()
preprocessed_penguins_df
```

	species object	island object	bill_length_mm fl...	bill_depth_mm flo...	flipper_length_m...	body_m
	Adelie 44.2% Gentoo 36% Chinstrap 19.8%	Biscoe 48.8% Dream 36% Torgersen 15.1%	32.1 - 59.6	13.1 - 21.5	172.0 - 231.0	2700.0 -
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	nan	nan	nan	
4	Adelie	Torgersen	36.7	19.3	193.0	
5	Adelie	Torgersen	39.3	20.6	190.0	
6	Adelie	Torgersen	38.9	17.8	181.0	
7	Adelie	Torgersen	39.2	19.6	195.0	
8	Adelie	Torgersen	34.1	18.1	193.0	
9	Adelie	Torgersen	42.0	20.2	190.0	

Utilizando los conjuntos de datos de `seaborn`

```
preprocessed_penguins_df = sns.load_dataset("penguins")
```

Utilizando la interfaz de Deepnote

Links de importación de datos:

- Conjunto de datos crudos.
- Conjunto de datos previamente procesados.

```
raw_penguins_df2 = pd.read_csv('penguins_raw.csv')
raw_penguins_df2
```

	studyName object	Sample Number i...	Species object	Region object	Island object	Stage o
	PAL0910 34.9% PAL0809 33.1% PAL0708 32%	1 - 152	Adelie Pen... 44.2% Gentoo peng... 36% Chinstrap p... 19.8%	Anvers 100%	Biscoe 48.8% Dream 36% Torgersen 15.1%	Adult, 1 I
0	PAL0708	1	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
1	PAL0708	2	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
2	PAL0708	3	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
3	PAL0708	4	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
4	PAL0708	5	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
5	PAL0708	6	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
6	PAL0708	7	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
7	PAL0708	8	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
8	PAL0708	9	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1
9	PAL0708	10	Adelie Penguin (P...	Anvers	Torgersen	Adult, 1

```
preprocessed_penguins_df2 = pd.read_csv('penguins.csv')
preprocessed_penguins_df2
```

	species object	island object	bill_length_mm fl...	bill_depth_mm flo...	flipper_length_m...	body_m
	Adelie 44.2% Gentoo 36% Chinstrap 19.8%	Biscoe 48.8% Dream 36% Torgersen 15.1%	32.1 - 59.6	13.1 - 21.5	172.0 - 231.0	2700.0 -
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	nan	nan	nan	
4	Adelie	Torgersen	36.7	19.3	193.0	
5	Adelie	Torgersen	39.3	20.6	190.0	
6	Adelie	Torgersen	38.9	17.8	181.0	
7	Adelie	Torgersen	39.2	19.6	195.0	
8	Adelie	Torgersen	34.1	18.1	193.0	
9	Adelie	Torgersen	42.0	20.2	190.0	

Colecta y validación de datos

¿Qué tipo de dato son las variables del conjunto de datos?

```
preprocessed_penguins_df.dtypes
```

¿Cuántas variables de cada tipo de dato tenemos en el conjunto

```
(
    preprocessed_penguins_df
    .dtypes
    .value_counts()
)
```

¿Cuántas variables y observaciones tenemos en el conjunto de d

```
preprocessed_penguins_df.shape
```

¿Existen valores nulos explicitos en el conjunto de datos?

```
(
    preprocessed_penguins_df
    .isnull()
    .any()
)
```

De tener observaciones con valores nulos, ¿cuántas tenemos po

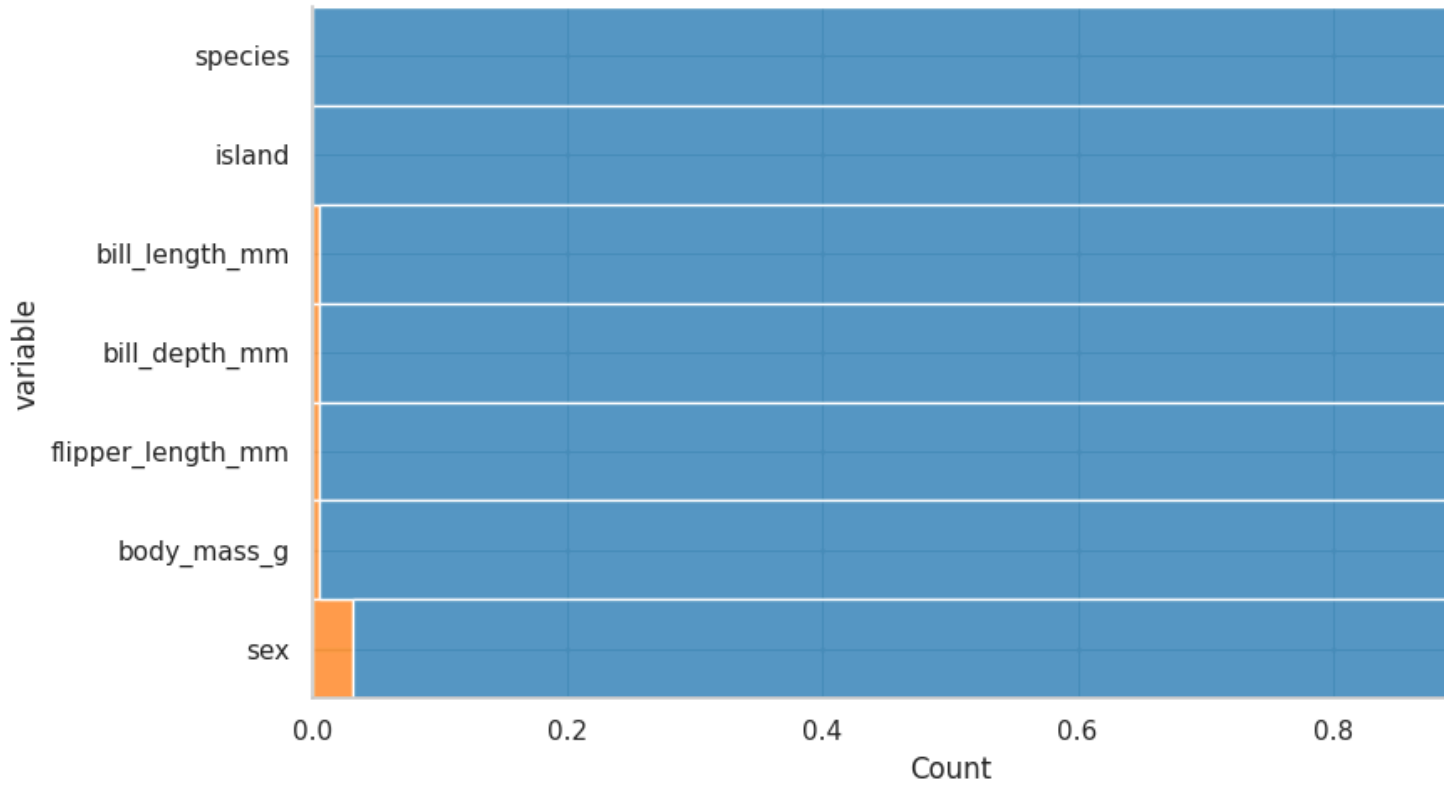
```
(
    preprocessed_penguins_df
    .isnull()
    .sum()
    .sort_values(ascending=False)
)
```

¿Cuántos valores nulos tenemos en total en el conjunto de datos

```
(
    preprocessed_penguins_df
    .isnull()
    .sum()
    .sum()
)
```

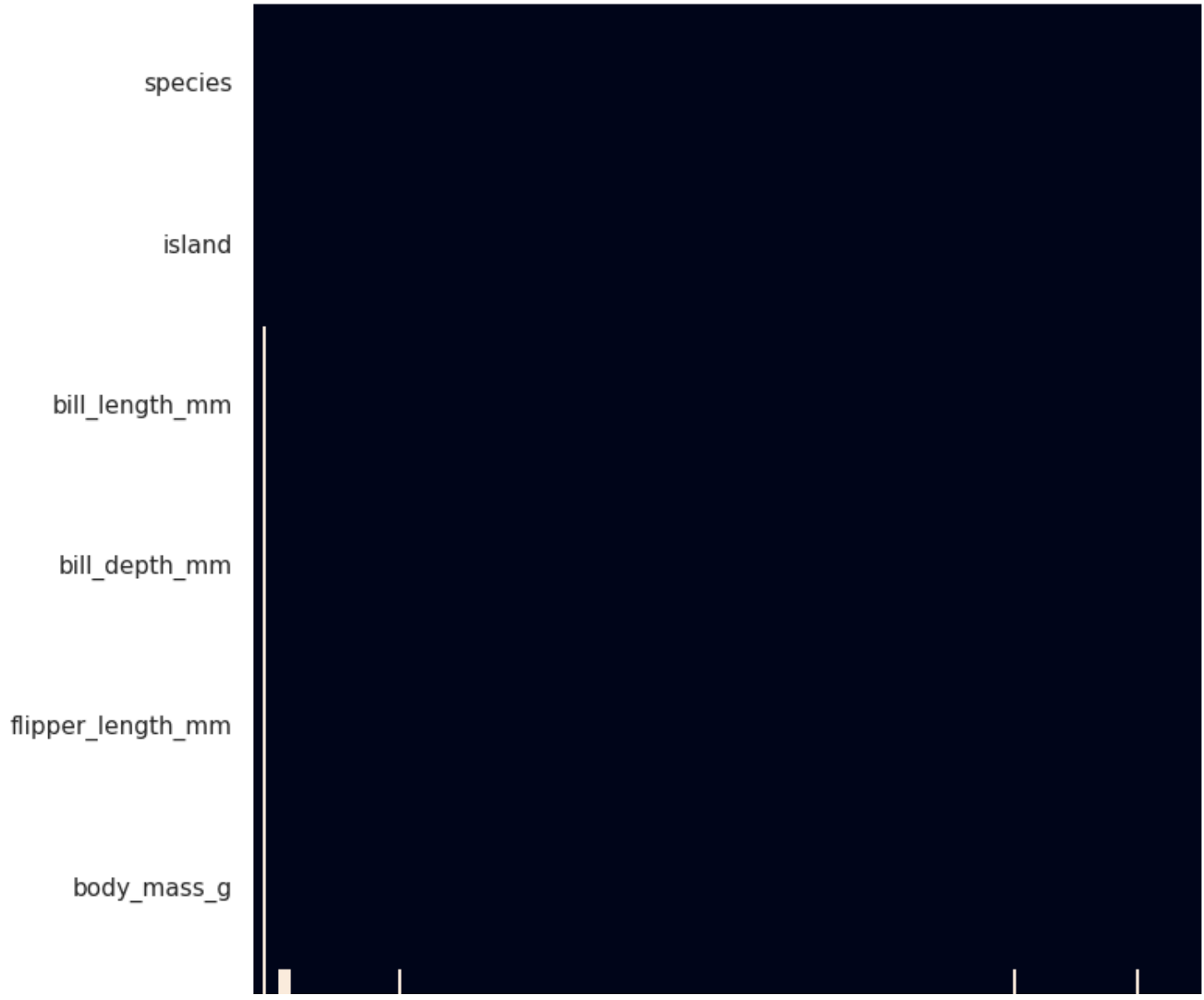
¿Cuál es la proporción de valores nulos por cada variable?

```
(
    preprocessed_penguins_df
    .isnull()
    .melt(value_name='missing')
    .pipe(
        lambda df: (
            sns.displot(
                data=df,
                y='variable',
                hue='missing',
                multiple='fill',
                aspect=2
            )
        )
    )
)
```



¿Cómo podemos visualizar los valores nulos en todo el conjunto

```
(
    preprocessed_penguins_df
    .isnull()
    .transpose()
    .pipe(
        lambda df: sns.heatmap(data=df)
    )
)
```



¿Cuántas observaciones perdemos si eliminamos los datos faltar

```
processed_penguins_df = (
    preprocessed_penguins_df
    .dropna()
)

processed_penguins_df
```

	species object Adelie 43.8% Gentoo 35.7% Chinstrap 20.4%	island object Biscoe 48.9% Dream 36.9% Torgersen 14.1%	bill_length_mm fl... 32.1 - 59.6	bill_depth_mm flo... 13.1 - 21.5	flipper_length_m... 172.0 - 231.0	body_m 2700.0 -
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
4	Adelie	Torgersen	36.7	19.3	193.0	
5	Adelie	Torgersen	39.3	20.6	190.0	
6	Adelie	Torgersen	38.9	17.8	181.0	
7	Adelie	Torgersen	39.2	19.6	195.0	
12	Adelie	Torgersen	41.1	17.6	182.0	
13	Adelie	Torgersen	38.6	21.2	191.0	
14	Adelie	Torgersen	34.6	21.1	198.0	

Conteos y proporciones

Preludio: ¿Qué estadísticos describen el conjunto de datos?

Todas las variables

```
processed_penguins_df.describe(include='all')
```

	species object	island object	bill_length_mm fl...	bill_depth_mm flo...	flipper_length_m...	body_m
	333 9.1% 3 others 27.3% Missing 63.6%	333 9.1% 3 others 27.3% Missing 63.6%	5.4686683426475...	1.9692354633199...	14.0157652882878...	333.0 - 6
cou...	333	333	333.0	333.0	333.0	
uni...	3	3	nan	nan	nan	
top	Adelie	Biscoe	nan	nan	nan	
freq	146	163	nan	nan	nan	
me...	nan	nan	43.99279279279..	17.16486486486...	200.9669669669..	4207.05
std	nan	nan	5.468668342647..	1.969235463319..	14.015765288287..	805.215
min	nan	nan	32.1	13.1	172.0	
25%	nan	nan	39.5	15.6	190.0	
50%	nan	nan	44.5	17.3	197.0	
75%	nan	nan	48.6	18.7	213.0	

Solo las numéricas

```
processed_penguins_df.describe(include=[np.number])
```

	bill_length_mm fl...	bill_depth_mm flo...	flipper_length_m...	body_mass_g floa...	
cou...	333.0	333.0	333.0	333.0	
me...	43.99279279279..	17.16486486486...	200.9669669669..	4207.057057057...	
std	5.468668342647..	1.969235463319..	14.015765288287..	805.2158019428...	
min	32.1	13.1	172.0	2700.0	
25%	39.5	15.6	190.0	3550.0	
50%	44.5	17.3	197.0	4050.0	
75%	48.6	18.7	213.0	4775.0	
max	59.6	21.5	231.0	6300.0	

Solo categóricas - 1

```
processed_penguins_df.describe(include=object)
```

	species object	island object	sex object	
cou...	333	333	333	
uni...	3	3	2	
top	Adelie	Biscoe	Male	
freq	146	163	168	

Solo categóricas - 2

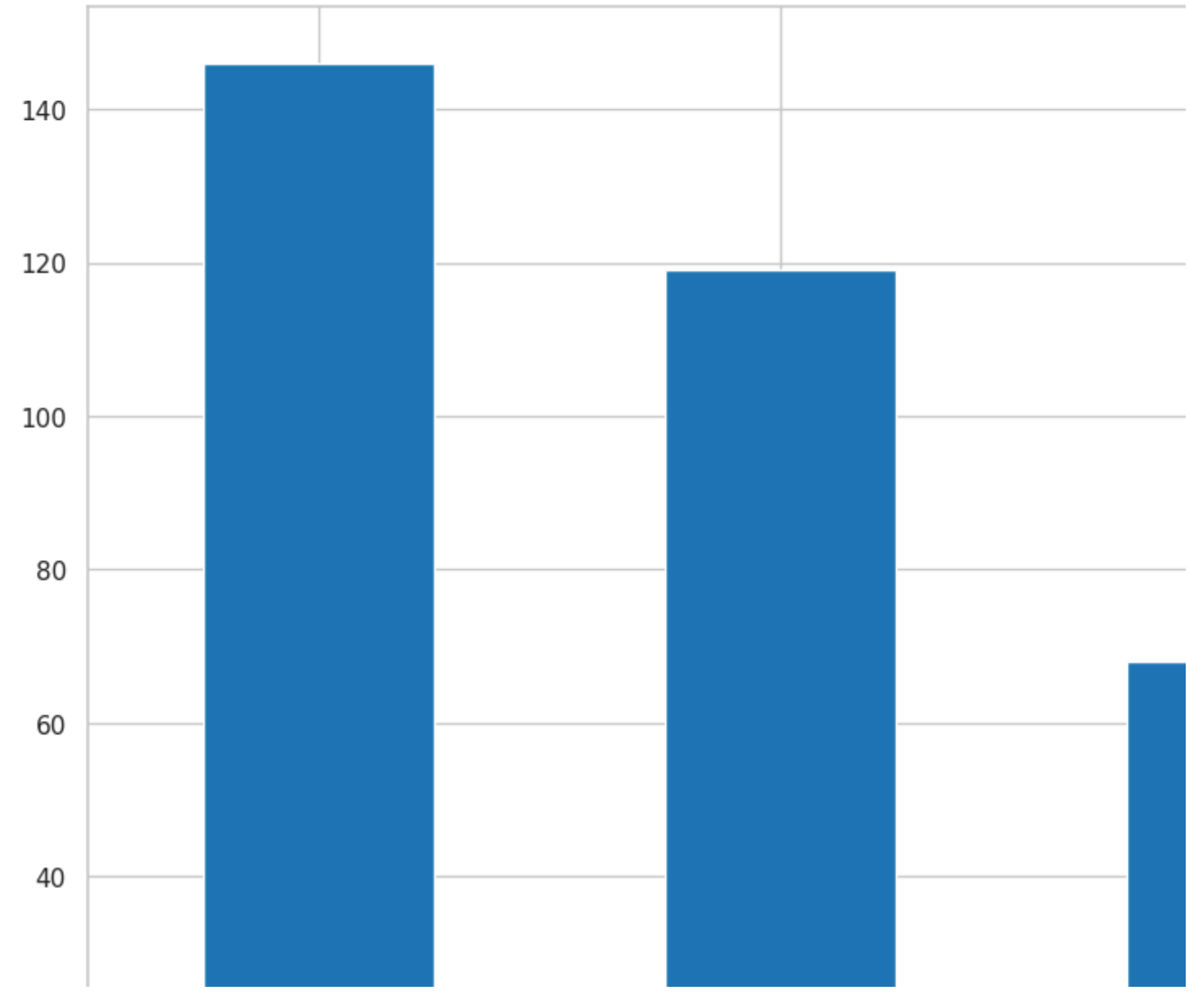
```
(
    processed_penguins_df
    .astype(
        {
            'species': 'category',
            'island': 'category',
            'sex': 'category'
        }
    )
    .describe(include=['category', object])
)
```

	species object	island object	sex object	
cou...	333	333	333	
uni...	3	3	2	
top	Adelie	Biscoe	Male	
freq	146	163	168	

¿Cómo visualizar los conteos?

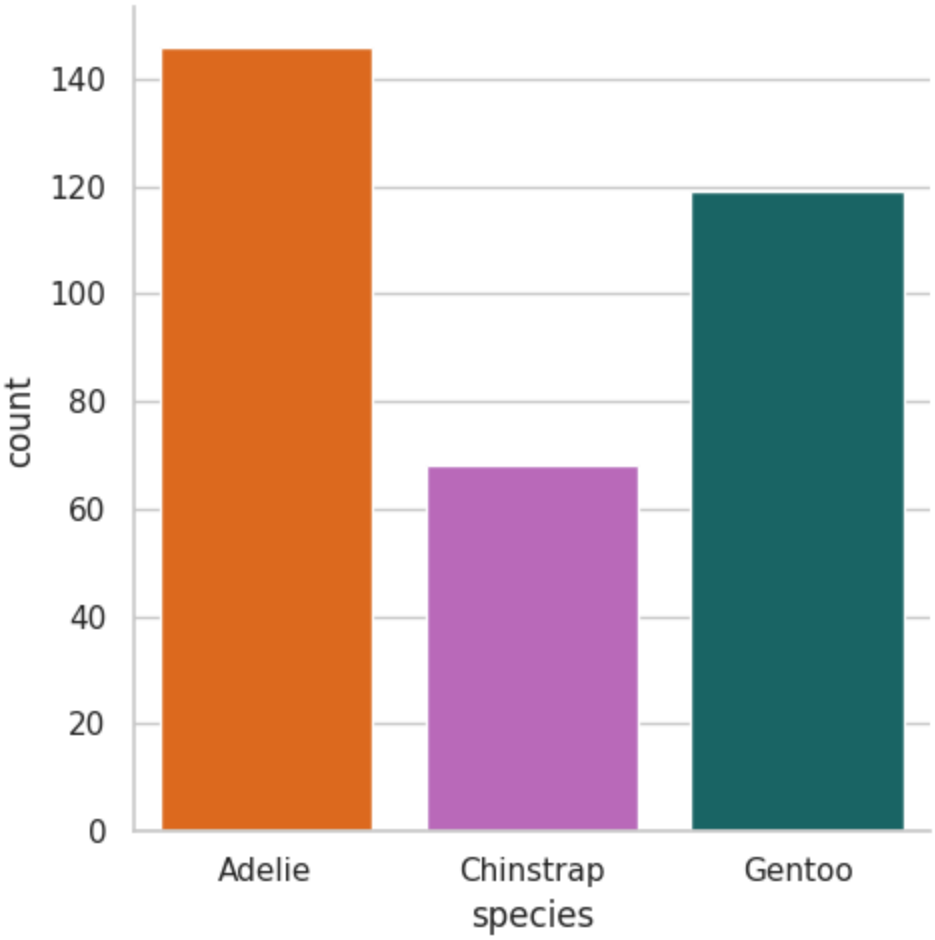
Pandas

```
(
    processed_penguins_df
    .species
    .value_counts()
    .plot(
        kind='bar',
        # color=penguin_color.values()
    )
)
```

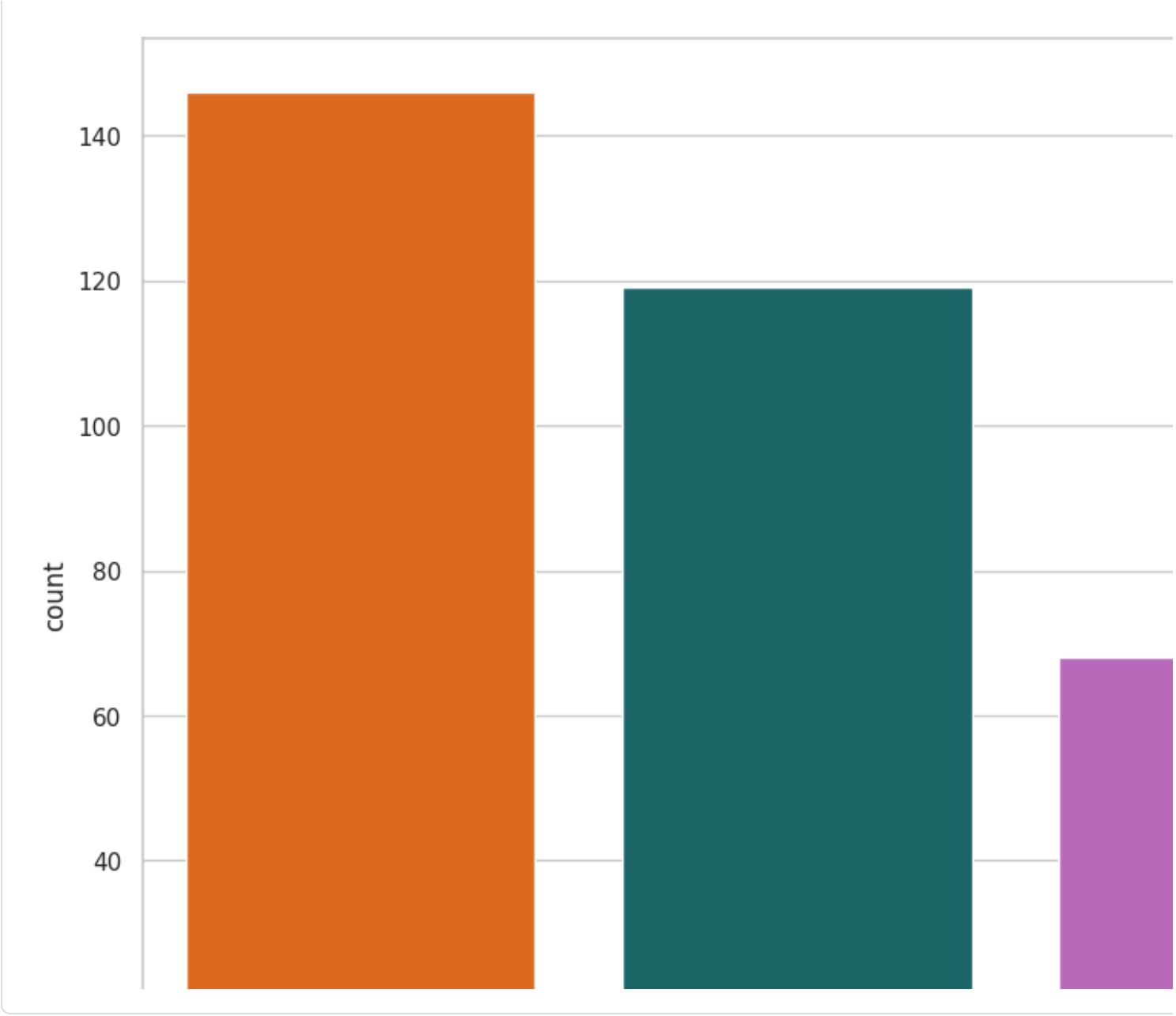


Seaborn

```
sns.catplot(
    data=processed_penguins_df,
    x='species',
    kind='count',
    palette=penguin_color,
    # order=processed_penguins_df.value_counts('species', sort=True).index
)
```

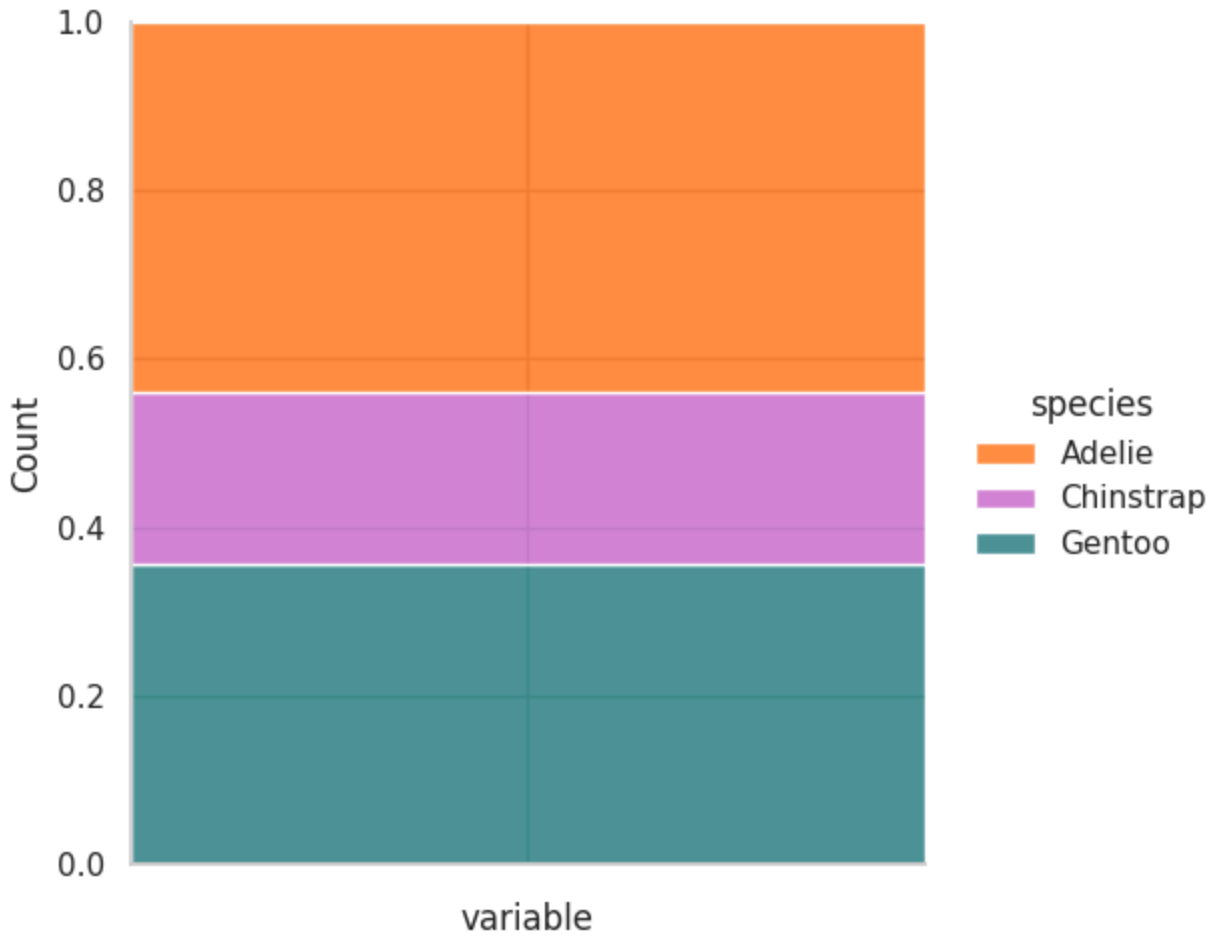


```
(
    processed_penguins_df
    .value_counts('species', sort=True)
    .reset_index(name='count')
    .pipe(
        lambda df: (
            sns.barplot(
                data=df,
                x='species',
                y='count',
                palette=penguin_color
            )
        )
    )
)
```



¿Cómo visualizar las proporciones?

```
(
    processed_penguins_df
    .add_column('variable', '')
    .pipe(
        lambda df: (
            sns.displot(
                data=df,
                x='variable',
                hue='species',
                multiple='fill',
                palette=penguin_color
            )
        )
    )
)
```



Medidas de tendencia central

Media o promedio

```
processed_penguins_df.bill_depth_mm.mean()
np.mean(processed_penguins_df.bill_depth_mm)
processed_penguins_df.mean()

processed_penguins_df.median()
processed_penguins_df.mode()
processed_penguins_df.describe(include=object)
```

```
/tmp/ipykernel_144/1680988049.py:3: FutureWarning: The default value of numeric_only in DataFrame.mean is depr
  processed_penguins_df.mean()
/tmp/ipykernel_144/1680988049.py:5: FutureWarning: The default value of numeric_only in DataFrame.median is de
  processed_penguins_df.median()
```

	species object	island object	sex object	
cou...	333	333	333	
uni...	3	3	2	
top	Adelie	Biscoe	Male	
freq	146	163	168	

```
np.mean(processed_penguins_df.bill_depth_mm)
```

```
processed_penguins_df.mean()
```

```
/tmp/ipykernel_144/1618060137.py:1: FutureWarning: The default value of numeric_only in DataFrame.mean is depr
  processed_penguins_df.mean()
```

Mediana

```
processed_penguins_df.median()
```

```
/tmp/ipykernel_144/3242987746.py:1: FutureWarning: The default value of numeric_only in DataFrame.median is de
  processed_penguins_df.median()
```

Moda

```
processed_penguins_df.mode()
```

	species object	island object	bill_length_mm fl...	bill_depth_mm flo...	flipper_length_m...	body_m
0	Adelie	Biscoe	41.1	17.0	190.0	

```
processed_penguins_df.describe(include=object)
```

	species object	island object	sex object
cou...	333	333	333
uni...	3	3	2
top	Adelie	Biscoe	Male
freq	146	163	168

Medidas de dispersión

¿Cuál es el valor máximo de las variables?

```
processed_penguins_df.max()
```

¿Cuál es el valor mínimo de las variables?

```
processed_penguins_df.min()
```

¿Cuál es el rango de las variables?

```
processed_penguins_df.max(numeric_only=True) - processed_penguins_df.min(numeric_only=True)
```

¿Cuál es la desviación estándar de las variables?

```
processed_penguins_df.std()
```

```
/tmp/ipykernel_144/4261057176.py:1: FutureWarning: The default value of numeric_only in DataFrame.std is depre
  processed_penguins_df.std()
```

¿Cuál es el rango intercuartílico?

```
processed_penguins_df.quantile(0.25)
```

```
/tmp/ipykernel_144/2653252021.py:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is
  processed_penguins_df.quantile(0.25)
```

```
processed_penguins_df.quantile(0.75) - processed_penguins_df.quantile(0.25)
```

```
/tmp/ipykernel_144/1337602098.py:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is
  processed_penguins_df.quantile(0.75) - processed_penguins_df.quantile(0.25)
```

```
(
    processed_penguins_df
    .quantile(q=[0.75, 0.50, 0.25])
    .transpose()
    .rename_axis('metric')
    .reset_index()
    .assign(
        iqr = lambda df: df[0.75] - df[0.25]
    )
)
```

```
/tmp/ipykernel_144/610954116.py:2: FutureWarning: The default value of numeric_only in DataFrame.quantile is d
  processed_penguins_df
```

	metric object	0.75 float64	0.5 float64	0.25 float64	iqr float64
0	bill_length_mm	48.6	44.5	39.5	9.100000000000..
1	bill_depth_mm	18.7	17.3	15.6	3.099999999999..
2	flipper_length_mm	213.0	197.0	190.0	23.0
3	body_mass_g	4775.0	4050.0	3550.0	1225.0

¿Cómo puedo visualizar la distribución de una variable?

Histograma

```
sns.histplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.mean(),
    color='red',
    linestyle='dashed',
    linewidth=2
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.median(),
    color='blue',
    linestyle='dashed',
    linewidth=2
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.mode().values[0],
    color='black',
    linestyle='dashed',
    linewidth=4
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.quantile(0.25),
    color='yellow',
    linestyle='dashed',
    linewidth=2
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.quantile(0.75),
    color='yellow',
    linestyle='dashed',
    linewidth=2
)
```

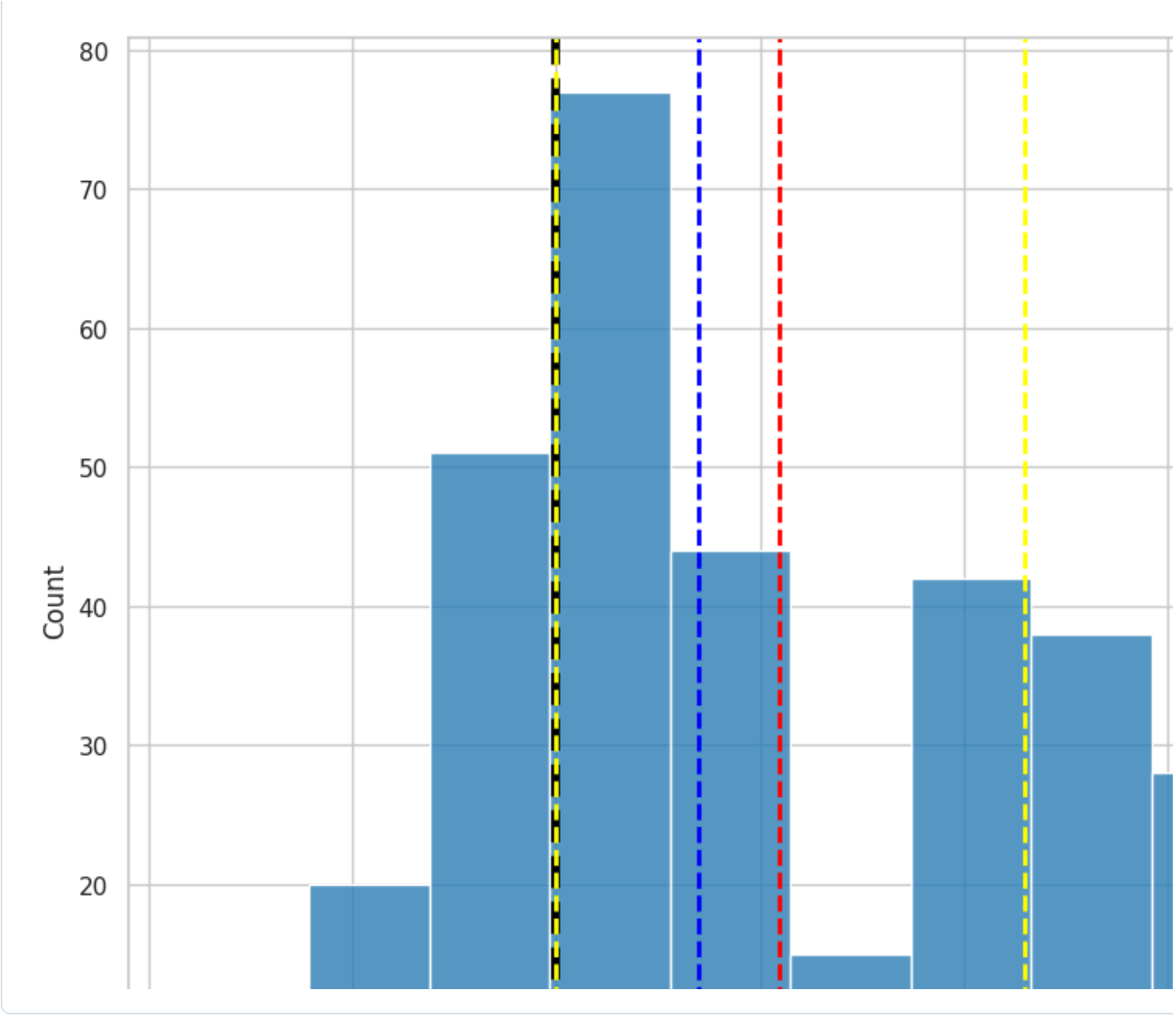
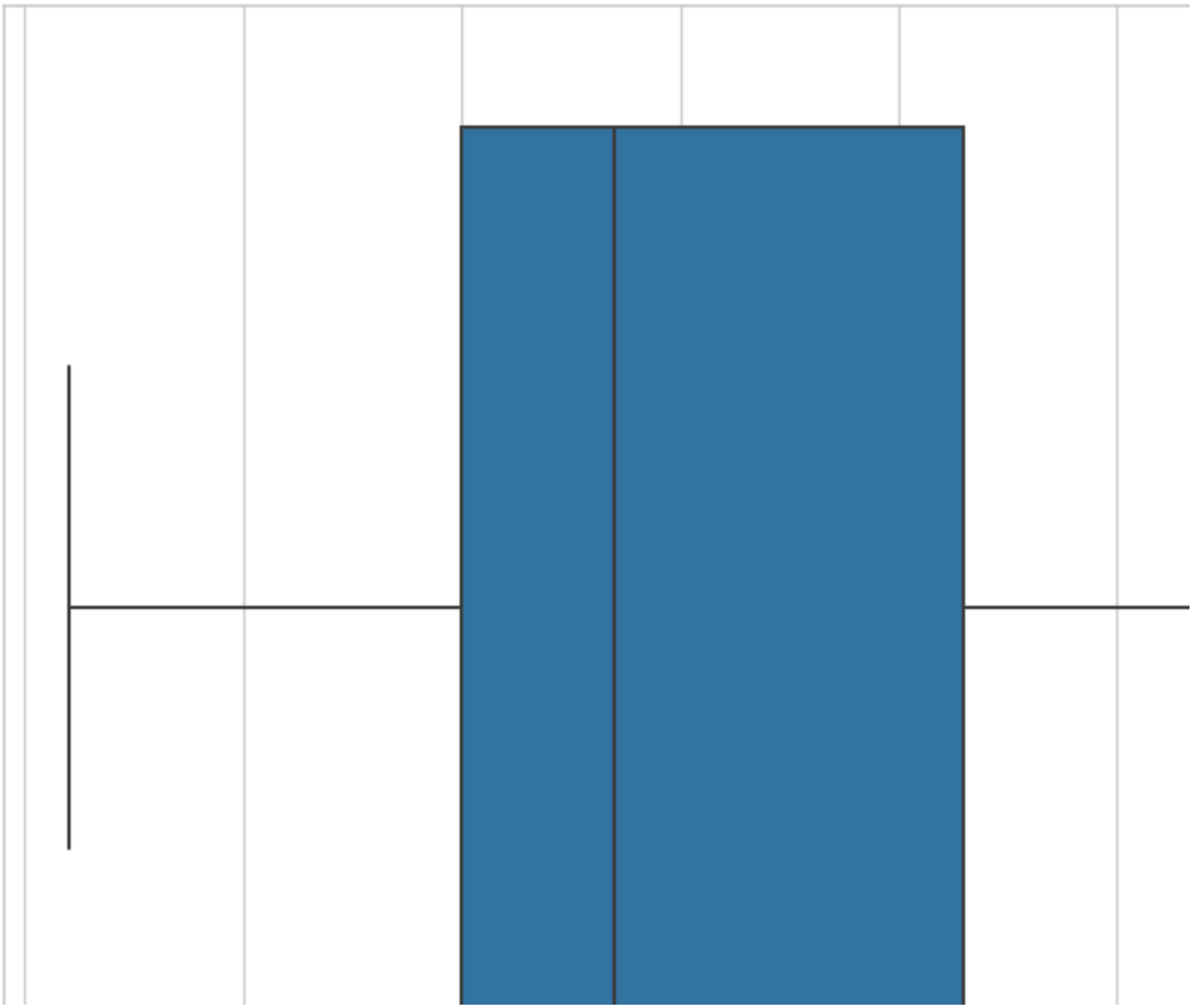


Diagrama de caja / boxplot

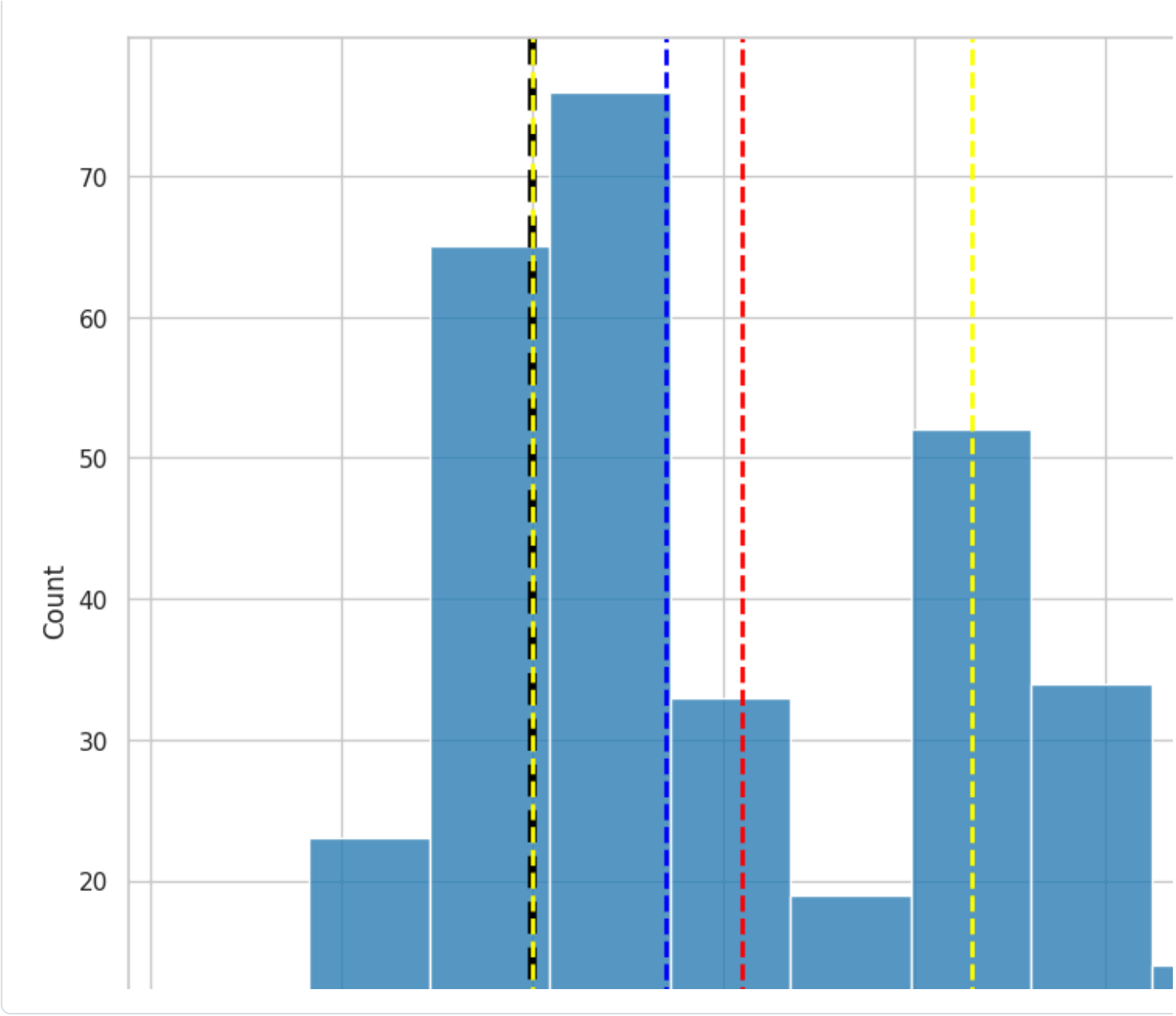
```
sns.boxplot(
    x=processed_penguins_df.flipper_length_mm,
)
```



Limitaciones

```python
def freedman_diaconis_bindwidth(x: pd.Series) -> float:
    """Find optimal bindwidth using Freedman-Diaconis rule."""

    IQR = x.quantile(0.75) - x.quantile(0.25)
    N = x.size

    return 2 * IQR / N ** (1 / 3)
```

```
sns.histplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    binwidth=6.3
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.mean(),
    color='red',
    linestyle='dashed',
    linewidth=2
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.median(),
    color='blue',
    linestyle='dashed',
    linewidth=2
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.mode().values[0],
    color='black',
    linestyle='dashed',
    linewidth=4
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.quantile(0.25),
    color='yellow',
    linestyle='dashed',
    linewidth=2
)

plt.axvline(
    x=processed_penguins_df.flipper_length_mm.quantile(0.75),
    color='yellow',
    linestyle='dashed',
    linewidth=2
)
```

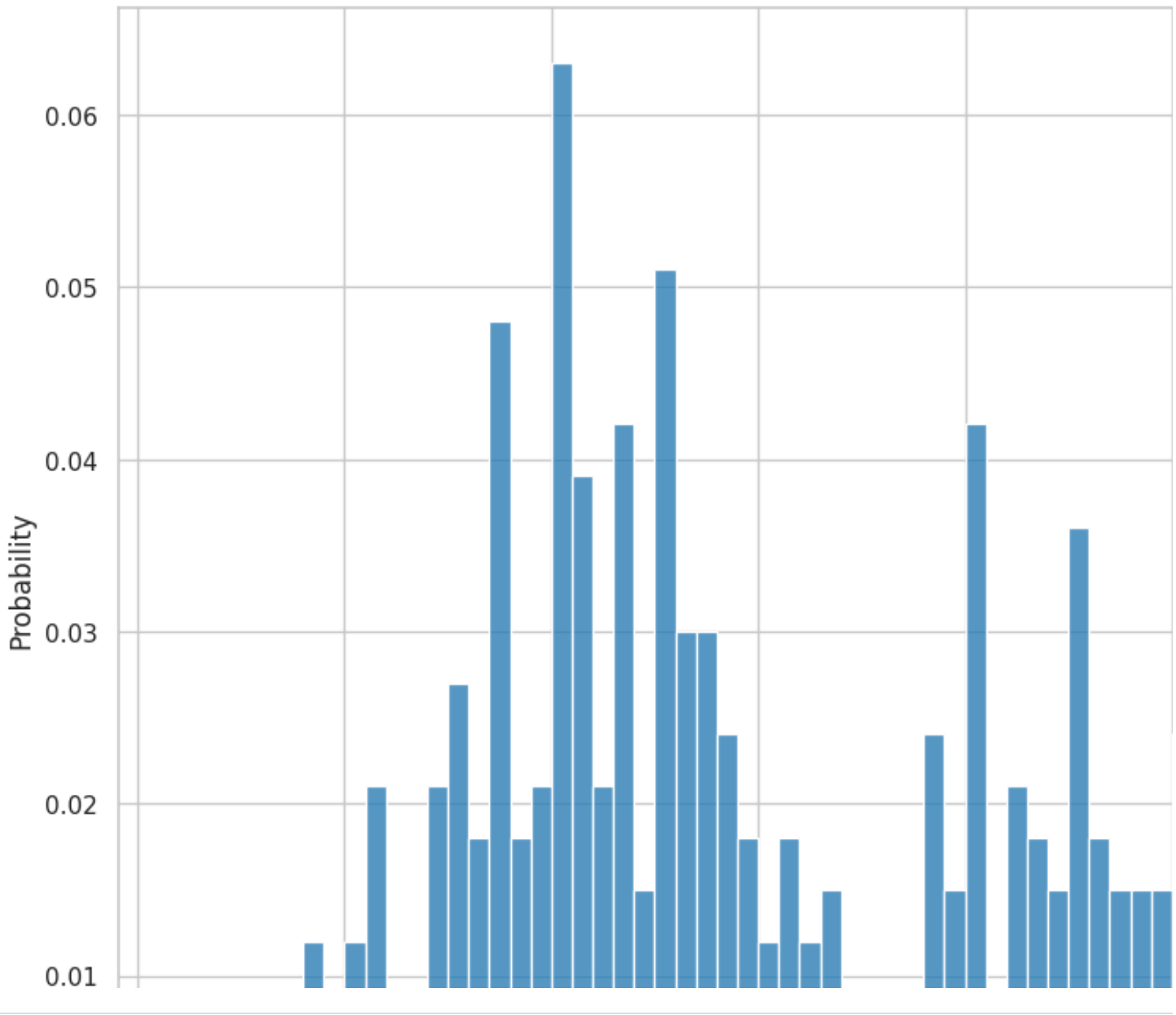


Distribuciones: PMFs, CDFs y PDFs

Funciones de probabilidad de masas (PMFs)

Utilizando `seaborn`

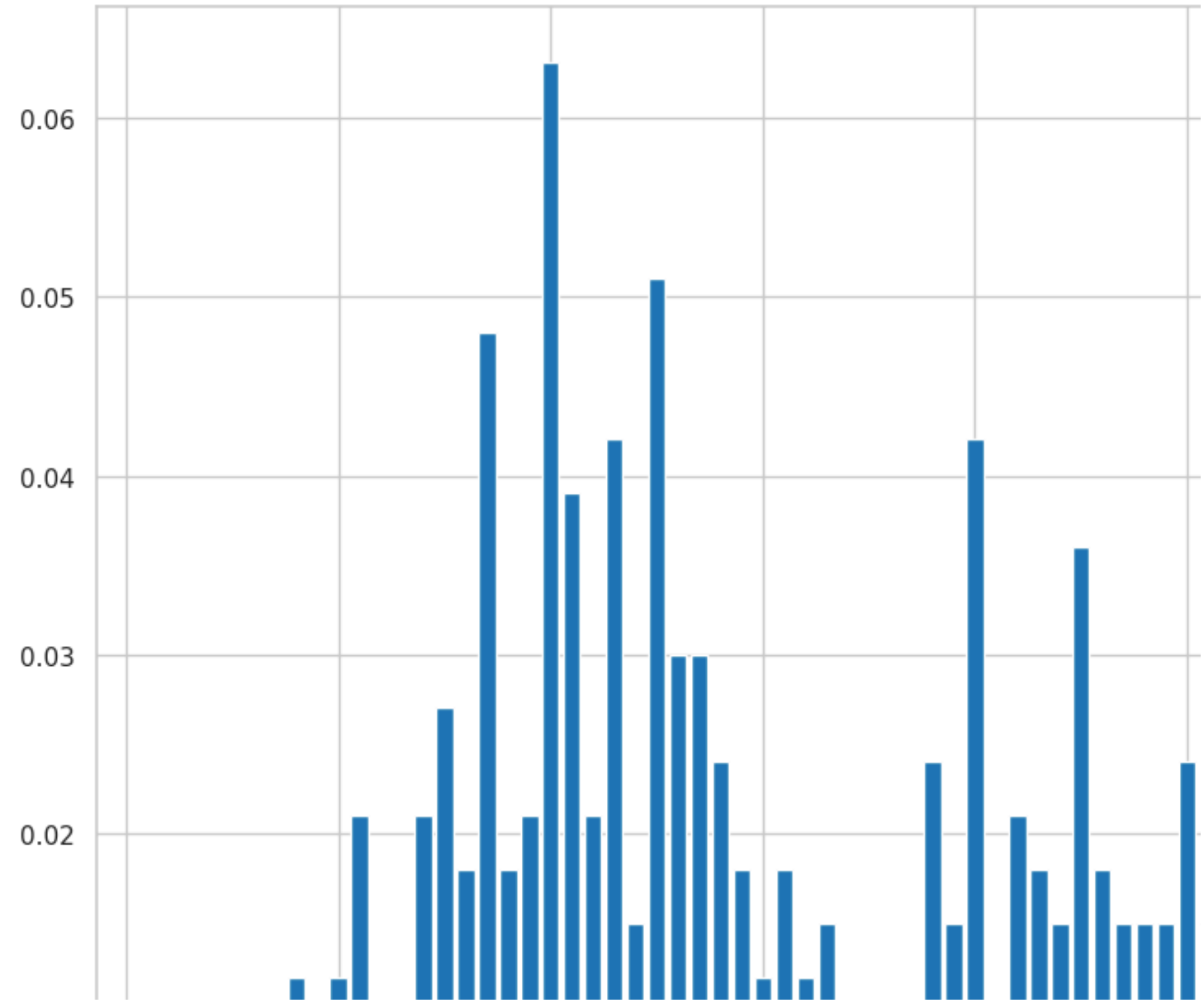
```
sns.histplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    binwidth=1,
    stat='probability'
)
```



Utilizando `empiricaldist`

```
pmf_flipper_length_mm = empiricaldist.Pmf.from_seq(
    processed_penguins_df.flipper_length_mm,
    normalize=True
)
```

```
pmf_flipper_length_mm.bar()
```



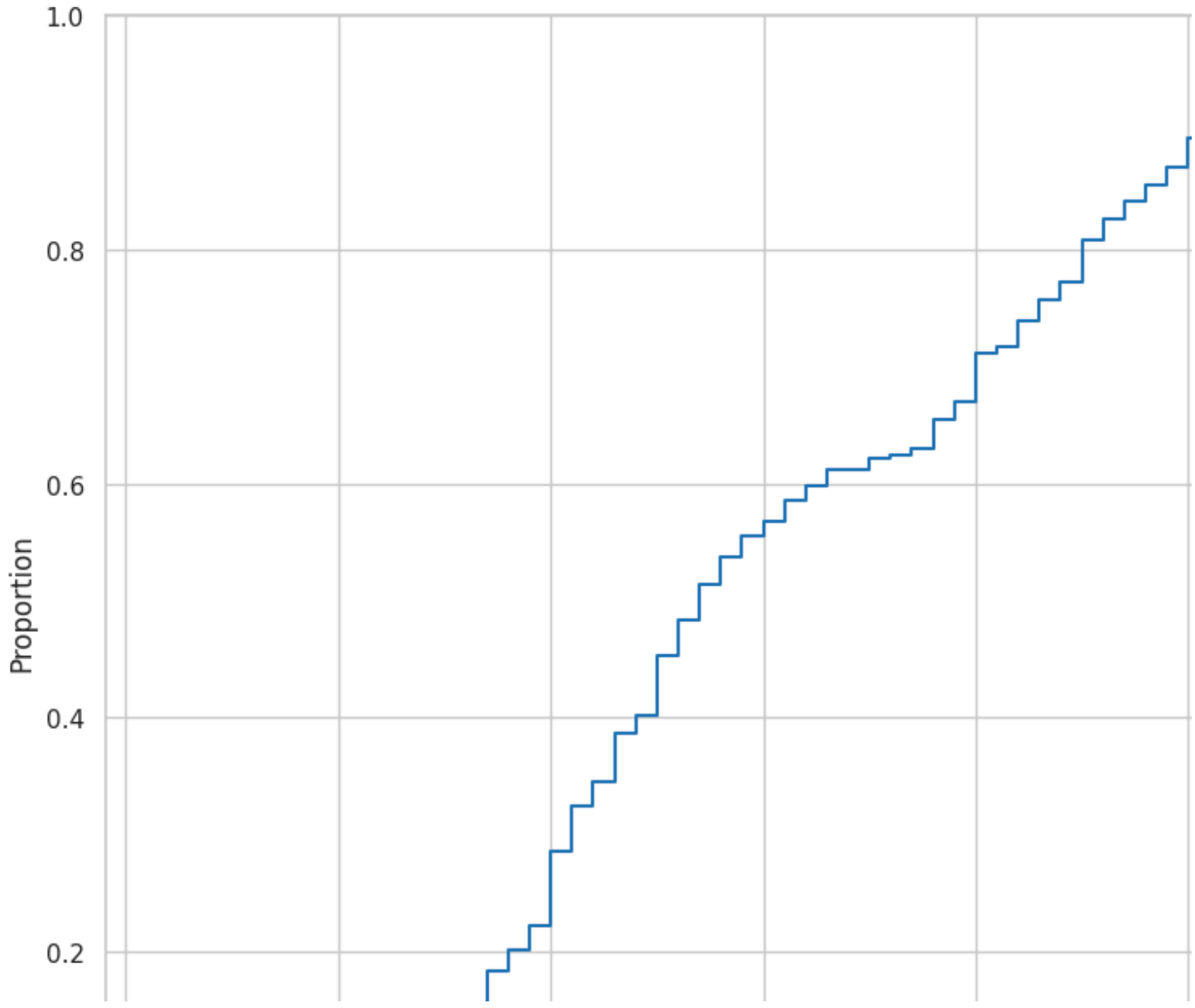

```
pmf_flipper_length_mm(231)
```

```
processed_penguins_df.flipper_length_mm.max()
```

Funciones empirícas de probabilidad acumulada (ECDFs)

Utilizando **seaborn**

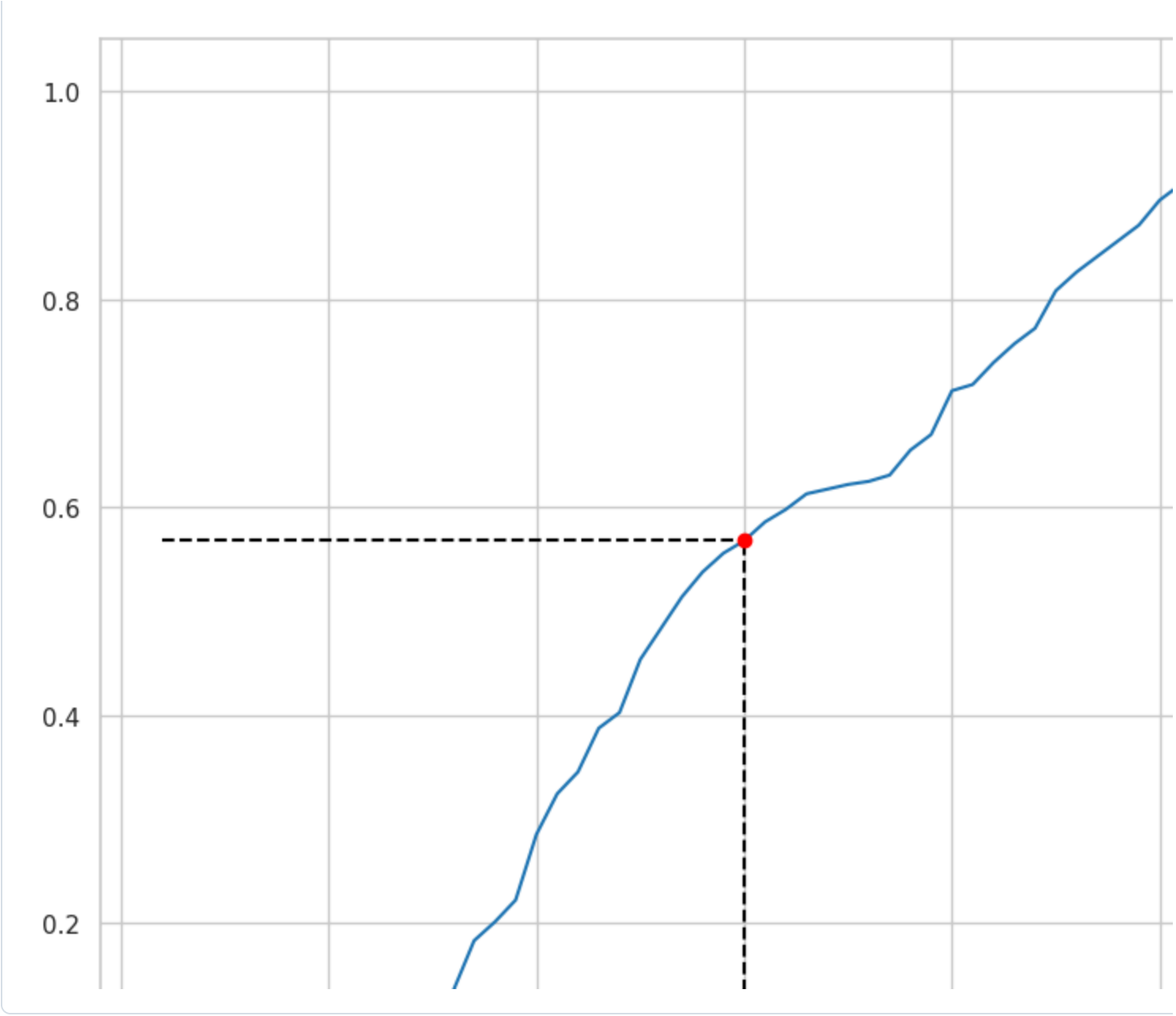
```
sns.ecdfplot(
    data=processed_penguins_df,
    x="flipper_length_mm"
)
```



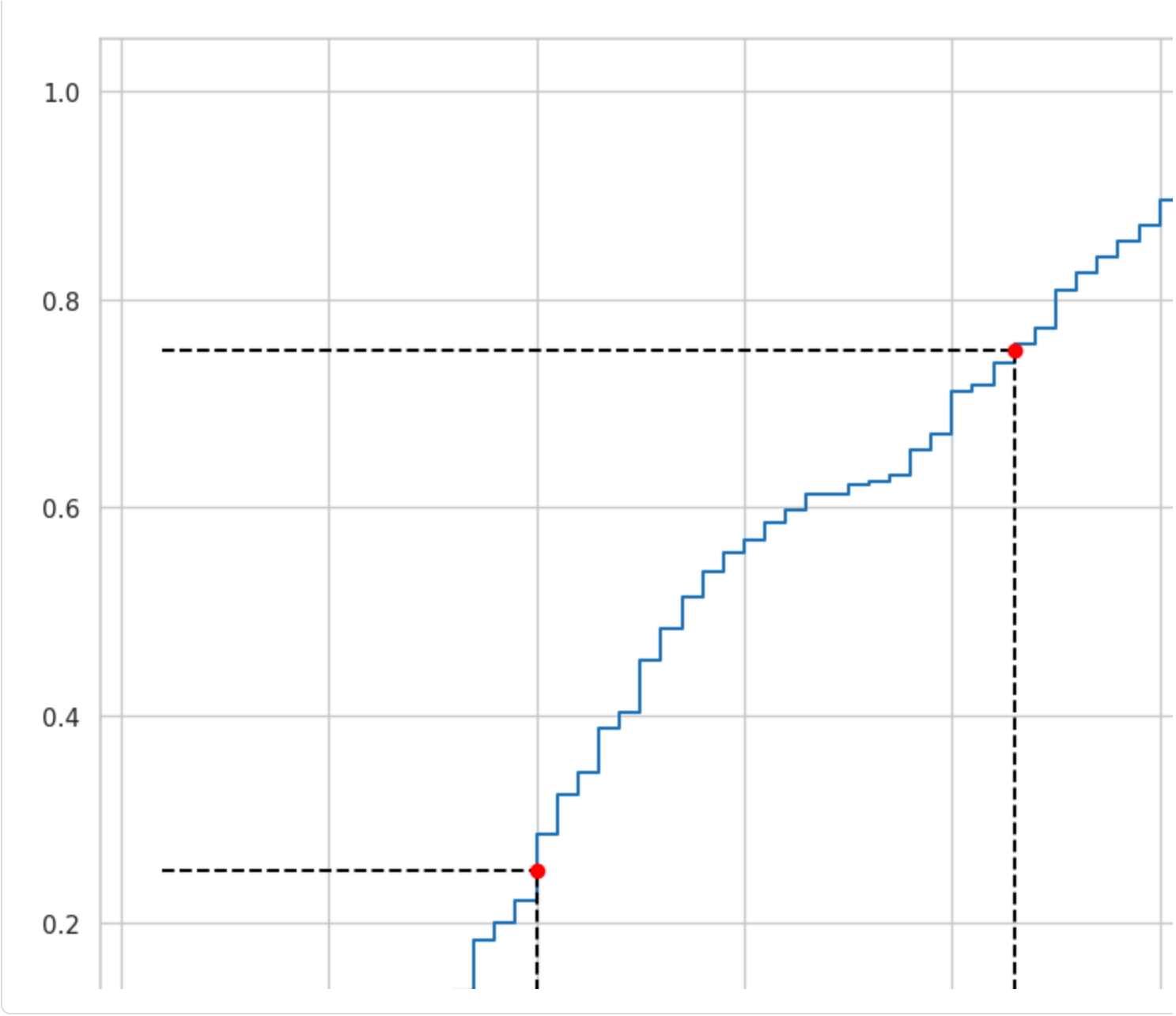
Utilizando `empiricaldist`

```
cdf_flipper_length_mm = empiricaldist.Cdf.from_seq(
    processed_penguins_df.flipper_length_mm,
    normalize=True
)
```

```
cdf_flipper_length_mm.plot()

q = 200 # Specify quantity
p = cdf_flipper_length_mm.forward(q)

plt.vlines(
    x=q,
    ymin=0,
    ymax=p,
    color = 'black',
    linestyle='dashed'
)

plt.hlines(
    y=p,
    xmin=pmf_flipper_length_mm.qs[0],
    xmax=q,
    color='black',
    linestyle='dashed'
)

plt.plot(q, p, 'ro')
```

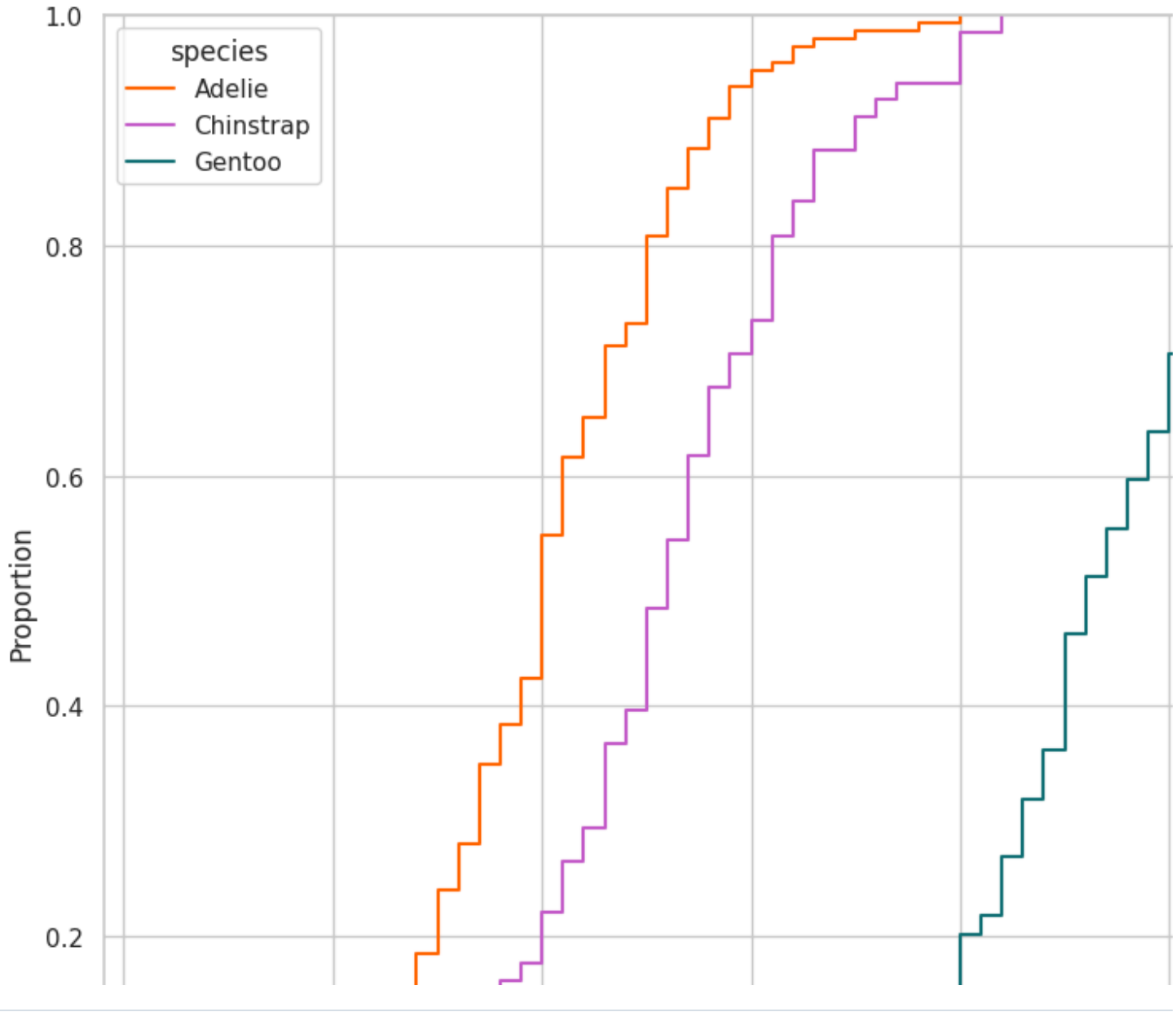

1.0
0.8
0.6
0.4
0.2

```
cdf_flipper_length_mm.step()

p_1 = 0.25 # Specify probability
p_2 = 0.75

ps = (0.25, 0.75) # IQR
qs = cdf_flipper_length_mm.inverse(ps)

plt.vlines(
    x=qs,
    ymin=0,
    ymax=ps,
    color = 'black',
    linestyle='dashed'
)

plt.hlines(
    y=ps,
    xmin=pmf_flipper_length_mm.qs[0],
    xmax=qs,
    color='black',
    linestyle='dashed'
)

plt.scatter(
    x=qs,
    y=ps,
    color='red',
    zorder=2
)
```



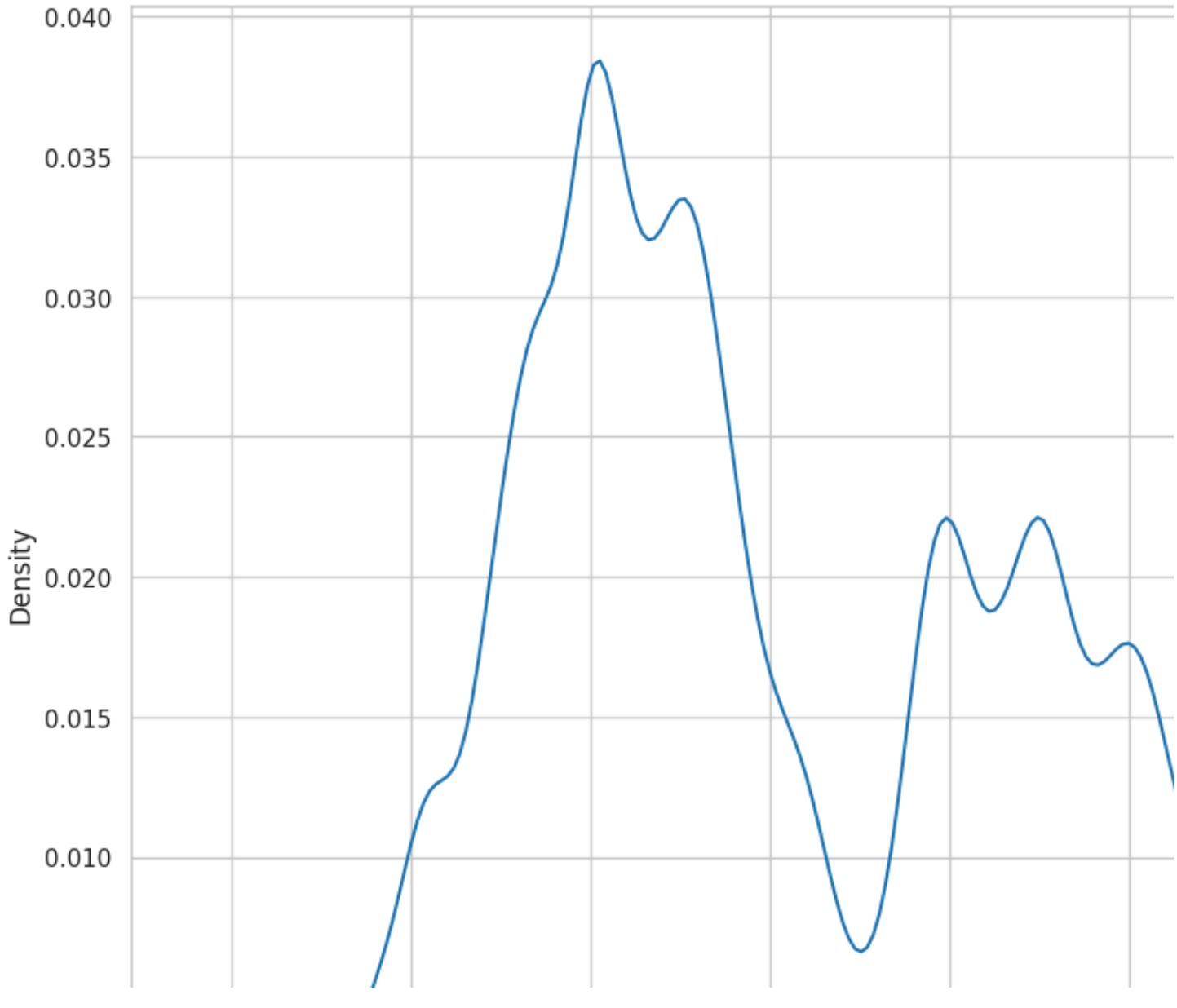
Comparando distribuciones

```
sns.ecdfplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    hue='species',
    palette=penguin_color
)
```



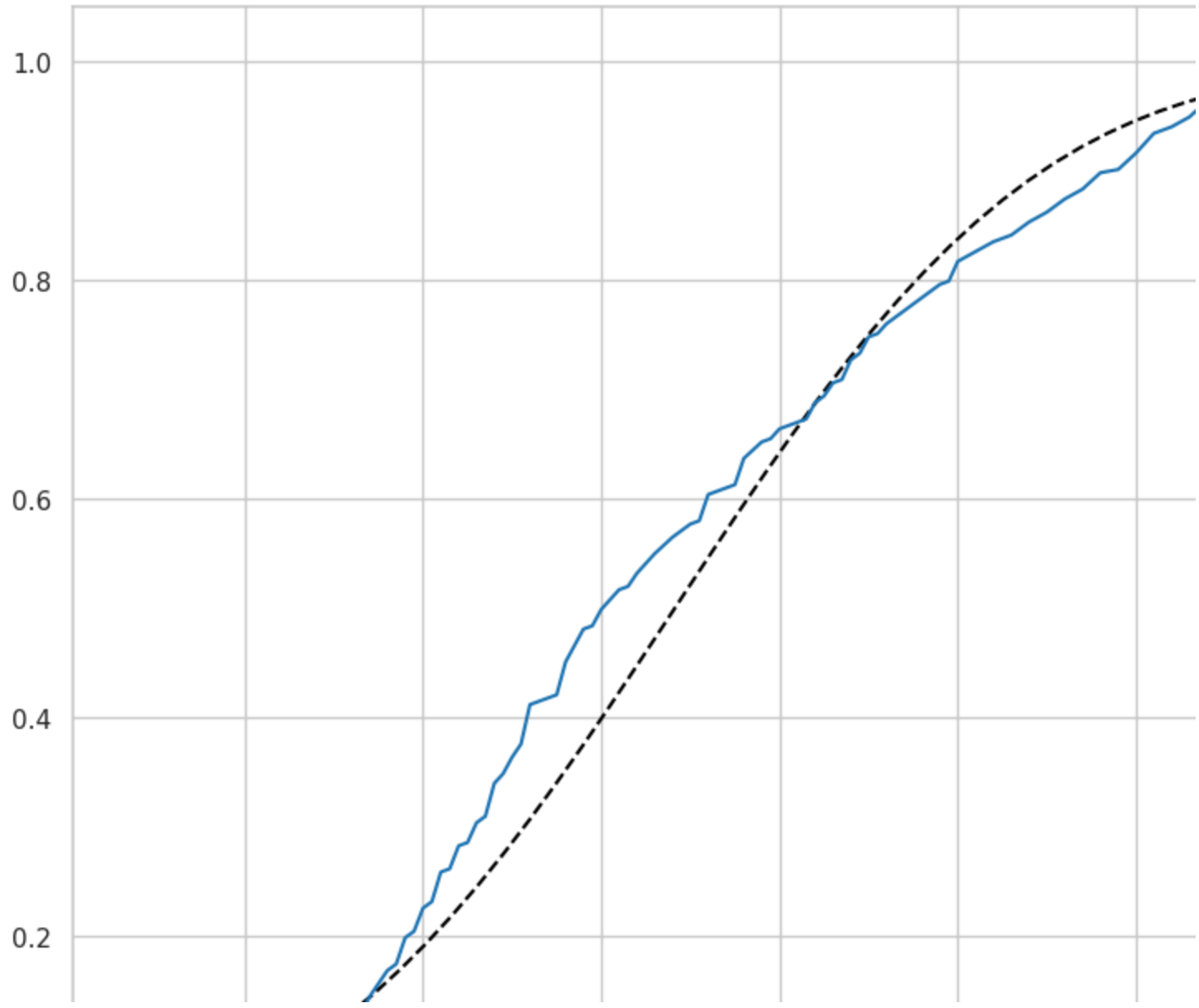
Funciones de densidad de probabilidad

```
sns.kdeplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    bw_method=0.1
)
```

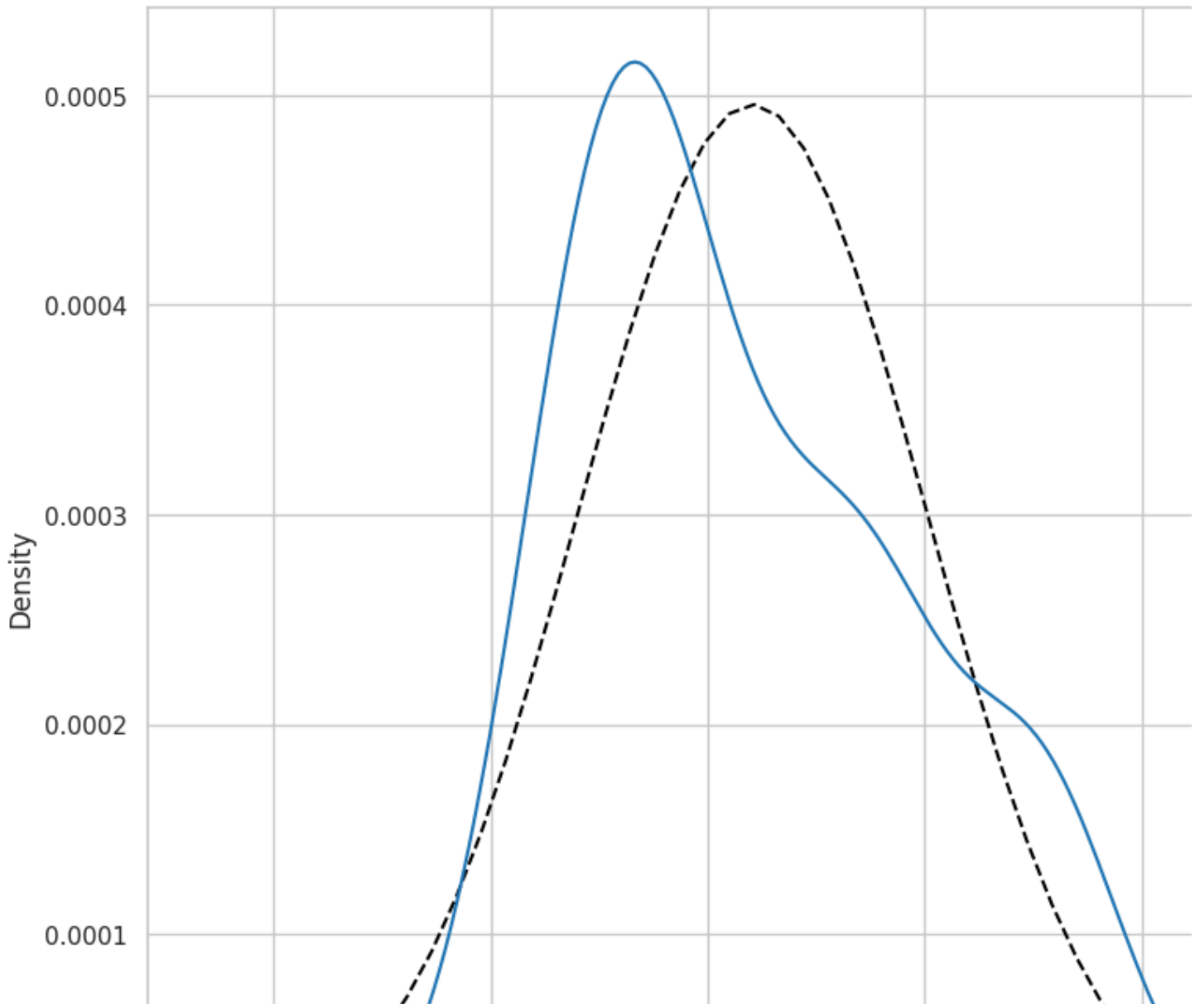



```
stats = processed_penguins_df.body_mass_g.describe()
stats
```

```
np.random.seed(42)
xs = np.linspace(stats['min'], stats['max'])
ys = scipy.stats.norm(stats['mean'], stats['std']).cdf(xs)
plt.plot(xs, ys, color='black', linestyle='dashed')

empiricaldist.Cdf.from_seq(
    processed_penguins_df.body_mass_g,
    normalize=True
).plot()
```



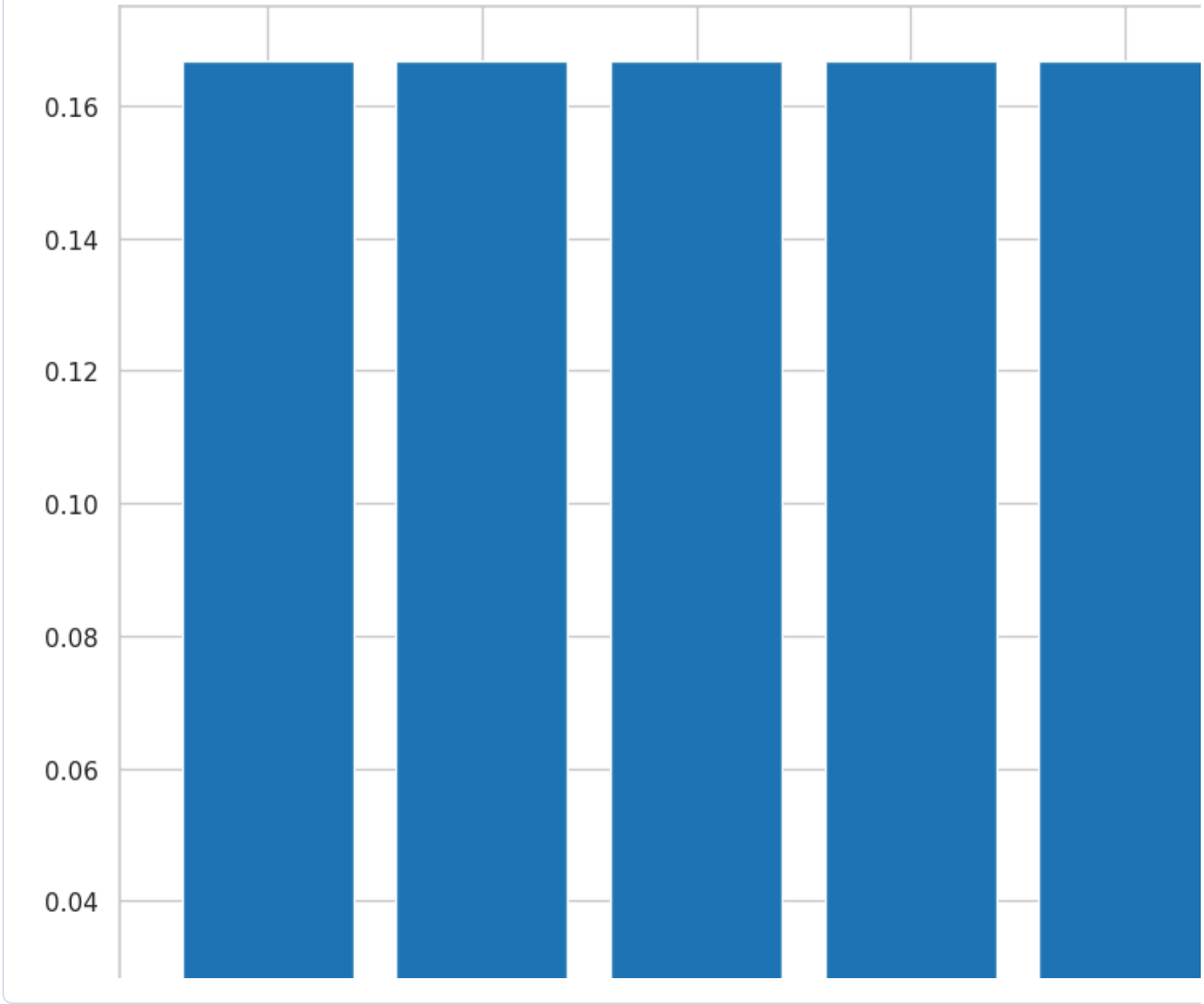
```
xs = np.linspace(stats['min']-1000, stats['max'] + 1000)
ys = scipy.stats.norm(stats['mean'], stats['std']).pdf(xs)
plt.plot(xs, ys, color='black', linestyle='dashed')

sns.kdeplot(
    data=processed_penguins_df,
    x='body_mass_g'
)
```



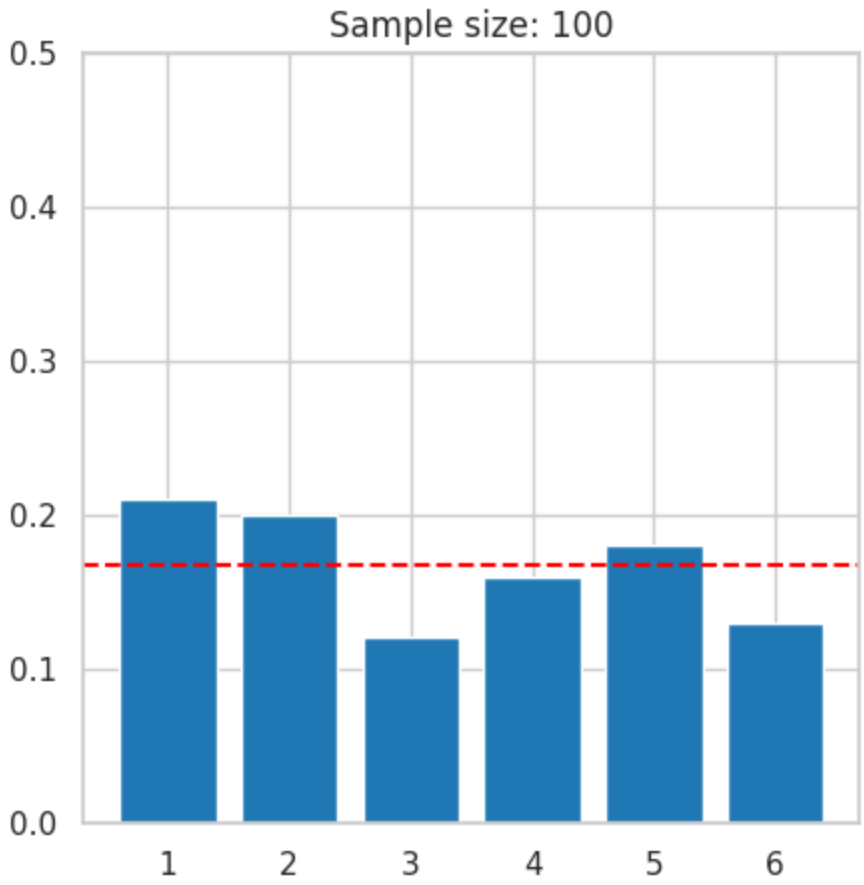
Ley de los Grandes Números y Teorema del Límite C

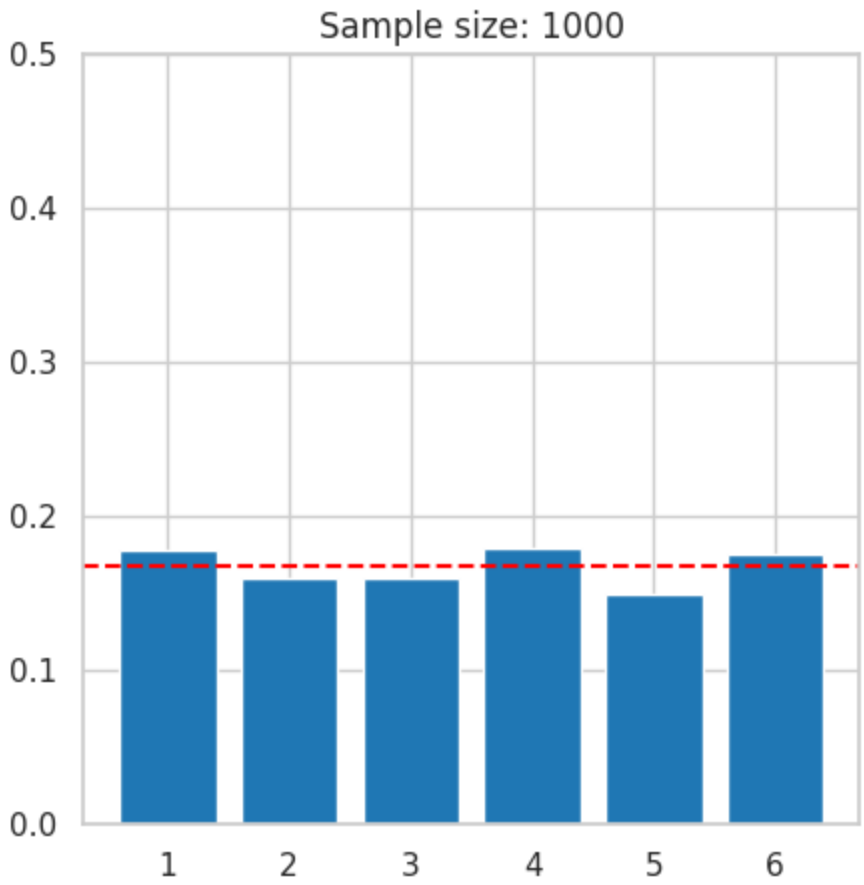
Ley de los Grandes Números

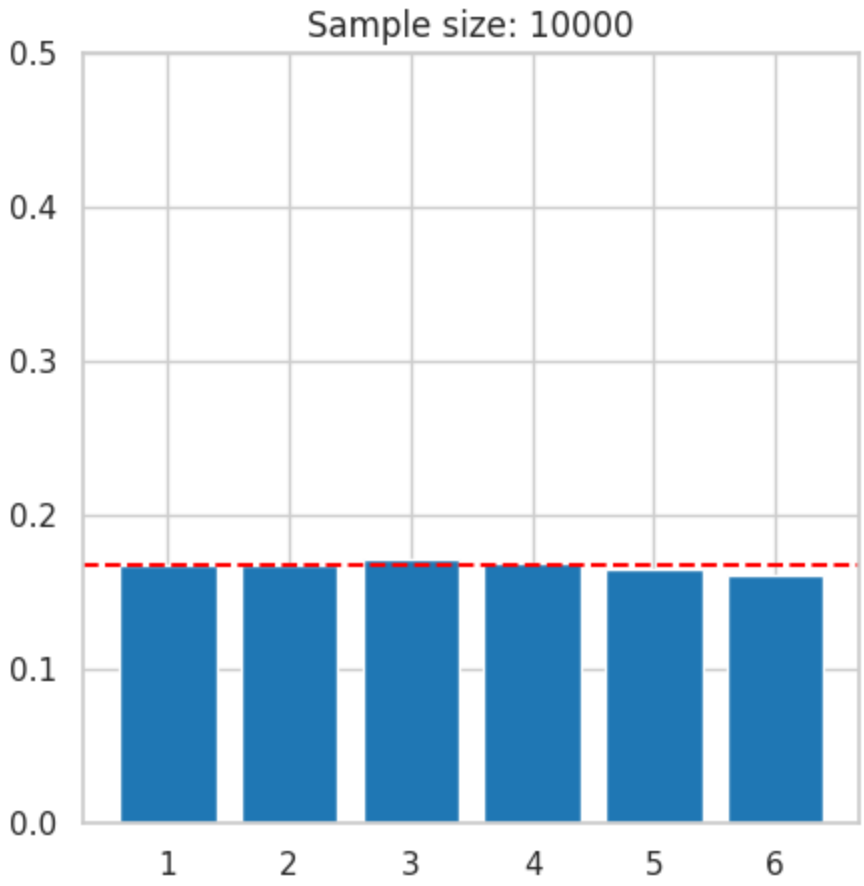
```
dice = empiricaldist.Pmf.from_seq([1, 2, 3, 4, 5, 6])
dice.bar()
```



```
for sample_size in (1e2, 1e3, 1e4):
    sample_size = int(sample_size)
    values = dice.sample(sample_size)
    sample_pmf = empiricaldist.Pmf.from_seq(values)
    plt.figure(figsize=(5,5))
    sample_pmf.bar()
    plt.axhline(y=1/6, color = 'red', linestyle='dashed')
    plt.ylim([0, 0.50])
    plt.title(f"Sample size: {sample_size}")
```




Sample size: 1000
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6


Sample size: 10000
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6

Teorema del límite central

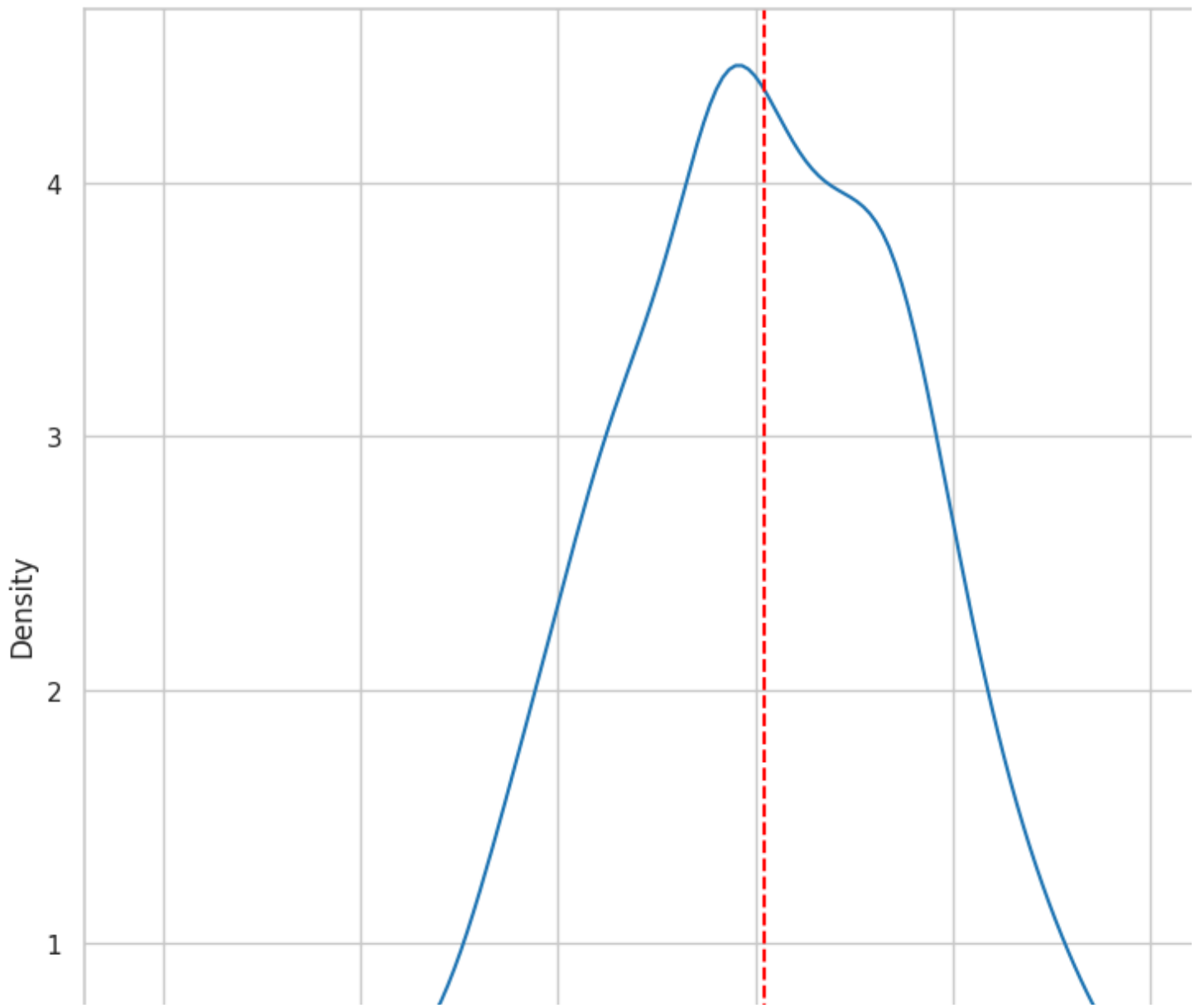
```
processed_penguins_df.sex.value_counts(normalize=True)
```

```
sex_numeric = processed_penguins_df.sex.replace(['Male', 'Female'], [1, 0])
```

```
number_samples = 1000
sample_size=35

samples_df = pd.DataFrame()

np.random.seed(42)
for i in range(1, number_samples + 1):
    sex_numeric_sample = sex_numeric.sample(sample_size, replace=True).to_numpy()
    sample_name = f"sample_{i}"
    samples_df[sample_name] = sex_numeric_sample


male_population_mean = samples_df.mean().mean()
print(f"Estimated percentage of male penguins in population is: {male_population_mean * 100:.
```

```
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
/tmp/ipykernel_144/1550112934.py:10: PerformanceWarning: DataFrame is highly fragmented.  This is usually the
  samples_df[sample_name] = sex_numeric_sample
```

```
sample_means_binomial = pd.DataFrame(samples_df.mean(), columns=['sample_mean'])
sns.kdeplot(data=sample_means_binomial)
plt.axvline(x=sex_numeric.mean(), color='red', linestyle='dashed')
```

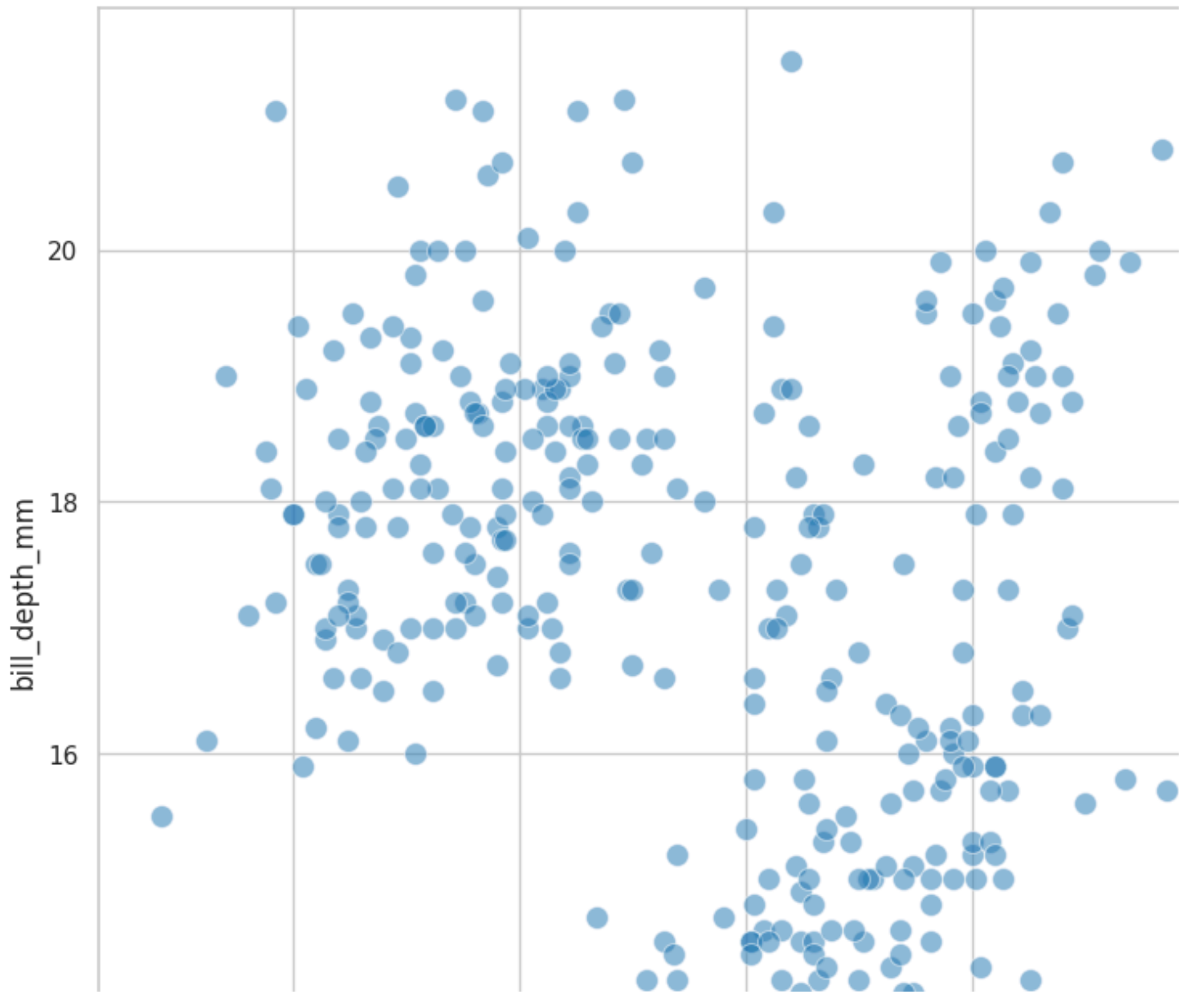



```
sample_size_experiment = pd.DataFrame(
    [[i, samples_df.iloc[:, 0:i].mean().mean().mean()] for i in range(1, number_samples + 1)],
    columns=['sample_size', 'estimated_mean']
)
```

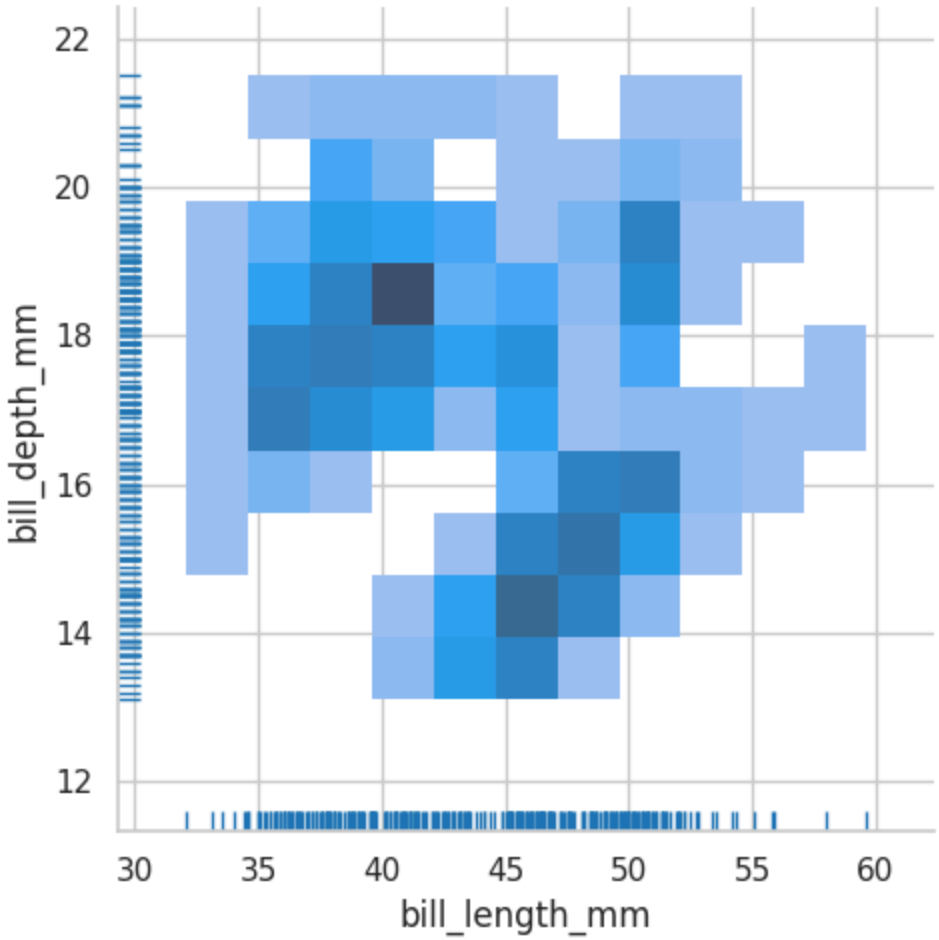
```
sns.scatterplot(
    data=sample_size_experiment,
    x='sample_size',
    y='estimated_mean'
)

plt.axhline(
    y=sex_numeric.mean(),
    color='red',
    linestyle='dashed'
)

plt.ylim([sex_numeric.mean() - 0.20, sex_numeric.mean() + 0.20])
```

Estableciendo relaciones: Gráfica de puntos

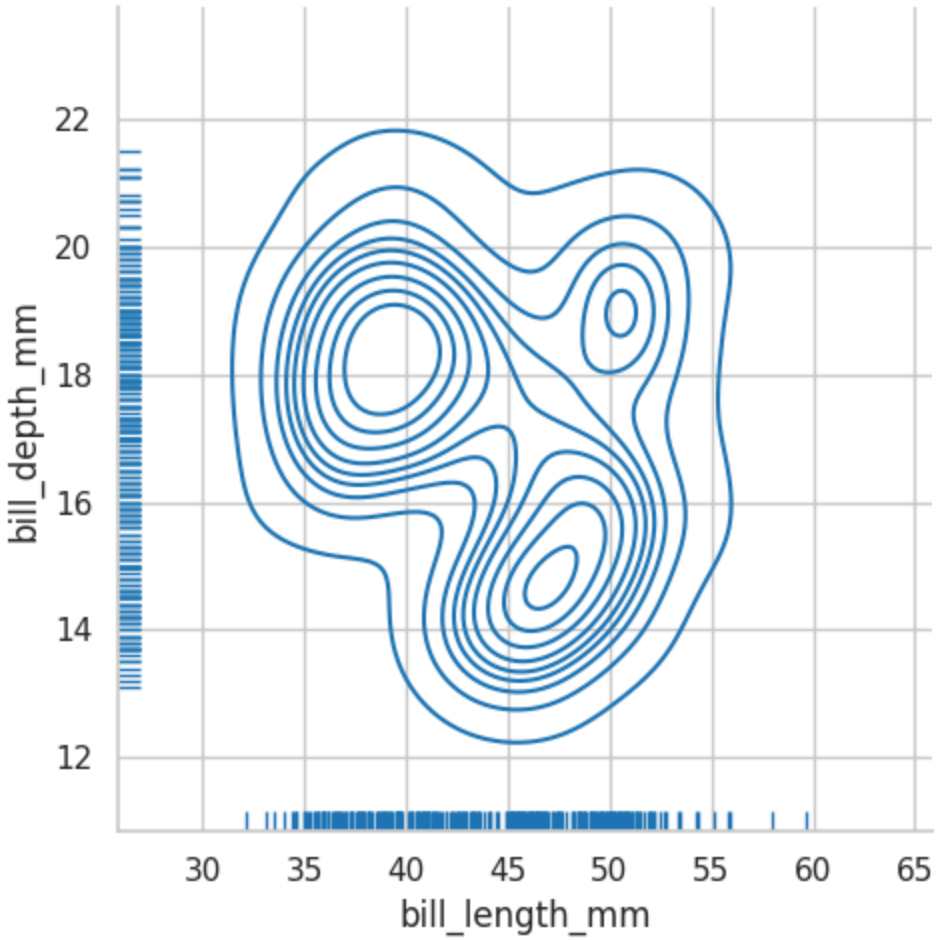
```
sns.scatterplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    alpha=1/2,
    s=100
)
```



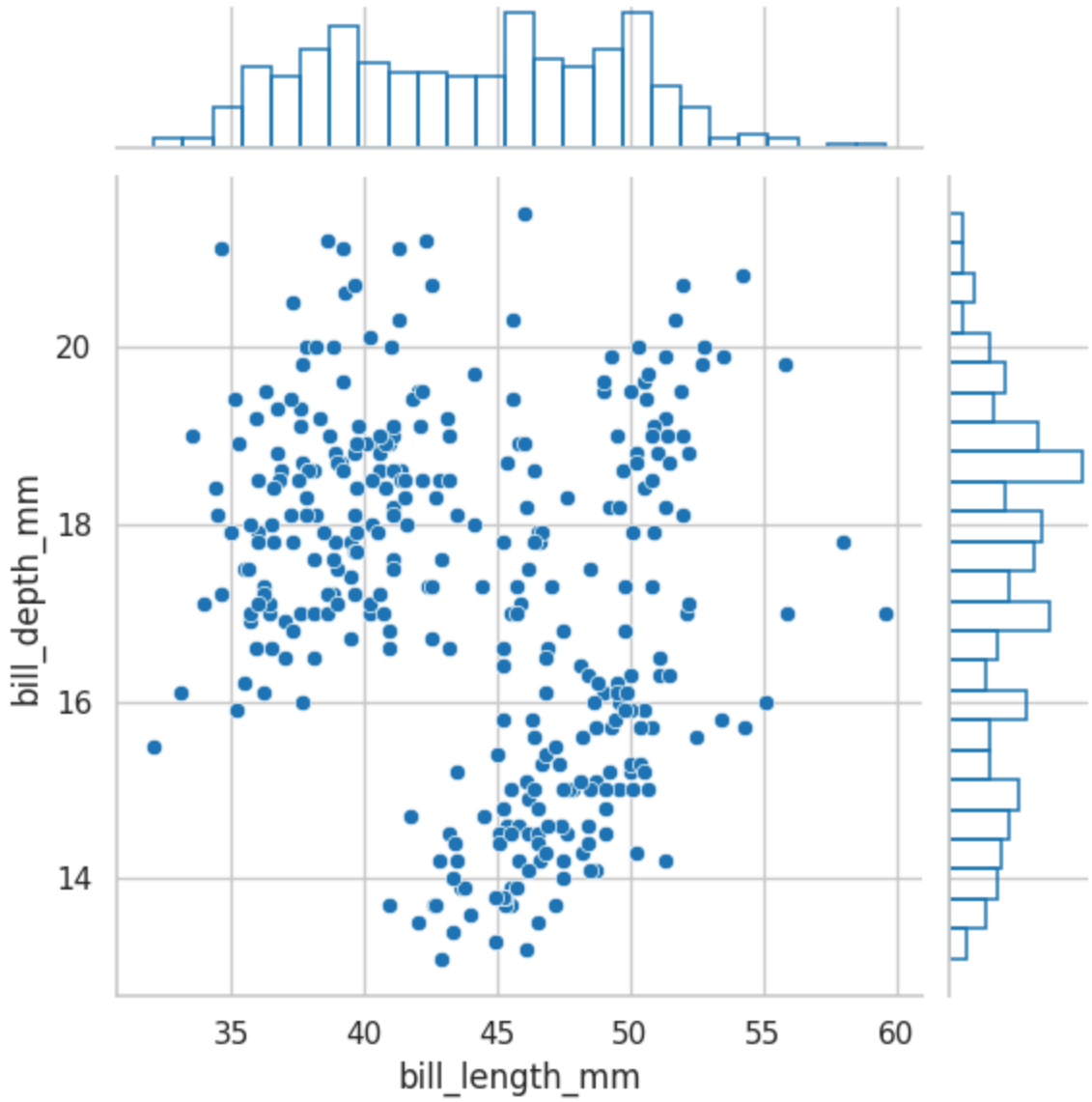
```
sns.displot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    rug=True
)
```



```
sns.displot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    kind='kde',
    rug=True
)
```

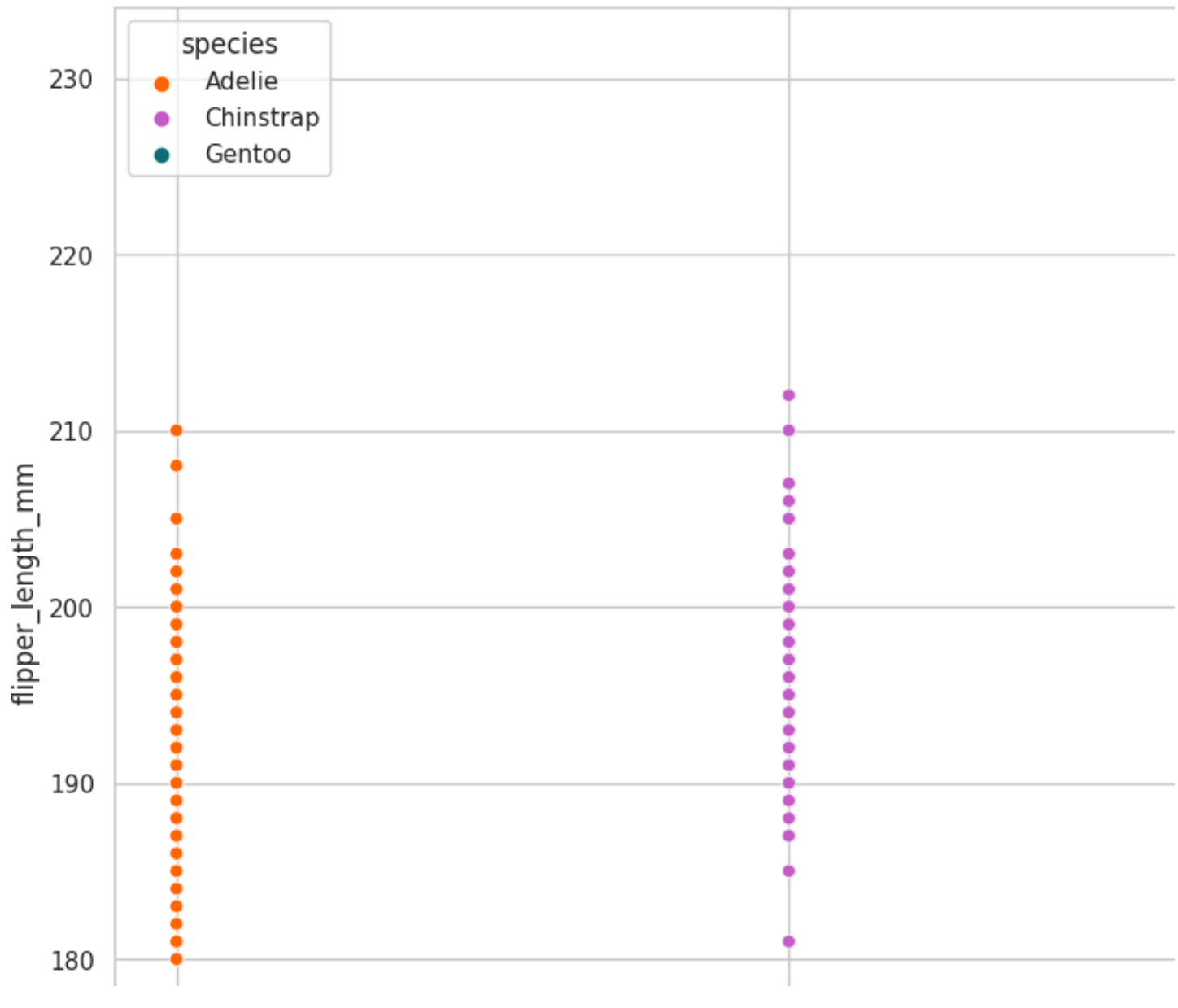


```
sns.jointplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    marginal_kws=dict(bins=25, fill=False)
)
```



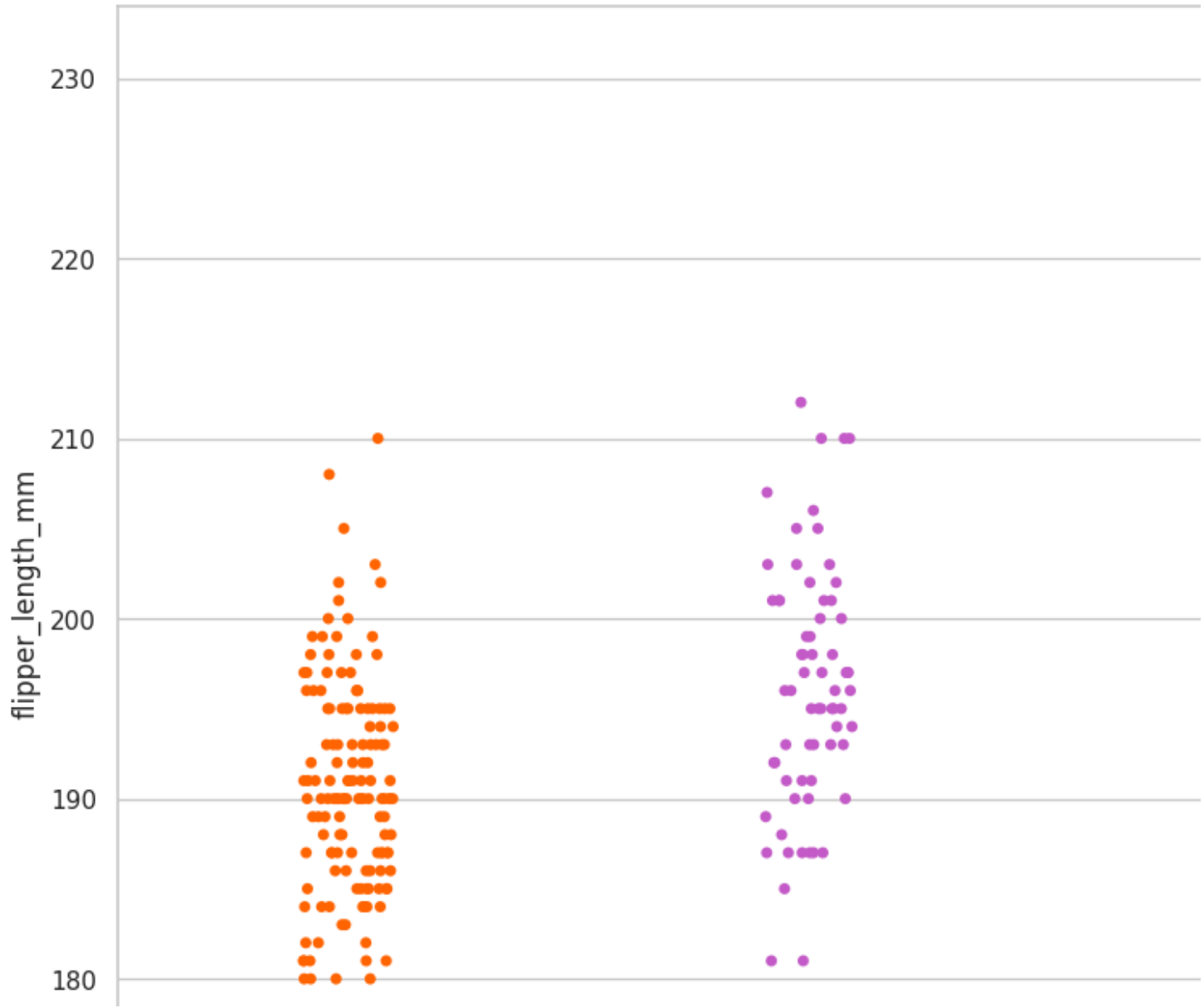
Estableciendo relaciones: Gráficos de violín y boxp

```
sns.scatterplot(
    data=processed_penguins_df,
    x='species',
    y='flipper_length_mm',
    hue='species',
    palette=penguin_color
)
```

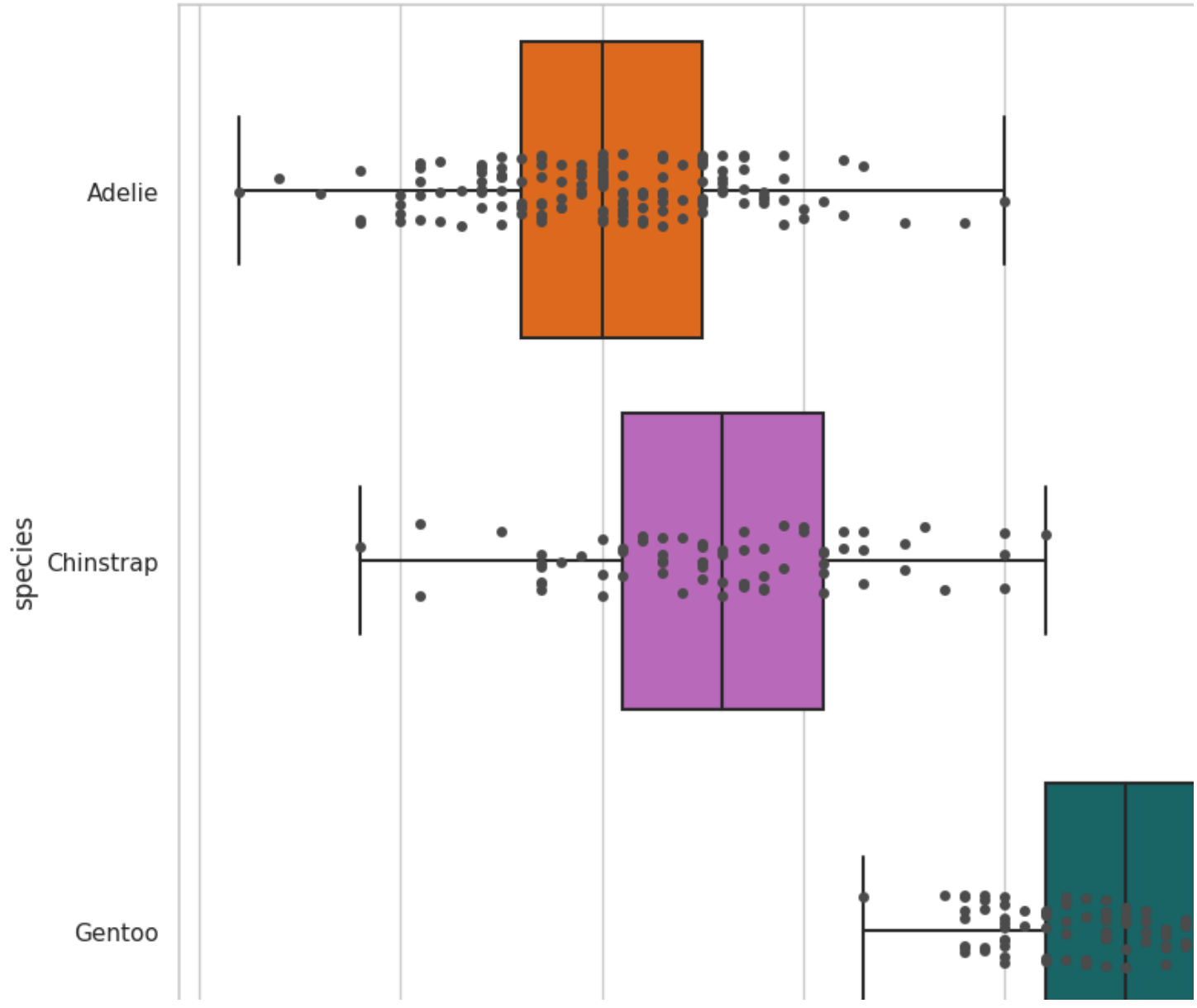


```
sns.stripplot(
    data=processed_penguins_df,
    x='species',
    y='flipper_length_mm',
    palette=penguin_color
)
```

```
/tmp/ipykernel_144/826111948.py:1: FutureWarning: Passing `palette` without assigning `hue` is deprecated.
  sns.stripplot(
```

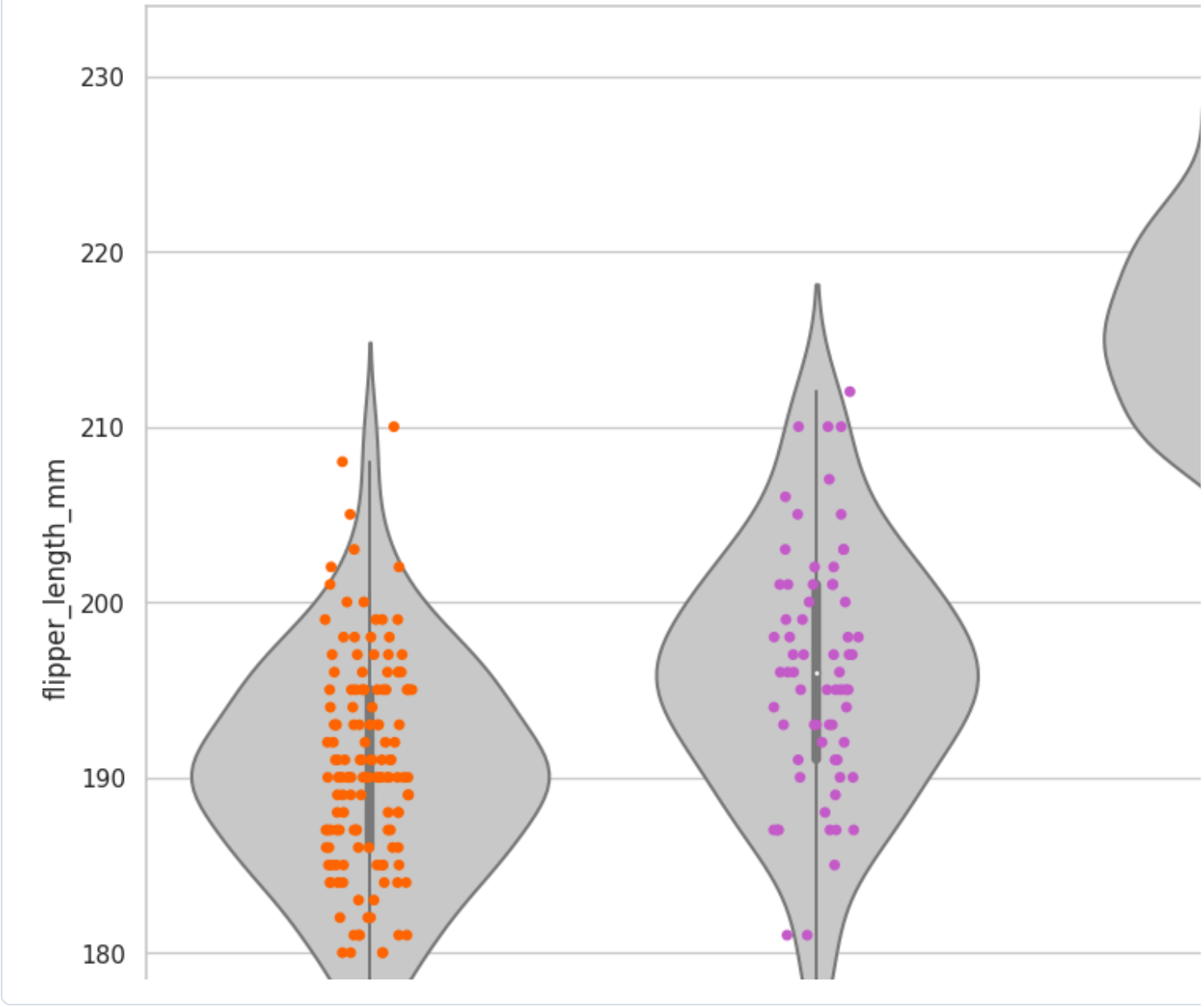


```
ax = sns.boxplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    y='species',
    palette=penguin_color,
    whis=np.inf
)

ax = sns.stripplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    y='species',
    color='.3'
)
```

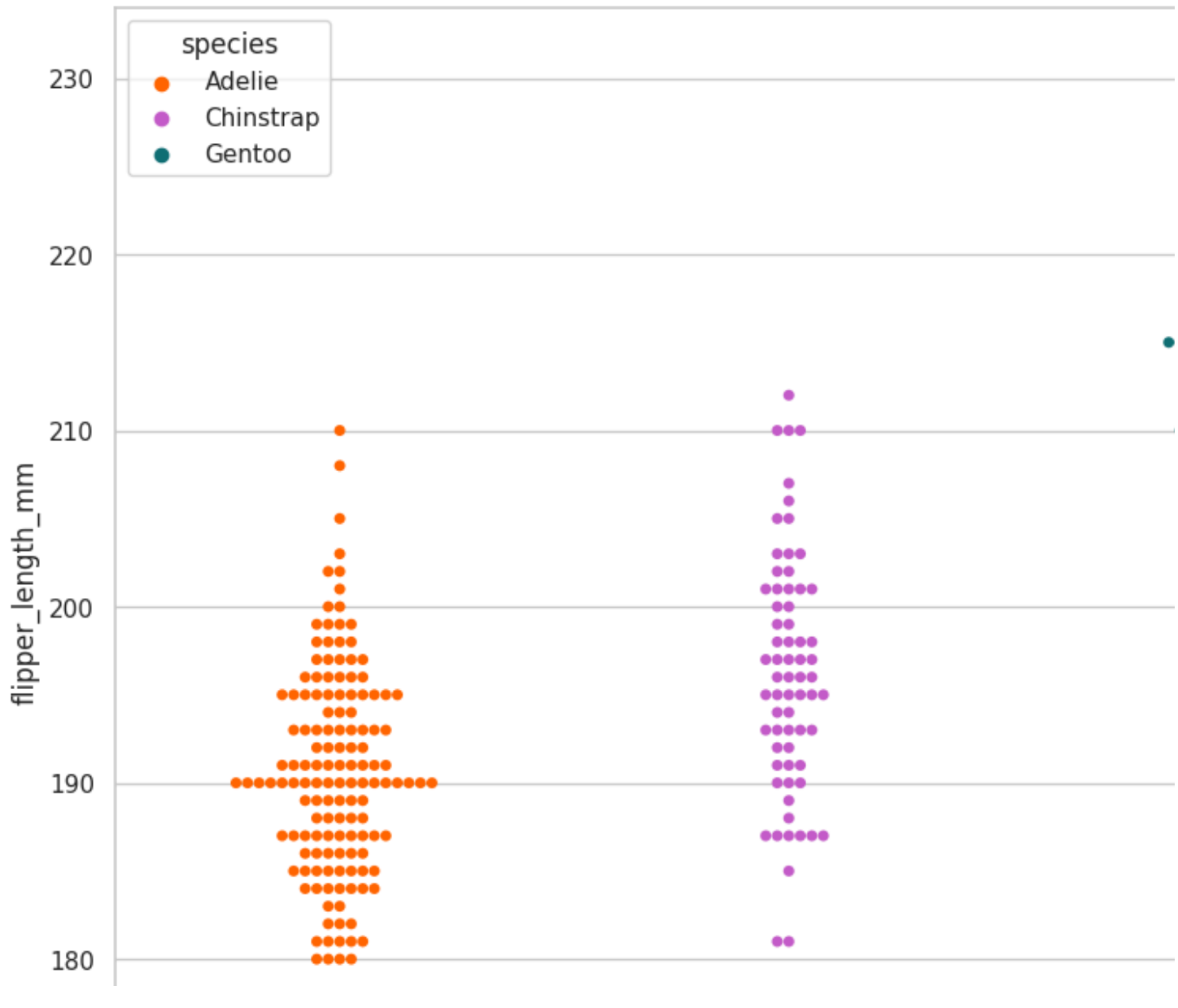


```
ax = sns.violinplot(
    data=processed_penguins_df,
    x='species',
    y='flipper_length_mm',
    color='.8'
)

ax = sns.stripplot(
    data=processed_penguins_df,
    x='species',
    y='flipper_length_mm',
    palette=penguin_color
)
```

```
/tmp/ipykernel_144/1926847696.py:8: FutureWarning: Passing `palette` without assigning `hue` is deprecated.
  ax = sns.stripplot(
```


230
220
210
200
190
180
flipper_length_mm

```
sns.swarmplot(
    data=processed_penguins_df,
    x='species',
    y='flipper_length_mm',
    hue='species',
    palette=penguin_color
)
```



Estableciendo relaciones: Matrices de correlación

¿Existe una correlación lineal entre alguna de nuestras variables?

```
processed_penguins_df.corr()
```

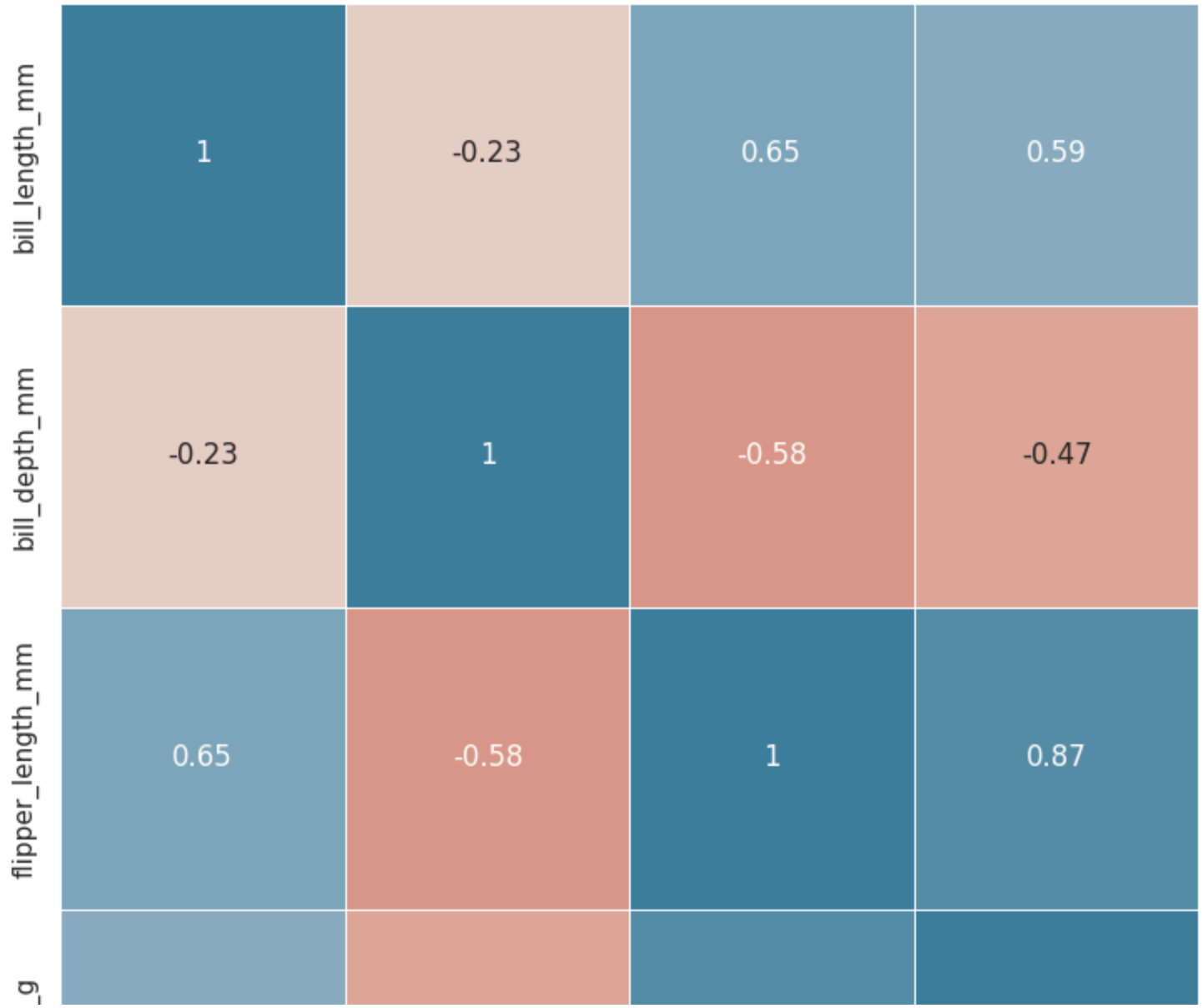
```
/tmp/ipykernel_144/4090656914.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is depr
  processed_penguins_df.corr()
```

	bill_length_mm fl...	bill_depth_mm flo...	flipper_length_m...	body_mass_g floa...	
bill...	1.0	-0.22862563591...	0.653095638667..	0.5894511101769..	
bill...	-0.22862563591...	1.0	-0.57779169633...	-0.472015660195..	
flip...	0.653095638667..	-0.57779169633...	1.0	0.872978898565..	
bo...	0.5894511101769..	-0.472015660195..	0.872978898565..	1.0	

¿Como puedo visualizar los coeficientes de correlación?

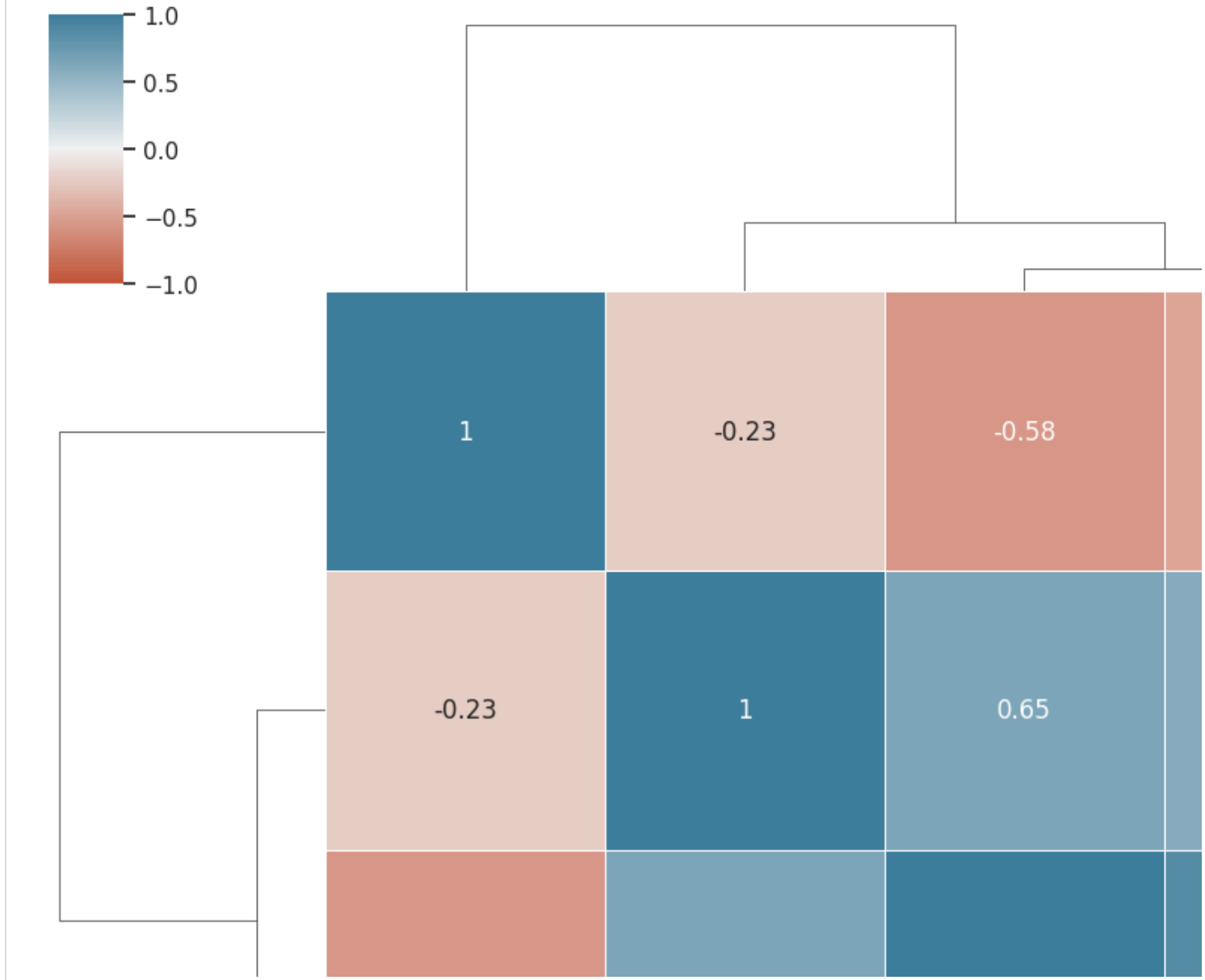
```
sns.heatmap(
    data=processed_penguins_df.corr(),
    cmap=sns.diverging_palette(20, 230, as_cmap=True),
    center=0,
    vmin=-1,
    vmax=1,
    linewidths=0.5,
    cbar_kws={"shrink": 0.5},
    annot=True
)
```

```
/tmp/ipykernel_144/1293620812.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is depr
  data=processed_penguins_df.corr(),
```


bill_length_mm
bill_depth_mm
flipper_length_mm
_g
1
-0.23
0.65
0.59
-0.23
1
-0.58
-0.47
0.65
-0.58
1
0.87

```
sns.clustermap(
    data=processed_penguins_df.corr(),
    cmap= sns.diverging_palette(20, 230, as_cmap=True), # 'BrBG'
    center=0,
    vmin=-1,
    vmax=1,
    linewidths=0.5,
    cbar_kws={"shrink": 0.5},
    annot=True
)
```

```
/tmp/ipykernel_144/873204341.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is depre
  data=processed_penguins_df.corr(),
```

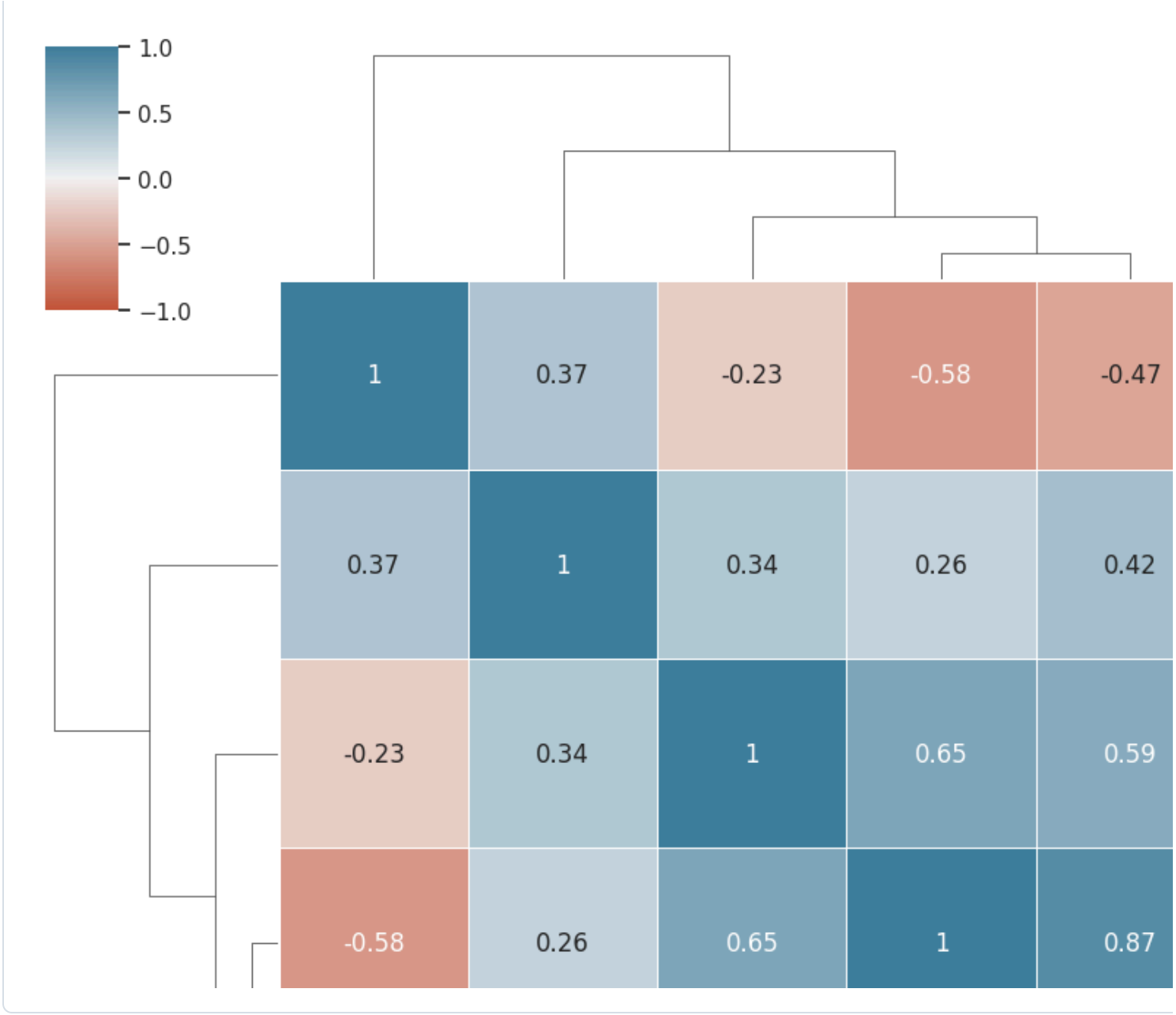


¿Cómo podría representar una variable categórica como numéric

```
processed_penguins_df = (
    processed_penguins_df
    .assign(
        numeric_sex=lambda df: df.sex.replace(['Female', 'Male'], [0, 1])

    )
)
```

```
sns.clustermap(
    data=processed_penguins_df.corr(),
    cmap= sns.diverging_palette(20, 230, as_cmap=True), # 'BrBG'
    center=0,
    vmin=-1,
    vmax=1,
    linewidths=0.5,
    cbar_kws={"shrink": 0.5},
    annot=True
)
```

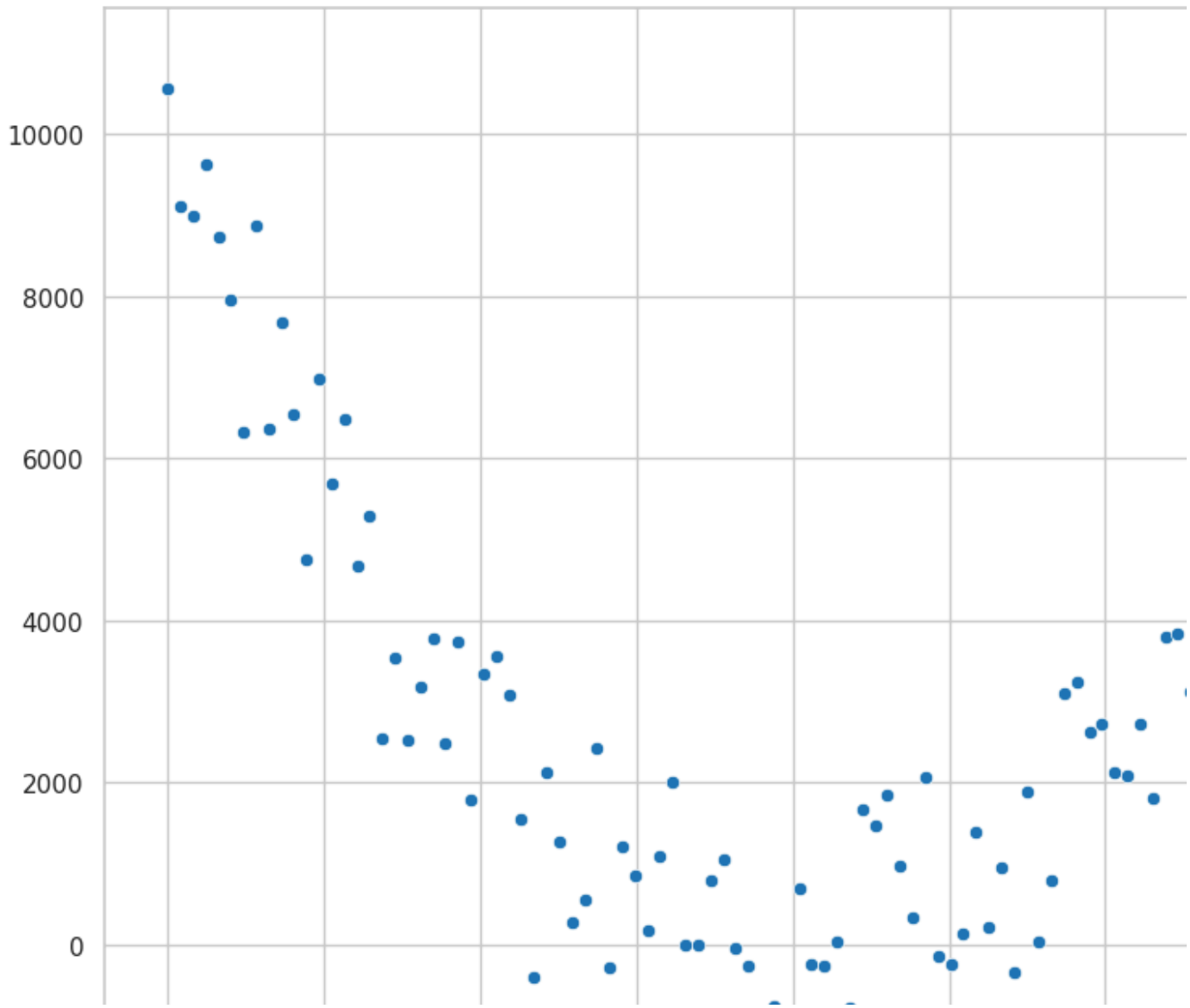
```
/tmp/ipykernel_144/873204341.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is depre
  data=processed_penguins_df.corr(),
```



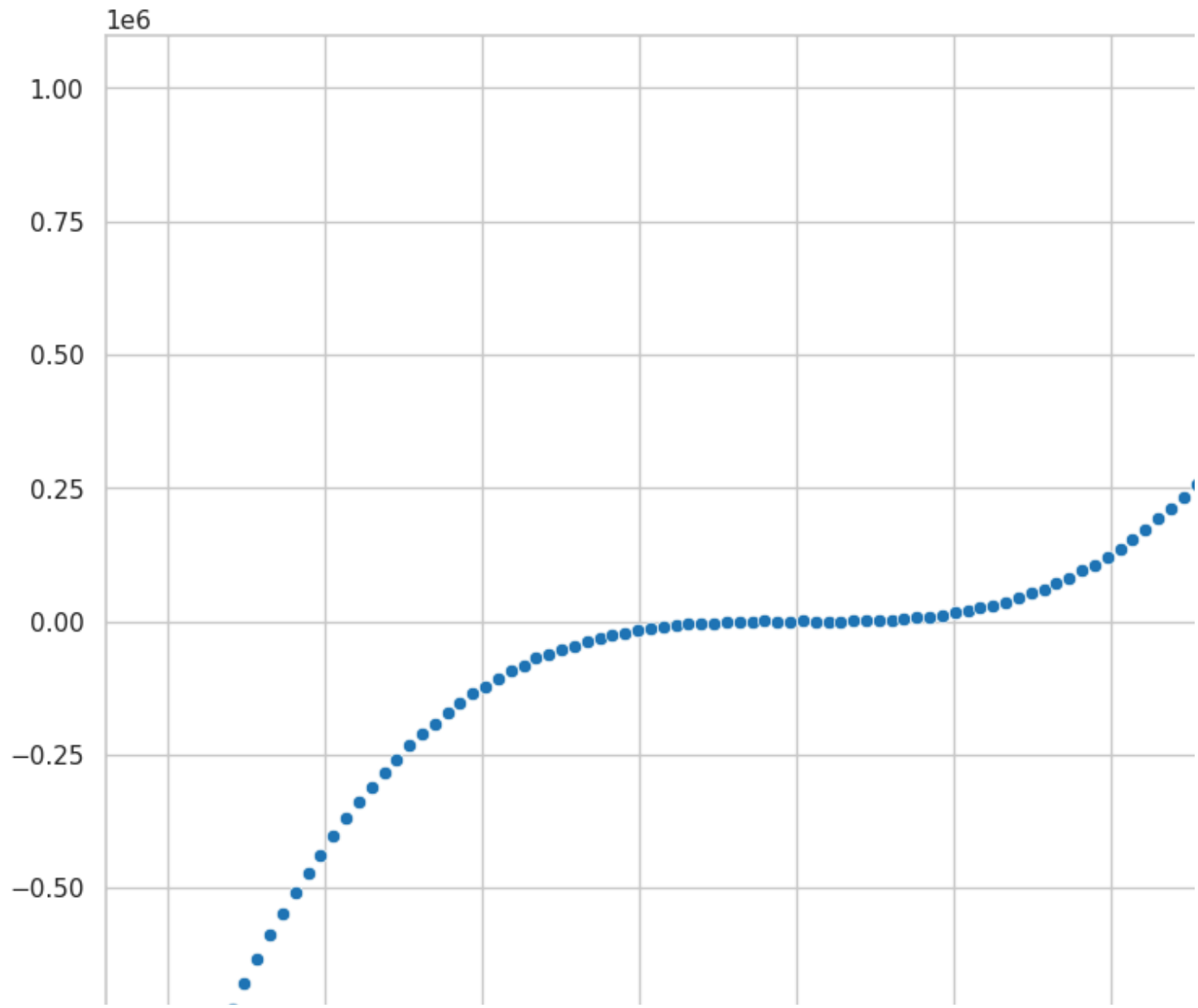
¿Cuál es una límitante de los coeficientes de correlación lineal?

Sólo nos ayuda a determinar la posible existencia de una correlación lineal, sir ausencia no significa que no exista otro tipo de correlación

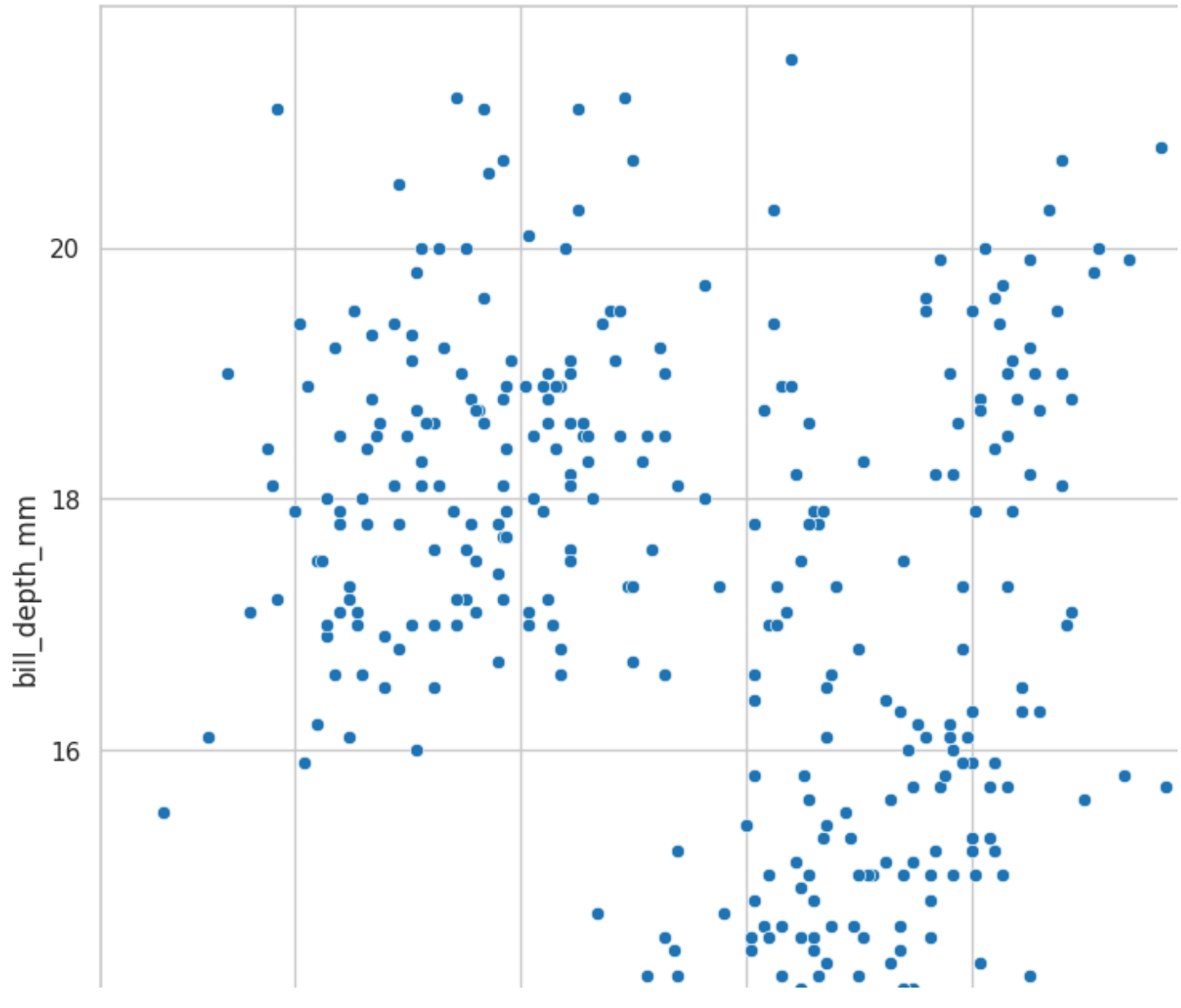
```
x = np.linspace(-100, 100, 100)
y = x ** 2
y += np.random.normal(0, 1000, x.size)

sns.scatterplot(x=x, y=y)
np.corrcoef(x, y)
```



```
x = np.linspace(-100, 100, 100)
y = x ** 3
y += np.random.normal(0, 1000, x.size)

sns.scatterplot(x=x, y=y)
np.corrcoef(x, y)
```



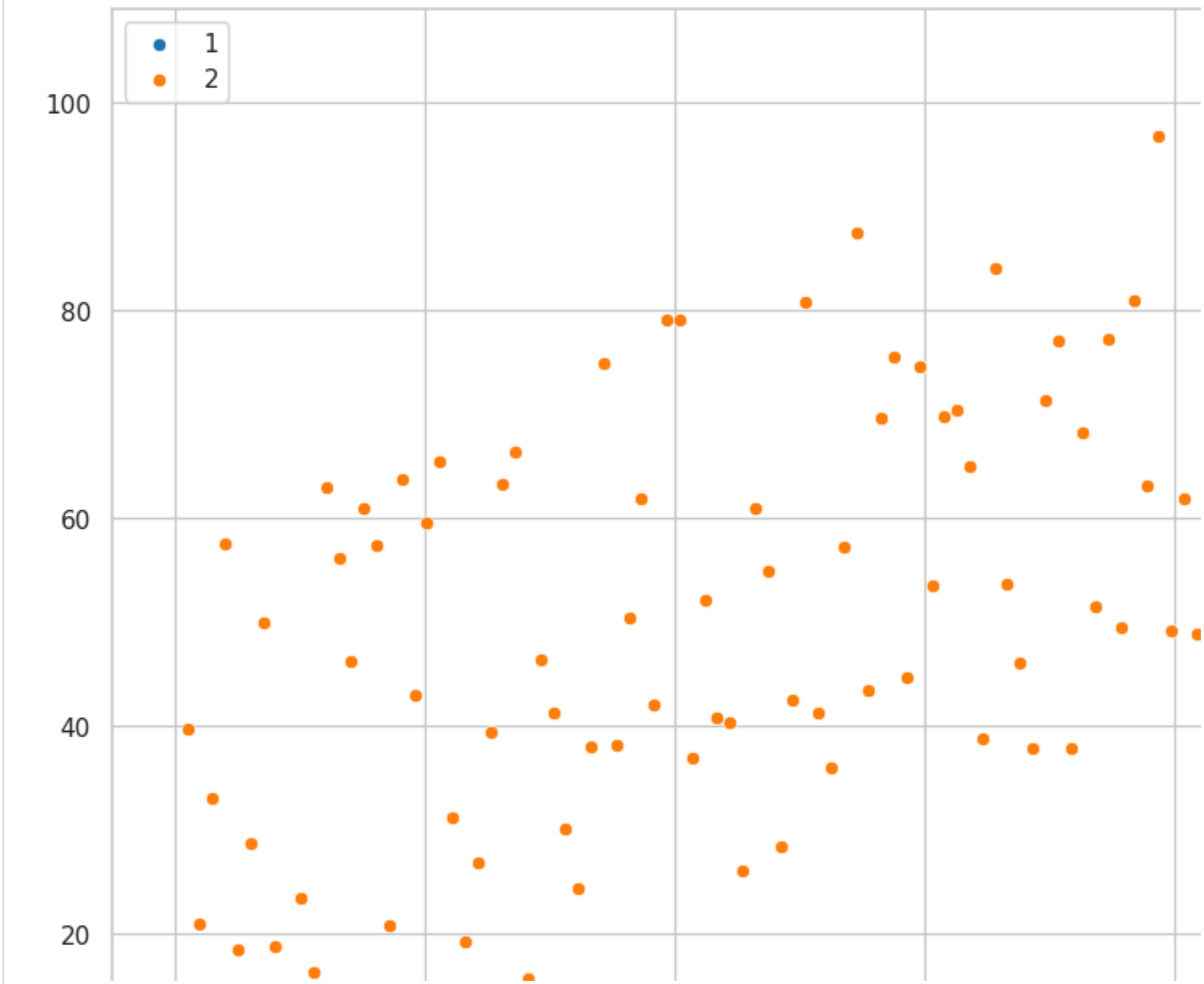
```
sns.scatterplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm'
)
```



El coeficiente de correlación no nos habla del impacto de la relación

```
np.random.seed(42)
x_1 = np.linspace(0, 100, 100)
y_1 = 0.1 * x_1 + 3 + np.random.uniform(-2, 2, size=x_1.size)

sns.scatterplot(
    x=x_1,
    y=y_1
)

x_2 = np.linspace(0, 100, 100)
y_2 = 0.5 * x_2 + 1 + np.random.uniform(0, 60, size=x_2.size)

sns.scatterplot(
    x=x_2,
    y=y_2
)

plt.legend(labels=['1', '2'])

print(np.corrcoef(x_1, y_1))
print(np.corrcoef(x_2, y_2))
```

```
[[1.         0.92761617]
 [0.92761617 1.        ]]
[[1.         0.67476343]
 [0.67476343 1.        ]]
```

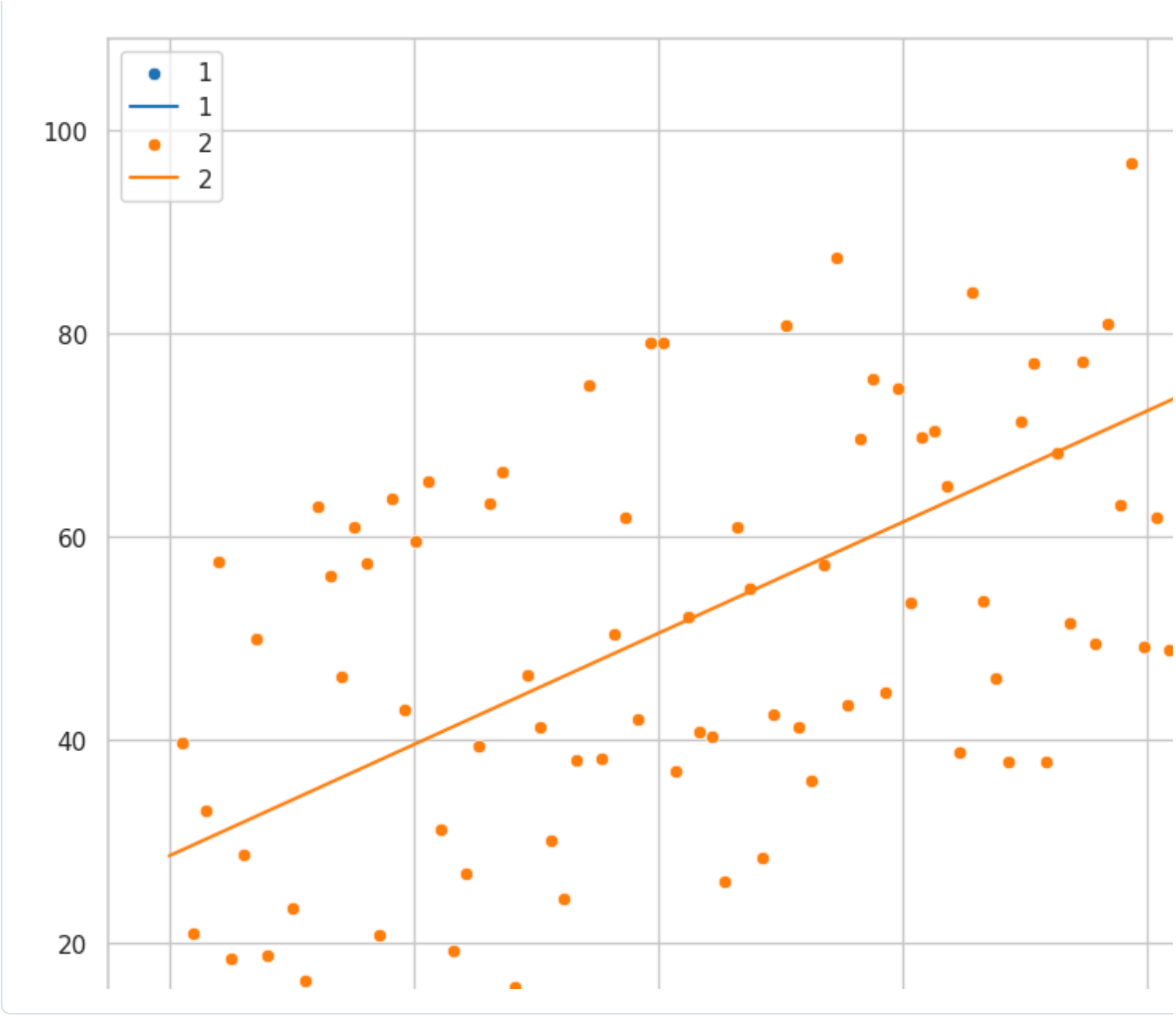


Estableciendo relaciones: Análisis de regresión sim

```
res_1 = scipy.stats.linregress(x=x_1, y=y_1)
res_2 = scipy.stats.linregress(x=x_2, y=y_2)

print(res_1, res_2, sep="\n")
```

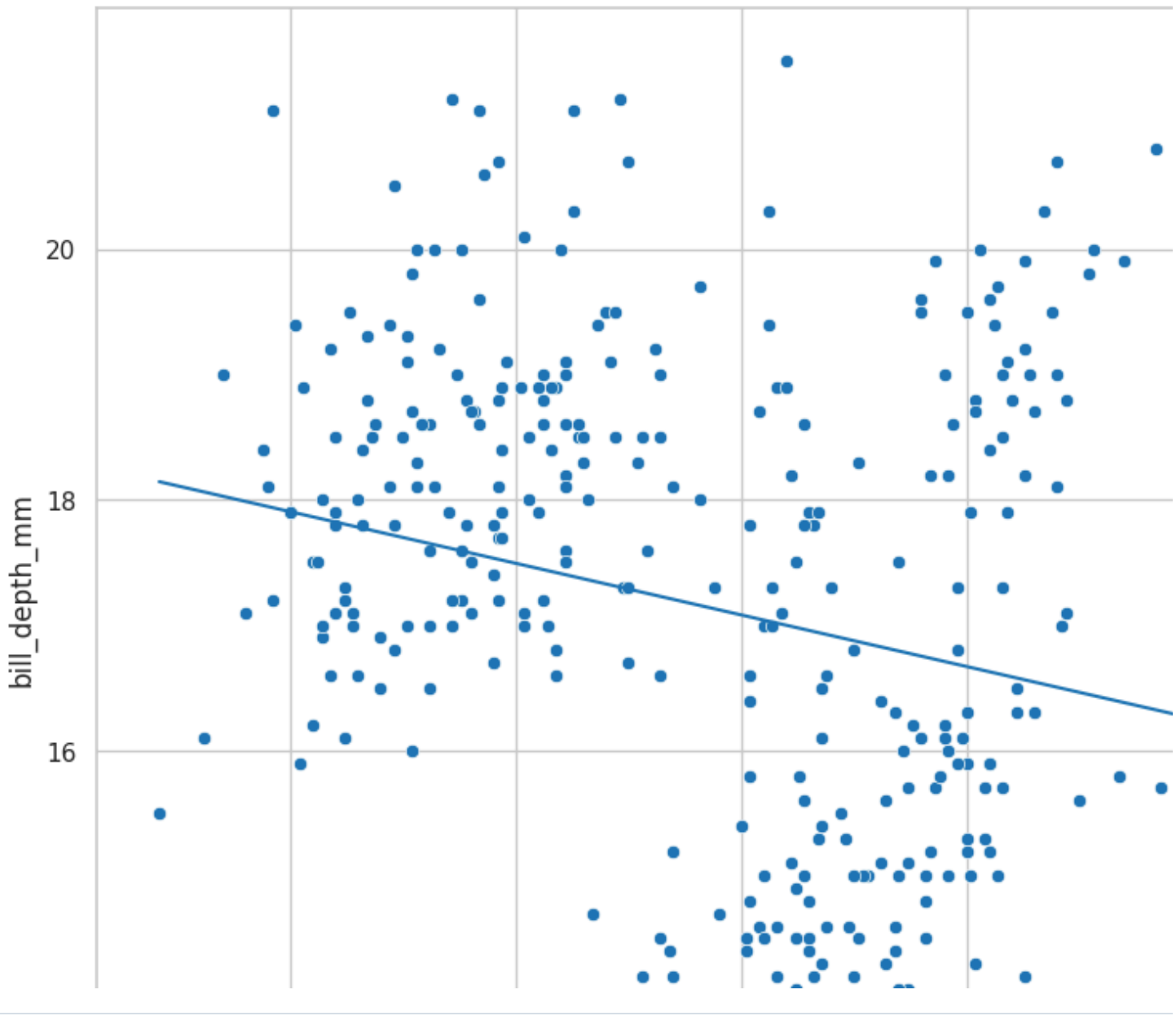
```
LinregressResult(slope=0.10081969280979615, intercept=2.8397383330230292, rvalue=0.9276161661149585, pvalue=1.
LinregressResult(slope=0.5470008424819226, intercept=28.519861265205236, rvalue=0.6747634267657527, pvalue=1.3
```

```
sns.scatterplot(
    x=x_1,
    y=y_1
)

fx_1 = np.array([x_1.min(), x_1.max()])
fy_1 =  res_1.intercept + res_1.slope *  fx_1

plt.plot(fx_1, fy_1)

sns.scatterplot(
    x=x_2,
    y=y_2
)

fx_2 = np.array([x_2.min(), x_2.max()])
fy_2 =  res_2.intercept + res_2.slope *  fx_2

plt.plot(fx_2, fy_2)

plt.legend(labels=['1', '1', '2', '2'])
```

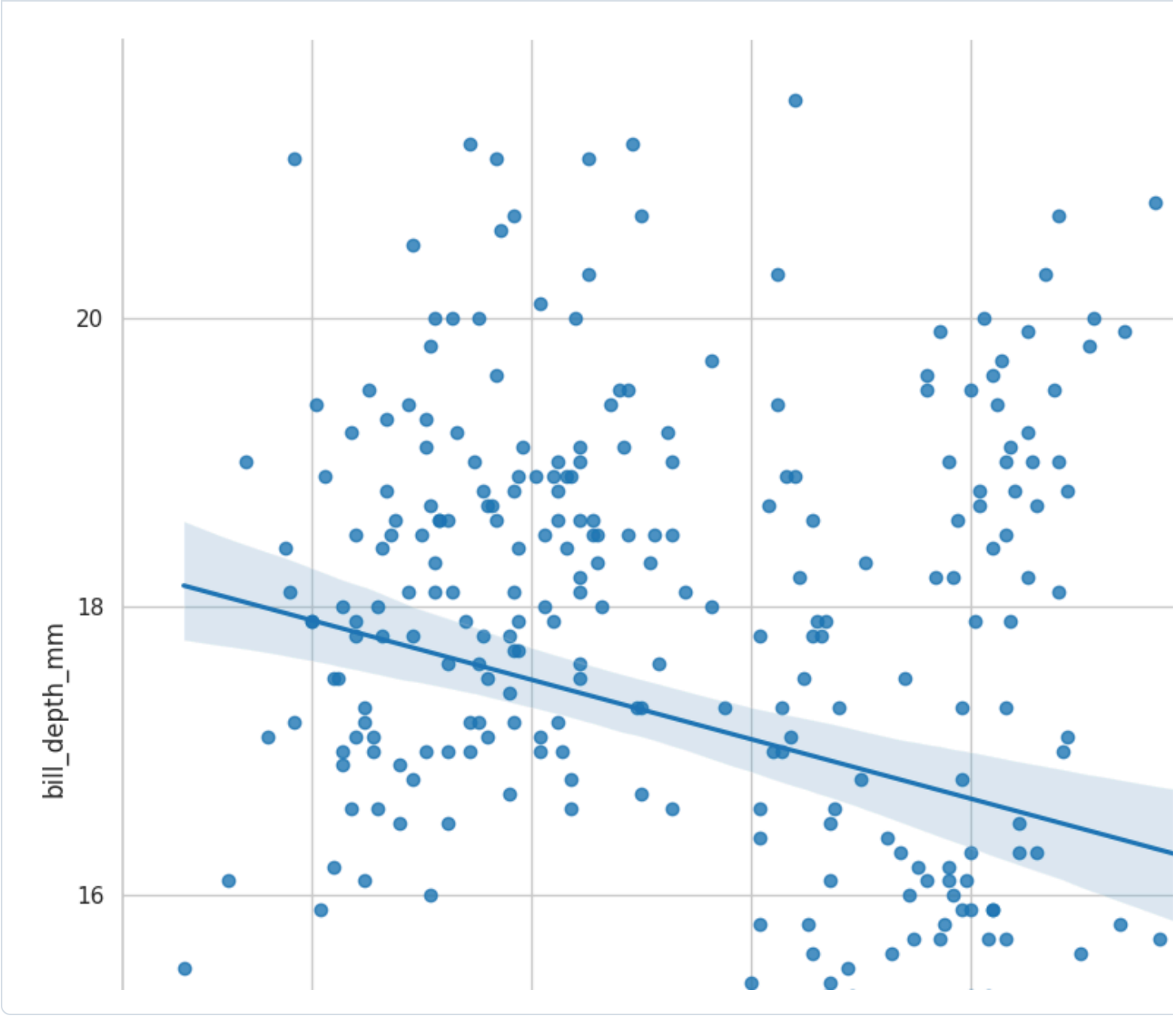

1
1
2
2
100
80
60
40
20

```
sns.scatterplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm'
)

res_penguins = scipy.stats.linregress(x=processed_penguins_df.bill_length_mm, y=processed_pen
print(res_penguins)

fx_1 = np.array([processed_penguins_df.bill_length_mm.min(), processed_penguins_df.bill_lengt
fy_1 =  res_penguins.intercept + res_penguins.slope *  fx_1

plt.plot(fx_1, fy_1)
```

```
LinregressResult(slope=-0.08232675339862285, intercept=20.786648668433827, rvalue=-0.22862563591302928, pvalue
```


20
18
16
bill_depth_mm

```
sns.lmplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    height=10
)
```



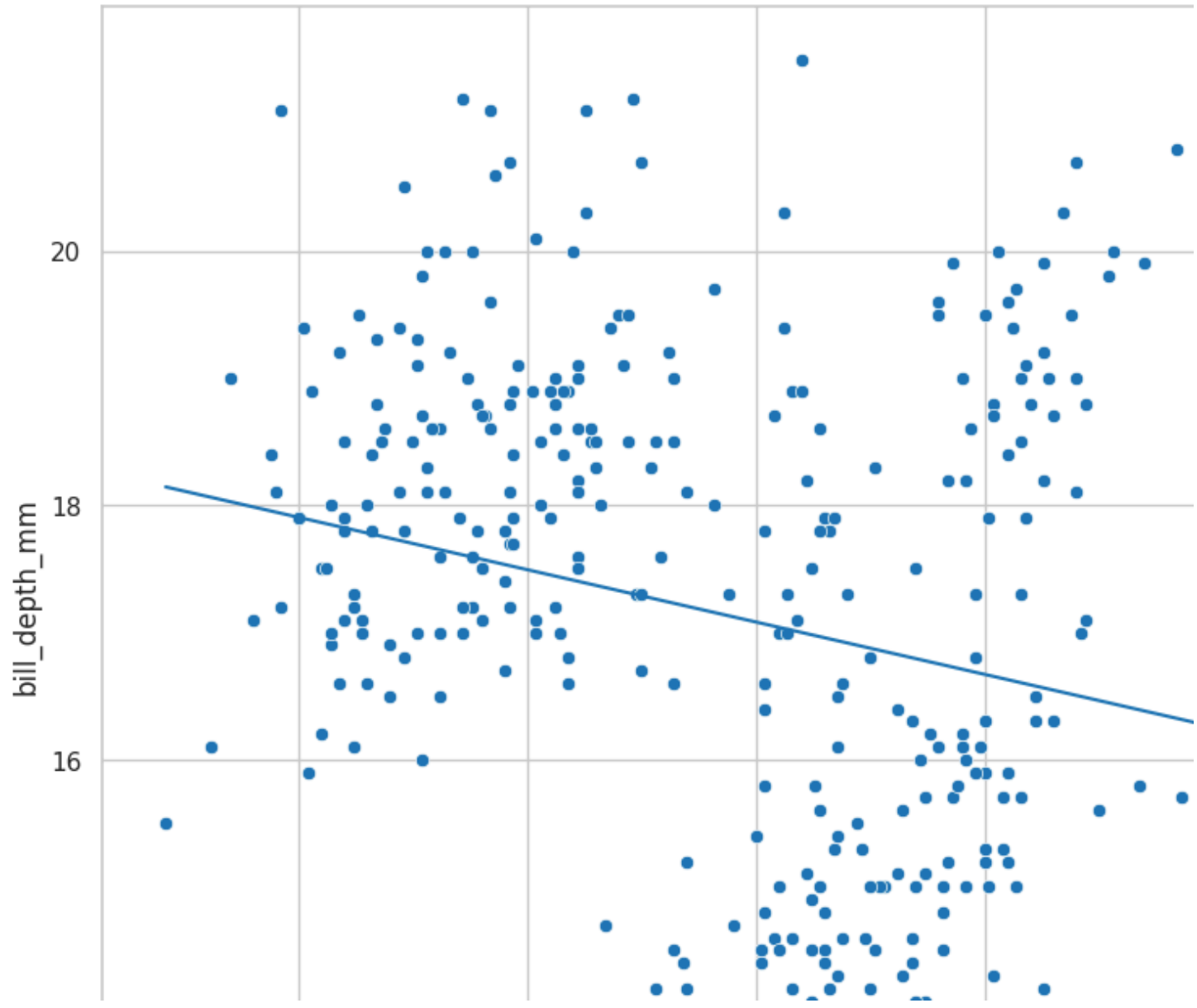
Limitaciones del análisis de regresión simple

La regresión lineal simple no es simétrica

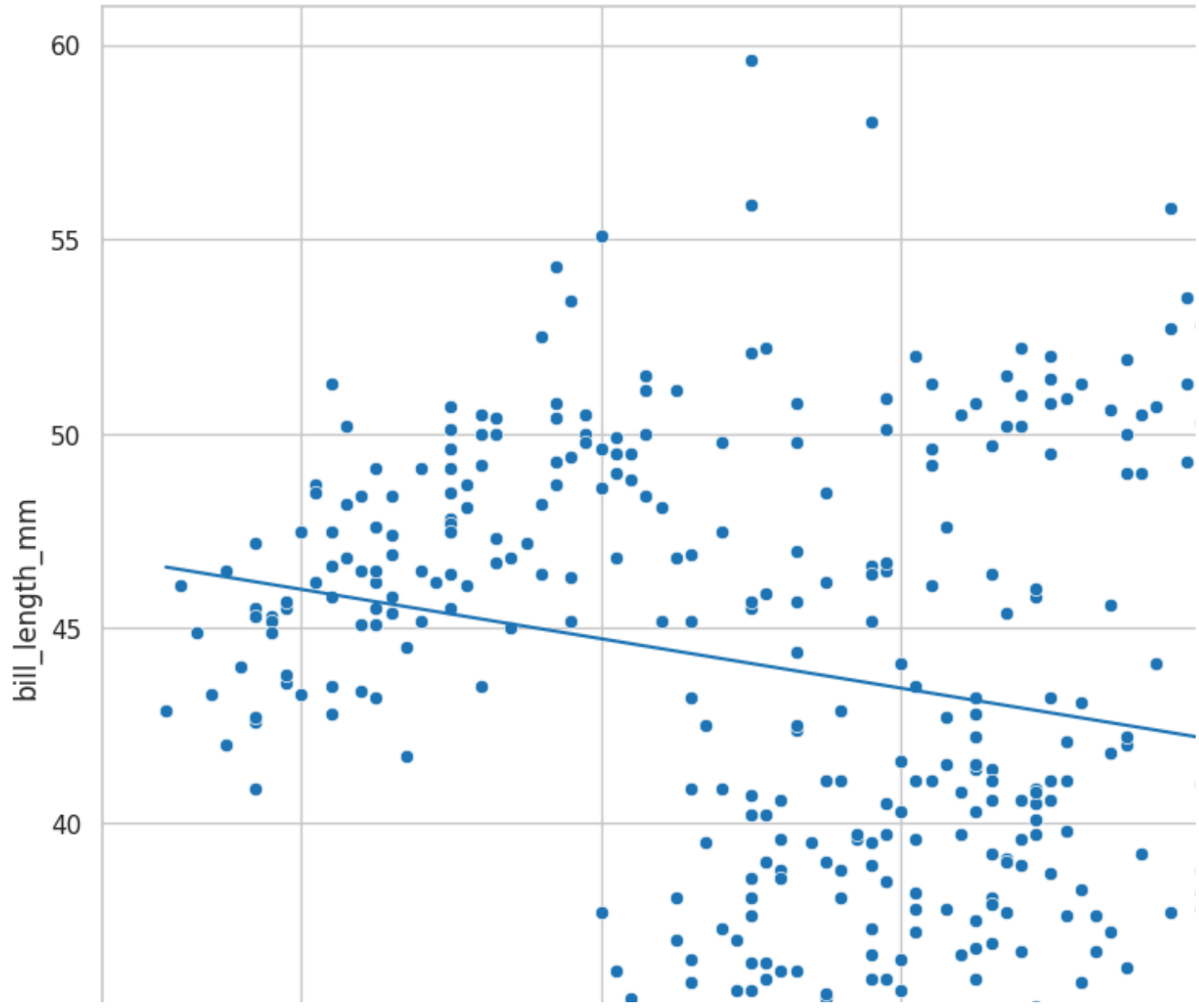
```
x = processed_penguins_df.bill_length_mm
y = processed_penguins_df.bill_depth_mm

res_x_y = scipy.stats.linregress(x=x, y=y)
res_y_x = scipy.stats.linregress(y=x, x=y)

print(res_x_y, res_y_x, sep="\n")
```

```
LinregressResult(slope=-0.08232675339862285, intercept=20.786648668433827, rvalue=-0.22862563591302928, pvalue
LinregressResult(slope=-0.6349051704195029, intercept=54.89085424504756, rvalue=-0.22862563591302928, pvalue=2
```

```
sns.scatterplot(
    x=x,
    y=y
)

fx_1 = np.array([x.min(), x.max()])
fy_1 =  res_x_y.intercept + res_x_y.slope *  fx_1

plt.plot(fx_1, fy_1)
```



```
sns.scatterplot(
    x=y,
    y=x
)

fx_1 = np.array([y.min(), y.max()])
fy_1 =  res_y_x.intercept + res_y_x.slope *  fx_1

plt.plot(fx_1, fy_1)
```



La regresión no nos dice nada sobre la causalidad, pero existen h separar las relaciones entre varias variables

La pendiente es **-0.634905**, lo que significa que cada milímetro adicional de profundidad del pico es asociado a u milímetros de la longitud del pico de un pingüino.

```
(
    smf.ols(
        formula="bill_length_mm ~ bill_depth_mm",
         data=processed_penguins_df
    )
    .fit()
    .params
)
```

```
(
    smf.ols(
        formula="bill_depth_mm ~ bill_length_mm",
         data=processed_penguins_df
    )
    .fit()
    .summary()
)
```

Análisis de regresión múltiple

Olvidé mi báscula para pesar a los pingüinos, ¿Cuál sería la mejo ese dato?

Creando modelos

```
model_1 = (
    smf.ols(
        formula="body_mass_g ~ bill_length_mm",
         data=processed_penguins_df
    )
    .fit()
)

model_1.summary()
```

```
model_2 = (
    smf.ols(
        formula="body_mass_g ~ bill_length_mm + bill_depth_mm ",
         data=processed_penguins_df
    )
    .fit()
)

model_2.summary()
```

```
model_3 = (
    smf.ols(
        formula="body_mass_g ~ bill_length_mm + bill_depth_mm + flipper_length_mm",
         data=processed_penguins_df
    )
    .fit()
)

model_3.summary()
```

```
model_4 = (
    smf.ols(
        formula="body_mass_g ~ bill_length_mm + bill_depth_mm + flipper_length_mm + C(sex)",
         data=processed_penguins_df
    )
    .fit()
)

model_4.summary()
```

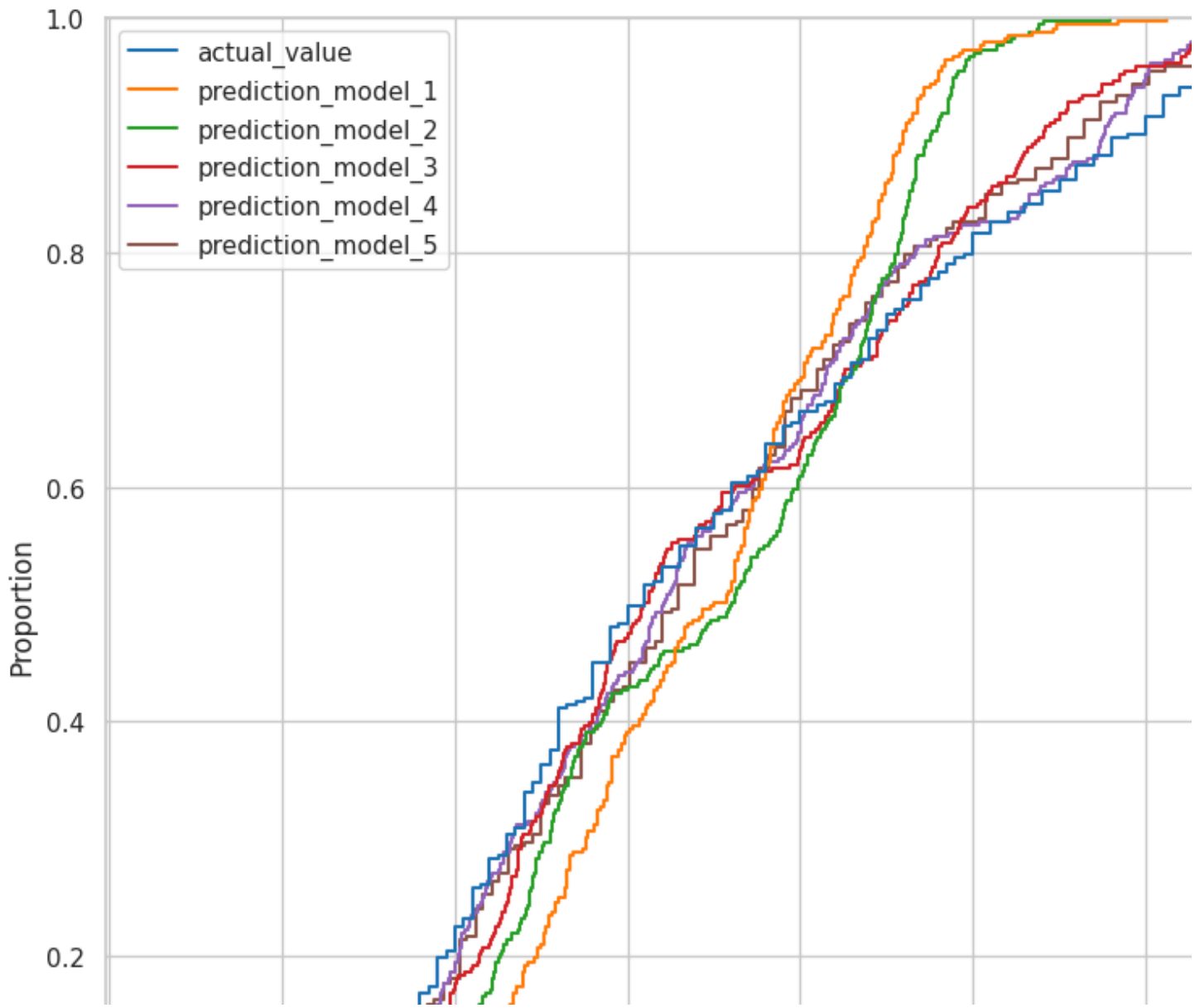
```
model_5 = (
    smf.ols(
        formula="body_mass_g ~ flipper_length_mm + C(sex)",
         data=processed_penguins_df
    )
    .fit()
)

model_5.summary()
```

Visualizando resultados

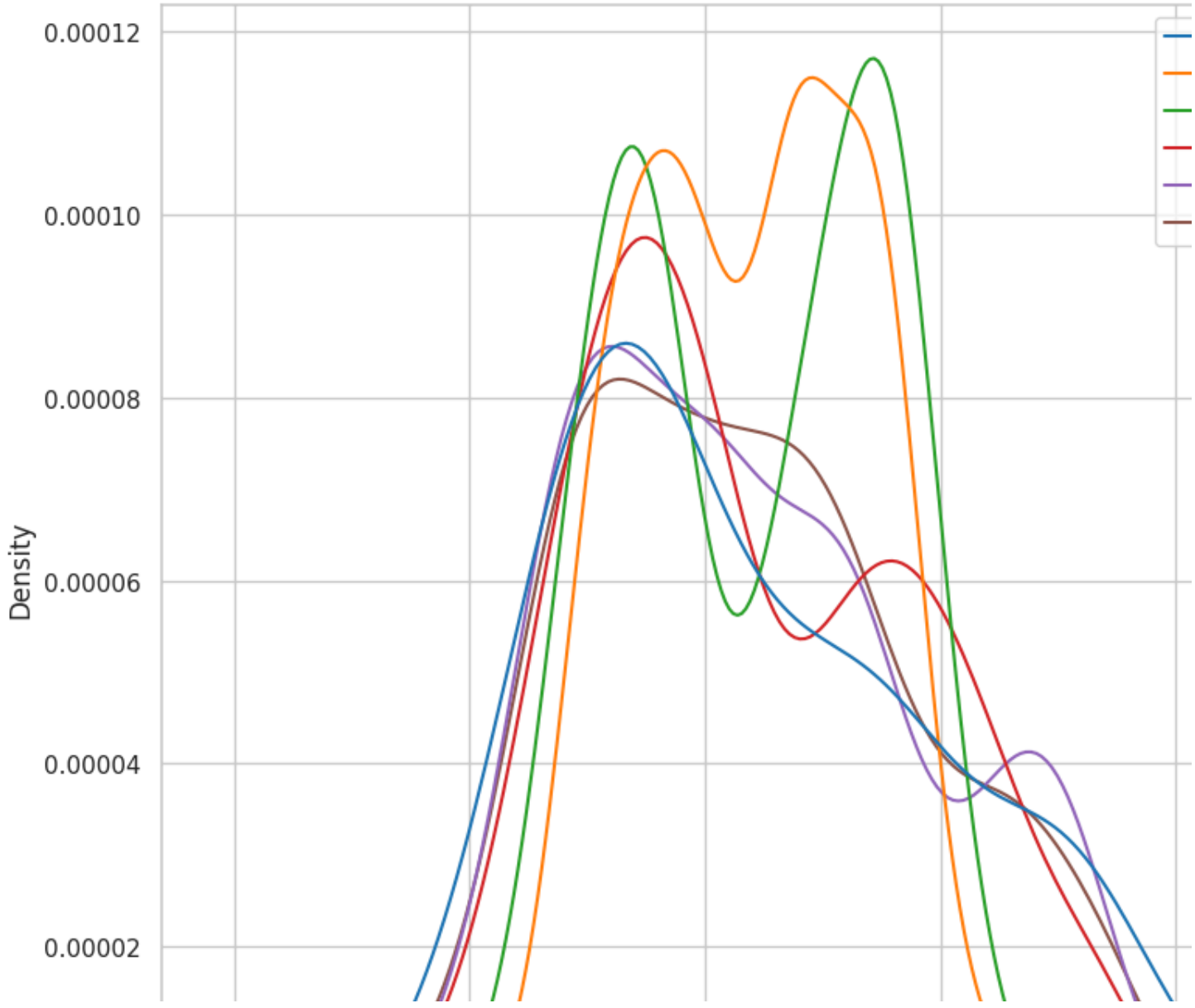
```
models_results = pd.DataFrame(
    dict(
        actual_value=processed_penguins_df.body_mass_g,
        prediction_model_1 = model_1.predict(),
        prediction_model_2 = model_2.predict(),
        prediction_model_3 = model_3.predict(),
        prediction_model_4 = model_4.predict(),
        prediction_model_5 = model_5.predict(),
        species=processed_penguins_df.species,
        sex=processed_penguins_df.sex
    )
)

models_results
```

	actual_value float... 2700.0 - 6300.0	prediction_mod... 3174.8606434468...	prediction_mod... 2931.76713745199...	prediction_mod... 2742.16696809388...	prediction_mod... 2700.0792083920...	predicti 2670.63
0	3750.0	3782.402960979..	3617.6411922647..	3204.761227286...	3579.136945849...	3441.32
1	3800.0	3817.1196648387..	3836.725580481..	3436.701722298..	3343.220771658...	3328.3
2	3250.0	3886.553072556..	3809.2713711970..	3906.897031997...	3639.137335064..	3751.22
4	3450.0	3574.102737825...	3350.786580829..	3816.7057718755..	3457.954242592..	3657.25
5	3650.0	3799.761312909...	3356.140069628..	3696.1681278221..	3764.536023252..	3864.16
6	3625.0	3765.044609050..	3733.635131863...	3188.049902837..	3116.053553638...	3093.4
7	4675.0	3791.08213694445	3494.165990098..	3931.8131088785..	4044.986255638..	4099.07
12	3200.0	3955.986480274..	3927.324344949..	3242.489053884..	3166.973924441...	3140.4
13	3800.0	3739.007081155...	3216.466921836...	3755.327090343..	3753.338992054..	3911.14
14	4400.0	3391.840042565..	2931.7671374519..	4095.706922783..	4043.043337112..	4240.0

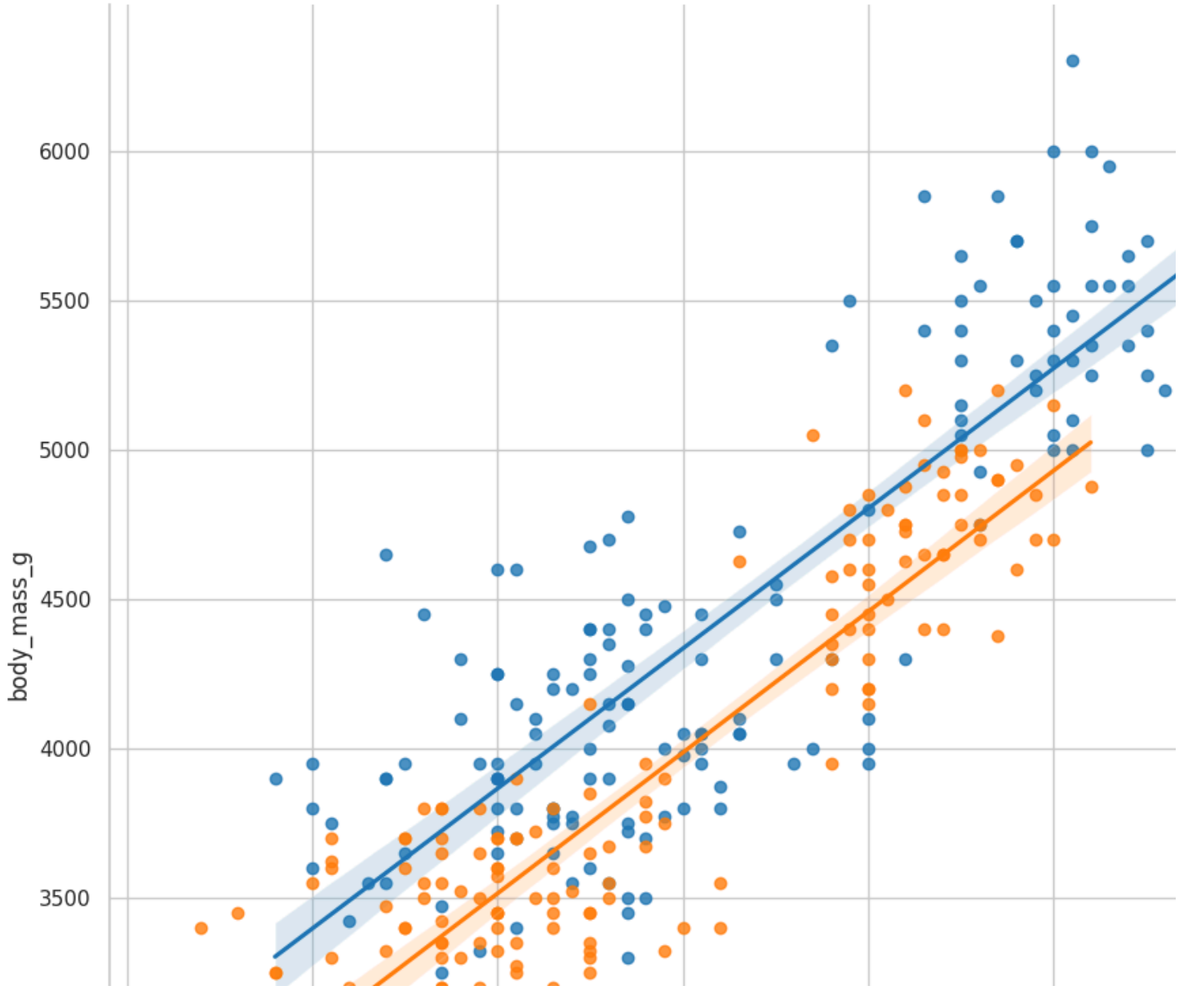
```
sns.ecdfplot(
    data=models_results#.select_columns(['actual_value', 'prediction_model_5'])
)
```



```
sns.kdeplot(
    data=models_results,
    cumulative=False
)
```



```
sns.lmplot(
    data=processed_penguins_df,
    x='flipper_length_mm',
    y='body_mass_g',
    height=10,
    hue='sex'
)
```



Análisis de regresión logística

```
smf.logit(
    formula='sex_numeric ~ flipper_length_mm + bill_length_mm + bill_depth_mm + C(island)',
    data=processed_penguins_df
).fit().summary()
```

```
Optimization terminated successfully.
         Current function value: 0.360900
         Iterations 7
```

```
(
    processed_penguins_df
    .value_counts(['island', 'sex', 'species'])
    .reset_index(name='count')
)
```

	island object	sex object	species object	count int64
	Biscoe 40% Dream 40% Torgersen 20%	Male 50% Female 50%	Adelie 60% Gentoo 20% Chinstrap 20%	22 - 61
0	Biscoe	Male	Gentoo	61
1	Biscoe	Female	Gentoo	58
2	Dream	Female	Chinstrap	34
3	Dream	Male	Chinstrap	34
4	Dream	Male	Adelie	28
5	Dream	Female	Adelie	27
6	Torgersen	Female	Adelie	24
7	Torgersen	Male	Adelie	23
8	Biscoe	Female	Adelie	22
9	Biscoe	Male	Adelie	22

```
processed_penguins_df.species.unique()
```

```
processed_penguins_df = (
    processed_penguins_df
    .assign(is_adelie=lambda df: df.species.replace(['Adelie', 'Chinstrap', 'Gentoo'], [1, 0,
)
```

```
model_is_adelie = smf.logit(
    formula='is_adelie ~ flipper_length_mm + C(sex)',
    data=processed_penguins_df
).fit(maxiter=100)

model_is_adelie.params
```

```
Optimization terminated successfully.
         Current function value: 0.355225
         Iterations 8
```

```
is_adelie_df_predictions = pd.DataFrame(
    dict(
        actual_adelie = processed_penguins_df.species.replace(['Adelie', 'Chinstrap', 'Gentoo
        predicted_values = model_is_adelie.predict().round()
    )
)

is_adelie_df_predictions
```

	actual_adelie int64 0 - 1	predicted_values f.. 0.0 - 1.0
0	1	1.0
1	1	1.0
2	1	0.0
4	1	1.0
5	1	1.0
6	1	1.0
7	1	1.0
12	1	1.0
13	1	1.0
14	1	1.0

```
(
    is_adelie_df_predictions
    .value_counts(['actual_adelie', 'predicted_values'])
    .reset_index(name='count')
    .pivot_wider(
        index='actual_adelie',
        names_from='predicted_values',
        values_from='count'
    )
    .rename_column('actual_adelie', 'actual / predicted')
)
```

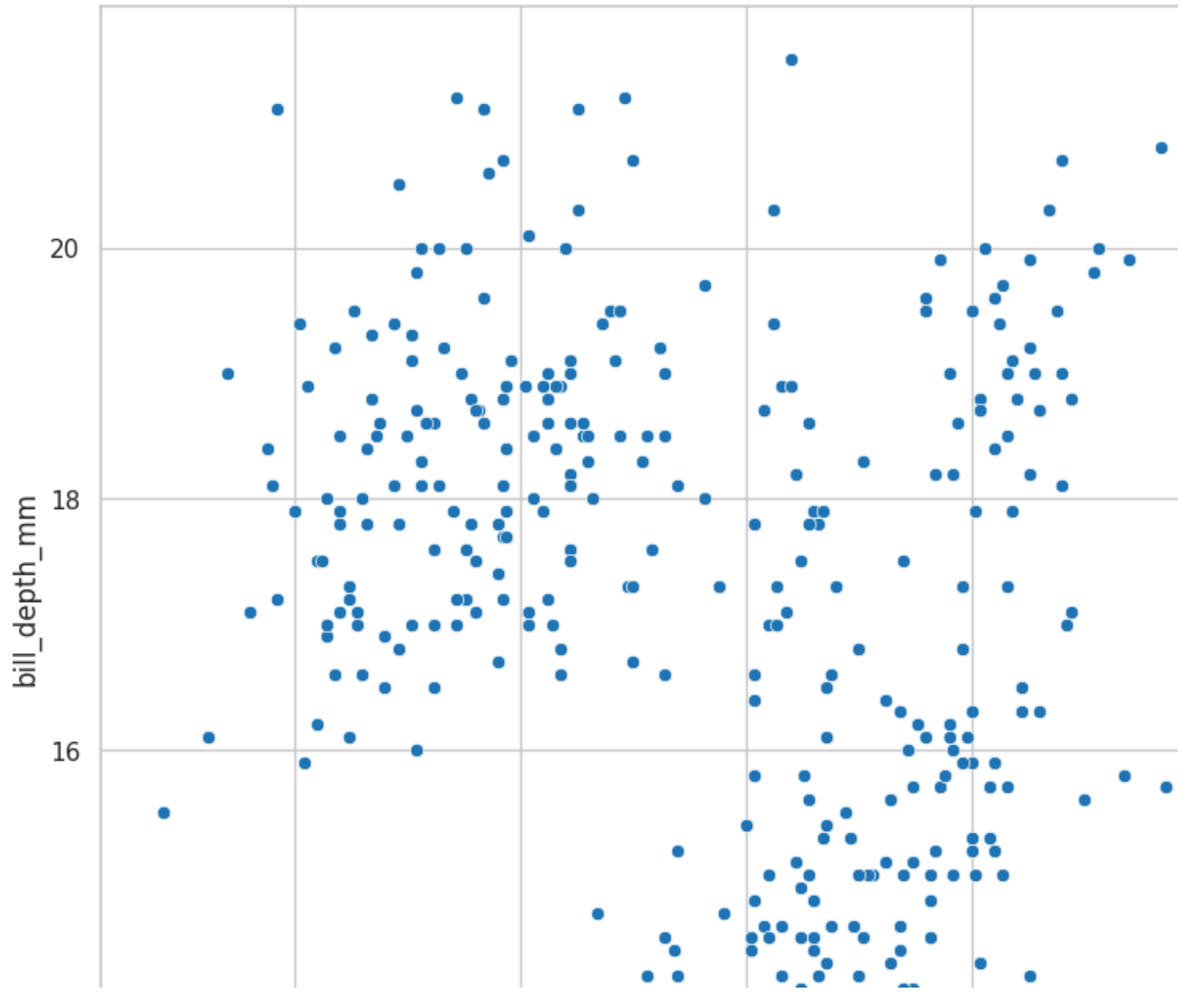
	actual / predicted i.	0.0 int64	1.0 int64	
0	0	151	36	
1	1	17	129	

```
print(
    sklearn.metrics.confusion_matrix(
        is_adelie_df_predictions.actual_adelie,
        is_adelie_df_predictions.predicted_values
    )
)

sklearn.metrics.accuracy_score(
    is_adelie_df_predictions.actual_adelie,
    is_adelie_df_predictions.predicted_values
)
```

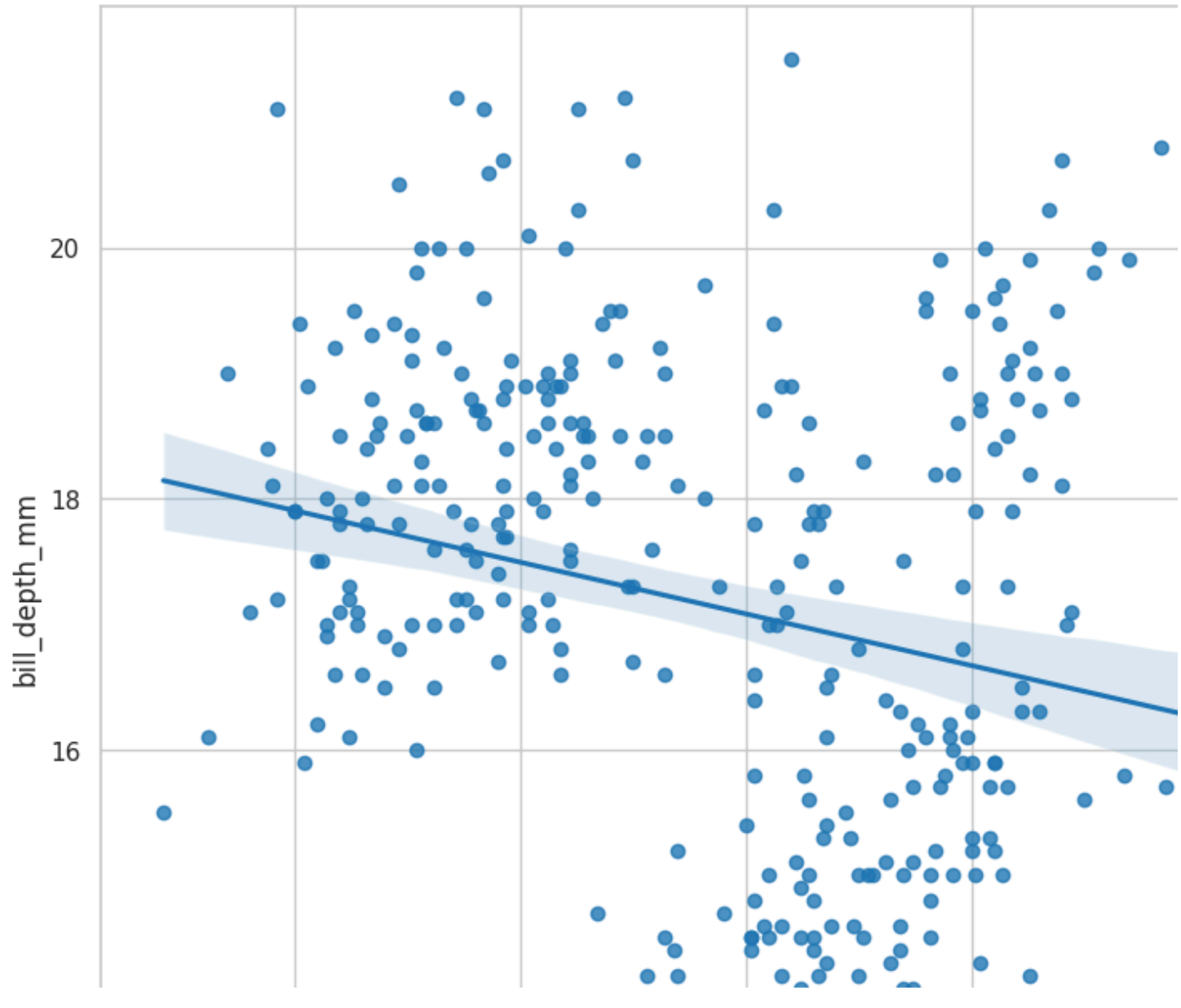
```
[[151  36]
 [ 17 129]]
```

Paradoja de Simpson

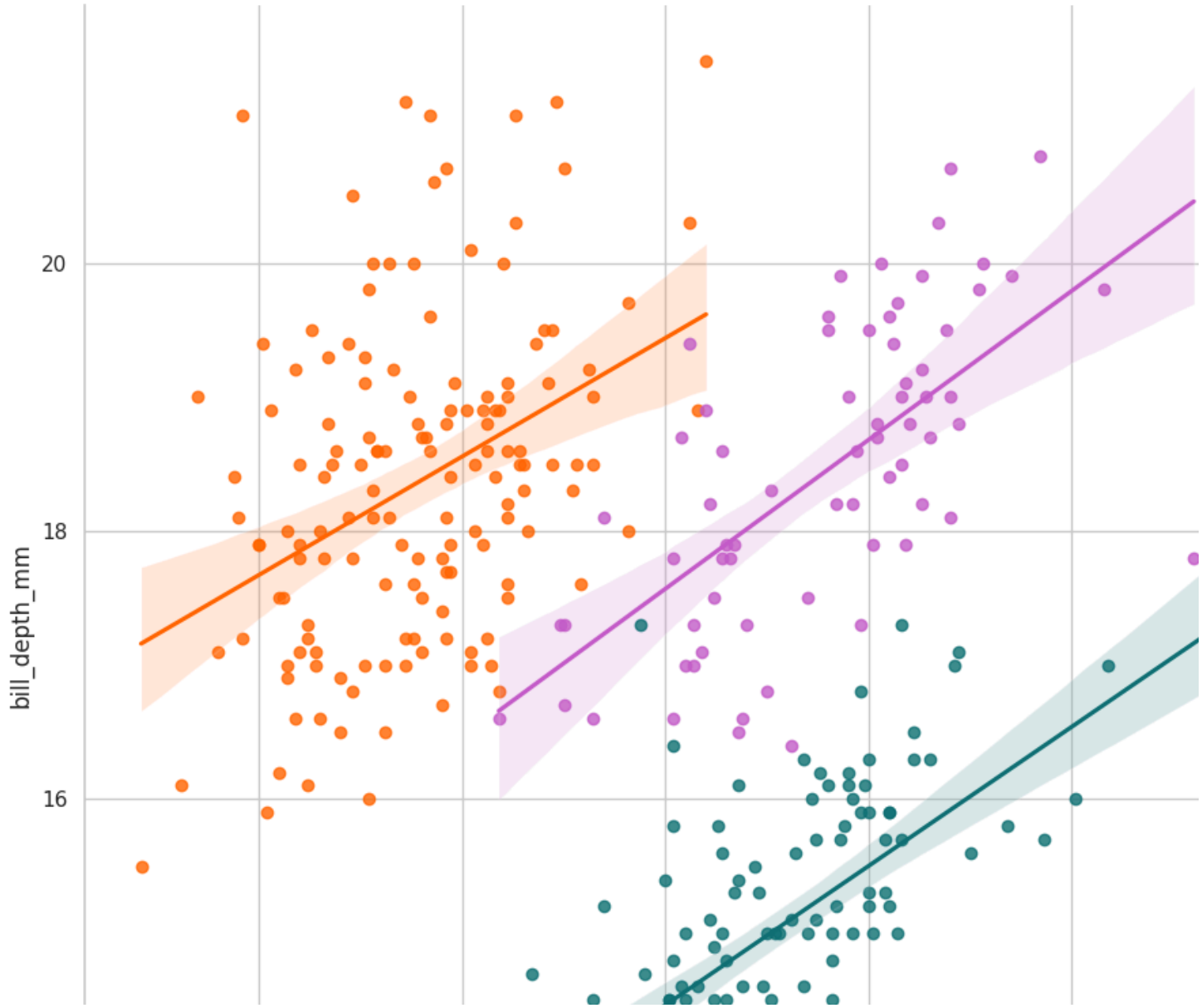
```
sns.scatterplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm'
)
```



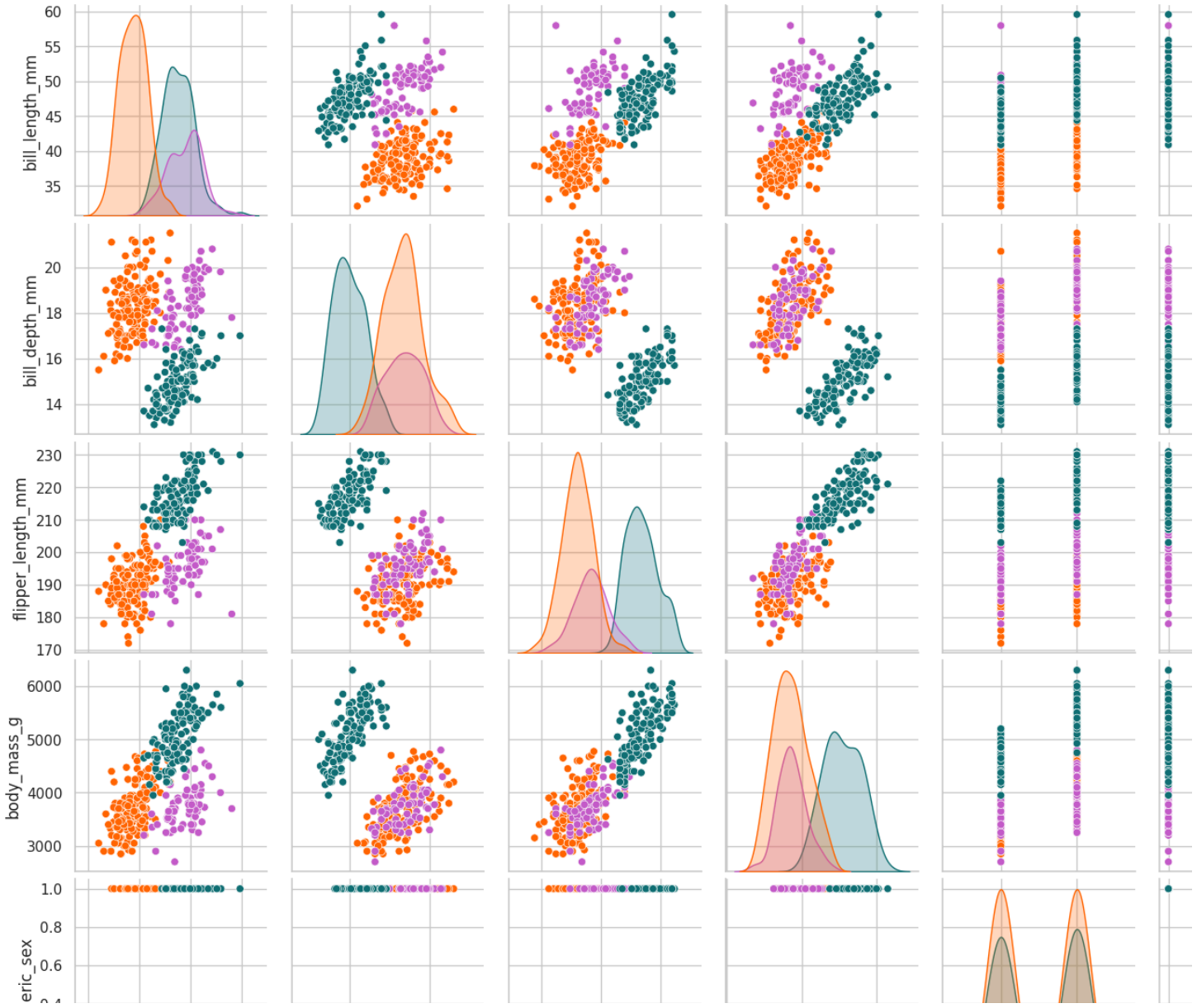
```
sns.regplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm'
)
```



```
sns.lmplot(
    data=processed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    hue='species',
    height=10,
    palette=penguin_color
)
```



```
sns.pairplot(data=processed_penguins_df, hue='species', palette=penguin_color)
```



Información de sesión

```
session_info.show()
```