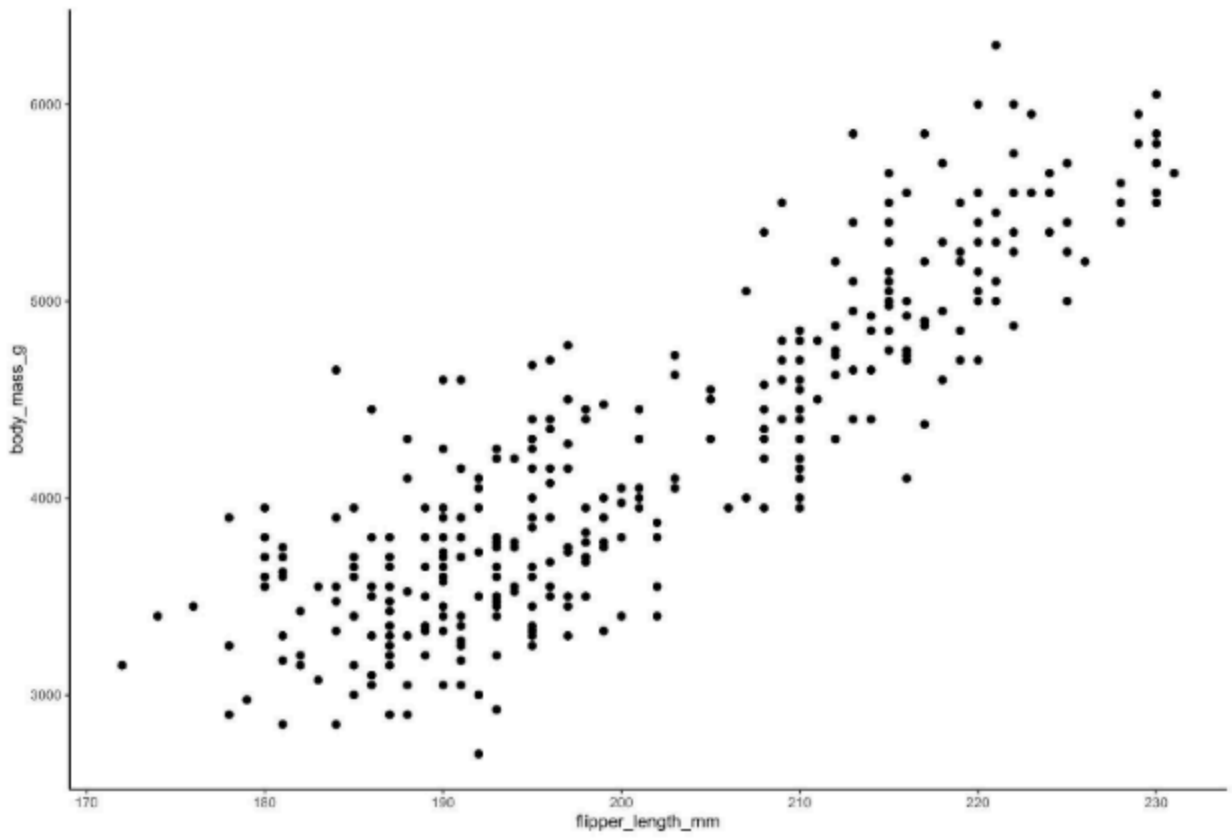
Estableciendo relaciones: Gráficos de puntos

Aprenderemos a establecer relaciones de 2 variables.

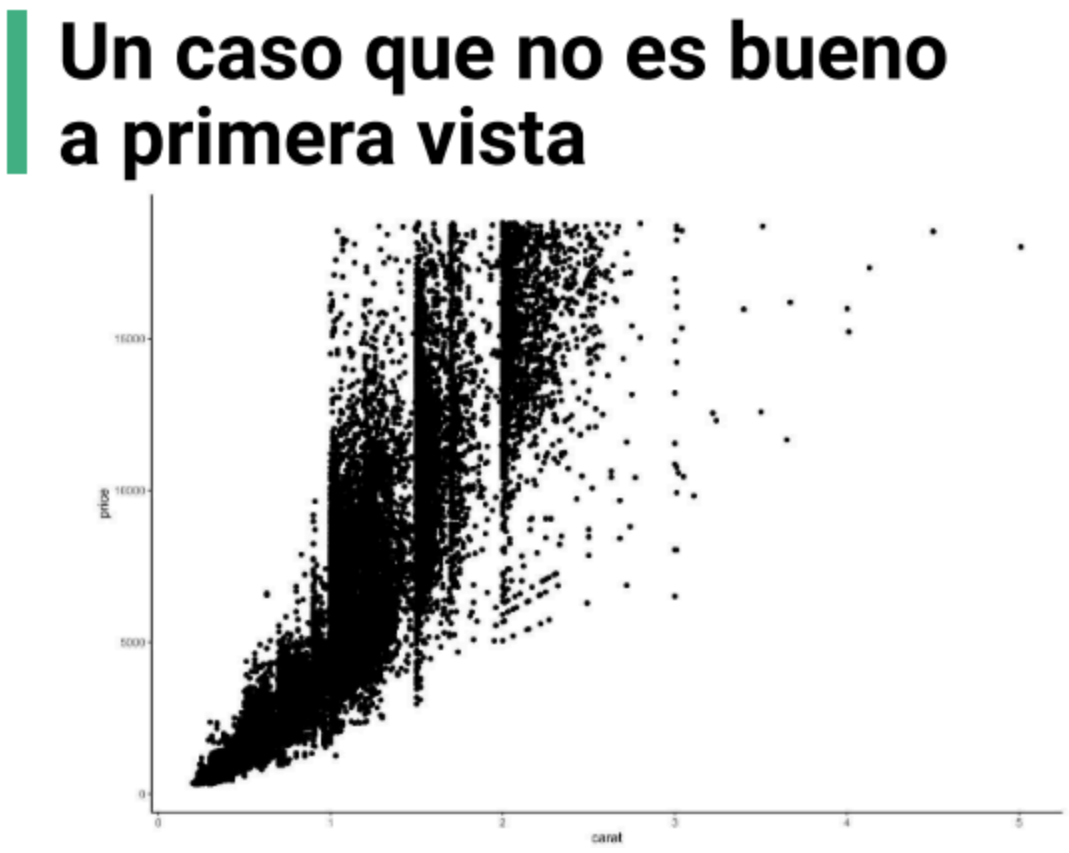
Vamos a graficar en un plano y observar donde intersectan entre ellas.

Por ejemplo tenemos la longitud de las alas contra el peso. Vemos una tendencia creciente. ¿tiene un significado?



Esta gráfica nos va ayudar a entender como se distribuyen los puntos.

Pero existen casos en donde los datos son demasiados y no es verá bien.

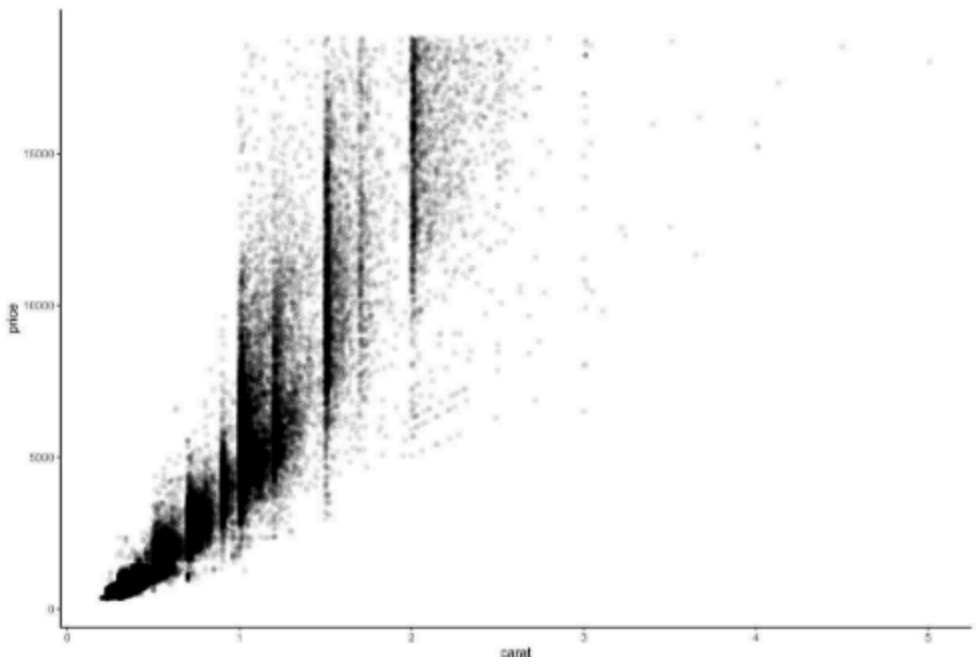


Esto pasa cuando tenemos demasiados datos, y hay zonas donde hay muchos datos, o donde no los hay. Estas situaciones pasan y nos encontraremos con ellas.

Aquí va una lista de recomendaciones para mejorar la gravitación de `scatter`.

Recomendaciones

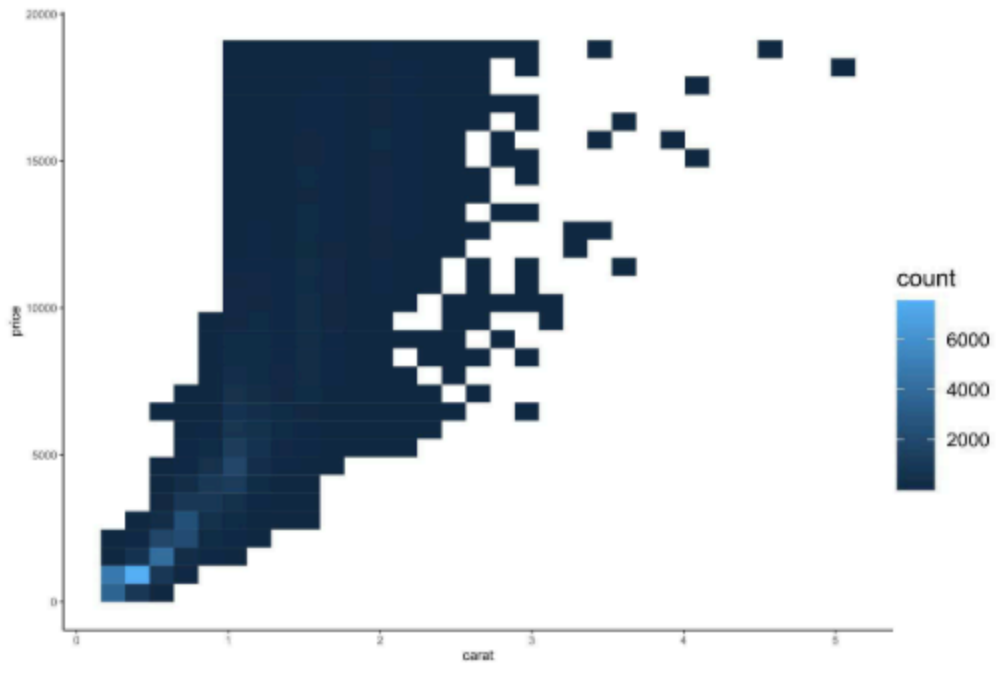
Modificar la transparencia



Es importante ajustarlo, se conoce como **alpha**. Podemos decir que la transparencia puede ir en función de la cantidad de puntos. Por ejemplo, para que el color sea completamente solido, tendremos que tener 30 puntos, entonces el alpha lo podemos definir como $\text{alpha} = \frac{1}{30}$.

Entonces la transparencia nos va a ayudar a observar en que zonas se acumulan demasiado los datos.

Histograma de dos dimensiones

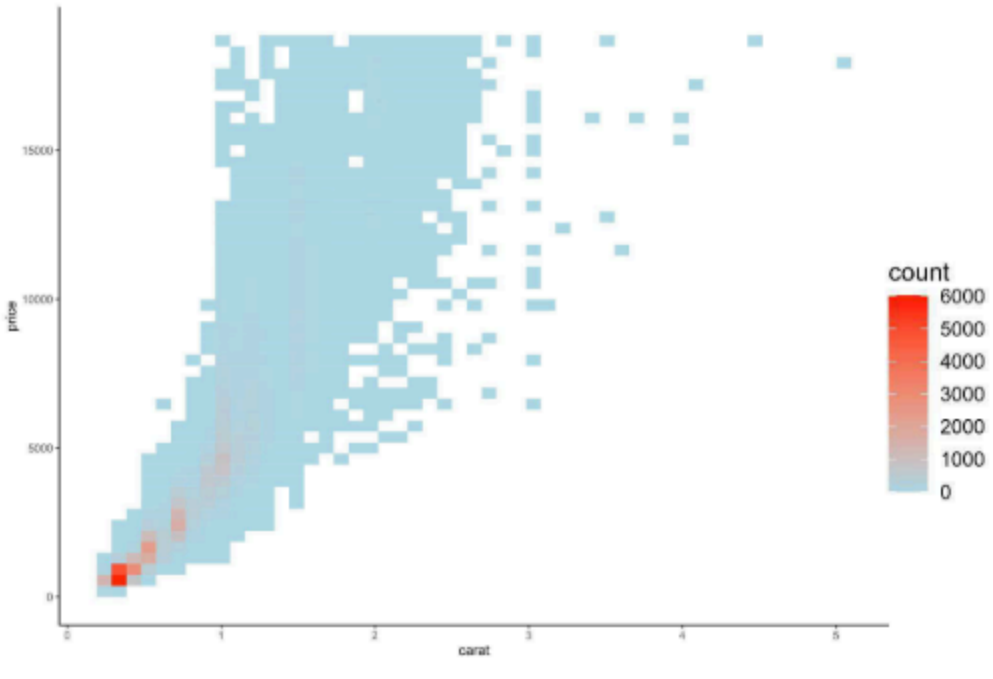


Otra forma de visualizarlo es a traves de **Histogramas**, ya vimos en la sección anterior que el **histograma** lo realizamos con una variable, pero podemos extender su uso a 2 dimensiones y calcular los valores que están entre mi variable X e Y y finalmente contar la frecuencia.

Por ejemplo en este caso es más util porque nos permite apreciar por el color, la concentración de los puntos, en el caso anterior no era tan claro.

También podemos cambiar el color.

Cambio de color



Mediante el cambio de color podemos tener una mejor percepción de lo que está sucediendo, en este caso; entre más rojo significa que la frecuencia es mayor, es decir se repite más ese número.

Entonces es más fácil ver que se acumulan los datos.

In [ ]:
```
# Importando 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 las gráficas

In [ ]:
```
%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

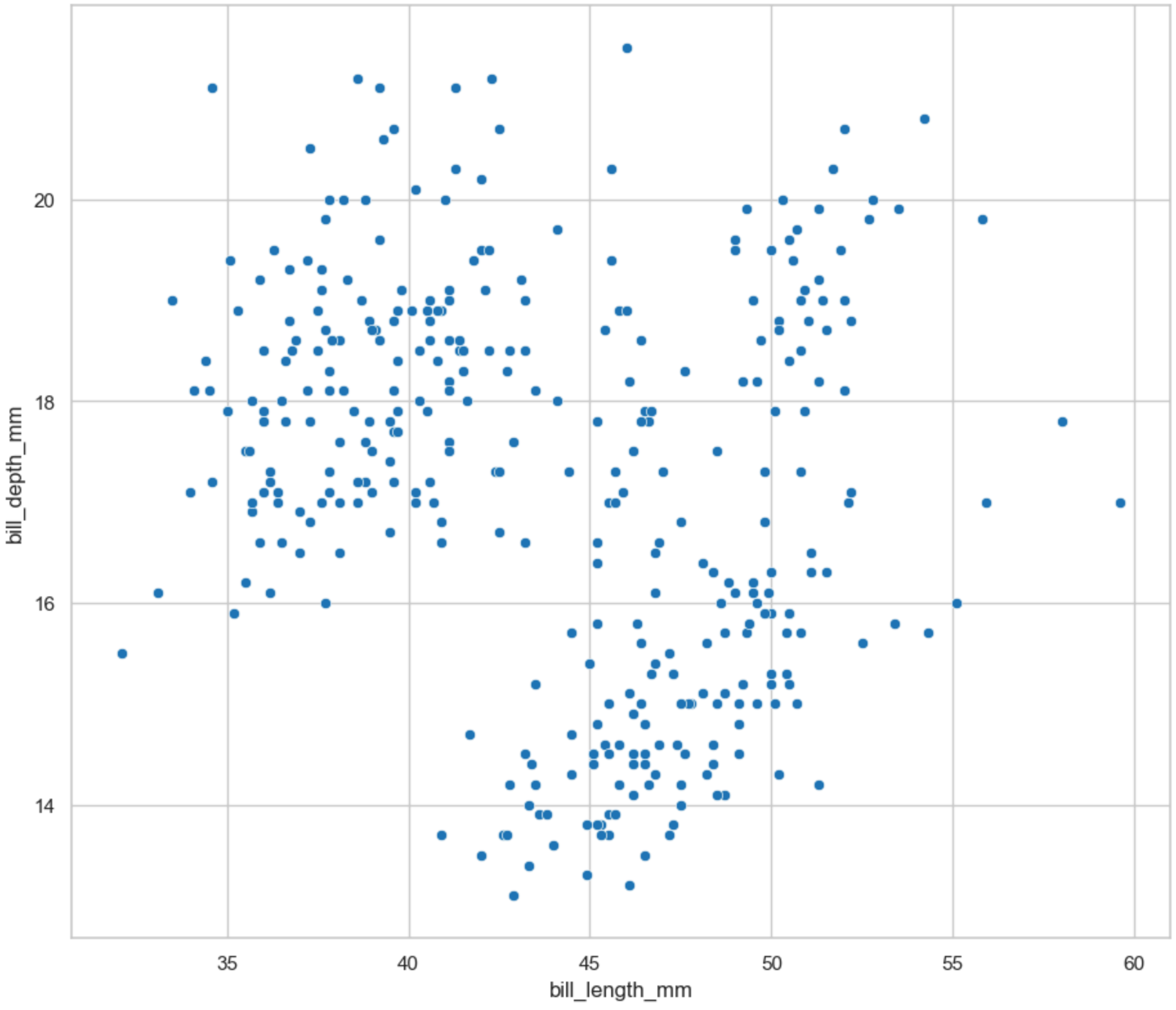
Datos Preprocesados

In [ ]:
```
preprocessed_penguins_df = pd.read_csv('dataset/penguins.csv')
```

Scatter plot

In [ ]:
```
sns.scatterplot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm"
)
```

Out[ ]:
```
<AxesSubplot: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```

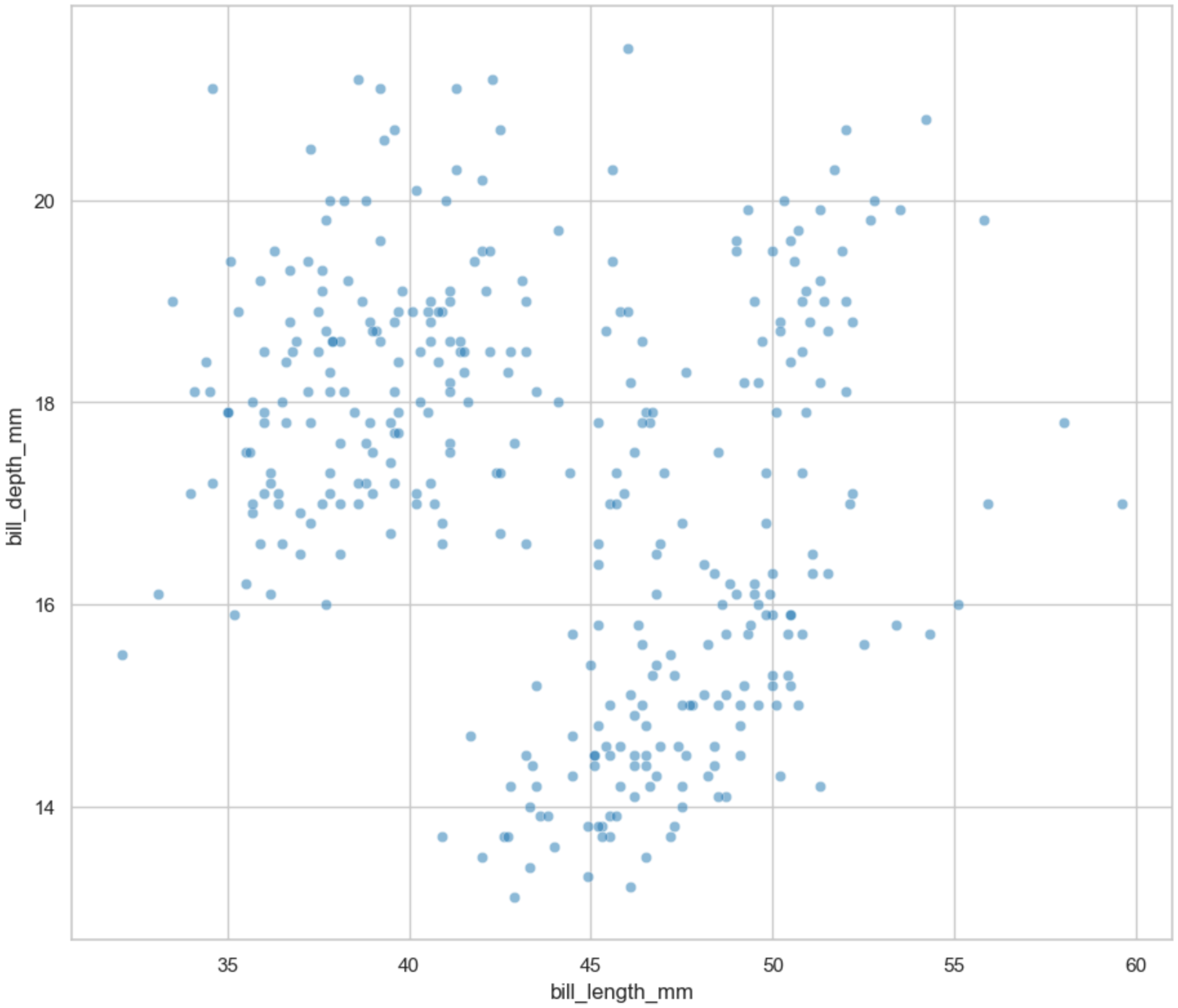


Hay que fijarnos que tenemos unos puntos que están dispersos, y tenemos un pequeño hueco.

Visualizar los datos nos da mayor información.

In [ ]:
```
sns.scatterplot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm",
    alpha=0.5
)
```

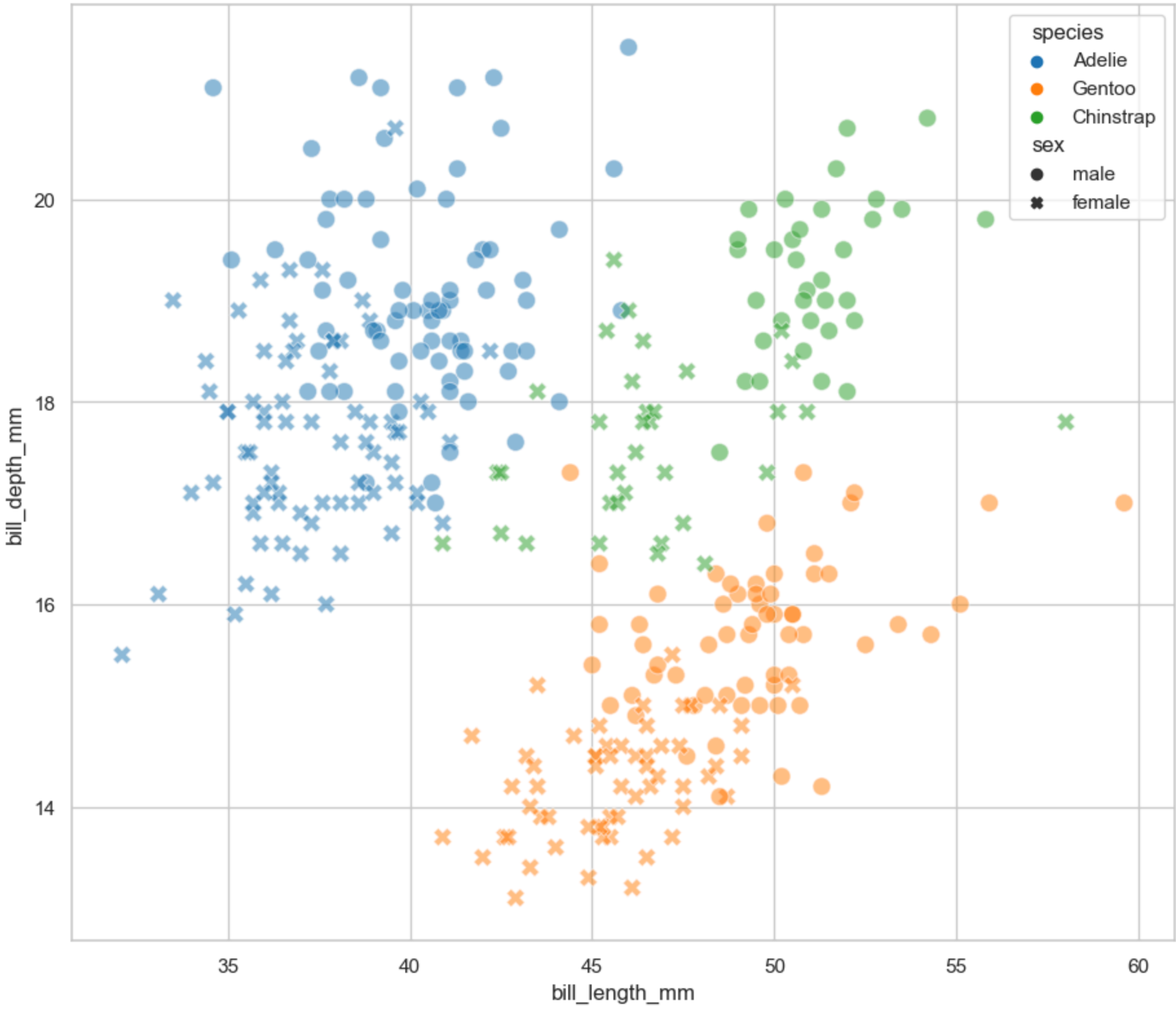
Out[ ]:
```
<AxesSubplot: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



Cambiando alpha y tamaño de los puntos. En algunos casos es conveniente disminuir el tamaño de los puntos para poder ver mejor el patrón, para este caso aumentaremos el tamaño.

In [ ]:
```
sns.scatterplot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm",
    alpha=0.5,
    hue='species',
    style='sex',
    s=100
)
```

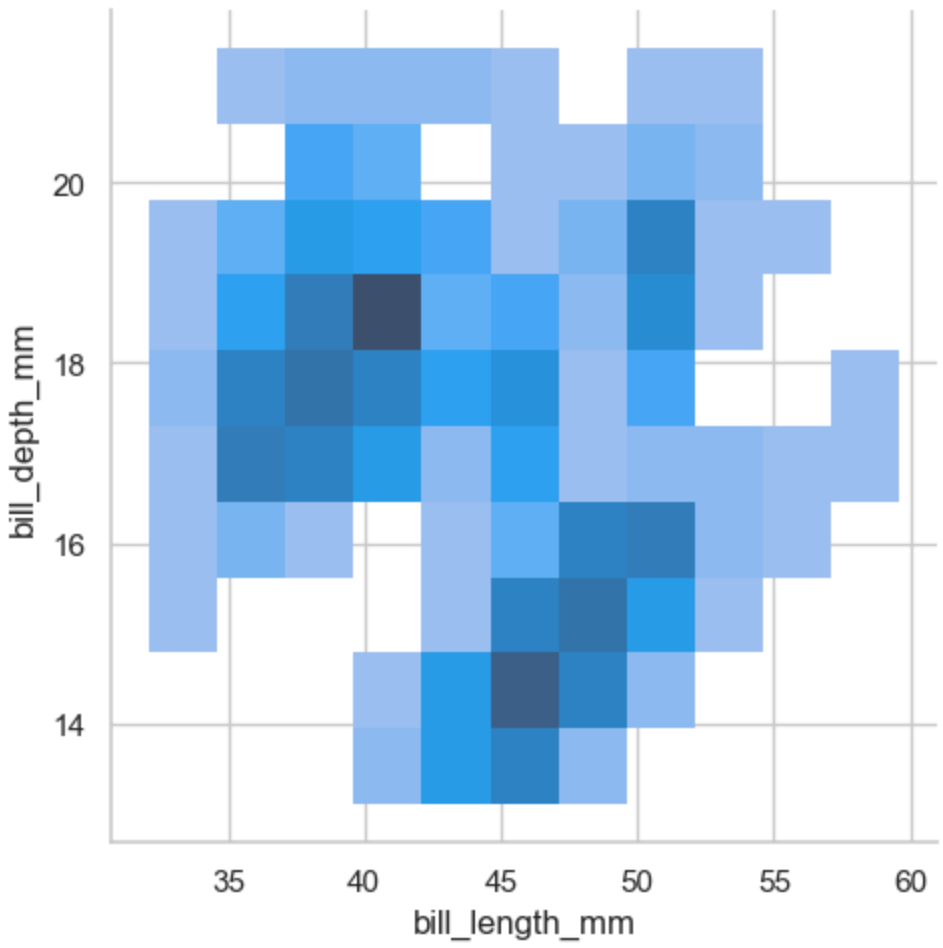
Out[ ]:
```
<AxesSubplot: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



De esta forma visualizamos una relación de 2 variables.

In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm"
)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd9c0d09db0>
```

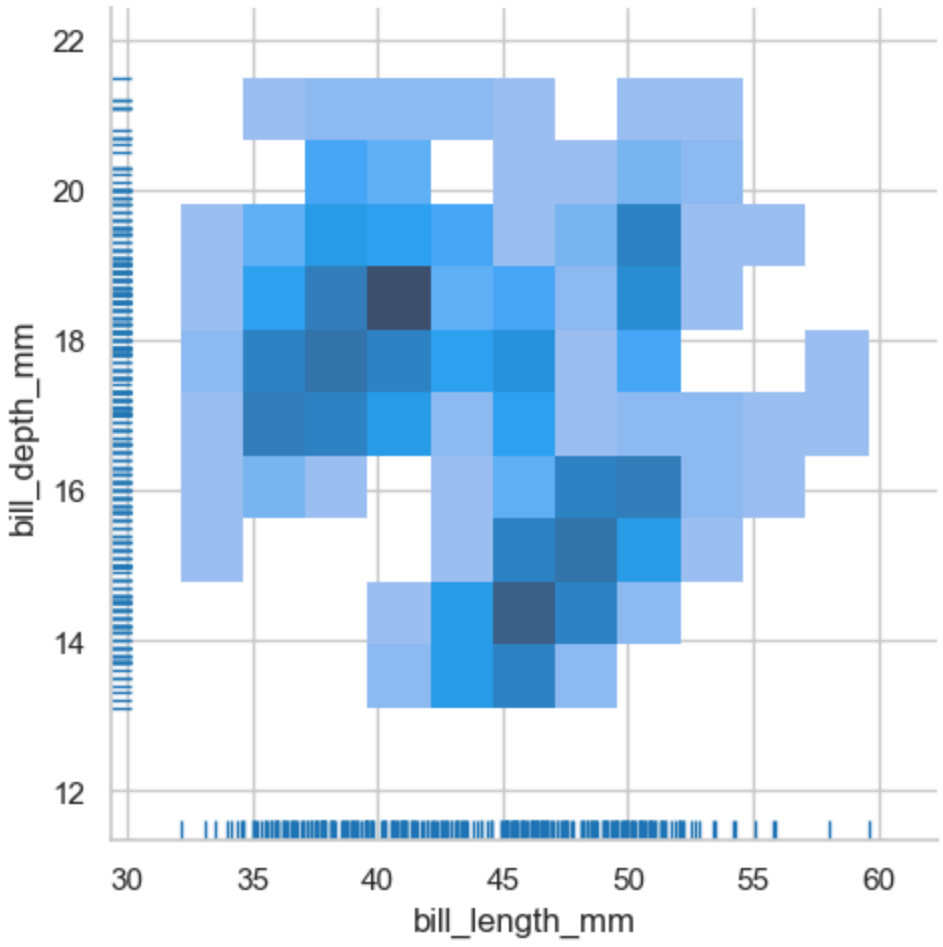


En este momento ya tenemos un histograma de frecuencias, de la interacción de las 2 variables.

El histograma nos ayuda a darnos un resumen de como están los datos, tenemos varios datos acumulados por zonas

In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm",
    rug=True
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7fd9afbf3d60>



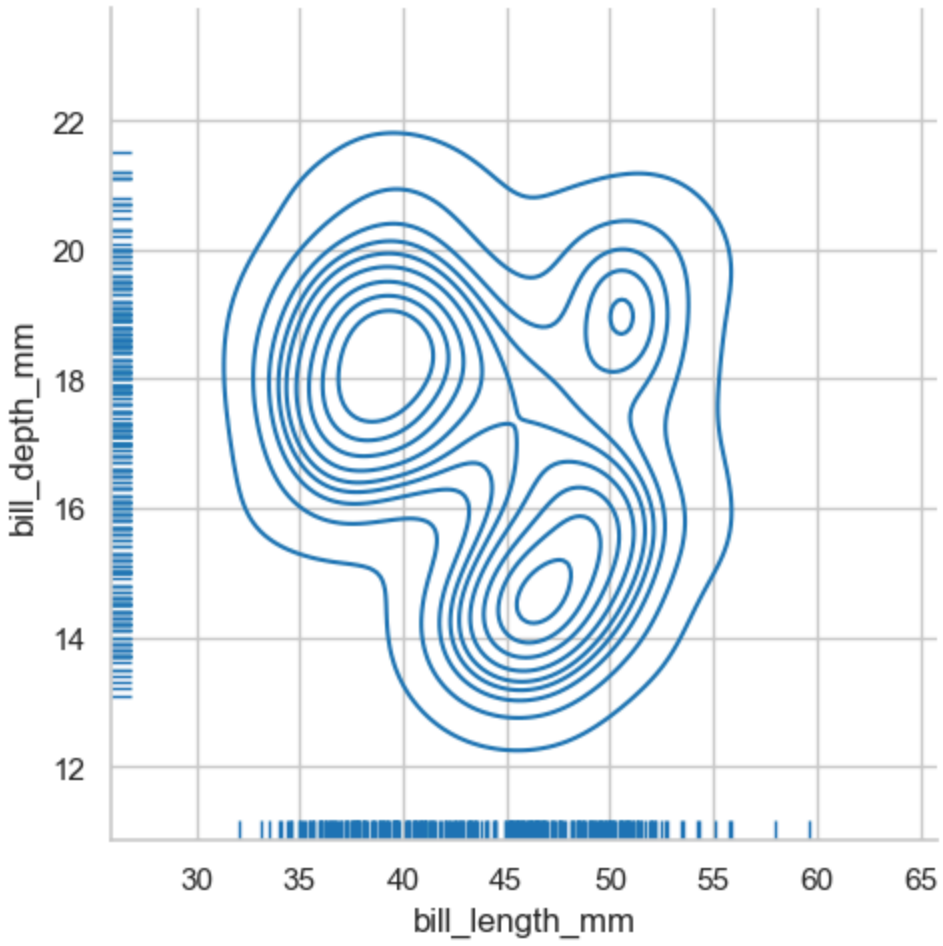
Ahora ajustaremos un parámetro que nos permite visualizar como se distribuyen los datos de manera única o como si se estuviera analizando una única variable. Este parámetro es `rug`.

Lo que me dice es que tan acumulado están los datos en ciertas zonas.

KIND = KDE

In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm",
    rug=True,
    kind="kde"
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7fd9afa8b880>

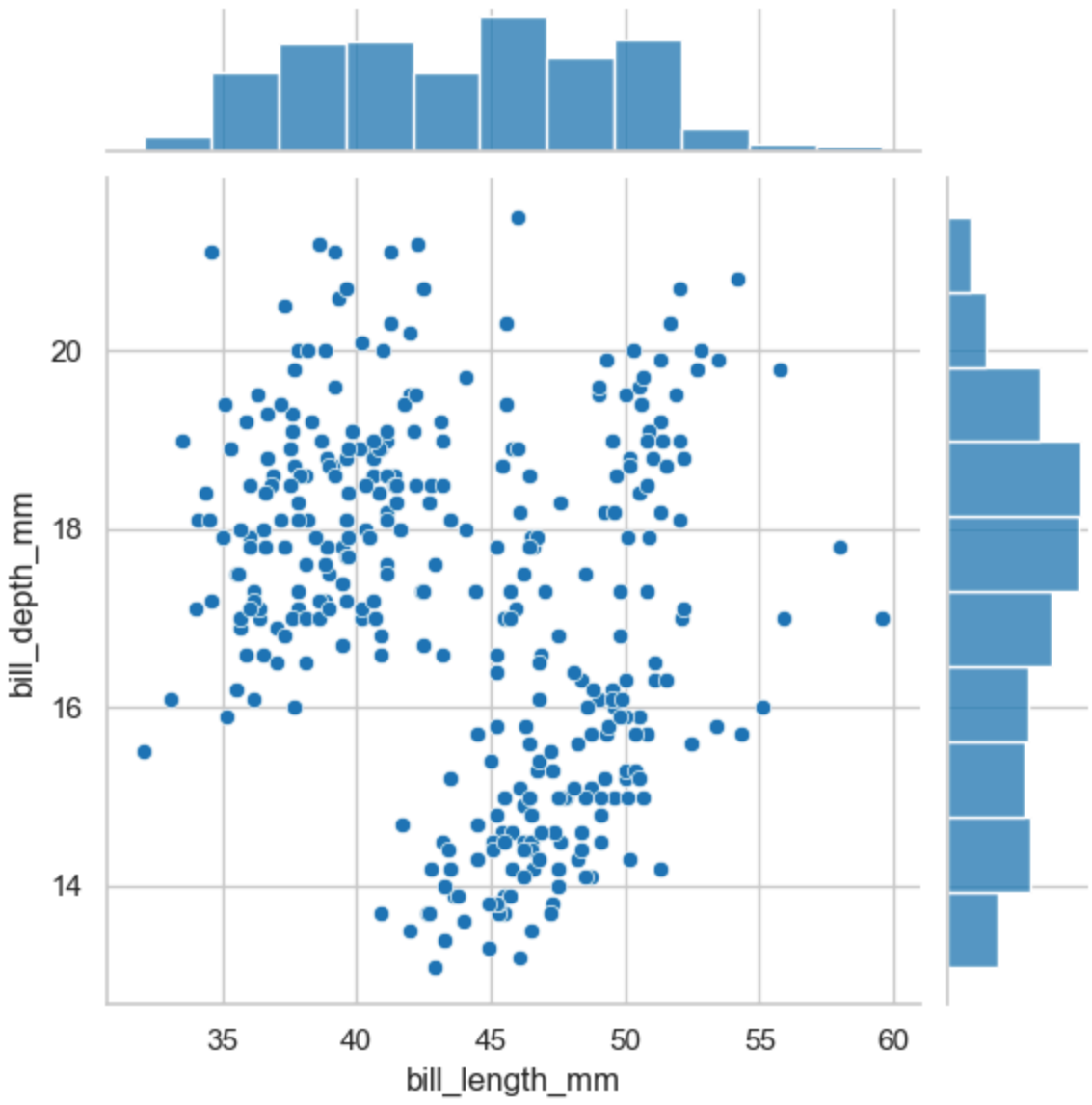


Vemos que los datos se acumulan en la especie de círculos concéntricos, son diferentes maneras de visualizar pero bastante util.

Combinando gráficas

In [ ]:
```
sns.jointplot(
    data=preprocessed_penguins_df,
    x="bill_length_mm",
    y="bill_depth_mm",
)
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x7fd9af9b3910>



Cómo se puede ver tengo mi gráfico de relaciones y ademas un histograma por cada variable, así me permite ver como se distribuyen los datos.
OJO: es muy similar que el `rug` que agregamos, pero con una gráfica.

In [ ]:
```
sns.jointplot(
    data=preprocessed_penguins_df,
    x='bill_length_mm',
    y='bill_depth_mm',
    palette=penguin_color,
    hue='species',
)
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x7fd9ad01f9a0>