Data visualization for life sciences

BIFO, 18.11.2025

Goals of this course

- Create engaging, scientifically accurate visualizations specifically tailored for biological research

Goals of this course

- Create engaging, scientifically accurate visualizations specifically tailored for biological research
- Using programming tools like R and Python

Goals of this course

- Create engaging, scientifically accurate visualizations specifically tailored for biological research
- Using programming tools like R and Python
- Use graphic editing software such as Inkscape

Goals of this course

- Create engaging, scientifically accurate visualizations specifically tailored for biological research
- Using programming tools like R and Python
- Use graphic editing software such as Inkscape
- Explore AI tools like ChatGPT and Gemini for coding support, troubleshooting, and inspiration

Goals of this course

- Create engaging, scientifically accurate visualizations specifically tailored for biological research
- Using programming tools like R and Python
- Use graphic editing software such as Inkscape
- Explore AI tools like ChatGPT and Gemini for coding support, troubleshooting, and inspiration
- Make figures ideal for publication

Course overview

This course will take place from 18 to 20 November 2025 (6 hours per day, 10am to 5pm) online via Zoom.

Course overview

This course will take place from 18 to 20 November 2025 (6 hours per day, 10am to 5pm) online via Zoom.

- Day 1 – Foundations and Basic Plotting
- Day 2 – Advanced Visualization Techniques
- Day 3 – From Plot to Publication

Day plan

- 10:00 - 11:00 Kickoff & tech checks
- 11:00 - 11:30 Data handling essentials
- 11.30 - 12.15 Hands-on
- 12.15 - 13.00 Lunch
- 13:00 - 13.55 Core plots
- 14.00 - 14.25 Color theory
- 14:30–14:45 Break (15 min)
- 14:45–16:15 Guided exercise (small groups, 3–4)
- 16:15–17:00 Share-out & feedback

Who we are

Katrina Norwood
PhD student
BIFO HZI

Philipp C. Münch
Research Associate,
Harvard School of Public Health & Helmholtz Centre for Infection Research

Georgios Kallergis
PhD student
BIFO HZI

Who are you people?

Time for a technical check!

Dataset description

The *Palmer Penguins* dataset provides real-world biological data collected from three penguin species inhabiting the Palmer Archipelago in Antarctica.

Source:
Data were collected by Dr. Kristen Gorman and the Palmer Station LTER (Long Term Ecological Research) project, supported by the U.S. National Science Foundation.

Purpose:
To explore species differences and relationships between physical measurements and habitat.

Types of variables

Numerical (Quantitative) Variables

Categorical (Qualitative) Variables

Types of variables

Numerical (Quantitative) Variables

- **Continuous:** Can take any value within a range (height, temperature, price)
- **Discrete:** Countable whole numbers (number of children, items sold, page views)

Categorical (Qualitative) Variables

Types of variables

Numerical (Quantitative) Variables

- **Continuous:** Can take any value within a range (height, temperature, price)
- **Discrete:** Countable whole numbers (number of children, items sold, page views)

Categorical (Qualitative) Variables

- **Nominal:** No inherent order (color, country, gender, product type)
- **Ordinal:** Natural ordering (education level, satisfaction rating, income bracket)

Dataset description

Variable Name	Type	Description
species	Categorical (Nominal)	Penguin species: *Adelie*, *Chinstrap*, *Gentoo*
island	Categorical (Nominal)	Island where the penguin was observed: *Torgersen*, *Biscoe*, *Dream*
sex	Categorical	Male or Female
bill_length_mm	Numeric (Continuous)	Length of the penguin's bill in millimeters
bill_depth_mm	Numeric (Continuous)	Depth of the penguin's bill in millimeters
flipper_length_mm	Numeric (Discrete/Continuous)	Length of the flipper in millimeters
body_mass_g	Numeric (Continuous)	Body mass in grams
year	Numeric (Discrete)	Year of observation (2007, 2008, 2009)

Key Python Libraries for Data Visualization

- **Pandas**
  Data manipulation and analysis library. Works with DataFrames—table-like structures with labeled rows and columns for organizing data.
- **Matplotlib:**
  Core plotting library for creating static, interactive, and publication-quality visualizations.
- **Seaborn**
  Built on Matplotlib. Provides high-level interface for creating attractive statistical graphics with less code.

Key Python Libraries for Data Visualization

- **Pandas**
  Data manipulation and analysis library. Works with DataFrames—table-like structures with labeled rows and columns for organizing data.
- **Matplotlib:**
  Core plotting library for creating static, interactive, and publication-quality visualizations.
- **Seaborn**
  Built on Matplotlib. Provides high-level interface for creating attractive statistical graphics with less code.

Magic spell: Import “library”

import pandas

Key Python Libraries for Data Visualization

- **Pandas**
  Data manipulation and analysis library. Works with DataFrames—table-like structures with labeled rows and columns for organizing data.
- **Matplotlib:**
  Core plotting library for creating static, interactive, and publication-quality visualizations.
- **Seaborn**
  Built on Matplotlib. Provides high-level interface for creating attractive statistical graphics with less code.

Magic spell: Import “library”

import pandas
import pandas as pd

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

Key components:

- Rows: Each observation/record
- Columns: Each variable/feature
- Index: Row labels (default: 0, 1, 2...)

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

1. **Easy data cleaning**, e.g., handle missing values

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

1. **Easy data cleaning**, e.g., handle missing values
2. **Fast vectorized operations,** .g., perform calculations on entire columns at once (e.g., `df["price"] * 1.2`).

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

1. **Easy data cleaning**, e.g., handle missing values
2. **Fast vectorized operations,** .g., perform calculations on entire columns at once (e.g., `df["price"] * 1.2`).
3. **Good for real-world messy data,** e.g., handle inconsistent or incomplete data

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

1. **Easy data cleaning**, e.g., handle missing values
2. **Fast vectorized operations,** .g., perform calculations on entire columns at once (e.g., `df["price"] * 1.2`).
3. **Good for real-world messy data,** e.g., handle inconsistent or incomplete data
4. **Integrates well with other tools** e.g., scikit or matplotlib

Dataframe

A 2-dimensional labeled data structure with columns of potentially different types

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

1. **Easy data cleaning**, e.g., handle missing values
2. **Fast vectorized operations,** .g., perform calculations on entire columns at once (e.g., `df["price"] * 1.2`).
3. **Good for real-world messy data,** e.g., handle inconsistent or incomplete data
4. **Integrates well with other tools** e.g., scikit or matplotlib
5. **Clear and intuitive structure**

Processing a dataset

Load a dataset:

```
df =pd.read_csv(dataset_name, sep = ',') # for csv file
df =pd.read_csv(dataset_name, sep = '\t') # for tsv file
df =pd.read_excel(dataset_name) # for excel files
```

Inspect data

```
df.head() # preview of the first 5 rows
df.head(10) # preview of the first 10 rows
df.tail() # preview of last 5 rows
df.describe() # statistical description of data
df.info() # overview of data (variables, population, variable type)
```

Data types

Type Conversion

- df['price'] = pd.to_numeric(df['price'])
- pd.to_numeric(df['col'], errors='coerce')
- df['col'].astype('int64')

At Read Time

- pd.read_csv('data.csv', dtype={'price': float})

Check types

- df.dtypes

Handling NA values

Find NA values in each column

```
df.isna().sum()  # Count NAs per column
df.isnull().sum()  # Same thing (isna and isnull are aliases)
```

	0
species	0
island	0
bill_length_mm	2
bill_depth_mm	2
flipper_length_mm	2
body_mass_g	2
sex	11

dtype: int64

Filling NA values

Use mean, median or specific value:

```
df['bill_length_mm'].fillna(df['bill_length_mm'].median(),
inplace=True)
```

Dropping NA values

Drop columns with NA values

```
df_col_all = df.dropna(axis = 1, inplace =False, how = 'all') #drops column if all values are missing
df_col_all = df.dropna(axis = 1, inplace =False, how = 'any') #drops columns even if one value is NA
```

Drop rows with NA values

```
df_col_all = df.dropna(axis = 0, inplace =False, how = 'all') #drops rows even if one value is NA
df_col_all = df.dropna(axis = 0, inplace =False, how = 'any') #drops rows even if one value is NA
```

Drop rows with less values than a threshold

```
df_drop= df.dropna(axis = 1, thresh=340)  # Keep rows with at least 340 non-NA values
```

Wide vs Long format

Wide format

- Each variable has its own column
- Human-readable, spreadsheet friendly

```
ID | 2021 | 2022 | 2023
---+------+------+-----
A  |  100 |  120 |  150
B  |   80 |   95 |  110
```

Long format

- Each observation is a row
- Analysis -friendly, tidy data

```
ID | Year | Value
---+------+------
A  | 2021 |  100
A  | 2022 |  120
A  | 2023 |  150
B  | 2021 |   80
B  | 2022 |   95
B  | 2023 |  110
```

Wide vs Long format

pandas - melt()

```
df_long = df.melt(
id_vars=['ID'],
value_vars=['2021', '2022', '2023'],
var_name='Year',
value_name='Value'
)
```

id_vars: Keep as identifiers
value_vars: Columns to unpivot
var_name: New variable column
value_name: New value column

R - pivot_longer()

```
library(tidyr)
df_long <- df %>%
pivot_longer(
cols = c('2021', '2022', '2023'),
names_to = 'Year',
values_to = 'Value'
)
```

cols: Columns to pivot
names_to: New names column
values_to: New values column

Pivot

```
df_wide = df_long.pivot(
index='ID',
columns='Year',
values='Value'
)
```

```
# For aggregating duplicates
df.pivot_table(
index='ID', columns='Year',
values='Value', aggfunc='mean')
```

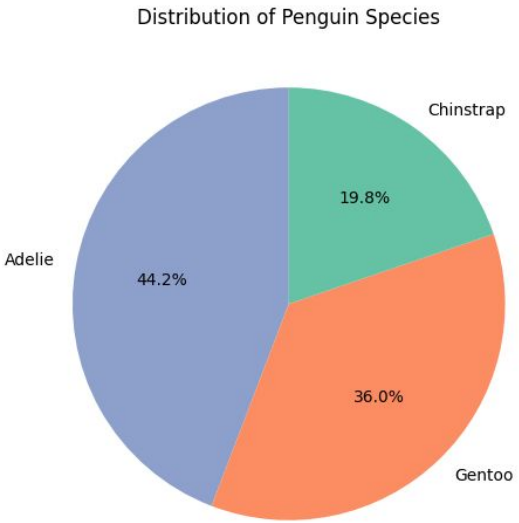
R - pivot_wider()

```
library(tidyr)
df_wide <- df_long %>%
pivot_wider(
names_from = Year,
values_from = Value
)
```

names_from: Values to become column names
values_from: Values to fill new columns

Pie charts

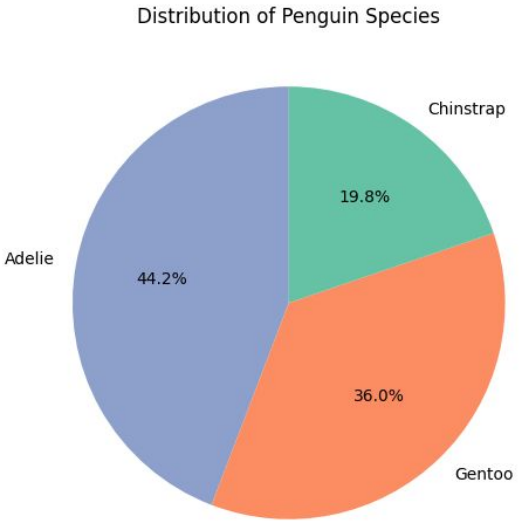
A circular graphic divided into wedge-shaped sectors, where each sector represents a proportion or percentage of a whole, making it easy to visualize the relationship between parts and their total.



- Quick, high-level view
- Highlight a dominant category
- Visualizing relative proportions
- Simple, visual appeal (good for infographics)

Pie charts

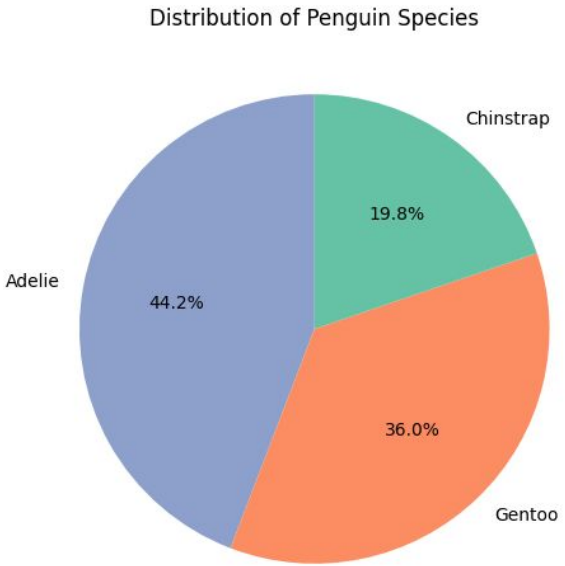
A circular graphic divided into wedge-shaped sectors, where each sector represents a proportion or percentage of a whole, making it easy to visualize the relationship between parts and their total.




```
species_counts = penguins["species"].value_counts()
# counts the number of penguins per species

# Create pie chart
plt.pie(species_counts, labels=species)
```

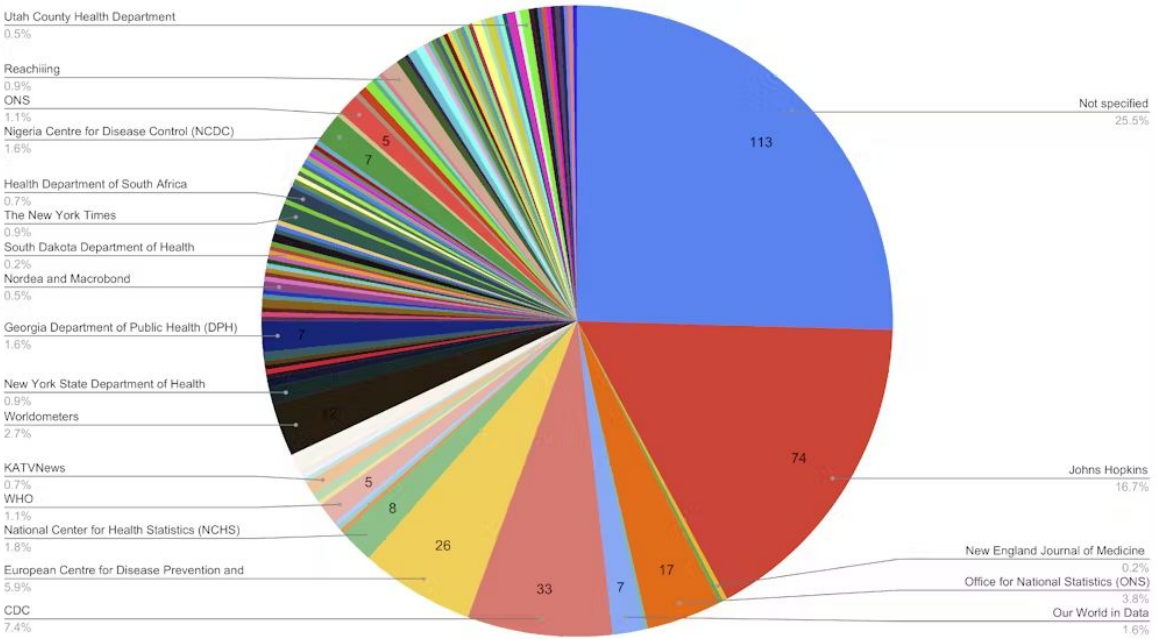
Pie charts



“There is no data that can be displayed in a pie chart, that cannot be displayed BETTER in some other type of chart.”

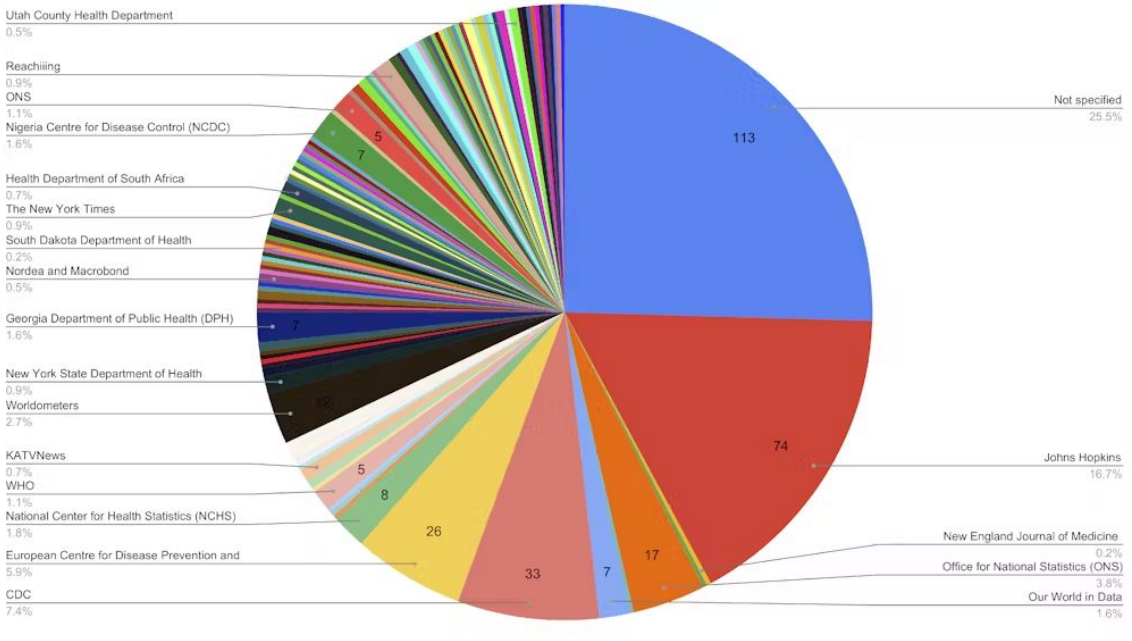
John Tukey

What’s wrong with that pie?



Trajkova et al., Informatics (2020); modified., CC BY https://theconversation.com/heres-why-you-should-almost-never-use-a-pie-chart-for-your-data-214576

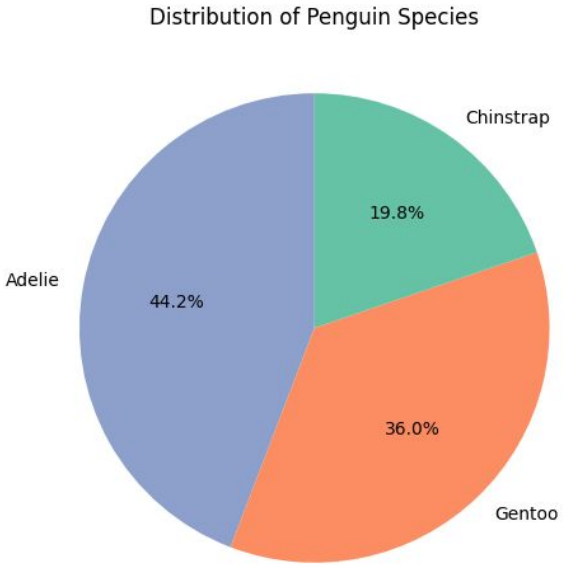
What’s wrong with that pie?



- Too many slices
- Too small slices
- Compare categories across multiple groups
- Precise comparison is important

Trajkova et al., Informatics (2020); modified., CC BY https://theconversation.com/heres-why-you-should-almost-never-use-a-pie-chart-for-your-data-214576

Pie charts



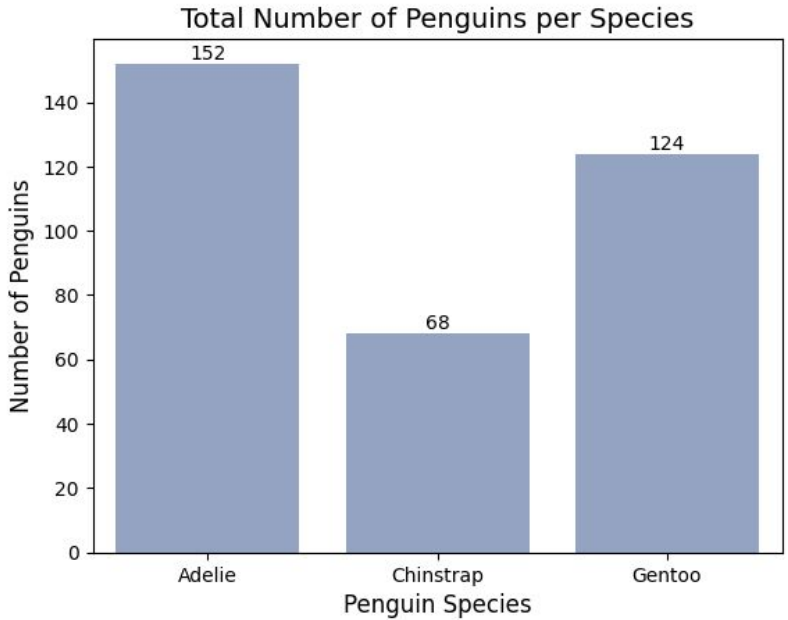
"There is no data that can be displayed in a pie chart, that cannot be displayed BETTER in some other type of chart."

John Tukey

Save pies for dessert!

Bar plots

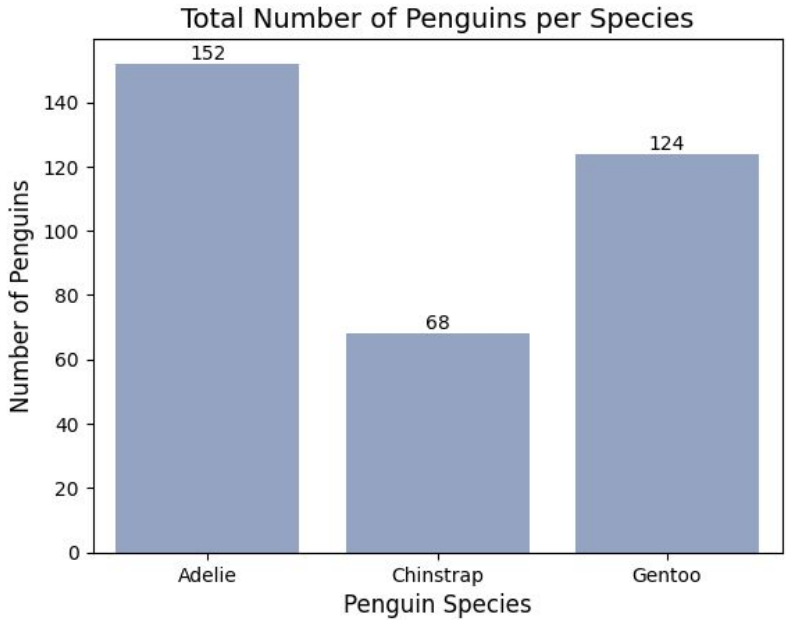
Bars indicate comparisons among discrete categories.



Bar plots

Bars indicate comparisons among discrete categories.

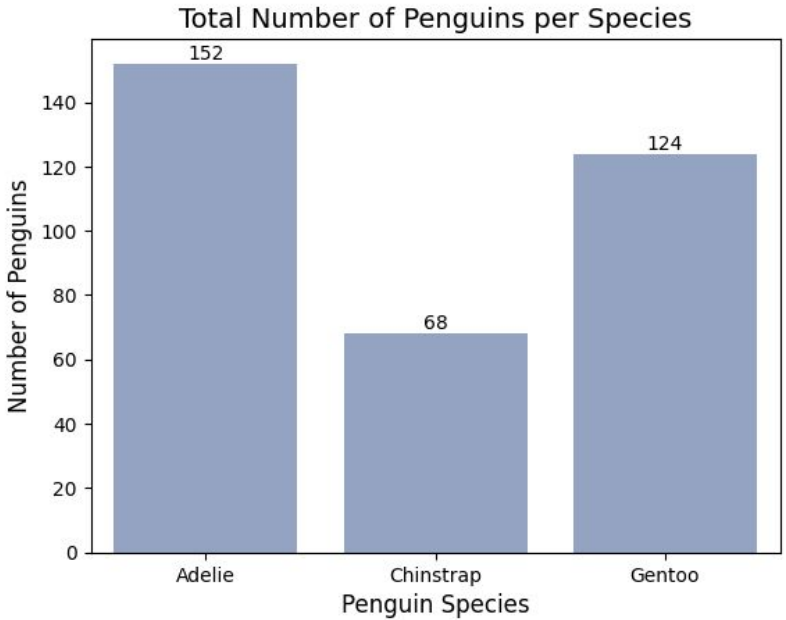
- x-axis typically represents different categories or qualitative variables.



Bar plots

Bars indicate shows comparisons among discrete categories.

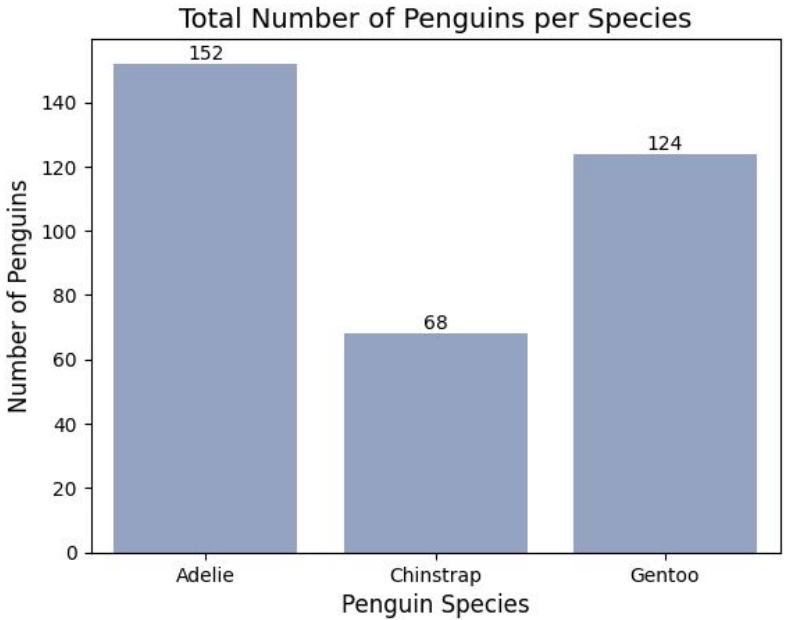
- x-axis typically represents different categories or qualitative variables.
- y-axis or length of the bars represents the quantitative values
    - counts, frequencies, or other numerical measures.



Bar plots

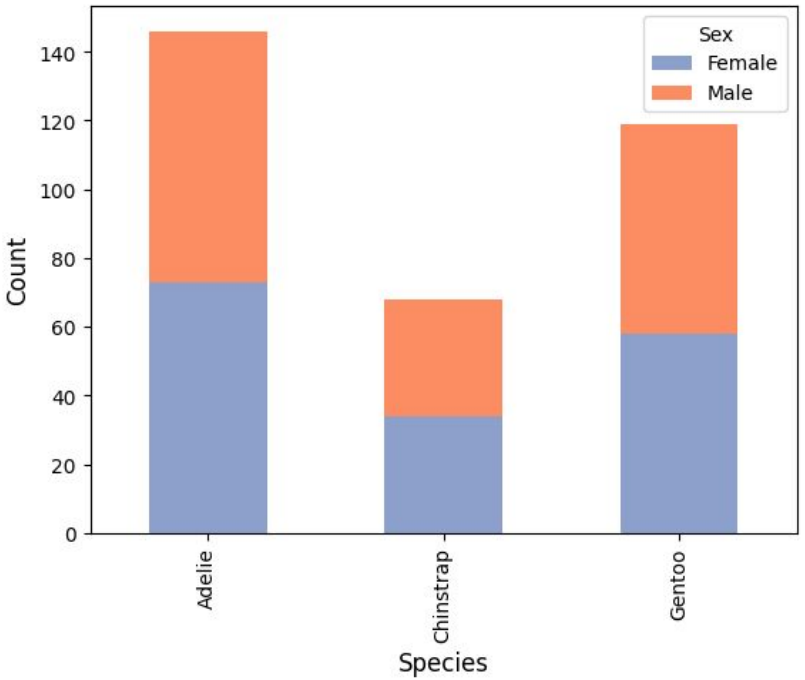
Bars indicate shows comparisons among discrete categories.

- x-axis typically represents different categories or qualitative variables.
- y-axis or length of the bars represents the quantitative values
    - counts, frequencies, or other numerical measures.



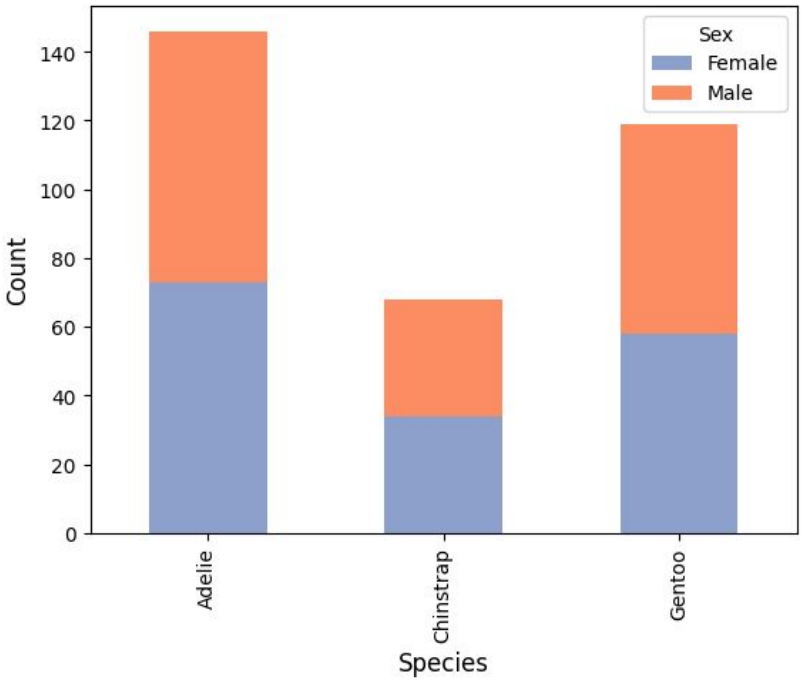

```
ax = sns.countplot(data=penguins, x="species")
```

Stacked barplot



Bars for different categories are stacked on top of each other (vertical) or side by side within one bar (horizontal).

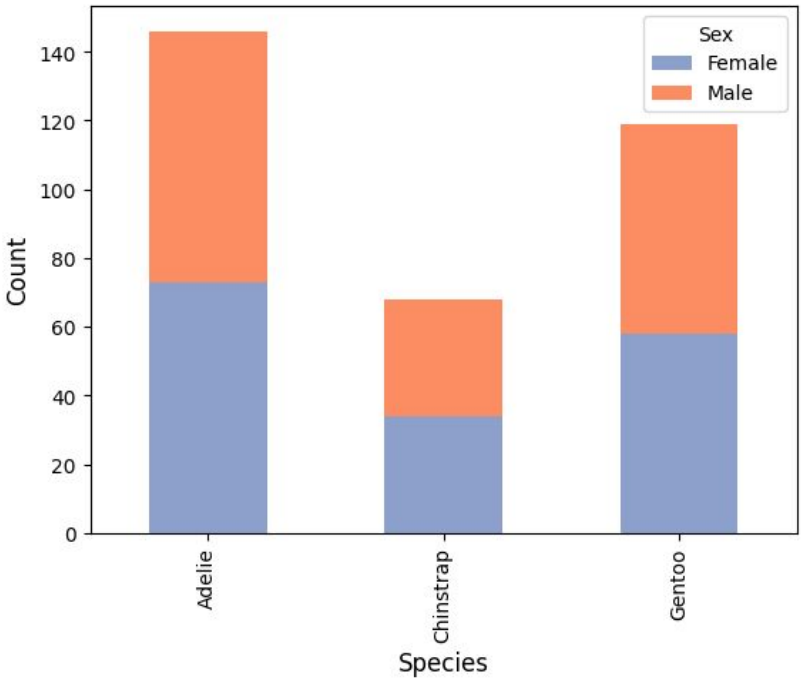
Stacked barplot



Bars for different categories are stacked on top of each other (vertical) or side by side within one bar (horizontal).

- Each bar represents the total for a group, with the internal sections showing how much each subcategory contributes.

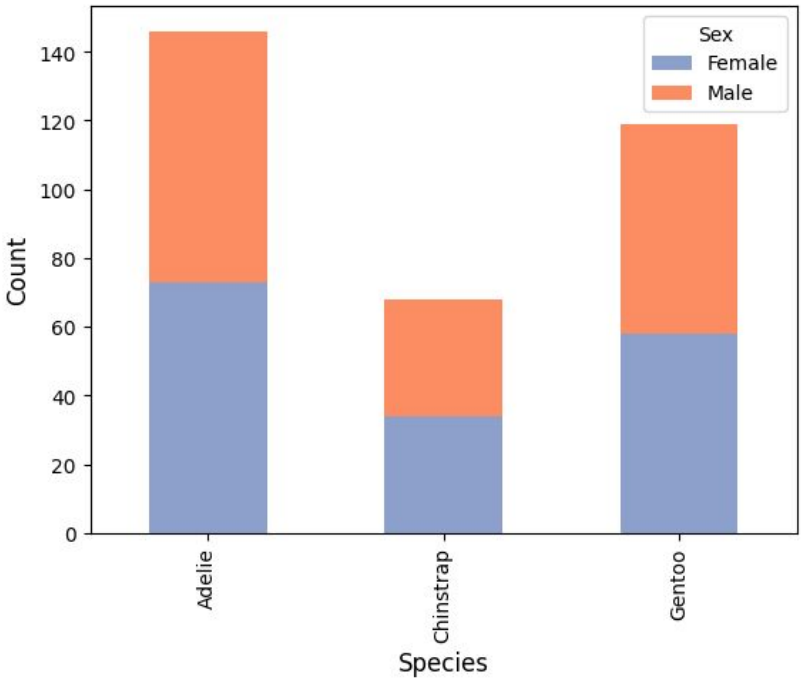
Stacked barplot



Bars for different categories are stacked on top of each other (vertical) or side by side within one bar (horizontal).

- Each bar represents the total for a group, with the internal sections showing how much each subcategory contributes.
- Use case: When you want to emphasize the overall total and also show the composition of that total.

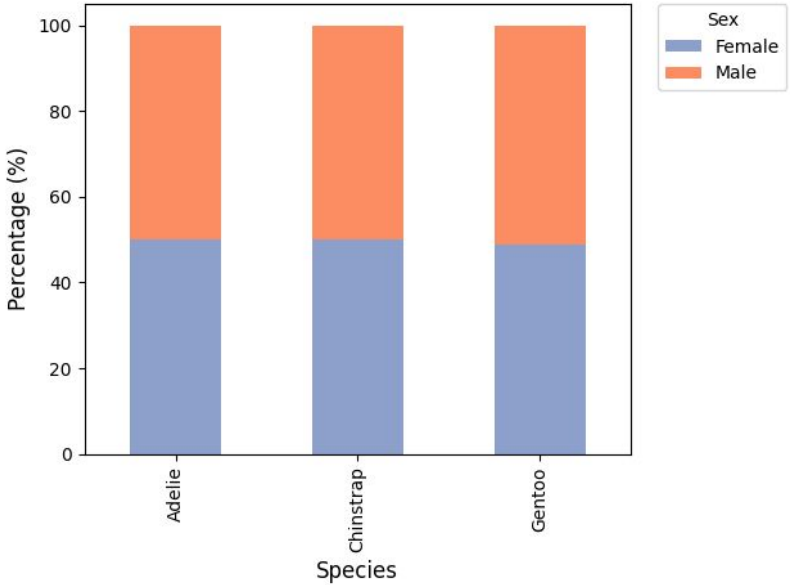
Stacked barplot



Bars for different categories are stacked on top of each other (vertical) or side by side within one bar (horizontal).

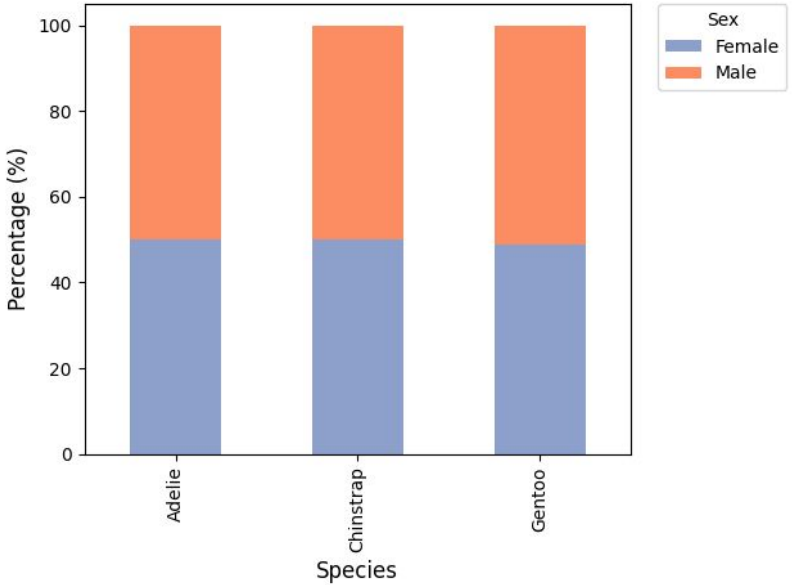
- Each bar represents the total for a group, with the internal sections showing how much each subcategory contributes.
- Use case: When you want to emphasize the overall total and also show the composition of that total.
- Hard to compare individual subcategories across groups (except the bottom one, since it has a fixed baseline).

Percentage stacked barplot



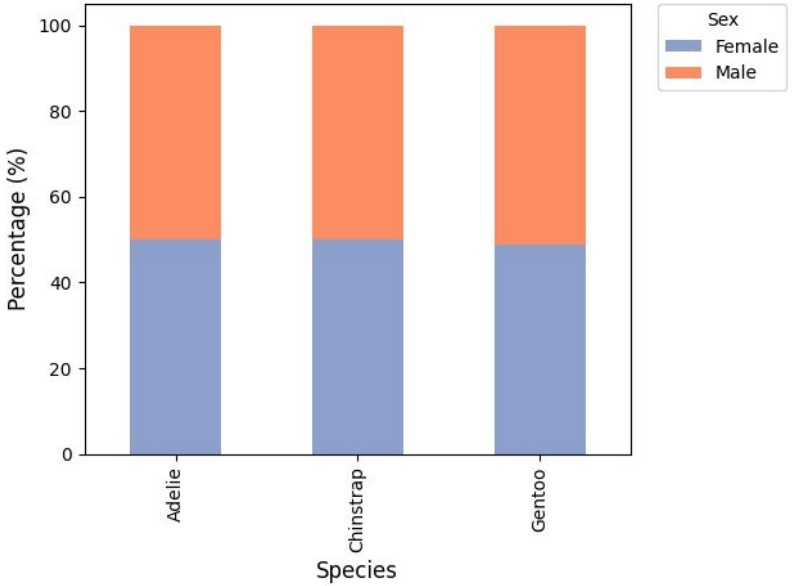
- Easy proportion comparison - All bars reach 100%, making it simple to compare relative sizes

Percentage stacked barplot



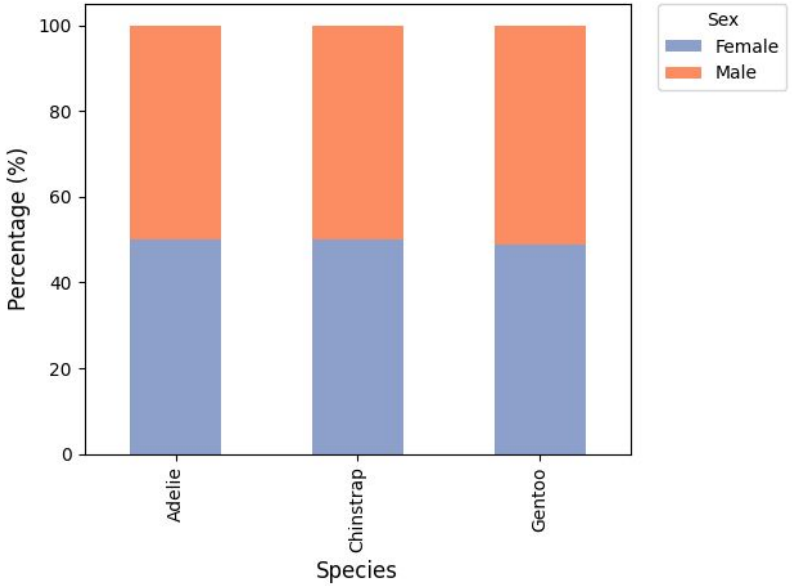
- Easy proportion comparison - All bars reach 100%, making it simple to compare relative sizes
- Standardizes comparison - Groups with different totals can be compared fairly

Percentage stacked barplot



- Easy proportion comparison - All bars reach 100%, making it simple to compare relative sizes
- Standardizes comparison - Groups with different totals can be compared fairly
- Shows composition - Clearly displays what percentage each subcategory contributes

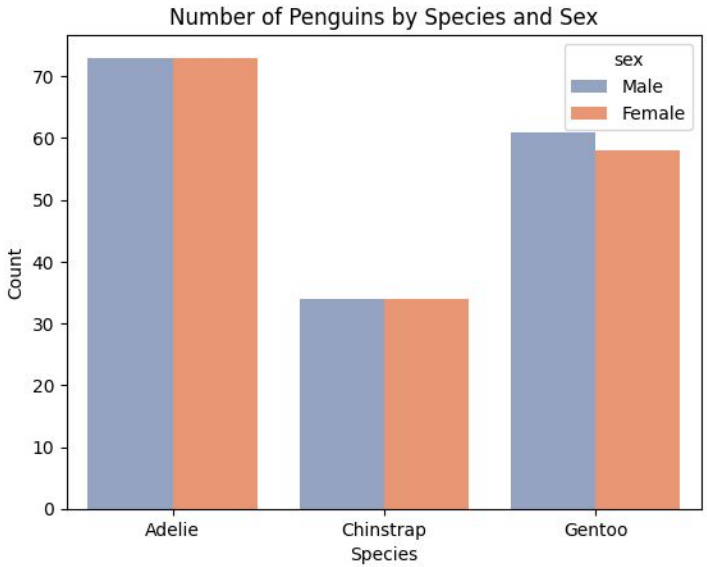
Percentage stacked barplot



- Easy proportion comparison - All bars reach 100%, making it simple to compare relative sizes
- Standardizes comparison - Groups with different totals can be compared fairly
- Shows composition - Clearly displays what percentage each subcategory contributes
- Space efficient - Fits multiple groups in limited space

Side-by-Side (Grouped) Bar Plot

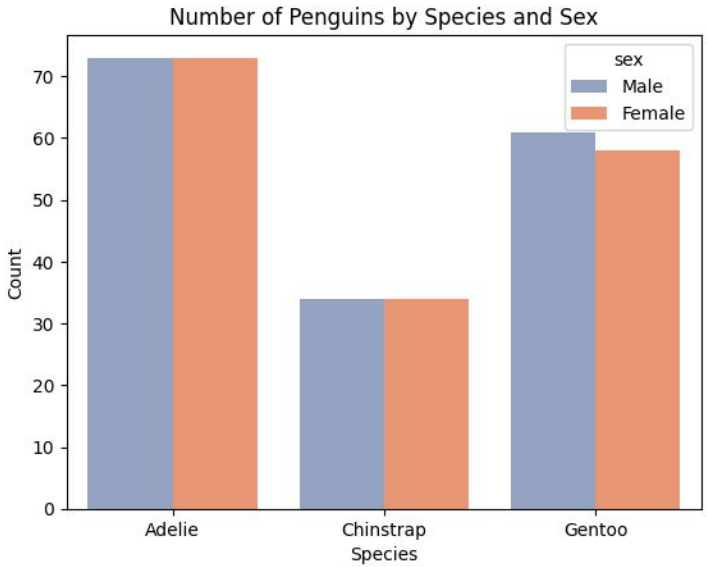
Bars for different categories are placed next to each other within a group, rather than stacked.



Side-by-Side (Grouped) Bar Plot

Bars for different categories are placed next to each other within a group, rather than stacked.

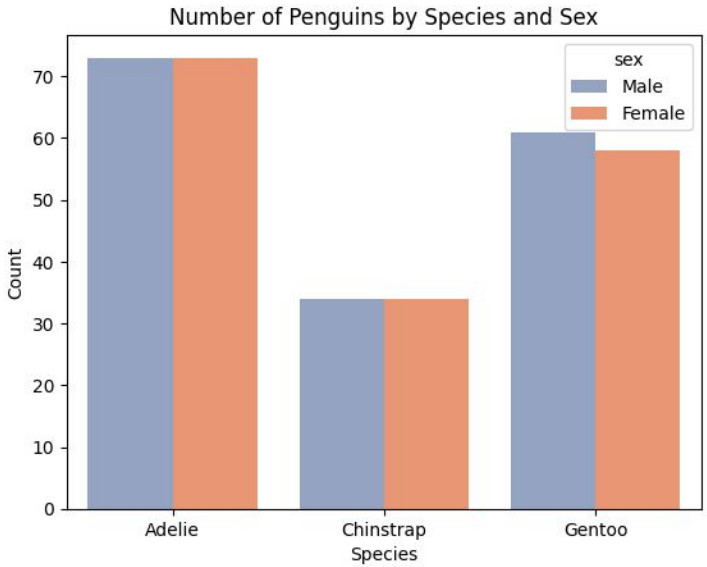
Each group of bars represents the categories for one group.



Side-by-Side (Grouped) Bar Plot

Bars for different categories are placed next to each other within a group, rather than stacked.

Each group of bars represents the categories for one group. Use case: When you want to compare subcategories across groups directly.

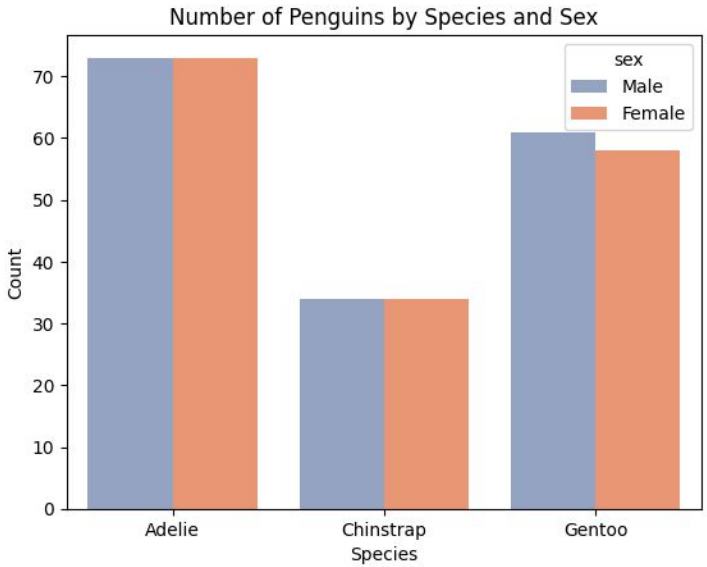


Side-by-Side (Grouped) Bar Plot

Bars for different categories are placed next to each other within a group, rather than stacked.

Each group of bars represents the categories for one group. Use case: When you want to compare subcategories across groups directly.

- Easy to compare subcategories between groups.
- Clear separation of categories.

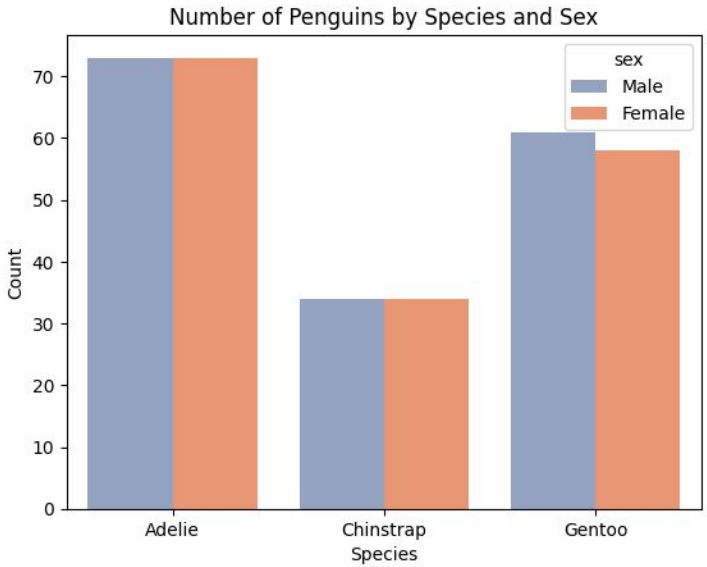


Side-by-Side (Grouped) Bar Plot

Bars for different categories are placed next to each other within a group, rather than stacked.

Each group of bars represents the categories for one group.
Use case: When you want to compare subcategories across groups directly.

- Easy to compare subcategories between groups.
- Clear separation of categories.
- Harder to see overall totals at a glance.
- Can get cluttered if there are too many subcategories.

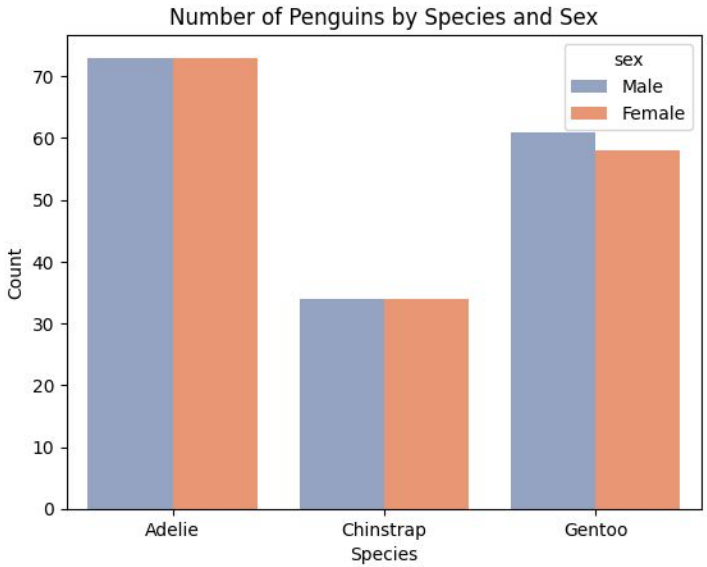


Side-by-Side (Grouped) Bar Plot

Bars for different categories are placed next to each other within a group, rather than stacked.

Each group of bars represents the categories for one group. Use case: When you want to compare subcategories across groups directly.

- Easy to compare subcategories between groups.
- Clear separation of categories.
- Harder to see overall totals at a glance.
- Can get cluttered if there are too many subcategories.




```
sns.countplot(data=penguins, x="species", hue="sex")
```

Stacked vs side-by-side barplot

Stacked vs side-by-side barplot

Need to compare specific values? →

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side

Need to show totals + breakdown? → ?

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side

Need to show totals + breakdown? → Stacked

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side
Need to show totals + breakdown? → Stacked
Need to compare proportions? → ?

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side
Need to show totals + breakdown? → Stacked
Need to compare proportions? → Percentage stacked

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side
Need to show totals + breakdown? → Stacked
Need to compare proportions? → Percentage stacked
Space constrained? → ?

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side
Need to show totals + breakdown? → Stacked
Need to compare proportions? → Percentage stacked
Space constrained? → Stacked

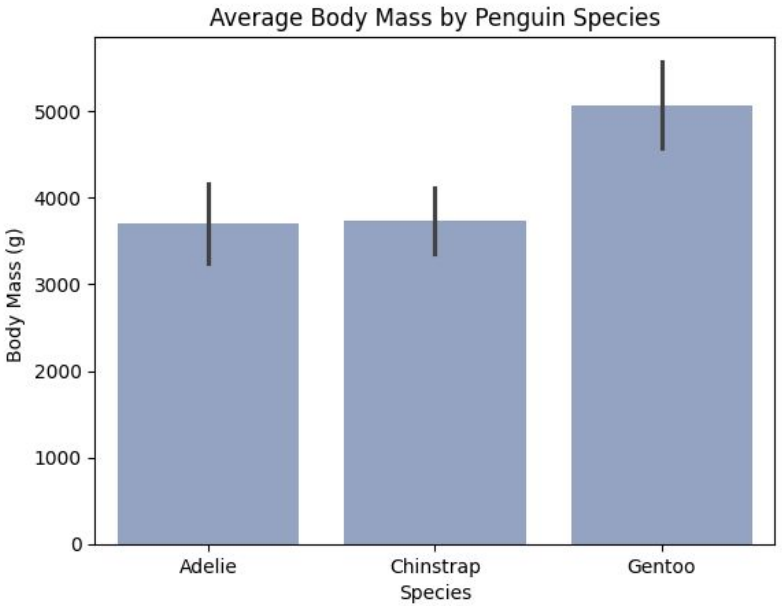
Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side
Need to show totals + breakdown? → Stacked
Need to compare proportions? → Percentage stacked
Space constrained? → Stacked
Precision matters? → ?

Stacked vs side-by-side barplot

Need to compare specific values? → Side-by-side
Need to show totals + breakdown? → Stacked
Need to compare proportions? → Percentage stacked
Space constrained? → Stacked
Precision matters? → Side-by-side

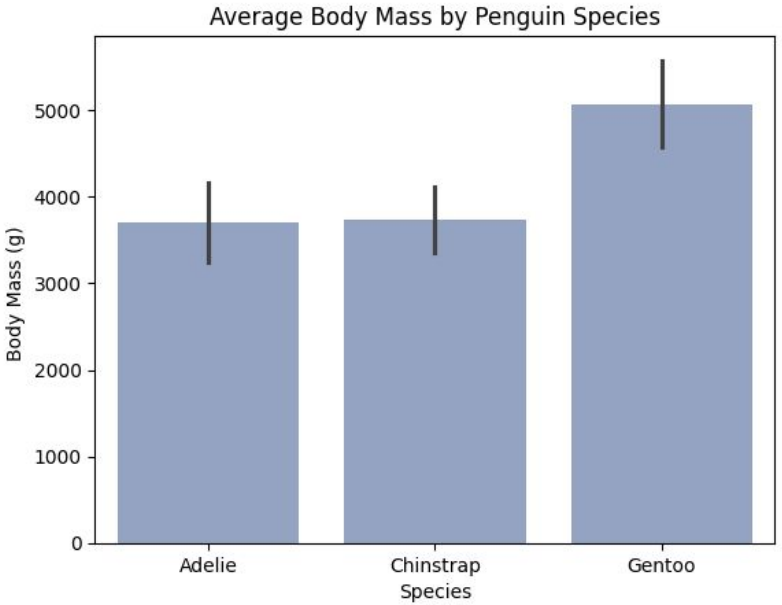
Error bar



A graphical representation of the variability or uncertainty in data in which vertical or horizontal lines extending from data points (like bars, points, or means).

- Show how much a **measured value** may vary from the **true value**.

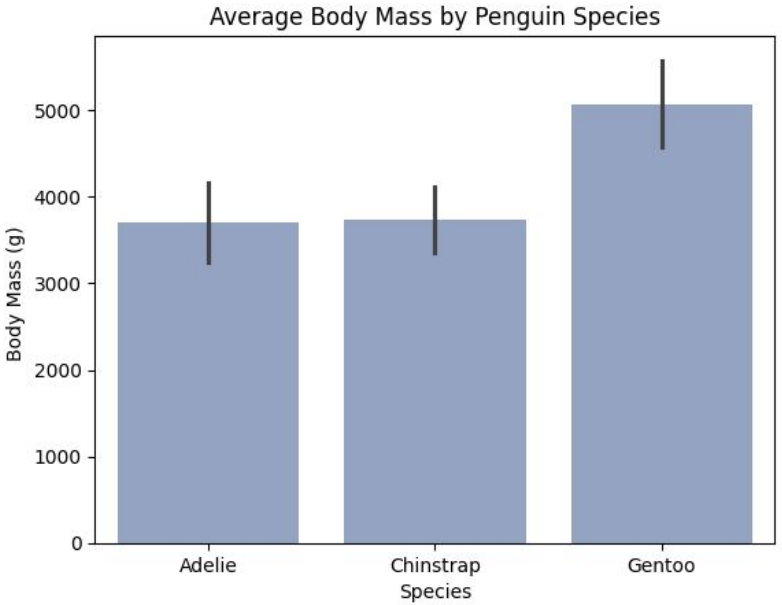
Error bar



A graphical representation of the variability or uncertainty in data in which vertical or horizontal lines extending from data points (like bars, points, or means).

- Show how much a **measured value** may vary from the **true value**.
- Can represent **standard deviation (SD)**, **standard error (SE)**, or **confidence intervals (CI)** — depending on the context.

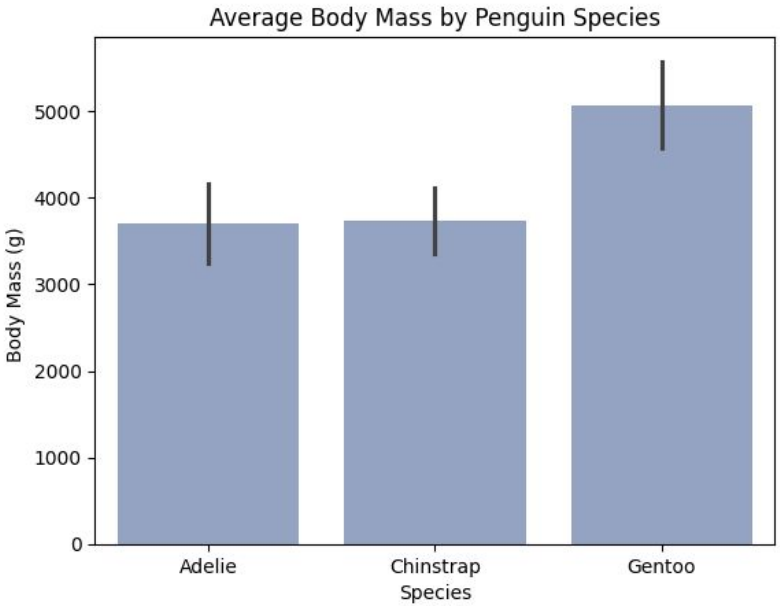
Error bar



A graphical representation of the variability or uncertainty in data in which vertical or horizontal lines extending from data points (like bars, points, or means).

- Show how much a **measured value** may vary from the **true value**.
- Can represent **standard deviation (SD)**, **standard error (SE)**, or **confidence intervals (CI)** — depending on the context.
- Help indicate the **reliability** and **precision** of measurements.
    - **Short error bars** → more precise or less variable data.
    - **Long error bars** → more variability or higher uncertainty.

Error bar

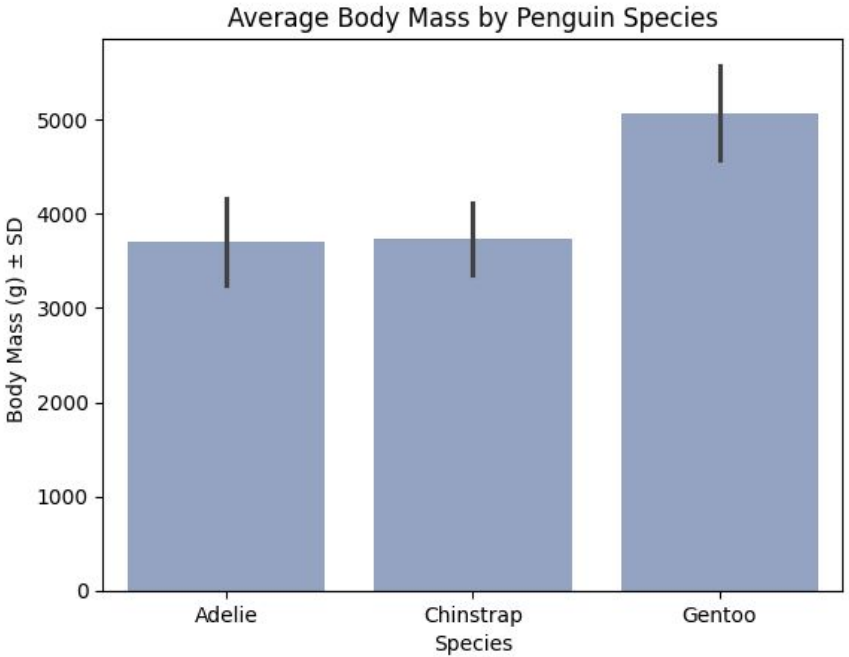


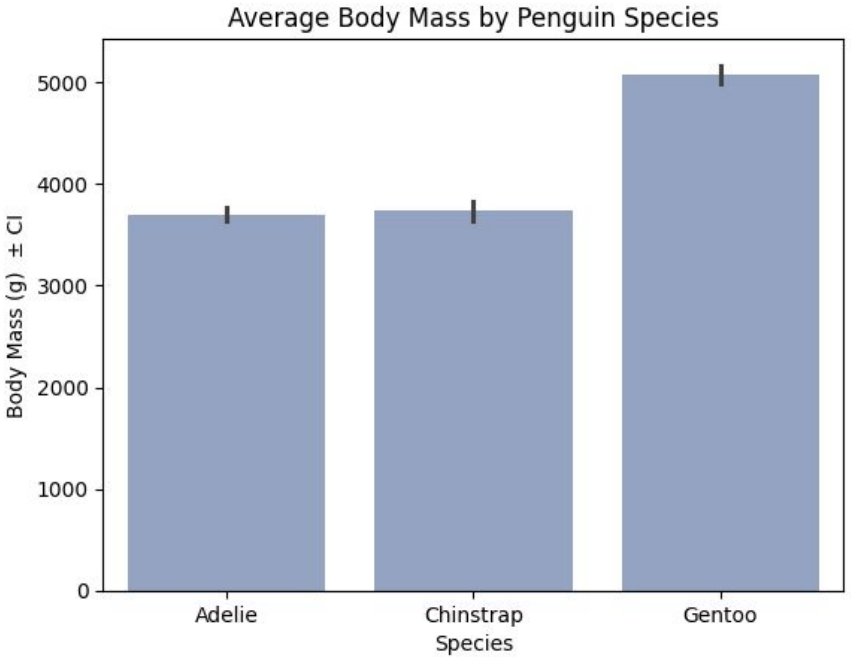
A graphical representation of the variability or uncertainty in data in which vertical or horizontal lines extending from data points (like bars, points, or means).

- Show how much a **measured value** may vary from the **true value**.
- Can represent **standard deviation (SD)**, **standard error (SE)**, or **confidence intervals (CI)** — depending on the context.
- Help indicate the **reliability** and **precision** of measurements.
    - **Short error bars** → more precise or less variable data.
    - **Long error bars** → more variability or higher uncertainty.

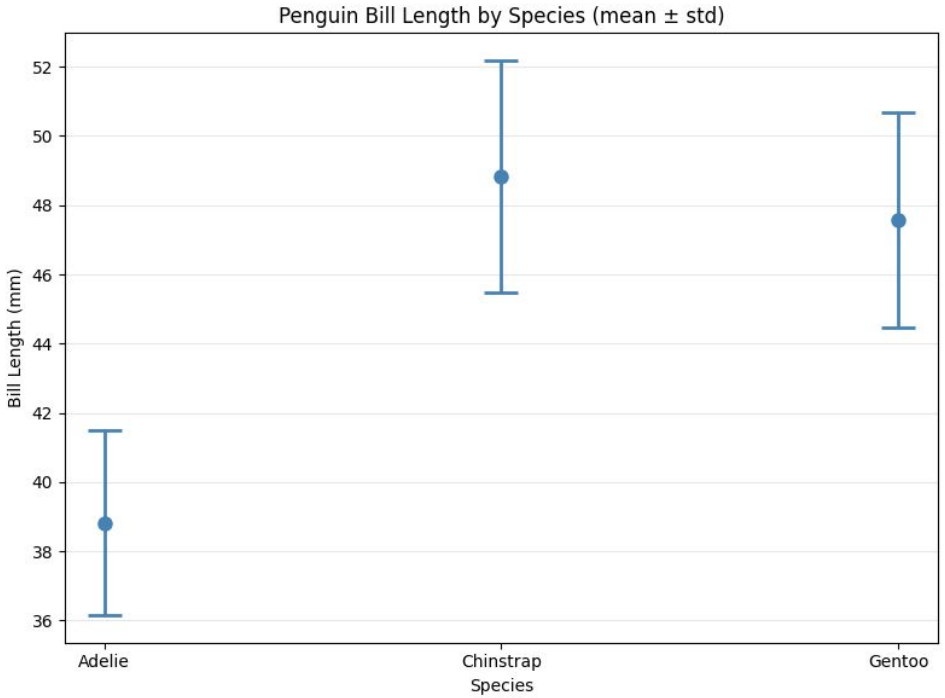
Should always include a **legend or note** explaining what the error bars represent

Always state what the error bars represent!





More on error bars




```
plt.errorbar(x_positions, means, yerr=sds,
             fmt='o',      # circle markers
             capsize=10,         # cap width
             capthick=2,         # cap thickness
             markersize=8,
             linewidth=2,
             color='steelblue')
```

Standard Deviation vs Standard Error

Standard Deviation (SD)

What it Measures

Variability of individual data points from the mean

Formula

$$SD = \sqrt{[\Sigma(x - \mu)^2 / N]}$$

When to Use

- Describing data spread
- Comparing variability
- Unchanged by sample size

Standard Error (SE)

What it Measures

Precision of sample mean as estimate of population mean

Formula

$$SE = SD / \sqrt{n}$$

When to Use

- Making inferences
- Confidence intervals
- Decreases with larger n

Key Distinction: SD describes your data | SE describes the uncertainty in your estimate

When to use scatterplots

- **Show the relationship between two numeric variables**
- **Reveal trends, patterns, or clusters**
- **Spot outliers**
- **Compare multiple groups**
- **Explore correlations**

- **One or both variables are categorical**
- **You have very few or too many data points**

Data distribution

Graphical representation of the distribution of numerical data to visualize the frequency or count of data points within specified ranges (bins).

Key Features:

- Bars represent intervals of a continuous variable.
- Height of bars shows the frequency or density of data in each bin.
- No gaps between bars (unless no data in a bin), indicating continuous data.

Applications:

- Analyze data spread, central tendency, and variability.
- Identify patterns like skewness, modality (uni-, bi-, multi-modal), or outliers.

Example: Display a histogram of test scores to show how many students scored within specific grade ranges.

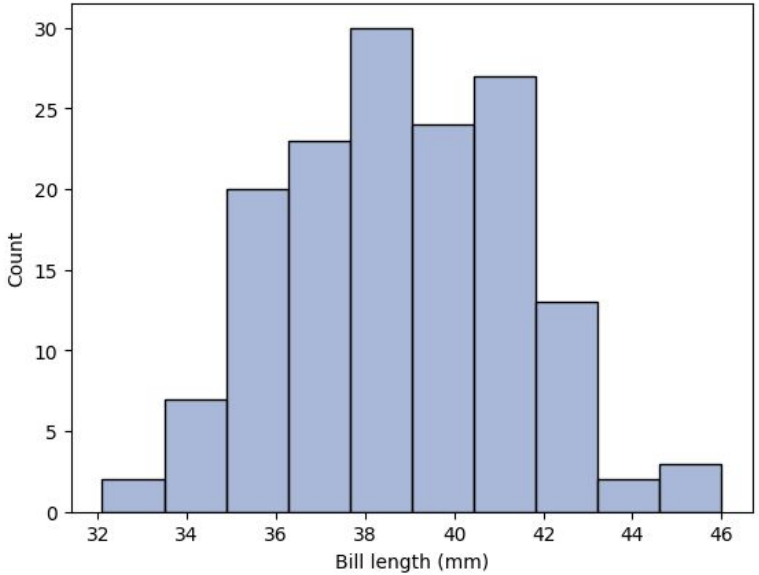
Histogram

A **histogram** is a graphical representation of the **distribution of numerical data**.

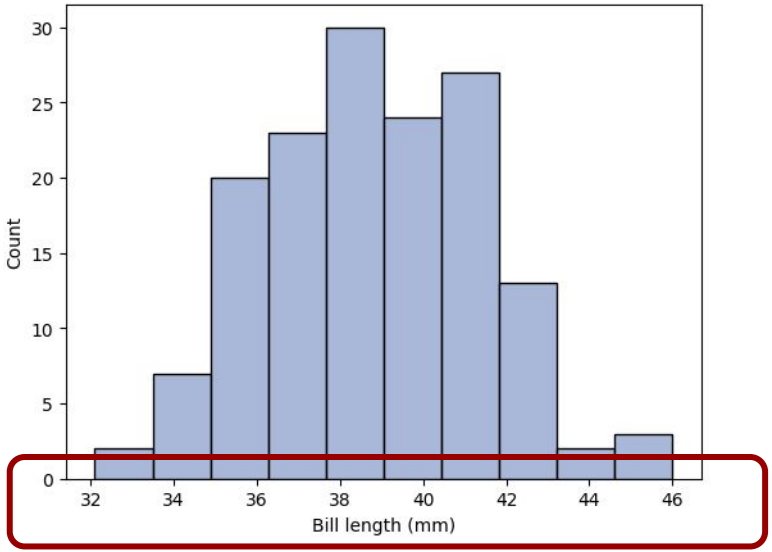
- It displays data by grouping values into **bins** (intervals) and showing the **frequency** of observations within each bin using **rectangular bars**.
- The **height of each bar** corresponds to the number (or proportion) of data points in that range.

Histograms are commonly used to:

- Visualize the **shape** of data distributions (e.g., normal, skewed, bimodal).
- Identify **central tendency**, **variability**, and **outliers**.
- Compare distributions across different datasets or experimental conditions.

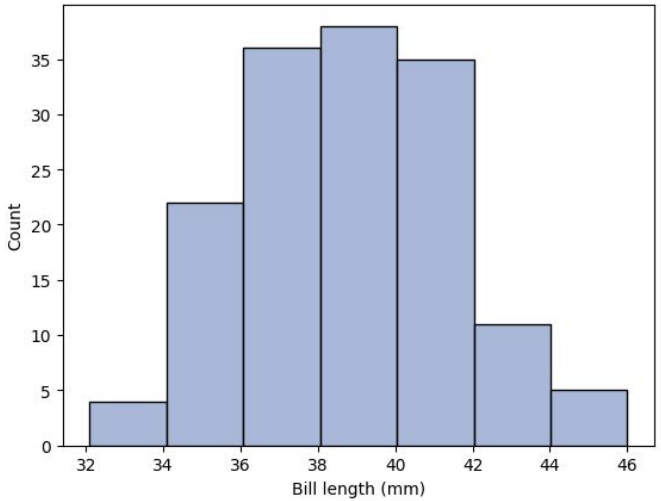


Histogram




```
sns.histplot(data_h, x = 'bill_length_mm')
```

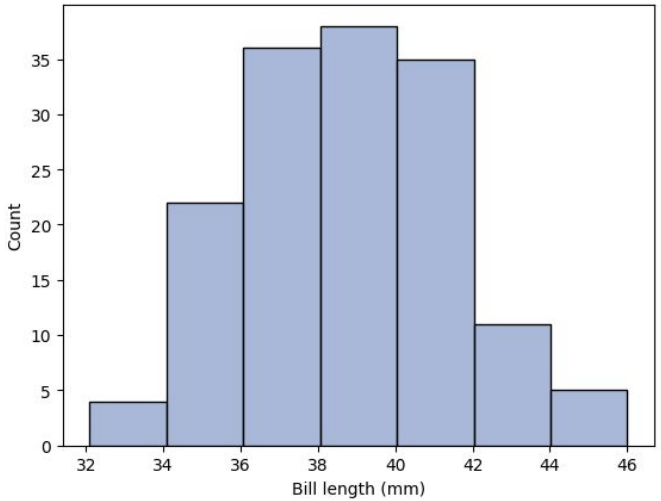
Histogram



a) Define the bin width (width=2)

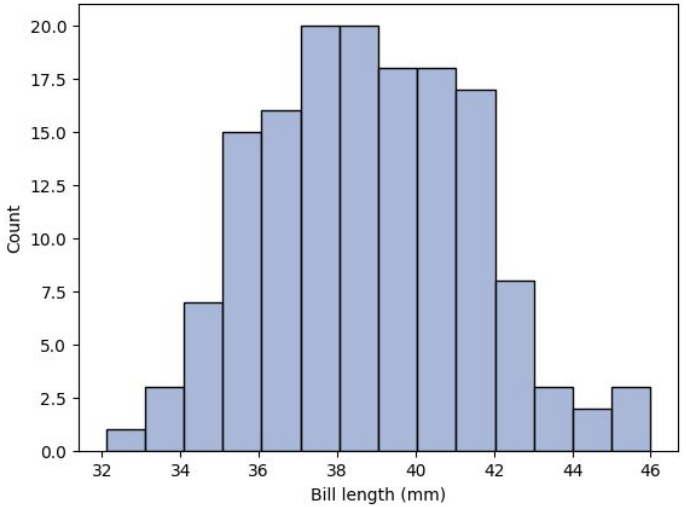
```
sns.histplot(data_h, x = 'bill_length_mm', binwidth = 2)
```

Histogram



a) Define the bin width (width=2)

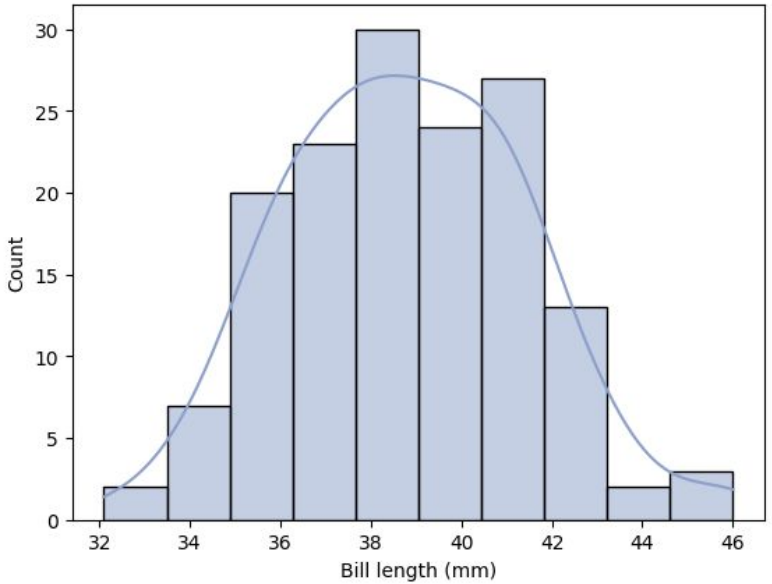
```
sns.histplot(data_h, x = 'bill_length_mm', binwidth = 2)
```



b) Define number of bins based on the task

```
sns.histplot(data, x = 'bill_length_mm', bins= 14)
```

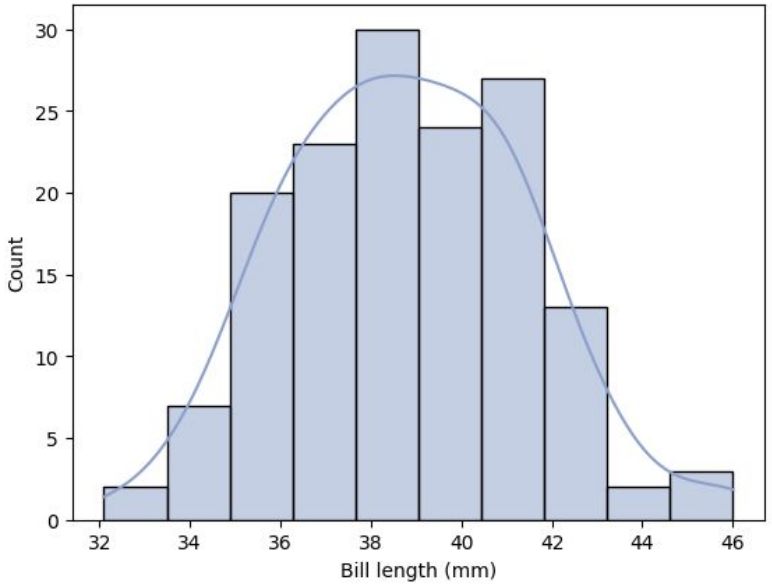
Kernel Density Estimate (KDE)



- A non-parametric way to estimate the probability density function of a continuous variable

```
sns.histplot(data_h, x = 'bill_length_mm', kde=True)
```

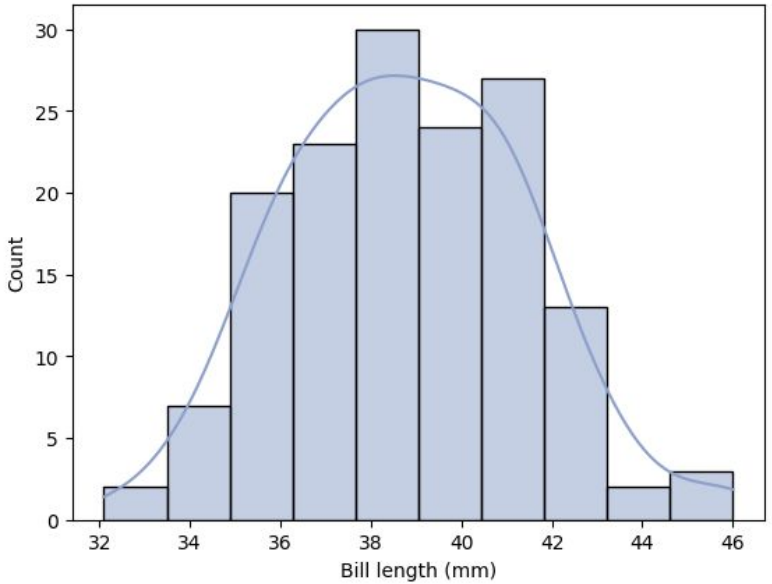
Kernel Density Estimate (KDE)



- A non-parametric way to estimate the probability density function of a continuous variable
- Uses "kernels" (small distributions) placed at each data point, then sums them up

```
sns.histplot(data_h, x = 'bill_length_mm', kde=True)
```

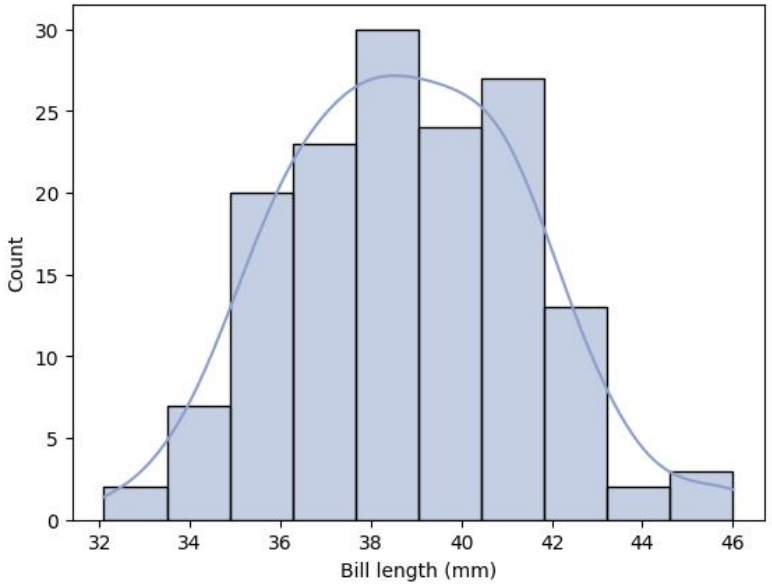
Kernel Density Estimate (KDE)



- A non-parametric way to estimate the probability density function of a continuous variable
- Uses "kernels" (small distributions) placed at each data point, then sums them up
- Smooths out data points into a continuous curve showing where values are concentrated

```
sns.histplot(data_h, x = 'bill_length_mm', kde=True)
```

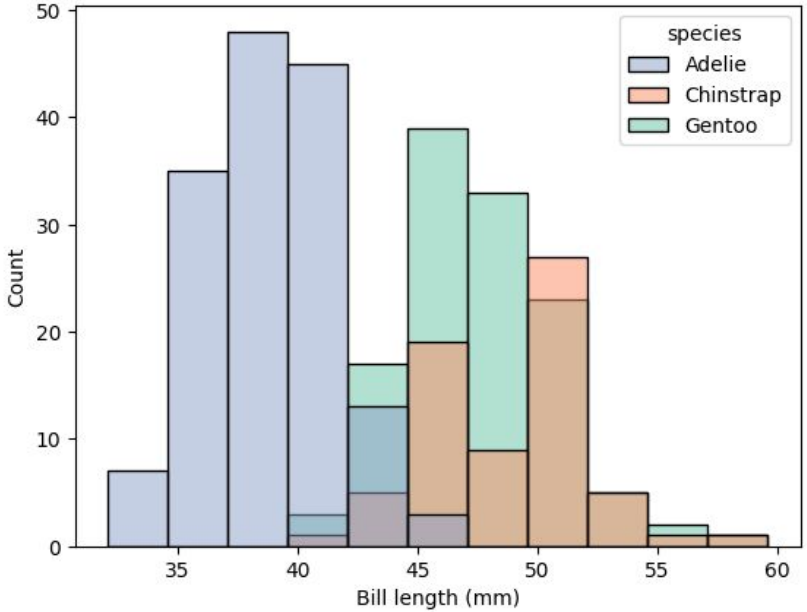
Kernel Density Estimate (KDE)



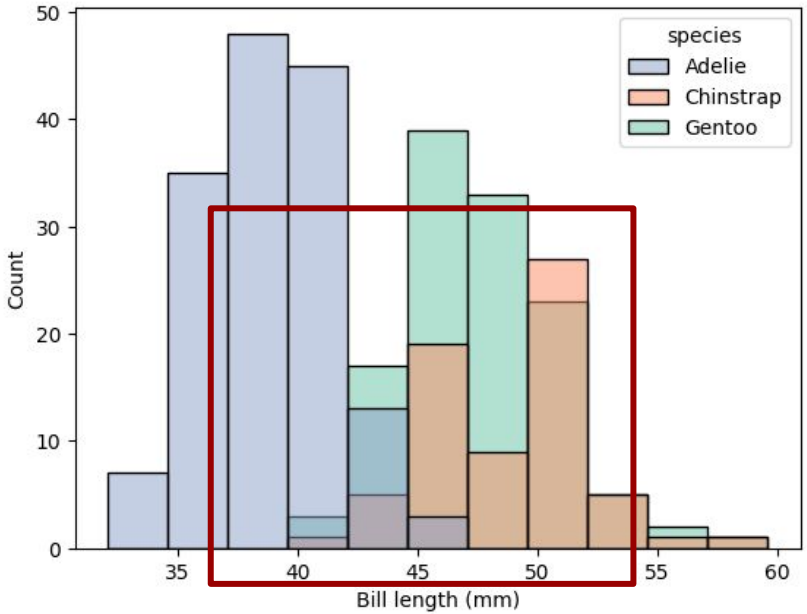

```
sns.histplot(data_h, x = 'bill_length_mm', kde=True)
```

- A non-parametric way to estimate the probability density function of a continuous variable
- Uses "kernels" (small distributions) placed at each data point, then sums them up
- Smooths out data points into a continuous curve showing where values are concentrated
- Histograms can look very different depending on bin width and positioning;
- Shows the true shape of the distribution more naturally

Comparing histograms

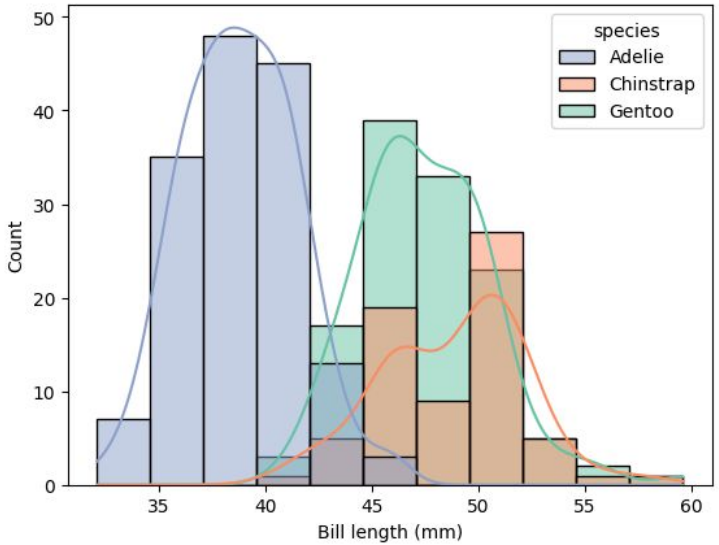


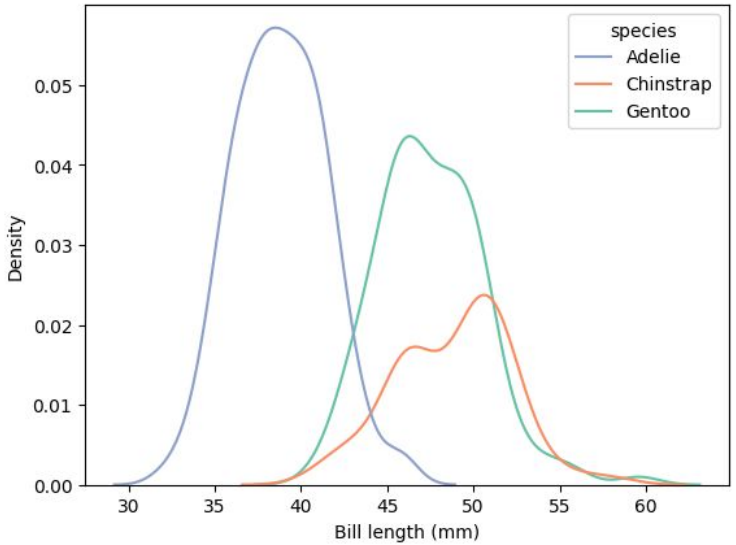
Comparing histograms



- Confusing to compare distributions
- Multiple KDE curves can be overlaid more clearly than multiple histograms

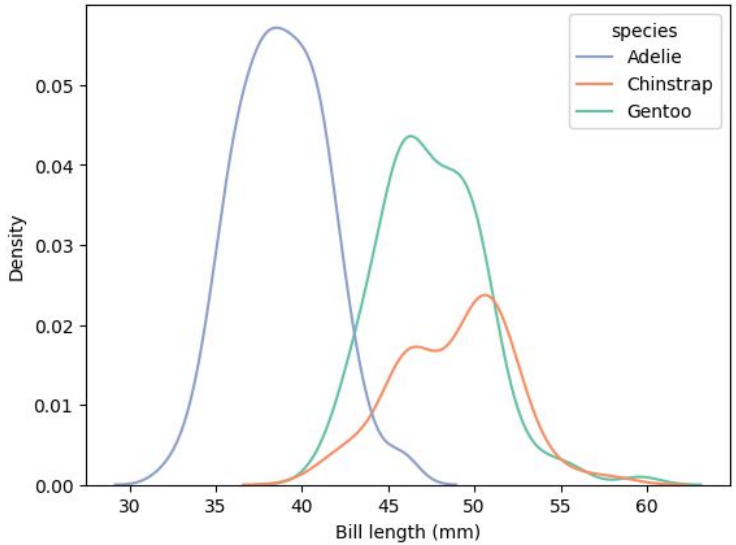
Comparing histograms





➢ KDE leads to a more clear outcome

KDE plot




```
sns.kdeplot(penguins, x = 'bill_length_mm', hue='species')
```

When NOT to use histograms?

- Comparing multiple groups

When NOT to use histograms?

- Comparing multiple groups
- With fewer than ~20-30 data points, histograms become unreliable and misleading

When NOT to use histograms?

- Comparing multiple groups
- With fewer than ~20-30 data points, histograms become unreliable and misleading
- When exact values matter - detecting duplicates/outliers

When NOT to use histograms?

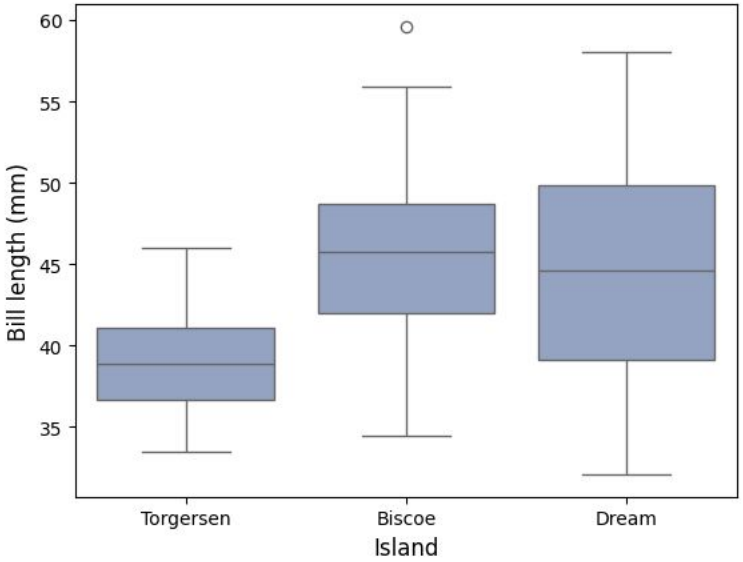
- Comparing multiple groups
- With fewer than ~20-30 data points, histograms become unreliable and misleading
- When exact values matter - detecting duplicates/outliers
- Categorical / Time Series Data

When NOT to use histograms?

- Comparing multiple groups
- With fewer than ~20-30 data points, histograms become unreliable and misleading
- When exact values matter - detecting duplicates/outliers
- Categorical / Time Series Data
- If your conclusions change drastically with different bin sizes, the histogram is unreliable

Box plots

Used to summarize the distribution of a numerical dataset. It displays lots of statistical characteristics in a very intuitive way.



Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}

- Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}

- Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)
- Interquartile range (IQR): represents the spread of the middle 50% of the data IQR = Q3-Q1

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}

- Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)
- Interquartile range (IQR): represents the spread of the middle 50% of the data IQR = Q3-Q1

2. Q2 is the median, Q1= 7, Q3 = 14

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}

- Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)
- Interquartile range (IQR): represents the spread of the middle 50% of the data IQR = Q3-Q1

2. Q2 is the median, Q1= 7, Q3 = 14

3. IQR = 14 - 7 = 7

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}
   - Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)
   - Interquartile range (IQR): represents the spread of the middle 50% of the data IQR = Q3-Q1
2. Q2 is the median, Q1= 7, Q3 = 14
3. IQR = 14 - 7 = 7
4. Lower whisker = min(datapoint, Q1 - 1.5 * IQR)

Boxplot decomposed

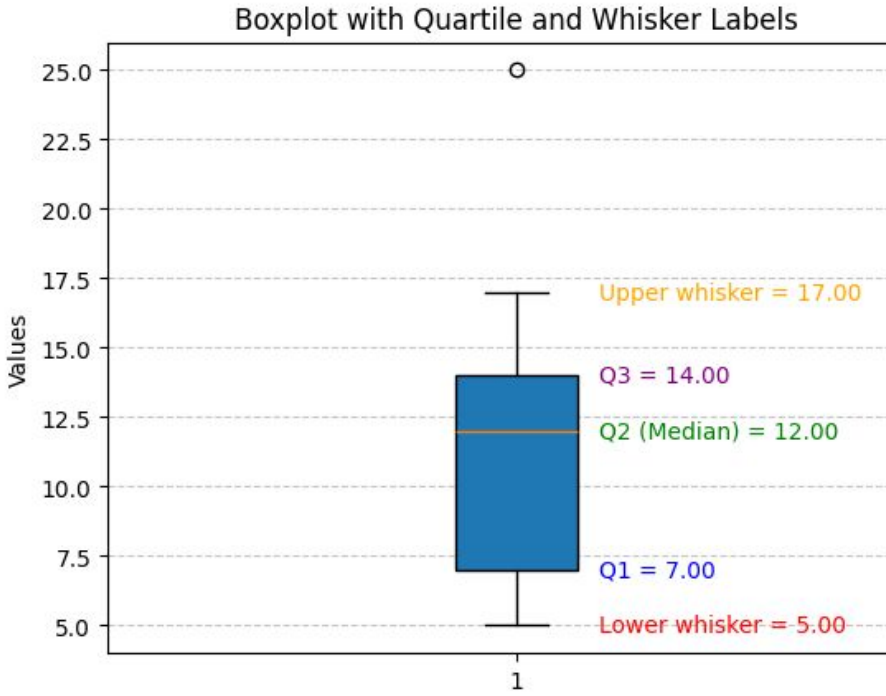
X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 60}
   - Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)
   - Interquartile range (IQR): represents the spread of the middle 50% of the data IQR = Q3-Q1
2. Q2 is the median, Q1= 7, Q3 = 14
3. IQR = 14 - 7 = 7
4. Lower whisker = min(datapoint, Q1 - 1.5 * IQR)
5. Upper whisker = max(datapoint, Q3 + 1.5 * IQR)

Boxplot decomposed

X = {5, 7, 7, 9, 12, 17, 60, 14, 14}

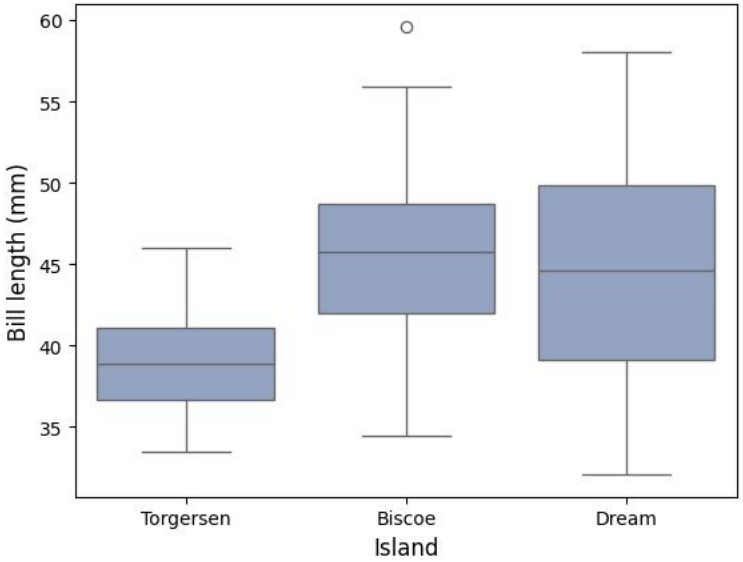
1. X_sort = {5, 7, 7, 9, 12, 14, 14, 17, 25}
   - Median: the middle value in a dataset that has been arranged in ascending or descending order. (median = 12)
   - Interquartile range (IQR): represents the spread of the middle 50% of the data IQR = Q3-Q1
2. Q2 is the median, Q1= 7, Q3 = 14
3. IQR = 14 - 7 = 7
4. Lower whisker = min(datapoint, Q1 - 1.5 * IQR)
5. Upper whisker = max(datapoint, Q3 + 1.5 * IQR)



Box plots

Used to summarize the distribution of a numerical dataset. It displays lots of statistical characteristics in a very intuitive way.

Why to use?

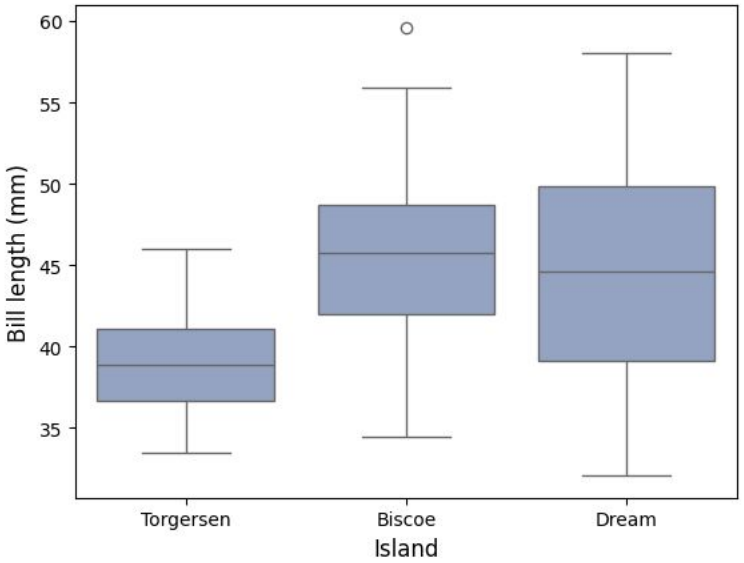


Box plots

Used to summarize the distribution of a numerical dataset. It displays lots of statistical characteristics in a very intuitive way.

Why to use?

- Compare spread, symmetry, and skewness between datasets

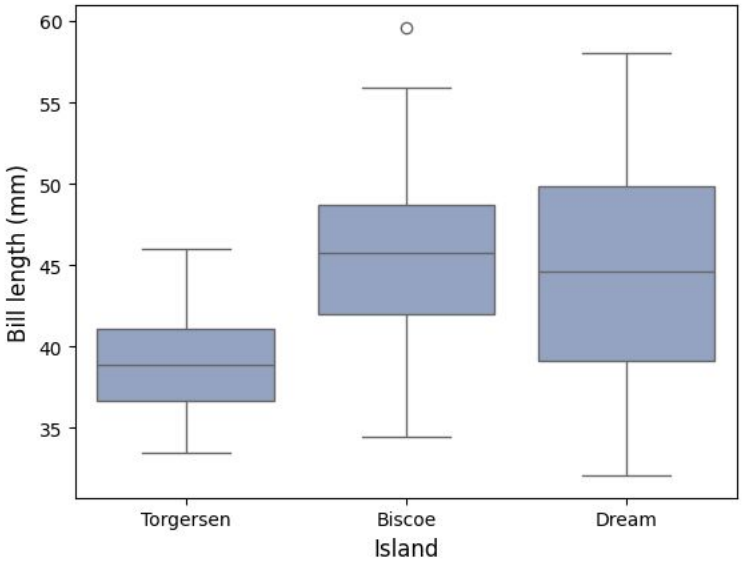


Box plots

Used to summarize the distribution of a numerical dataset. It displays lots of statistical characteristics in a very intuitive way.

Why to use?

- Compare spread, symmetry, and skewness between datasets
- Detect outliers quickly

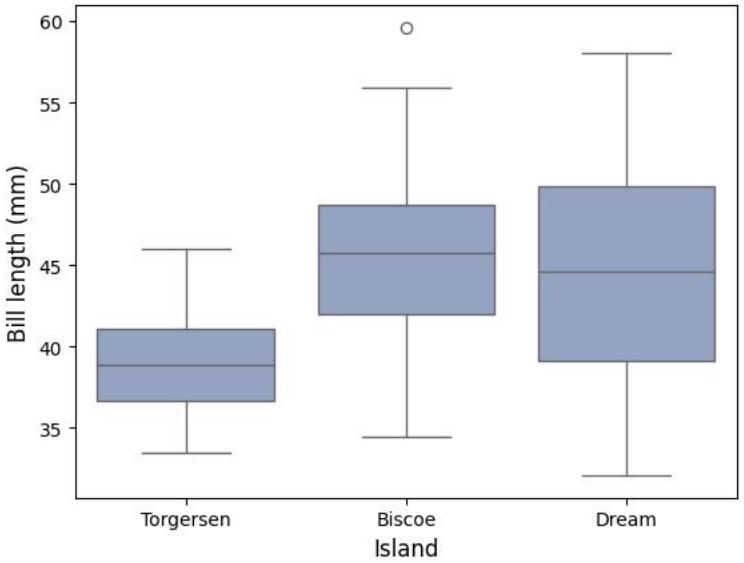


Box plots

Used to summarize the distribution of a numerical dataset. It displays lots of statistical characteristics in a very intuitive way.

Why to use?

- Compare spread, symmetry, and skewness between datasets
- Detect outliers quickly
- Show central tendency and variability at a glance



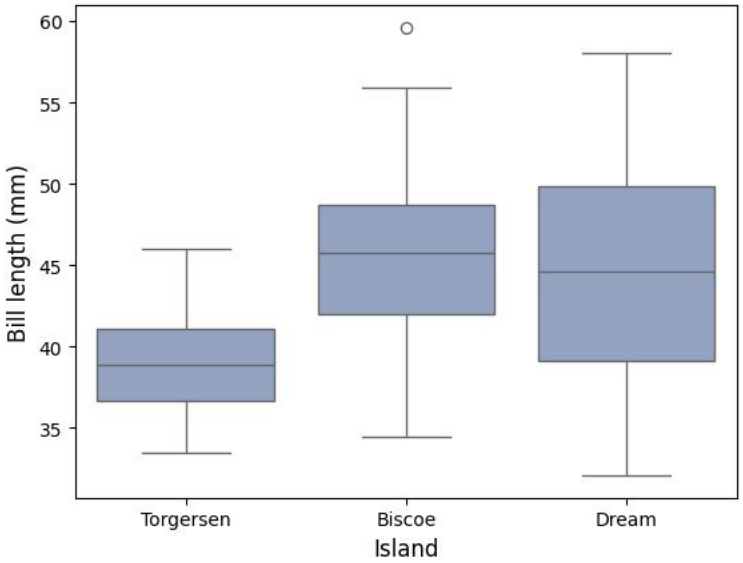
Box plots

Used to summarize the distribution of a numerical dataset. It displays lots of statistical characteristics in a very intuitive way.

Why to use?

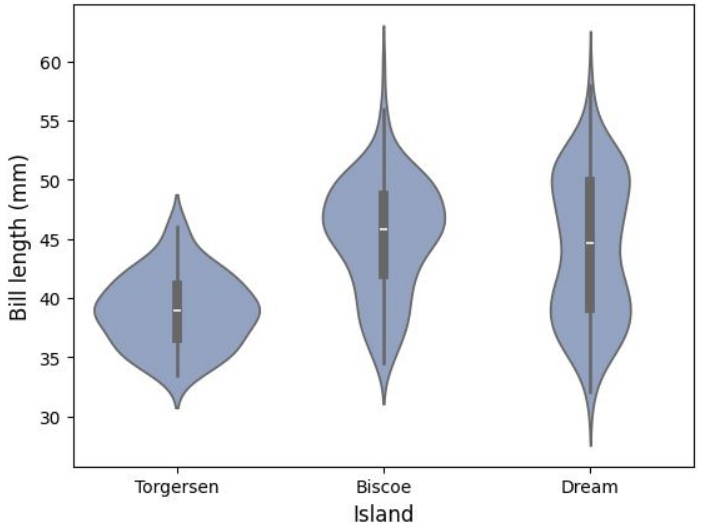
- Compare spread, symmetry, and skewness between datasets
- Detect outliers quickly
- Show central tendency and variability at a glance

```
sns.boxplot(data = penguins, x =  'island', y = 'bill_length_mm' )
```



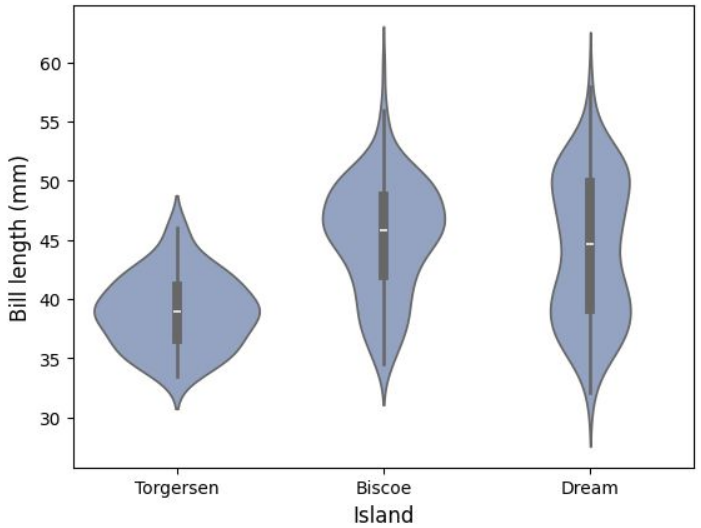
Violin plot

- Combines a box plot with a kernel density plot to display the full distribution shape of your data



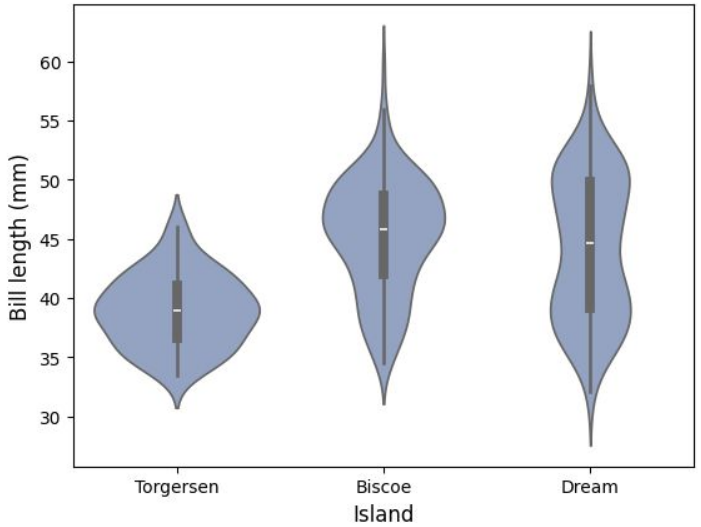
Violin plot

- Combines a box plot with a kernel density plot to display the full distribution shape of your data
- Reveals distribution characteristics that box plots miss - such as bimodal distributions, skewness, and multiple peaks



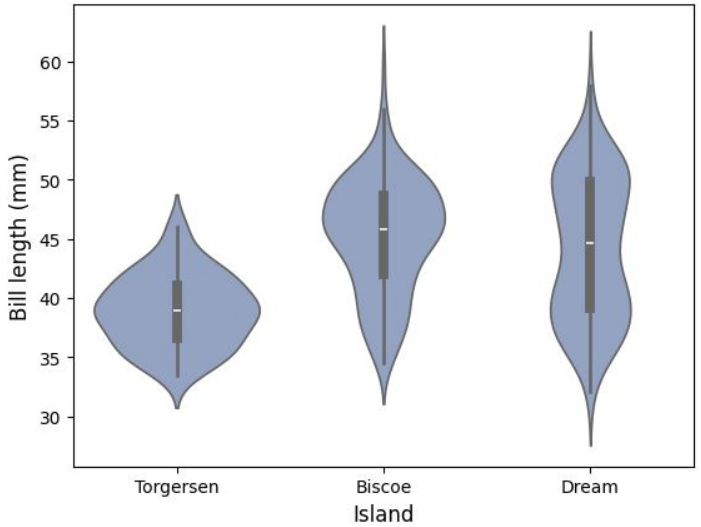
Violin plot

- Combines a box plot with a kernel density plot to display the full distribution shape of your data
- Reveals distribution characteristics that box plots miss - such as bimodal distributions, skewness, and multiple peaks
- Best for comparing distributions across categories when you need to see the complete data shape, not just summary statistics



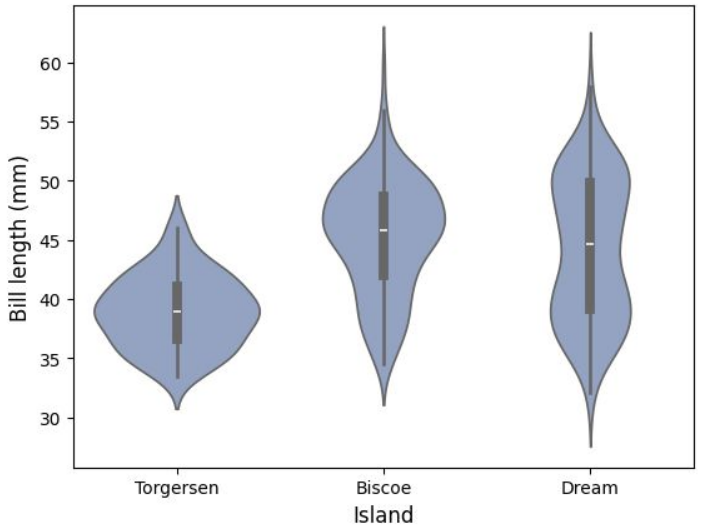
Violin plot

- Combines a box plot with a kernel density plot to display the full distribution shape of your data
- Reveals distribution characteristics that box plots miss - such as bimodal distributions, skewness, and multiple peaks
- Best for comparing distributions across categories when you need to see the complete data shape, not just summary statistics
- Width indicates data density - wider sections show where most data points concentrate



Violin plot

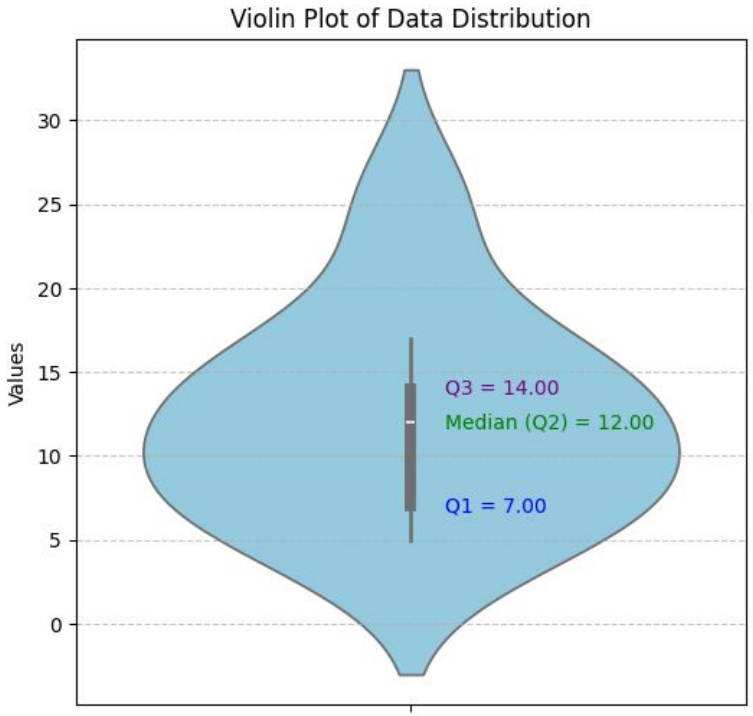
- Combines a box plot with a kernel density plot to display the full distribution shape of your data
- Reveals distribution characteristics that box plots miss - such as bimodal distributions, skewness, and multiple peaks
- Best for comparing distributions across categories when you need to see the complete data shape, not just summary statistics
- Width indicates data density - wider sections show where most data points concentrate



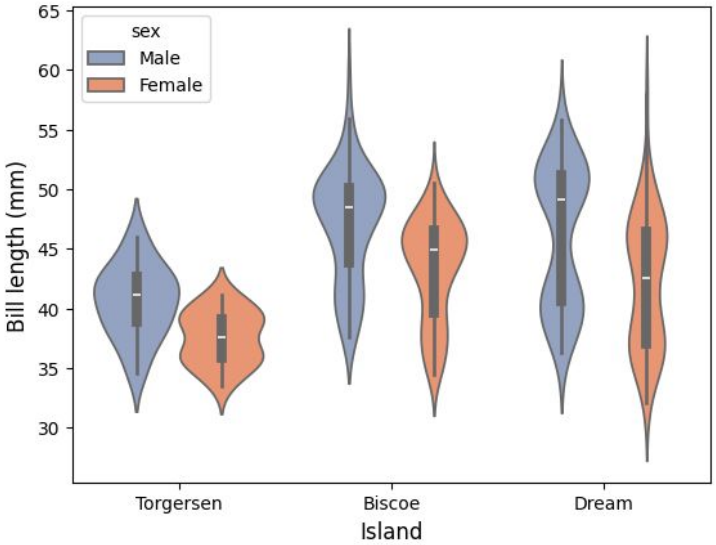

```
sns.violinplot(data = penguins, x = 'island',
y = 'bill_length_mm')
```

Analyzing violin plots

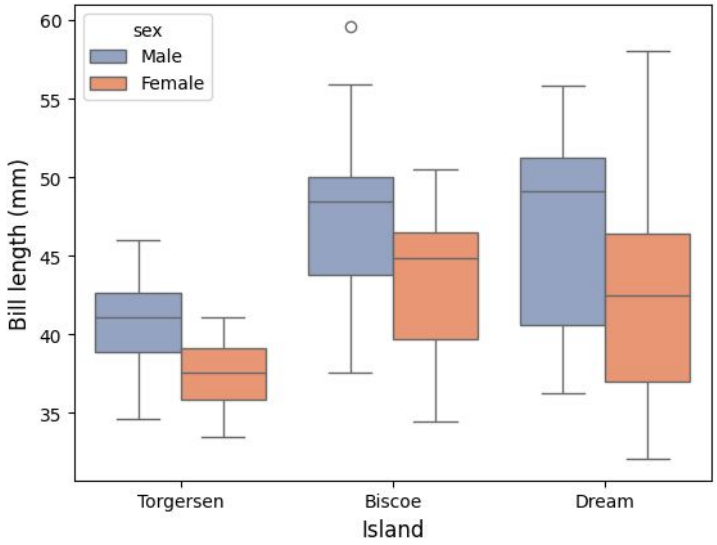
- Shows the same things a boxplot
- Shows the distribution of data



Violin and box plots side-by-side




```
sns.violinplot(data = penguins, x = 'island',
y = 'bill_length_mm', hue='sex')
```




```
sns.boxplot(data = penguins, x = 'island',
y = 'bill_length_mm', hue='sex')
```

Box vs violin plots

Aspect	Boxplot	Violin Plot
Purpose	Summarizes data using five-number summary	Combines boxplot with density to show full distribution

Box vs violin plots

Aspect	Boxplot	Violin Plot
Purpose	Summarizes data using five-number summary	Combines boxplot with density to show full distribution
Distribution Shape	Not shown	Clearly displayed (width = data density)

Box vs violin plots

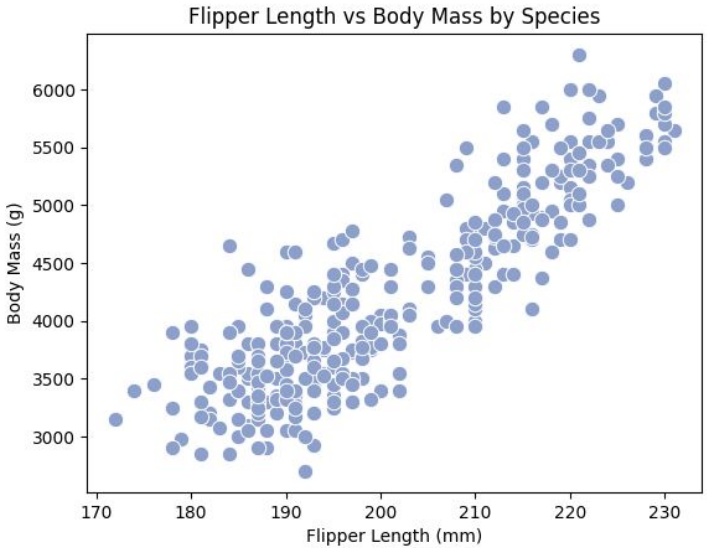
Aspect	Boxplot	Violin Plot
Purpose	Summarizes data using five-number summary	Combines boxplot with density to show full distribution
Distribution Shape	Not shown	Clearly displayed (width = data density)
Interpretation	Simple and easy to read	More detailed but slightly harder to interpret

Box vs violin plots

Aspect	Boxplot	Violin Plot
Purpose	Summarizes data using five-number summary	Combines boxplot with density to show full distribution
Distribution Shape	Not shown	Clearly displayed (width = data density)
Interpretation	Simple and easy to read	More detailed but slightly harder to interpret
Best Use	Quick comparison of medians and spread	Exploring data patterns, skewness, or multimodal distributions

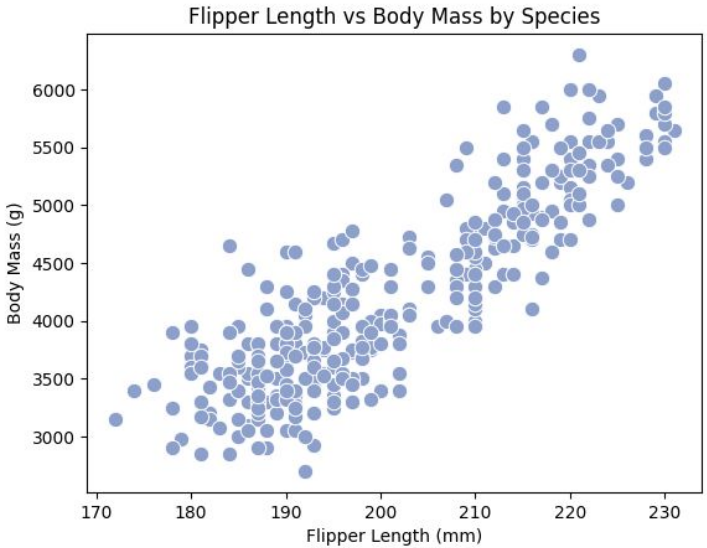
Scatterplot

Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



Scatterplot

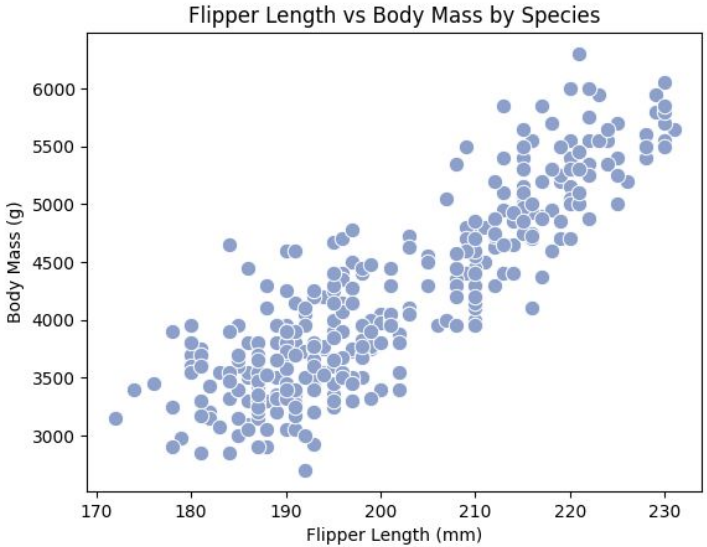
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships

Scatterplot

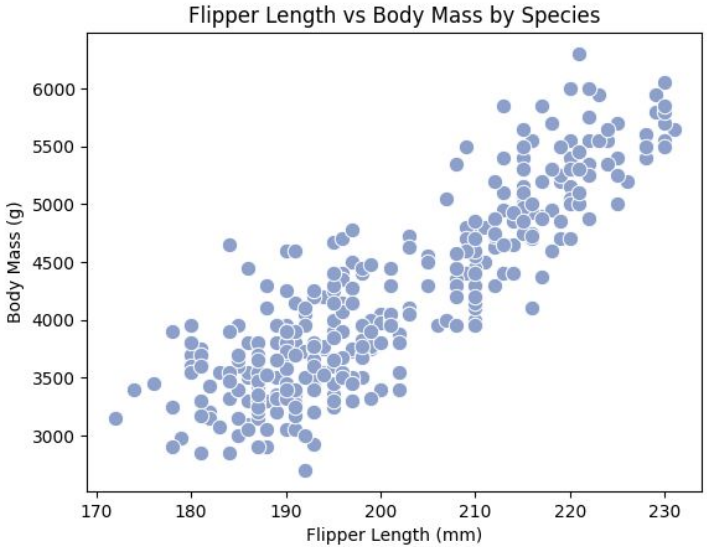
Scatterplot is a visualization that uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression

Scatterplot

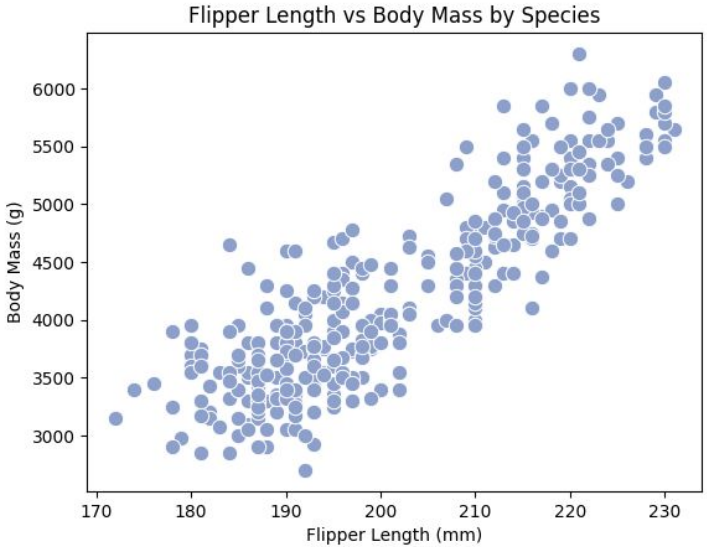
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression
  - Identifying clusters or outliers

Scatterplot

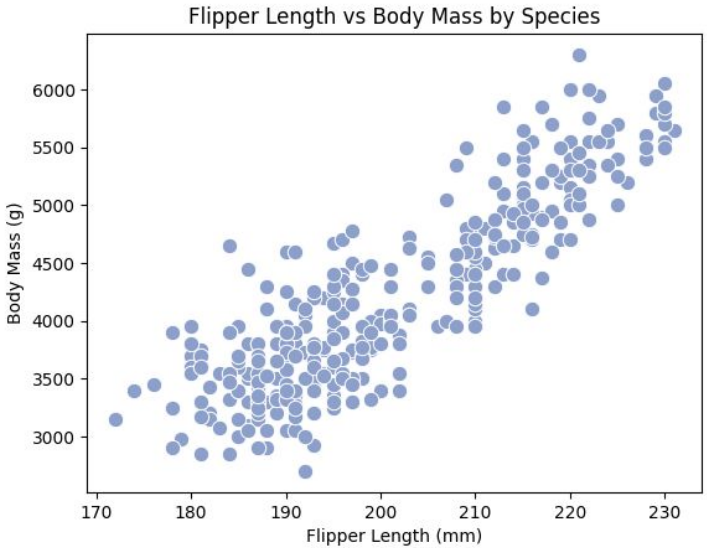
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression
  - Identifying clusters or outliers
  - Comparing groups (using color/shape)

Scatterplot

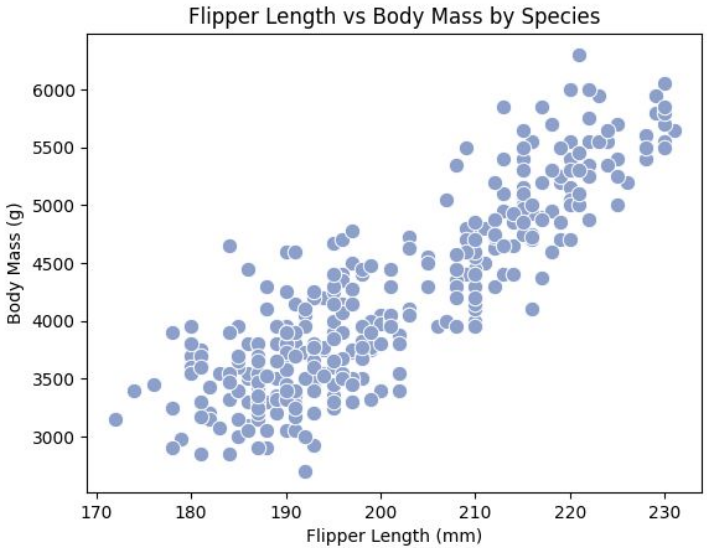
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression
  - Identifying clusters or outliers
  - Comparing groups (using color/shape)
- Best Practices:

Scatterplot

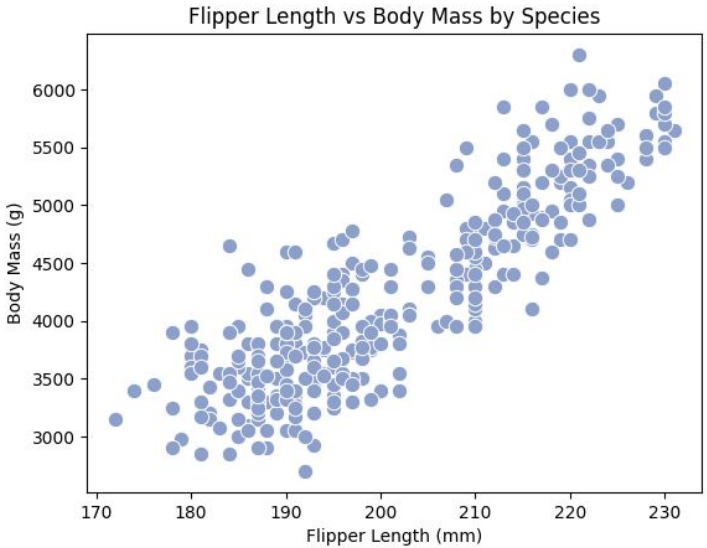
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression
  - Identifying clusters or outliers
  - Comparing groups (using color/shape)
- Best Practices:
  - Label axes clearly with units

Scatterplot

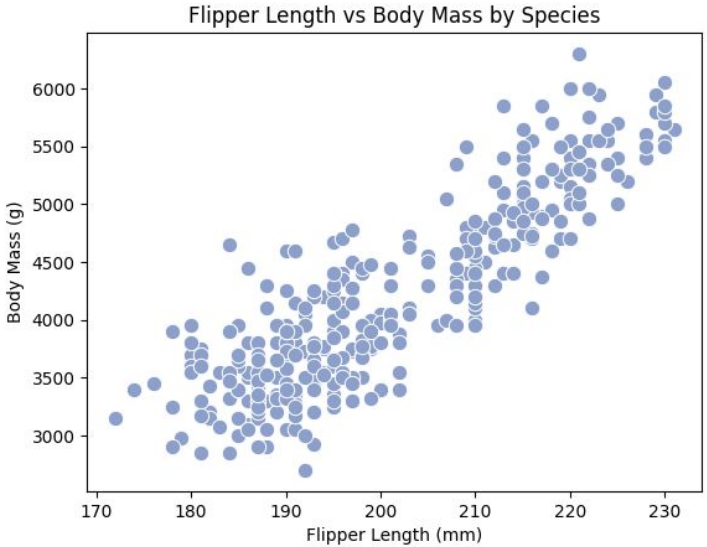
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression
  - Identifying clusters or outliers
  - Comparing groups (using color/shape)
- Best Practices:
  - Label axes clearly with units
  - Use appropriate scale ranges

Scatterplot

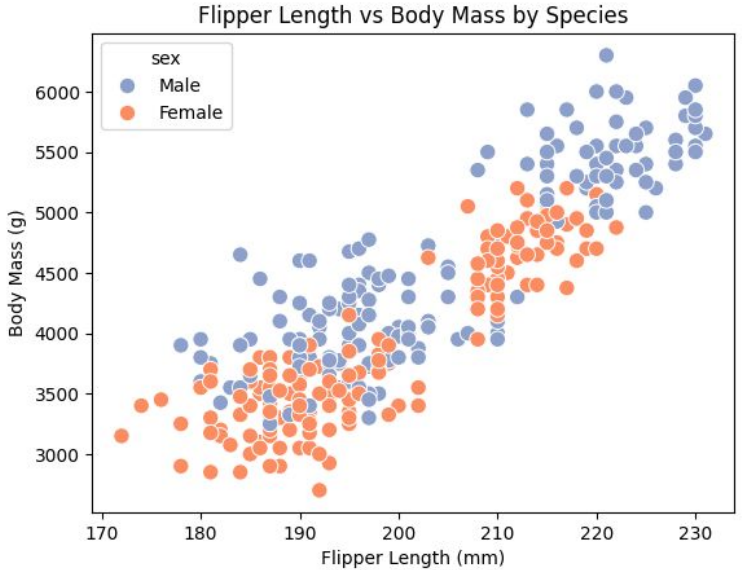
Scatterplot is a visualization uses dots to represent values for two different numeric variables. Scatter plots are used to observe relationships between variables.



- When to Use:
  - Exploring variable relationships
  - Checking for linearity before regression
  - Identifying clusters or outliers
  - Comparing groups (using color/shape)
- Best Practices:
  - Label axes clearly with units
  - Use appropriate scale ranges
  - Add trend lines when relevant

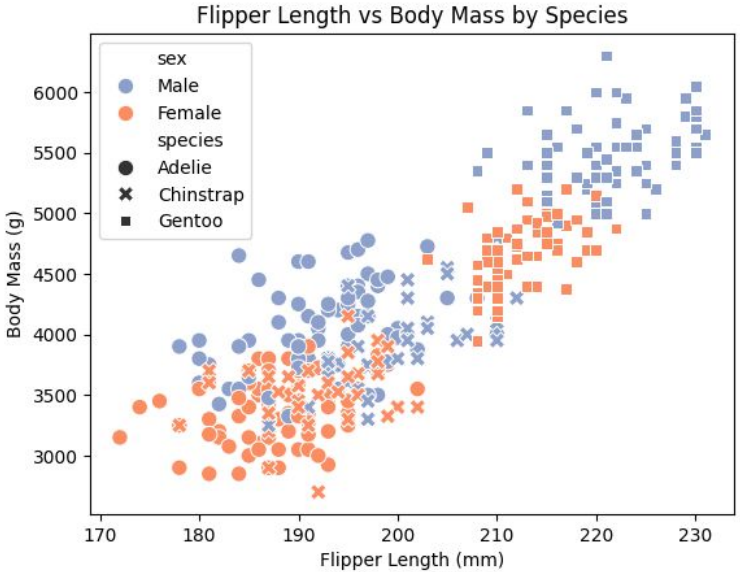
Comparing groups (using color/shape)

- Use color to distinguish between groups



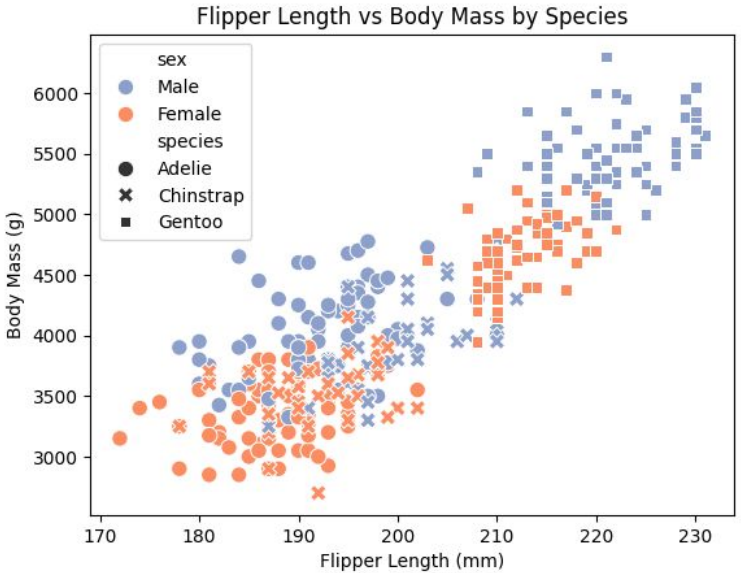
Comparing groups (using color/shape)

- Combine with markers to distinguish between groups



Comparing groups (using color/shape)

- Combine with markers to distinguish between groups




```
sns.scatterplot(
    data=penguins,
    x="flipper_length_mm",
    y="body_mass_g",
    hue="sex",          # color by species
    style="species",       # optional: different marker shapes
    s=80                   # point size
)
```

Basic functions for plotting

plt.xlabel(“ X label”, fontsize = 14) #to add an X, Y label and the title

plt.ylabel(“ Y label”, fontsize = 14)

plt.title(“Title”, fontsize = 14)

plt.show() # to show the figure

plt.savefig(“figure_name.png”, dpi=300) #to save the figure

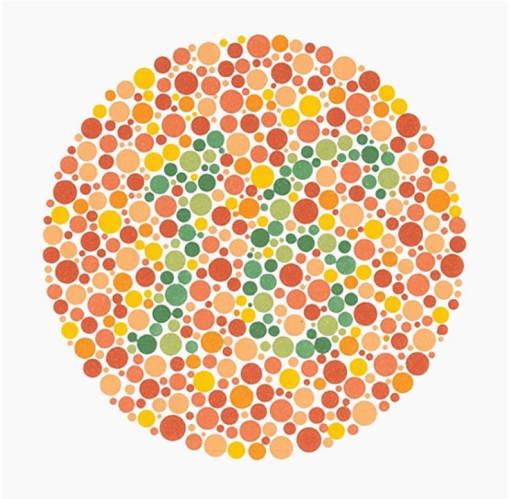
Group exercise

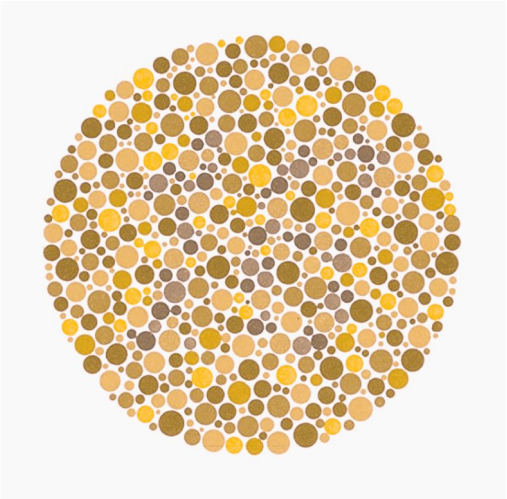
- Join breakout rooms with 3-4 colleagues
- Each room gets an assignment of visualizations
- Implement the assignment in python or R
- Instructors will visit your rooms for help and questions

Color choice

Colorblind

- Color Vision Deficiency (CVD)
  - ~8% of men, ~0.5% of women have some form of CVD (most commonly red–green).
  - Avoid relying solely on red/green or blue/purple distinctions.

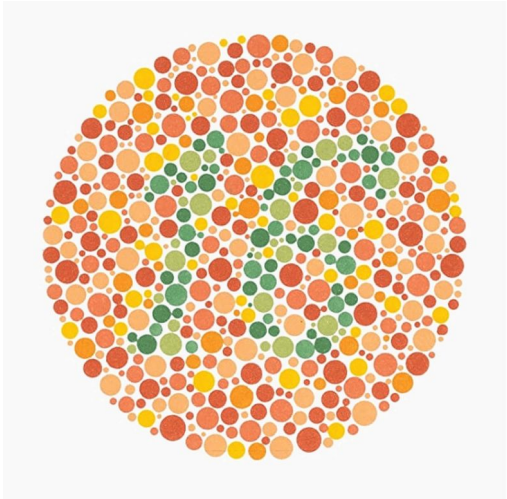


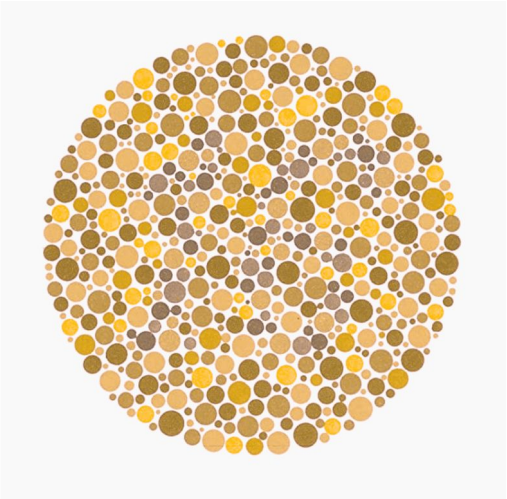


https://www.datawrapper.de/blog/colorblindness-part3

Colorblind

- Color Vision Deficiency (CVD)
  - ~8% of men, ~0.5% of women have some form of CVD (most commonly red–green).
  - Avoid relying solely on red/green or blue/purple distinctions.
- Use tools (e.g., Coblis, Color Oracle) to preview how palettes appear to different users.

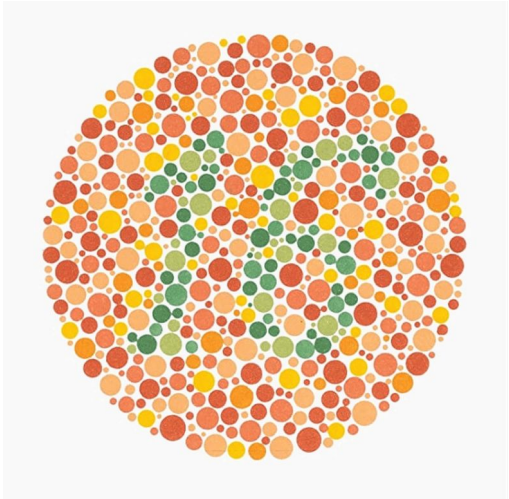


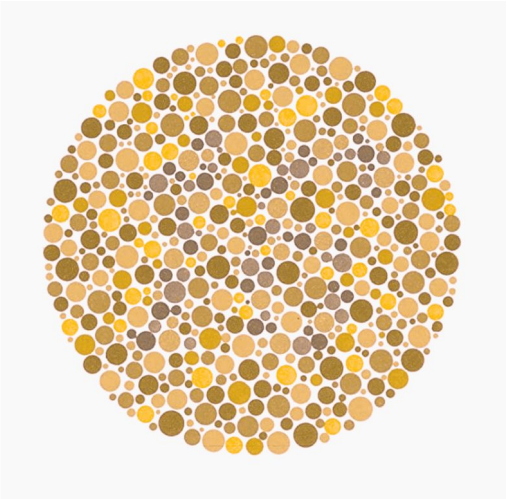


https://www.datawrapper.de/blog/colorblindness-part3

Colorblind

- Color Vision Deficiency (CVD)
  - ~8% of men, ~0.5% of women have some form of CVD (most commonly red–green).
  - Avoid relying solely on red/green or blue/purple distinctions.
- Use tools (e.g., Coblis, Color Oracle) to preview how palettes appear to different users.
- Redundancy

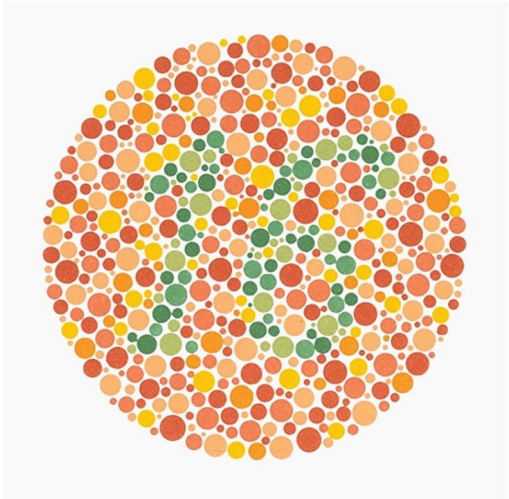


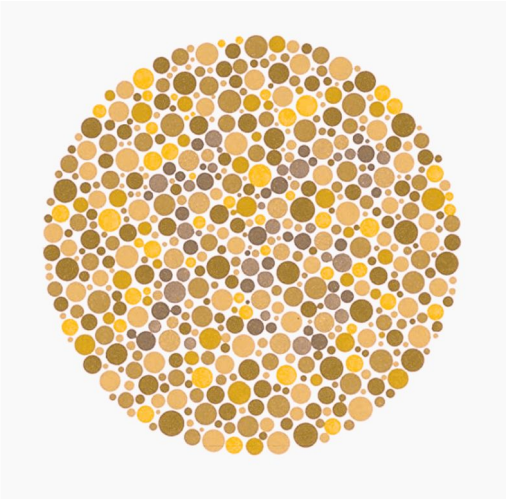


https://www.datawrapper.de/blog/colorblindness-part3

Colorblind

- Color Vision Deficiency (CVD)
  - ~8% of men, ~0.5% of women have some form of CVD (most commonly red–green).
  - Avoid relying solely on red/green or blue/purple distinctions.
- Use tools (e.g., Coblis, Color Oracle) to preview how palettes appear to different users.
- Redundancy
- Pair color with other encodings: shape, line style, texture, labels, e.g., use both color + marker type in scatterplots.





https://www.datawrapper.de/blog/colorblindness-part3

Contrast

Contrast is the difference in visual properties (like color, size, shape, or brightness) that makes elements distinguish from one another.

https://academy.datawrapper.de/article/140-what-to-consider-when-choosing-colors-for-data-visualization

Contrast

Contrast is the difference in visual properties (like color, size, shape, or brightness) that makes elements distinugish from one another.

- If the colors are too similar in brightness, people may struggle to tell them apart, especially in low lighting, poor projectors, or for viewers with color vision deficiencies.

https://academy.datawrapper.de/article/140-what-to-consider-when-choosing-colors-for-data-visualization

Contrast

Contrast is the difference in visual properties (like color, size, shape, or brightness) that makes elements distinugish from one another.

- If the colors are too similar in brightness, people may struggle to tell them apart, especially in low lighting, poor projectors, or for viewers with color vision deficiencies.
- Contrast Ratio: A measure of how different two colors appear in terms of brightness and visibility.
    - A ratio of 3:1 means that one color is at least three times as bright or dark relative to the other in a way the human eye can easily separate.

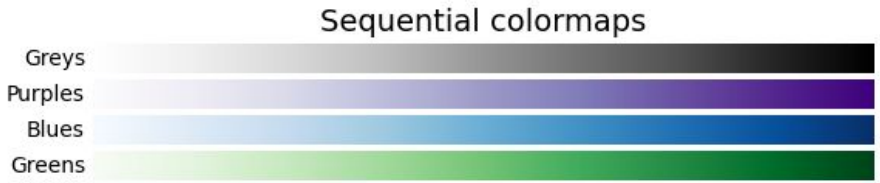
https://academy.datawrapper.de/article/140-what-to-consider-when-choosing-colors-for-data-visualization

Picking the right palettes

https://seaborn.pydata.org/tutorial/color_palettes.html

Picking the right palettes

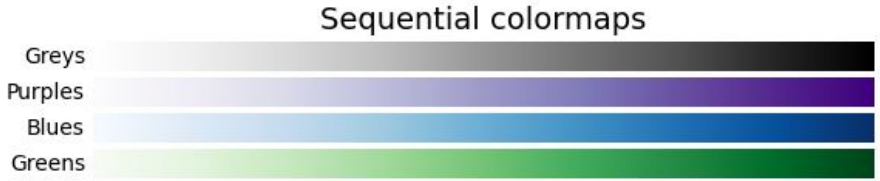
1. Sequential: change in lightness and often saturation of color incrementally, often using a single hue; should be used for representing information that has ordering.



https://seaborn.pydata.org/tutorial/color_palettes.html

Picking the right palettes

1. Sequential: change in lightness and often saturation of color incrementally, often using a single hue; should be used for representing information that has ordering.

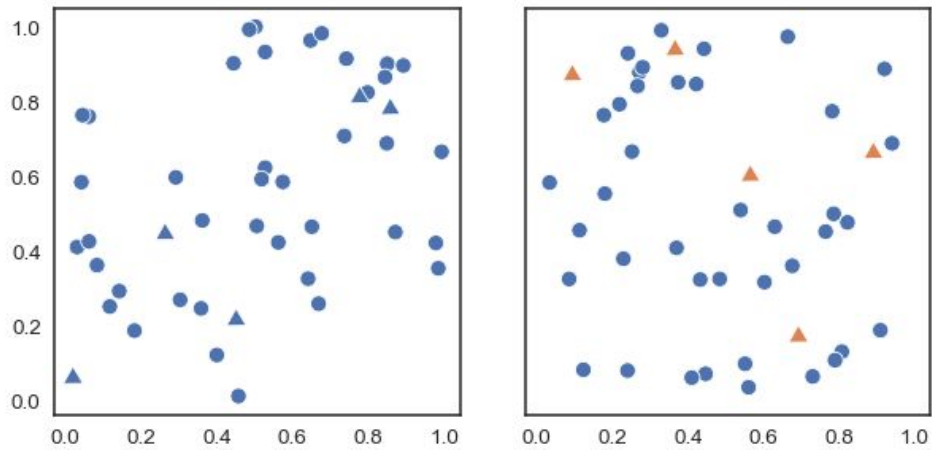


2. Diverging: change in lightness and possibly saturation of two different colors that meet in the middle at an unsaturated color; should be used when the information being plotted has a critical middle value, such as when the data deviates around zero.

https://seaborn.pydata.org/tutorial/color_palettes.html

Hue

- Used to visualize categorical data
- Easy distinction among different categories

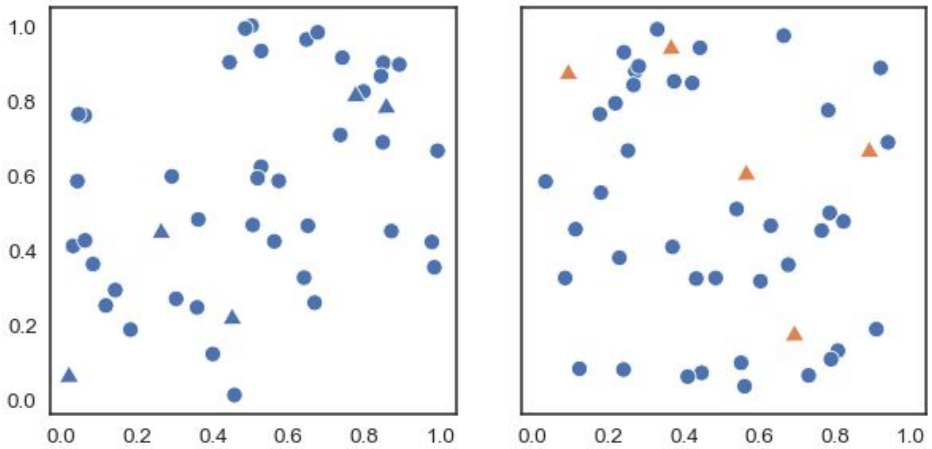


- Avoid too many hues; ~6–8 is usually the upper limit for clear perception.
- Be careful! Make colorblind-safe choices

https://seaborn.pydata.org/tutorial/color_palettes.html

Hue

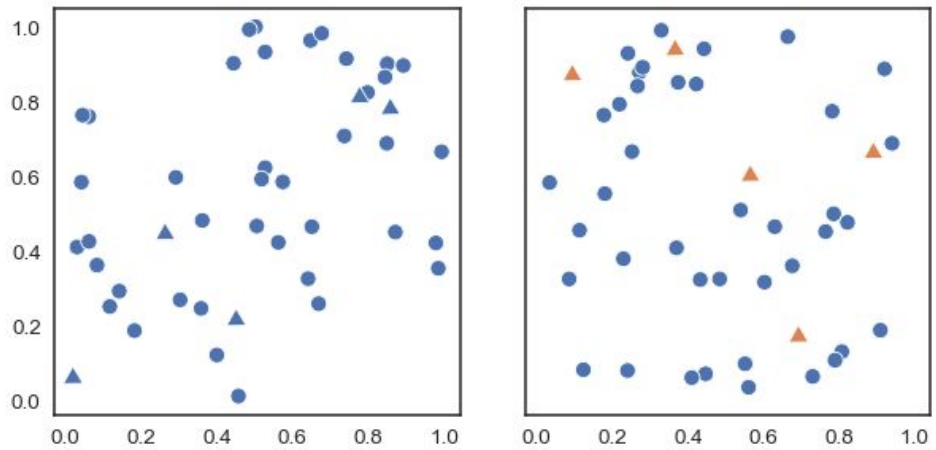
- Used to visualize categorical data



https://seaborn.pydata.org/tutorial/color_palettes.html

Hue

- Used to visualize categorical data
- Easy distinction among different categories

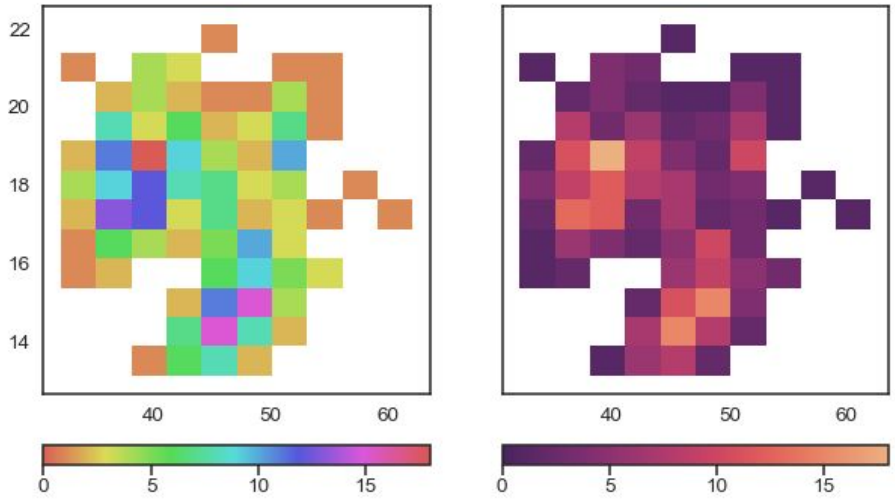


- Avoid too many hues; ~6–8 is usually the upper limit for clear perception.
- Be careful! Make colorblind-safe choices

https://seaborn.pydata.org/tutorial/color_palettes.html

Luminance

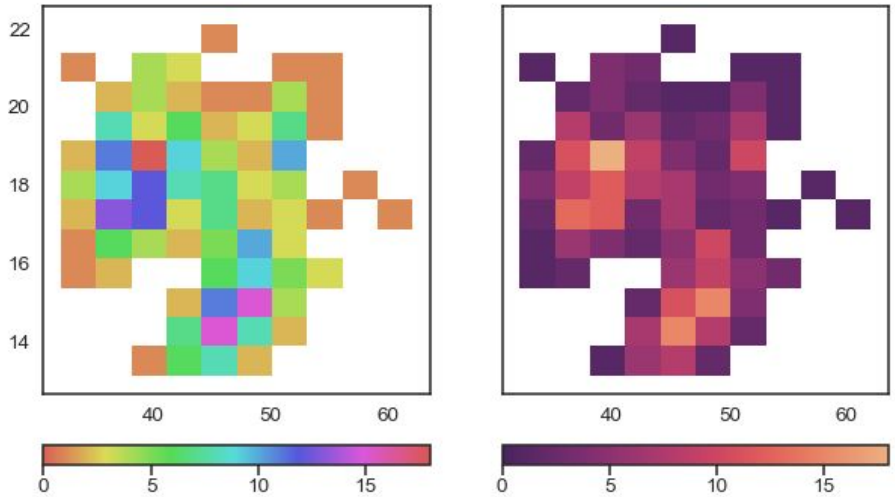
- Luminance is used to represent numeric data



https://seaborn.pydata.org/tutorial/color_palettes.html

Luminance

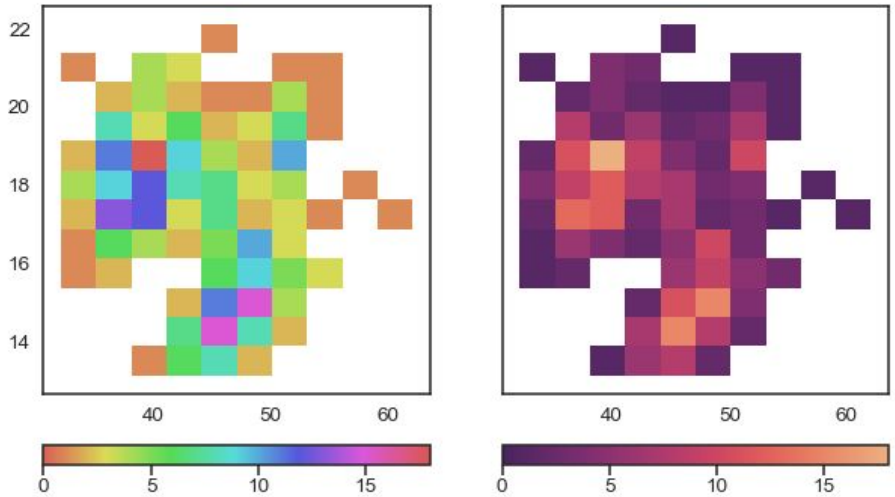
- Luminance is used to represent numeric data
- Darker/lighter values match intuitive expectations (e.g., darker = less, lighter = more).



https://seaborn.pydata.org/tutorial/color_palettes.html

Luminance

- Luminance is used to represent numeric data
- Darker/lighter values match intuitive expectations (e.g., darker = less, lighter = more).



- Watch for poor visibility in very light or dark colors, especially for projectors

https://seaborn.pydata.org/tutorial/color_palettes.html

Saturation

It refers to the purity and intensity of a color

High saturation:

- Colors are vibrant, strong, and intense.

Saturation

It refers to the purity and intensity of a color

High saturation:

- Colors are vibrant, strong, and intense.
- To make specific data points or categories stand out clearly from the background
  - create a strong contrast between different colors.
  - eye-catching or convey a sense of energy.

Saturation

It refers to the purity and intensity of a color

Low saturation:

- In complex figures with many elements, high saturation would make it hard to differentiate data.

Saturation

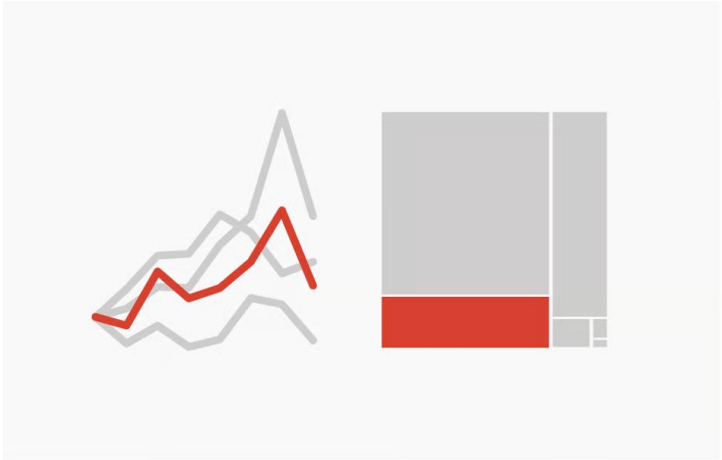
It refers to the purity and intensity of a color

Low saturation:

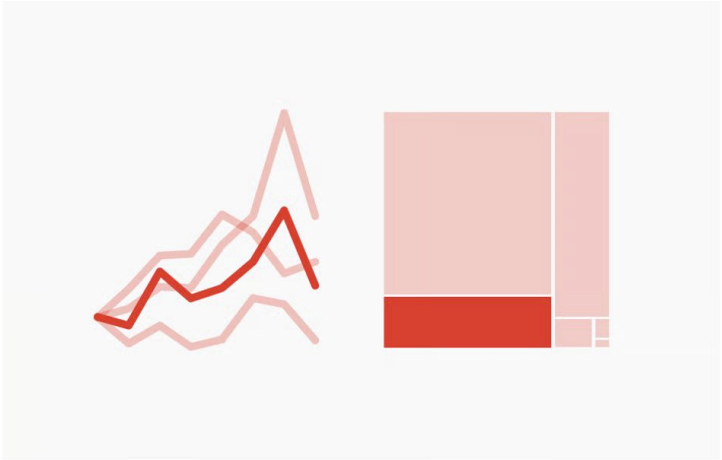
- In complex figures with many elements, high saturation would make it hard to differentiate data.
- Achieves a more subtle or sober aesthetic reducing visual noise and improving the clarity of essential information.

Saturation

It refers to the purity and intensity of a color



DEEMPHASIZE WITH GRAY



DEEMPHASIZE WITH A LESS SATURATED HIGHLIGHT COLOR

Take-home messages

- Load, always inspect data and preprocess data

Take-home messages

- Load, always inspect data and preprocess data
- There is a huge variety of data
  - choose according to your goals and variable types

Take-home messages

- Load, always inspect data and preprocess data
- There is a huge variety of data
  - choose according to your goals and variable types
- Make clear figures

Take-home messages

- Load, always inspect data and preprocess data
- There is a huge variety of data
  - choose according to your goals and variable types
- Make clear figures
- Take into account colorblindness, use color to make your figures more clear

Take-home messages

- Load, always inspect data and preprocess data
- There is a huge variety of data
  - choose according to your goals and variable types
- Make clear figures
- Take into account colorblindness, use color to make your figures more clear
- Be critical with your figures!

Plan for tomorrow

- PCA
- Heatmap

Thank you all for your attention!

- Load, always inspect data and preprocess data
- There is a huge variety of data
  - choose according to your goals and variable types
- Make clear figures
- Take into account colorblindness, use color to make your figures more clear
- Be critical with your figures!