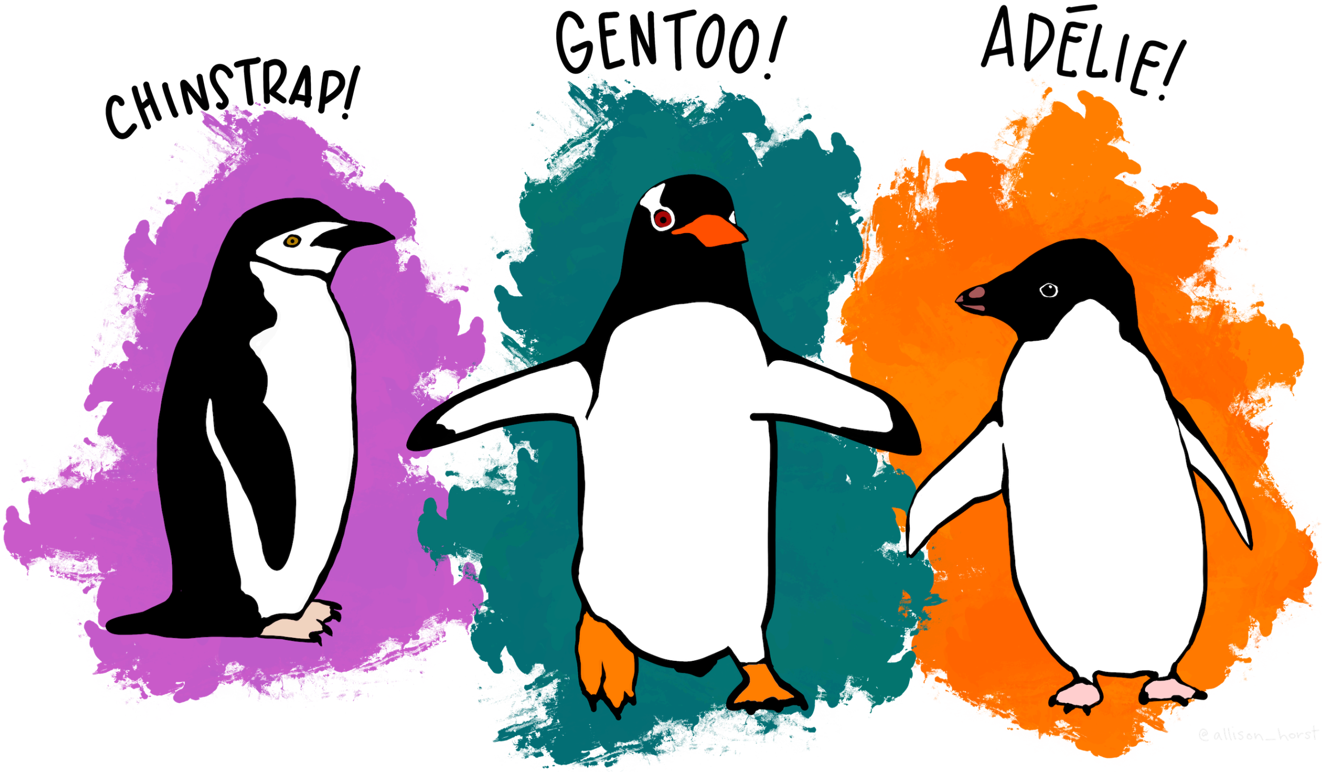
Sahana.D 2/7/2025 Clustering Antartic Penguin Species



source: @allison_horst https://github.com/allisonhorst/penguins

You have been asked to support a team of researchers who have been collecting data about penguins in Antartica! The data is available in csv-Format as `penguins.csv`

Origin of this data : Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.

The dataset consists of 5 columns.

Column	Description
culmen_length_mm	culmen length (mm)
culmen_depth_mm	culmen depth (mm)
flipper_length_mm	flipper length (mm)
body_mass_g	body mass (g)
sex	penguin sex

Unfortunately, they have not been able to record the species of penguin, but they know that there are **at least three** species that are native to the region: **Adelie**, **Chinstrap**, and **Gentoo**. Your task is to apply your data science skills to help them identify groups in the dataset!

Instruction:

Utilize your unsupervised learning skills to clusters in the penguins dataset!

Perform a cluster analysis based on a reasonable number of clusters and collect the average values for the clusters. The output should be a DataFrame named stat_penguins with one row per cluster that shows the mean of the original variables (or columns in "penguins.csv") by cluster. stat_penguins should not include any non-numeric columns.

Observation:

K-means clustering is a way to automatically group data points into clusters26. Imagine you have a bunch of scattered dots, and you want to find groups of dots that are close together. K-means helps you do this7.

The goal is to divide the penguin data into K number of groups (clusters)1. Each penguin will be assigned to the cluster with the nearest mean, which acts as the prototype of the cluster1.

```
# Import Required Packages
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Loading and examining the dataset
penguins_df = pd.read_csv("penguins.csv")
penguins_df.head()
```

index	culmen_length_mm	culmen_depth_mm	flipper_length_
0	39.1	18.7	
1	39.5	17.4	
2	40.3	18	
3	36.7	19.3	
4	39.3	20.6	

Rows: 5

```
#check datatypes
penguins_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 332 entries, 0 to 331
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   culmen_length_mm   332 non-null    float64
 1   culmen_depth_mm    332 non-null    float64
 2   flipper_length_mm  332 non-null    float64
 3   body_mass_g        332 non-null    float64
 4   sex                332 non-null    object
dtypes: float64(4), object(1)
memory usage: 13.1+ KB
```

Observation: 4 columns are float and one is an object

```
penguins_df.isnull().sum() #check the missing values
```

index	
culmen_length_mm	0
culmen_depth_mm	0
flipper_length_mm	0
body_mass_g	0
sex	0

Rows: 5

```
penguins_df.describe().T #stats summary
```

index		mean	std					
culmen_length_mm	332	44.0210843373	5.4524620702	32.1	39.5	44.7	48.625	59.6
culmen_depth_mm	332	17.1530120482	1.9602754199	13.1	15.6	17.3	18.7	21.5
flipper_length_mm	332	200.9759036145	14.0359709694	172	190	197	213	231
body_mass_g	332	4206.4759036145	806.3612775869	2700	3550	4025	4781.25	6300

Rows: 4

```
#Convert categorical variables into dummy/indicator variables
# dtype='int' ensure the output will be 0/1 instead of True/False
penguins_df = pd.get_dummies(penguins_df, dtype='int')
```

```
penguins_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 332 entries, 0 to 331
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   culmen_length_mm   332 non-null    float64
 1   culmen_depth_mm    332 non-null    float64
 2   flipper_length_mm  332 non-null    float64
 3   body_mass_g        332 non-null    float64
 4   sex_FEMALE         332 non-null    int64
 5   sex_MALE           332 non-null    int64
dtypes: float64(4), int64(2)
memory usage: 15.7 KB
```

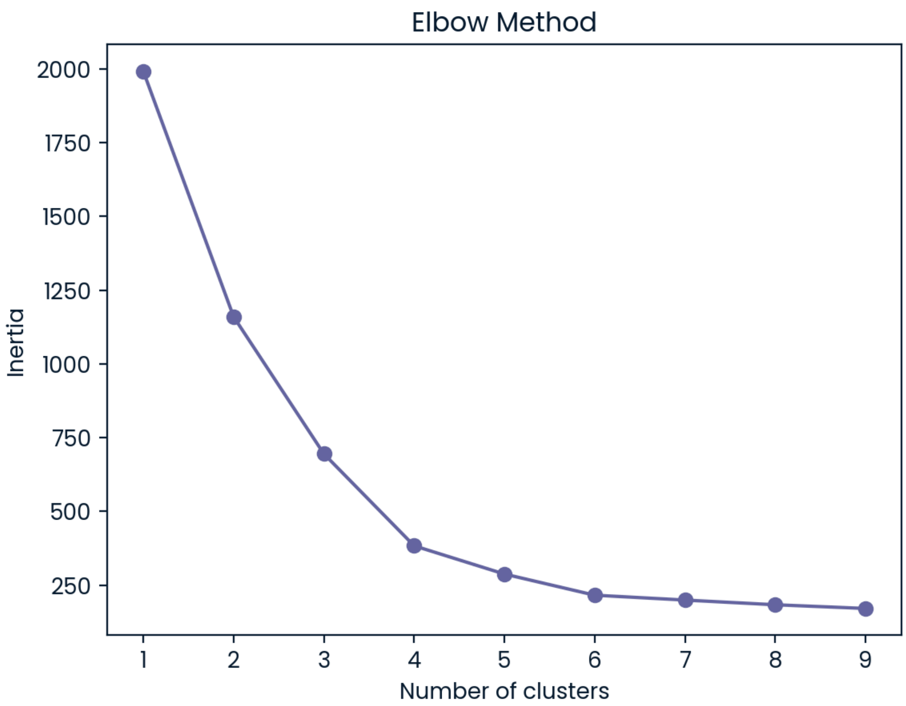
```
#Standardize the features
```

```python
# Scaling variables (also called standardizing) because this can increase the performance

scaler = StandardScaler()
penguins_scaled = scaler.fit_transform(penguins_df)

penguins_pprosd = pd.DataFrame(data=penguins_scaled, columns=penguins_df.columns)

penguins_pprosd.head()
```

	culmen_len...	culmen_d...	flipper_length...	body_...	sex_F...	sex_M...
0	-0.9039058557	0.7903598717	-1.4253417867	-0.5669480132	-0.9939939397	0.99399393
1	-0.8304337676	0.1261867396	-1.0685765023	-0.504847472	1.0060423511	-1.00604235
2	-0.6834895912	0.4327281852	-0.4263989905	-1.1879534252	1.0060423511	-1.00604235
3	-1.3447383847	1.0969013172	-0.5691051042	-0.9395512604	1.0060423511	-1.00604235
4	-0.8671698116	1.7610744492	-0.7831642748	-0.6911490956	-0.9939939397	0.99399393

Rows: 5

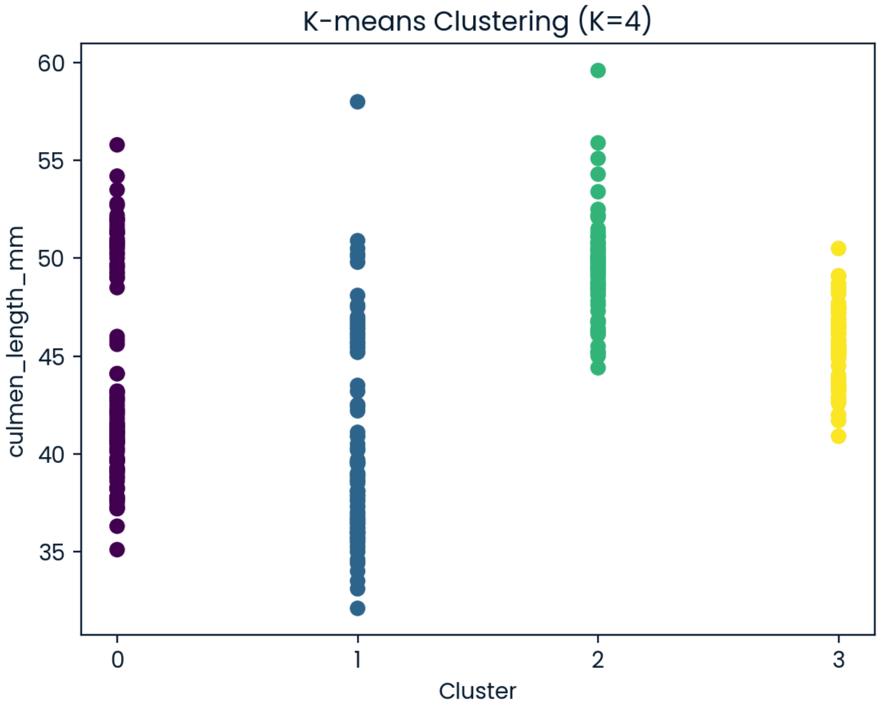
```python
#Detect the optimal number of clusters for k-means clustering
inertia = []
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=42).fit(penguins_pprosd)
    inertia.append(kmeans.inertia_)
plt.plot(range(1, 10), inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
n_clusters=4
```




```
# Perform K-means clustering with optimal k

kmeans = KMeans(n_clusters=n_clusters, random_state=42).fit(penguins_pprosd)
penguins_df['label'] = kmeans.labels_
```

```
#visualize the clusters (here for the 'culmen_length_mm' column)
plt.scatter(penguins_df['label'], penguins_df['culmen_length_mm'], c=kmeans.labels_, cmap='viridis')
```

```
plt.xlabel('Cluster')
plt.ylabel('culmen_length_mm')
plt.xticks(range(int(penguins_df['label'].min()), int(penguins_df['label'].max()) + 1))
plt.title(f'K-means Clustering (K={n_clusters})')
plt.show()
```




```
#create final `stat_penguins` DataFrame
numeric_columns = ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm','label']
stat_penguins = penguins_df[numeric_columns].groupby('label').mean()
```

stat_penguins

	culmen_len...	culmen_d...	flipper_length...
0	43.8783018868	19.1113207547	194.7641509434
1	40.2177570093	17.6112149533	189.046728972
2	49.4737704918	15.7180327869	221.5409836066