HW 03 Sep 26

September 27, 2024

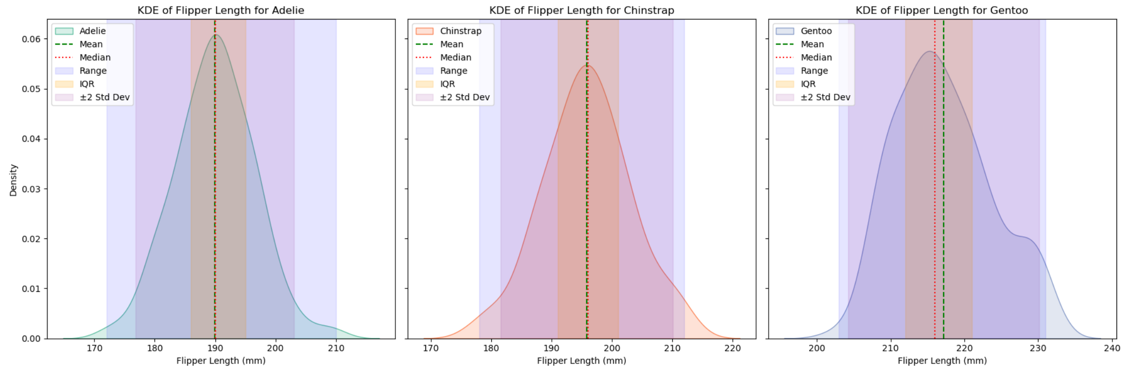
[1]:

```python
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import numpy as np

# Load the penguins dataset from seaborn
penguins = sns.load_dataset('penguins')

# Drop rows with missing values for simplicity
penguins = penguins.dropna(subset=['flipper_length_mm'])

# Create a figure with subplots for each species
fig = go.Figure()

species_list = penguins['species'].unique()

for species in species_list:
    # Filter the data for each species
    species_data = penguins[penguins['species'] == species]['flipper_length_mm']

    # Calculate statistics
    mean_fl = species_data.mean()
    median_fl = species_data.median()
    std_fl = species_data.std()
    min_fl = species_data.min()
    max_fl = species_data.max()
    q1_fl = species_data.quantile(0.25)
    q3_fl = species_data.quantile(0.75)

    # Create histogram for this species
    fig.add_trace(go.Histogram(x=species_data, name=species, opacity=0.6,␣
↪nbinsx=30))

    # Add mean and median lines (using hline for horizontal axis)
    fig.add_vline(x=mean_fl, line=dict(color='green', dash='dash'),␣
↪annotation_text=f'Mean ({species})', annotation_position="top right")
```

```
    fig.add_vline(x=median_fl, line=dict(color='red', dash='dot'),␣
↪annotation_text=f'Median ({species})', annotation_position="bottom right")

    # Add range using vrect (min-max)
    fig.add_vrect(x0=min_fl, x1=max_fl, line_width=0, fillcolor='blue',␣
↪opacity=0.1, annotation_text=f'Range ({species})')

    # Add interquartile range using vrect (IQR: Q1-Q3)
    fig.add_vrect(x0=q1_fl, x1=q3_fl, line_width=0, fillcolor='orange',␣
↪opacity=0.2, annotation_text=f'IQR ({species})')

    # Add two standard deviations range using vrect (mean ± 2*std)
    fig.add_vrect(x0=mean_fl - 2*std_fl, x1=mean_fl + 2*std_fl, line_width=0,␣
↪fillcolor='purple', opacity=0.1, annotation_text=f'±2 Std Dev ({species})')

# Update layout
fig.update_layout(
    title="Flipper Length Distribution for Each Penguin Species",
    xaxis_title="Flipper Length (mm)",
    yaxis_title="Count",
    barmode='overlay',
    showlegend=False
)

# Show the figure
fig.show()
```

[2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Load the penguins dataset from seaborn
penguins = sns.load_dataset('penguins')

# Drop rows with missing values for simplicity
penguins = penguins.dropna(subset=['flipper_length_mm'])

# Define a color palette
palette = sns.color_palette('Set2', 3)

# Create a 1x3 grid of subplots
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharey=True)

# List of unique species
species_list = penguins['species'].unique()
```

```python
# Loop through each species and create KDE plots
for i, species in enumerate(species_list):
    ax = axes[i]

    # Filter data for the current species
    species_data = penguins[penguins['species'] == species]['flipper_length_mm']

    # Calculate statistics
    mean_fl = species_data.mean()
    median_fl = species_data.median()
    std_fl = species_data.std()
    min_fl = species_data.min()
    max_fl = species_data.max()
    q1_fl = species_data.quantile(0.25)
    q3_fl = species_data.quantile(0.75)

    # Plot KDE for the current species
    sns.kdeplot(species_data, ax=ax, fill=True, color=palette[i], label=species)

    # Mark the mean and median using vertical lines
    ax.axvline(mean_fl, color='green', linestyle='--', label='Mean')
    ax.axvline(median_fl, color='red', linestyle=':', label='Median')

    # Shade the range (min to max)
    ax.axvspan(min_fl, max_fl, color='blue', alpha=0.1, label='Range')

    # Shade the interquartile range (Q1 to Q3)
    ax.axvspan(q1_fl, q3_fl, color='orange', alpha=0.2, label='IQR')

    # Shade the area within two standard deviations from the mean
    ax.axvspan(mean_fl - 2*std_fl, mean_fl + 2*std_fl, color='purple', alpha=0.
↪1, label='±2 Std Dev')

    # Set title and labels
    ax.set_title(f"KDE of Flipper Length for {species}")
    ax.set_xlabel("Flipper Length (mm)")
    if i == 0:  # Only show y-label on the first plot
        ax.set_ylabel("Density")

    # Add a legend
    ax.legend(loc='upper left')

# Adjust layout
plt.tight_layout()
plt.show()
```



[ ]:

```
from scipy import stats
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import numpy as np

n = 1500
data1 = stats.uniform.rvs(0, 10, size=n)
data2 = stats.norm.rvs(5, 1.5, size=n)
data3 = np.r_[stats.norm.rvs(2, 0.25, size=int(n/2)), stats.norm.rvs(8, 0.5, size=int(n/2))]
data4 = stats.norm.rvs(6, 0.5, size=n)

fig = make_subplots(rows=1, cols=4)

fig.add_trace(go.Histogram(x=data1, name='A', nbinsx=30, marker=dict(line=dict(color='black', width=1))), row=1, col=1)
fig.add_trace(go.Histogram(x=data2, name='B', nbinsx=15, marker=dict(line=dict(color='black', width=1))), row=1, col=2)
fig.add_trace(go.Histogram(x=data3, name='C', nbinsx=45, marker=dict(line=dict(color='black', width=1))), row=1, col=3)
fig.add_trace(go.Histogram(x=data4, name='D', nbinsx=15, marker=dict(line=dict(color='black', width=1))), row=1, col=4)

fig.update_layout(height=300, width=750, title_text="Row of Histograms")
fig.update_xaxes(title_text="A", row=1, col=1)
fig.update_xaxes(title_text="B", row=1, col=2)
fig.update_xaxes(title_text="C", row=1, col=3)
fig.update_xaxes(title_text="D", row=1, col=4)
fig.update_xaxes(range=[-0.5, 10.5])

for trace in fig.data:
    trace.xbins = dict(start=0, end=10)
```

```
# This code was produced by just making requests to Microsoft Copilot
# https://github.com/pointOfive/stat130chat130/blob/main/CHATLOG/wk3/COP/SLS/
↪0001_concise_makeAplotV1.md

fig.show() # USE `fig.show(renderer="png")` FOR ALL GitHub and MarkUs␣
↪SUBMISSIONS
```

4. Run the code below and look at the resulting figure of distrubutions and then answer the following questions

1) Which datasets have similar means and similar variances: None
2) Which datasets have similar means but quite different variances: B and D
3) Which datasets have similar variances but quite different means: C and D
4) Which datasets have quite different means and quite different variances: A and C

Pre-lecture Chatbot log: https://chatgpt.com/share/66f5f7e6-9b94-8009-b684-c6356832c597

[ ]:
```
from scipy import stats
import pandas as pd
import numpy as np

sample1 = stats.gamma(a=2,scale=2).rvs(size=1000)
fig1 = px.histogram(pd.DataFrame({'data': sample1}), x="data")
# USE `fig1.show(renderer="png")` FOR ALL GitHub and MarkUs SUBMISSIONS

sample1.mean()
np.quantile(sample1, [0.5]) # median

sample2 = -stats.gamma(a=2,scale=2).rvs(size=1000)
```

6.Go find an interesting dataset and use summary statistics and visualizations to understand and demonstate some interesting aspects of the data

Key Aspects Summary Calorie-Dense Items: Many fast-food items, particularly combo meals and fried foods, have very high calorie counts. Fat and Calories Correlation: High-fat items usually contain more calories. High Sodium Content: Fast food tends to be extremely high in sodium, contributing to unhealthy daily intakes. Sugary Items: Drinks and desserts are significant contributors to sugar intake. Protein vs. Calories: Many high-protein items also come with high calories. Serving Size Impact: Larger portions significantly impact calorie and fat consumption.

Histograms: Distribution of Nutritional Components Purpose: To show how specific nutritional components (like calories, fat, protein, sugar) are distributed across different fast-food items.

What You'll Learn:

Identify the most common range of values (e.g., most items have between 200-600 calories). Spot outliers—items that have unusually high or low values.

Post-lecture Chatbot log: https://chatgpt.com/share/66f60ed5-de78-8009-a9be-03fd9303f8e4

[ ]:
```

```