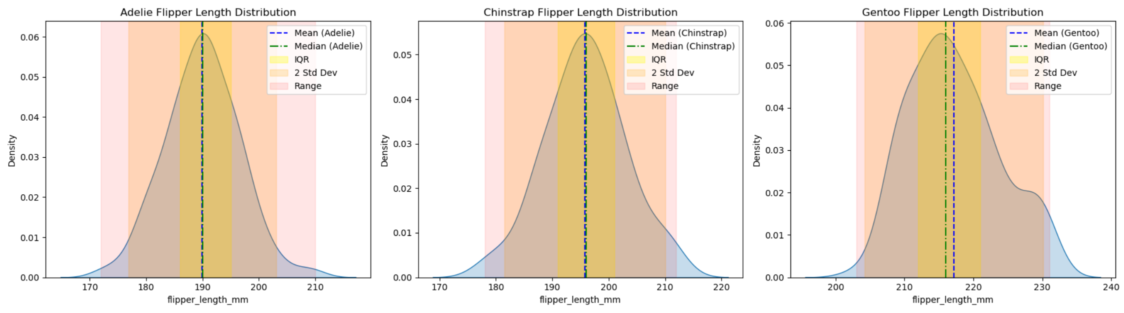
Untitled1

September 27, 2024

[1]:
```python
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
import seaborn as sns

# Load the penguins dataset
penguins = sns.load_dataset("penguins")

# Function to add lines and rectangles for the given species' data
def add_statistics_to_histogram(fig, species_data, species_name):
    # Calculate statistics
    mean = species_data['flipper_length_mm'].mean()
    median = species_data['flipper_length_mm'].median()
    std = species_data['flipper_length_mm'].std()
    min_val = species_data['flipper_length_mm'].min()
    max_val = species_data['flipper_length_mm'].max()
    q1 = species_data['flipper_length_mm'].quantile(0.25)
    q3 = species_data['flipper_length_mm'].quantile(0.75)

    # Range defined by two standard deviations from the mean
    lower_bound = mean - 2 * std
    upper_bound = mean + 2 * std

    # Add vertical lines for mean and median
    fig.add_vline(x=mean, line=dict(color='blue', dash='dash'),␣
↪annotation_text=f"Mean ({species_name})", annotation_position="top left")
    fig.add_vline(x=median, line=dict(color='green', dash='dot'),␣
↪annotation_text=f"Median ({species_name})", annotation_position="top right")

    # Add rectangle for IQR
    fig.add_vrect(x0=q1, x1=q3, fillcolor='yellow', opacity=0.3,␣
↪annotation_text=f"IQR ({species_name})", annotation_position="top left")

    # Add rectangle for the range of two standard deviations
    fig.add_vrect(x0=lower_bound, x1=upper_bound, fillcolor='orange', opacity=0.
↪2, annotation_text=f"2 Std Dev ({species_name})", annotation_position="top␣
↪right")
```

```
    # Add rectangle for the total range (min to max)
    fig.add_vrect(x0=min_val, x1=max_val, fillcolor='red', opacity=0.1,␣
 ↪annotation_text=f"Range ({species_name})", annotation_position="bottom␣
 ↪right")

# Create the histogram for each species
species_list = penguins['species'].unique()
fig = go.Figure()

for species in species_list:
    species_data = penguins[penguins['species'] == species]

    # Create histogram for current species
    fig.add_trace(go.Histogram(
        x=species_data['flipper_length_mm'],
        name=species,
        opacity=0.75
    ))

    # Add mean, median, and ranges (IQR, 2 Std Dev, min-max) to the plot
    add_statistics_to_histogram(fig, species_data, species)

# Update layout for better visualization
fig.update_layout(
    title='Flipper Length Distribution for Penguins Species',
    xaxis_title='Flipper Length (mm)',
    yaxis_title='Count',
    barmode='overlay'
)

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

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

# Load the penguins dataset
penguins = sns.load_dataset("penguins")

# Function to add lines and rectangles for the given species' data
def add_statistics_to_kde(ax, species_data, species_name):
    # Calculate statistics
    mean = species_data['flipper_length_mm'].mean()
    median = species_data['flipper_length_mm'].median()
    std = species_data['flipper_length_mm'].std()
    min_val = species_data['flipper_length_mm'].min()
    max_val = species_data['flipper_length_mm'].max()
    q1 = species_data['flipper_length_mm'].quantile(0.25)
    q3 = species_data['flipper_length_mm'].quantile(0.75)

    # Range defined by two standard deviations from the mean
    lower_bound = mean - 2 * std
    upper_bound = mean + 2 * std

    # Plot KDE
    sns.kdeplot(species_data['flipper_length_mm'], ax=ax, fill=True)

    # Add vertical lines for mean and median
    ax.axvline(mean, color='blue', linestyle='--', label=f"Mean␣
 ↪({species_name})")
    ax.axvline(median, color='green', linestyle='-.', label=f"Median␣
 ↪({species_name})")

    # Add rectangles for IQR and two standard deviation range
    ax.axvspan(q1, q3, alpha=0.3, color='yellow', label="IQR")
    ax.axvspan(lower_bound, upper_bound, alpha=0.2, color='orange', label="2␣
 ↪Std Dev")

    # Add the total range (min to max)
    ax.axvspan(min_val, max_val, alpha=0.1, color='red', label="Range")

    # Set the title and legend
    ax.set_title(f"{species_name} Flipper Length Distribution")
    ax.legend()

# Create the KDE plots for each species
species_list = penguins['species'].unique()

# Set up the figure with 3 subplots (one for each species) arranged in a row
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# Plot for each species
for i, species in enumerate(species_list):
    species_data = penguins[penguins['species'] == species]

    # Add statistics and plot KDE for current species
    add_statistics_to_kde(axes[i], species_data, species)

# Adjust layout for better visualization
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

# Generate data
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)

# Create subplot with 1 row and 4 columns
fig = make_subplots(rows=1, cols=4)

# Add histograms to each subplot
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)

# Update layout for axis titles and plot size
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])

# Adjust bin ranges for all histograms
```

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

# Display the figure
import ace_tools as tools; tools.display_plot(fig)
```

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

https://chatgpt.com/share/66f5fa4b-5480-800e-83db-4371ff1f4f11 - Links to Pre-lecture Question 1~4

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

Calorie Distribution: The mean number of calories across the food items is 296.1 calories, but the distribution is skewed with a few very high-calorie items (as shown in the histogram). There are food items with 0 calories, which likely represent drinks or very low-calorie options like water or diet beverages. Most food items fall into a range between 150 to 400 calories, but there are extreme outliers that exceed this, highlighting certain high-calorie meals.

https://chatgpt.com/share/66f60de9-882c-800e-9f00-6cc82eda3da0

[ ]:
```

```