PERFORM A DECISION MATRIX CALCULATION

```
import numpy as np
from scipy.spatial.distance import cdist
# Assuming we have the input arrays x, y, wx, wy, and size defined
# we need to define these arrays or replace them with your data
# Create a matrix containing the x and y coordinates
coordinates = np.column_stack((x, y))

# Create weight vectors for x and y
wx = np.array(wx)
wy = np.array(wy)

# Calculate the pairwise distances between all points
distances = cdist(coordinates, coordinates)

# Calculate the decision matrix
decision_matrix = np.zeros_like(distances)

for i in range(len(x)):
    for j in range(len(y)):
        decision_matrix[i, j] = wx[i] * wy[j] * distances[i, j]

# Adjust the decision matrix based on the specified size parameter
size = min(size, min(decision_matrix.shape))
decision_matrix[range(size), range(size)] = 0

# Now, the decision_matrix contains the result you need
import pyreadr
```

```
# Load the RData file
result = pyreadr.read_r("ausact-bic.RData")

# Access the R objects from the loaded file
# The objects will be stored as keys in the result dictionary
# For example, if you have an object named "my_data" in the RData file:
my_data = result["my_data"]

from bibitr import BiBit
# Assuming you have loaded or prepared your data in a suitable format
# Replace 'data_matrix' with your data matrix
# Replace 'row_names' and 'col_names' with row and column labels if available

# Create a BiBit object
bibit = BiBit()

# Fit the biclustering model
model = bibit.fit(data_matrix)

# Get the bicluster number
bcn = model.get_bicluster_number()

# bcn now contains the bicluster number for each row and column



import pandas as pd
from rpy2.robjects.packages import importr

# Load the required R package
MSA = importr("MSA")
```

```
# Load the "ausActiv" dataset from the MSA package
ausActiv = MSA.data("ausActiv")

# Create a list to store the results
cl12 = [None] * len(ausActiv)

# Assuming "bcn" is a list of data frames in R
# You'll need to define it or convert it to a suitable Python format
# Here, I'm assuming "bcn" is a list of pandas DataFrames

for k, df in enumerate(bcn, 1):  # Start k from 1 as R's seq_along starts from 1
    for row in df["Rows"]:
        cl12[row - 1] = k  # Adjust row index to start from 0

# Convert the result to a pandas Series
cl12 = pd.Series(cl12)

# Now, cl12 contains the desired values.
```

TO GENERATE A TABLE

```
import pandas as pd
# Assuming you have the 'cl12' Series defined
# Replace 'cl12' with your actual Series if needed
# Count the occurrences of each value, including NaN (equivalent to NA in R)
value_counts = cl12.value_counts(dropna=False)

# Convert the result to a DataFrame for a similar table format
table_df = pd.DataFrame({"cl12": value_counts})

# Rename the columns to match the R output
table_df.columns = ["count"]
```

```
# Sort the DataFrame by the 'cl12' values for consistency
table_df.sort_index(inplace=True)
# Display the table
print(table_df)
```

TO CREATE THE CL12.3 FACTOR VARIABLE

```
import pandas as pd
# Assuming you have the 'cl12' Series defined
# Replace 'cl12' with your actual Series if needed
# Create a boolean mask for "Not Segment 3"
not_segment_3_mask = ~cl12.isna() & (cl12 == 3)
```

TO CREATE A BOXPLOT IN PYTHON

```
# Create the 'cl12.3' factor variable
cl12_3 = pd.Series(pd.Categorical(not_segment_3_mask, categories=[False, True], labels=["Not Segment 3", "Segment 3"]))
# Now, 'cl12.3' contains the factor variable as specified
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Assuming you have the 'ausActivDesc' dataset loaded
# Replace 'ausActivDesc' with your actual dataset if needed

# Create the boxplot
plt.figure(figsize=(10, 6))
boxprops = dict(linewidth=2, color='blue')
medianprops = dict(linewidth=2, color='red')
flierprops = dict(marker='o', markersize=5, markerfacecolor='green', linestyle='none')
plt.boxplot([ausActivDesc[ausActivDesc['cl12.3'] == 'Not Segment 3']['spendpppd'].dropna(),
            ausActivDesc[ausActivDesc['cl12.3'] == 'Segment 3']['spendpppd'].dropna()],
        notch=True, vert=True, widths=0.7,
        boxprops=boxprops, medianprops=medianprops, flierprops=flierprops)

# Set y-axis to logarithmic scale
plt.yscale('log')

# Labels and title
plt.xticks([1, 2], ['Not Segment 3', 'Segment 3'])
plt.ylabel('AUD per person per day')
plt.title('Boxplot of spendpppd by cl12.3')

# Show the boxplot
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

PYTHON CODE TO CREATE THE PROPORTIONAL BARCHART:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have the 'ausActivDesc' dataset loaded
# Replace 'ausActivDesc' with your actual dataset if needed

# Subset the DataFrame to include only relevant columns (starting with 'book')
book_columns = [col for col in ausActivDesc.columns if col.startswith('book')]
subset_data = ausActivDesc[book_columns + ['cl12.3']]

# Calculate the proportion of each category for each group
prop_data = subset_data.groupby(['cl12.3']).apply(lambda x: x.mean() * 100)

# Reorder the columns based on their values
```

```
prop_data = prop_data[sorted(prop_data.columns, key=lambda x: prop_data['Segment 3'][x], reverse=True)]

# Create the barchart
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=prop_data, orient='h')

# Set the x-axis limits
plt.xlim(-2, 102)

# Set labels and title
plt.xlabel('Percent')
plt.ylabel('')
plt.title('Proportional Barchart of "book" Categories by cl12.3')

# Show the barchart
plt.show()
```

CREATE A PROPORTIONAL BARCHART IN PYTHON

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have the 'ausActivDesc' dataset loaded
# Replace 'ausActivDesc' with your actual dataset if needed

# Subset the DataFrame to include only relevant columns (starting with 'info')
info_columns = [col for col in ausActivDesc.columns if col.startswith('info')]
subset_data = ausActivDesc[info_columns + ['cl12.3']]

# Calculate the proportion of each category for each group
```

```
prop_data = subset_data.groupby(['cl12.3']).apply(lambda x: x.mean() * 100)

# Reorder the columns based on their values
prop_data = prop_data[sorted(prop_data.columns, key=lambda x: prop_data['Segment 3'][x], reverse=True)]

# Create the barchart
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=prop_data, orient='h')

# Set the x-axis limits
plt.xlim(-2, 102)

# Set labels and title
plt.xlabel('Percent')
plt.ylabel('')
plt.title('Proportional Barchart of "info" Categories by cl12.3')

# Show the barchart
plt.show()
```

PYTHON CODE TO CREATE THE MOSAIC PLOT

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm

# Assuming you have the 'ausActivDesc' dataset loaded
# Replace 'ausActivDesc' with your actual dataset if needed

# Create a contingency table
contingency_table = pd.crosstab(ausActivDesc['cl12.3'], ausActivDesc['TV.channel'])
```

```
# Fit the mosaic plot
mosaic = sm.graphics.mosaicplot(contingency_table.stack(), title="",
                              labelizer=lambda k: '')

# Set the x-axis labels to be vertical (las=2 equivalent in R)
plt.xticks(rotation=90)

# Show the mosaic plot
plt.show()
```