RAIV_Modeling_WITS

July 30, 2025

Utilize two interactive core functioning tools along with additional stagnant information + visualizations.

Cell 10- Bar chart displaying the sum of each countries total RAIV (Risk-Adjusted Import Value) across the 5 HS Codes sampled within my data.

Cell 11- Allows user to input country/economy of interest and analyze the 3 year progression of RAIV across the 5 HS Codes from 2022-2024. Helpful in determining and distinguishing trustworthy, stable exporters of pet-related products to the US.

Cell 16- Top recommendation scoring system and interactive calculator. User can input the weights they would like to associate with the 3 categories of RAIV (Total Import) Volume utilizing a logical risk multiplier, LPI 2023 Timeliness Scores, and a Risk Score using comprehensive LPI data. The top 10 countries will be output in descending order with higher scores indicating more favorable procurement conditions.

[3]:

```
#Packages
import pandas as pd
import numpy as np
import sqlite3
```

[4]:

```
# Load each sheet into its own DataFrame
import_2022 = pd.read_excel("RAIV.xlsx", sheet_name="Cleaned_Import_2022")
import_2023 = pd.read_excel("RAIV.xlsx", sheet_name="Cleaned_Import_2023")
import_2024 = pd.read_excel("RAIV.xlsx", sheet_name="Cleaned_Import_2024")
lpi_2023 = pd.read_excel("RAIV.xlsx", sheet_name="LPI_2023")
risk_multipliers = pd.read_excel("RAIV.xlsx", sheet_name="Risk_Multipliers")
```

[5]:

```
#Sanity check
print(import_2022.head())
print(risk_multipliers.head())
```

```
   HS Code product_category  year Reporter Name  PartnerCode PartnerISO3  \
0   230910         Pet Food  2022           USA            0         WLD
1   230910         Pet Food  2022           USA           31         AZE
2   230910         Pet Food  2022           USA           32         ARG
3   230910         Pet Food  2022           USA           36         AUS
4   230910         Pet Food  2022           USA           40         AUT

  PartnerName  TradeValuein1000USD
```

```
0       World         2161331.332
1  Azerbaijan             414.708
2   Argentina            4050.742
3   Australia            1647.953
4     Austria           10857.428
      Economy  TimelinessScore  RiskPremium
0    Singapore              4.3         0.05
1      Finland              4.3         0.05
2      Denmark              4.1         0.05
3      Germany              4.1         0.05
4  Netherlands              4.0         0.05
```

[6]:

```
#Core calculation of Risk Adjusted Import Values
def calculate_raiv(import_df, year, lpi_df, risk_df, base_year=2022):
    t = year - base_year

    # Merge LPI and Risk tables first (both use "Economy")
    lpi_with_risk = lpi_df.merge(risk_df[['Economy', 'RiskPremium']],␣
 ↪on='Economy', how='left')

    # Join to import data on PartnerName == Economy
    merged_df = import_df.merge(lpi_with_risk, left_on='PartnerName',␣
 ↪right_on='Economy', how='left')

    # Drop rows with missing data (optional or handle otherwise)
    merged_df = merged_df.dropna(subset=['TradeValuein1000USD',␣
 ↪'TimelinessScore', 'RiskPremium'])

    # RAIV formula
    merged_df['RAIV'] = (
        merged_df['TradeValuein1000USD'] *
        merged_df['TimelinessScore'] /
        (1 + merged_df['RiskPremium']) ** t
    )

    return merged_df[['PartnerName', 'HS Code', 'TradeValuein1000USD',␣
 ↪'TimelinessScore', 'RiskPremium', 'RAIV']]
```

[7]:

```
#RAIV calculations for each year of import information for every economy

raiv_2022 = calculate_raiv(import_2022, 2022, lpi_2023, risk_multipliers)
raiv_2023 = calculate_raiv(import_2023, 2023, lpi_2023, risk_multipliers)
raiv_2024 = calculate_raiv(import_2024, 2024, lpi_2023, risk_multipliers)
```

[8]:

```
#Combine/ Concatenate all RAIV values into single table
raiv_2022['Year'] = 2022
raiv_2023['Year'] = 2023
```

```
raiv_2024['Year'] = 2024

raiv_all_years = pd.concat([raiv_2022, raiv_2023, raiv_2024], ignore_index=True)
```

[9]:
```
#Country risk table (to be used in following visualization)
country_risk = raiv_all_years.groupby('PartnerName')['RAIV'].sum().
 ↪sort_values(ascending=False)
country_risk.head(10)
```

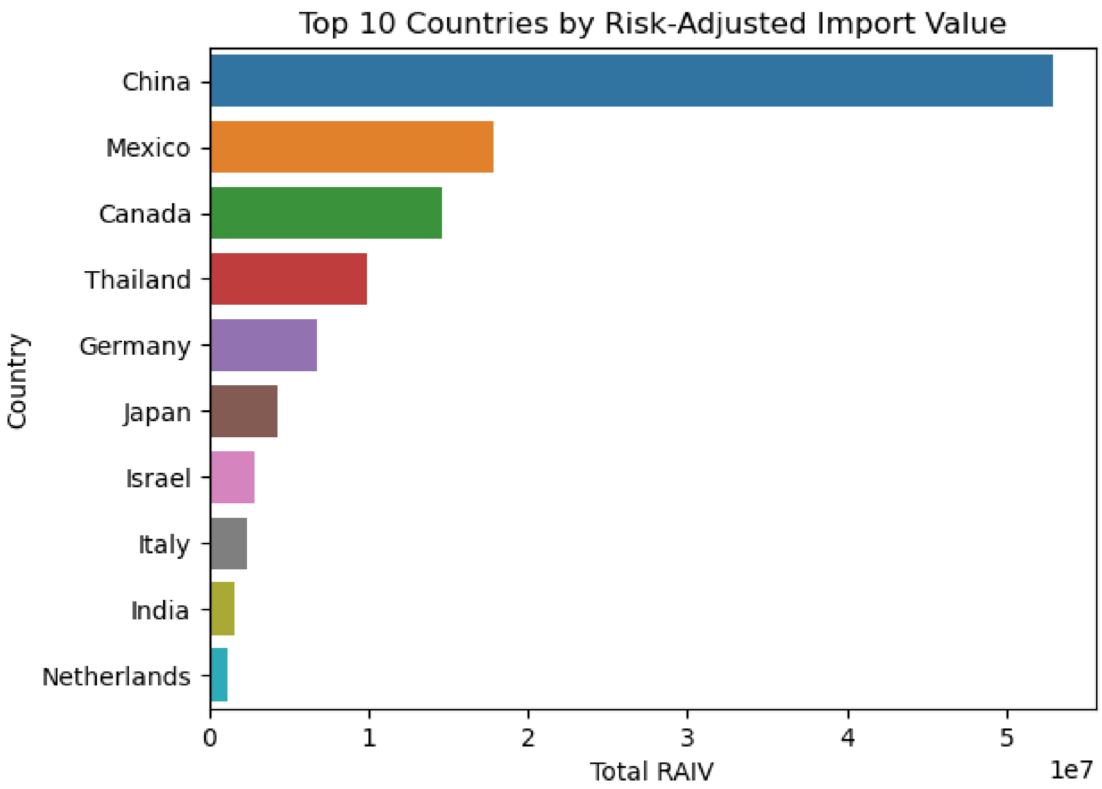
[9]:
```
PartnerName
China          5.295836e+07
Mexico         1.781599e+07
Canada         1.461248e+07
Thailand       9.913881e+06
Germany        6.791314e+06
Japan          4.239156e+06
Israel         2.806725e+06
Italy          2.323738e+06
India          1.546421e+06
Netherlands    1.129547e+06
Name: RAIV, dtype: float64
```

[10]:
```
#Stacked bar chart creation based on total RAIV value across 5 HS Categories␣
 ↪(red flag the data manipulation)
#Countries ranked in the same order as Total Import Values

import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(x=country_risk.values[:10], y=country_risk.index[:10])
plt.title("Top 10 Countries by Risk-Adjusted Import Value")
plt.xlabel("Total RAIV")
plt.ylabel("Country")
plt.show()

#China RAIV takes up 42% of the total concentration of total import value, the␣
 ↪HHI of 0.2243 additionally suggests that if this were to be a supplier␣
 ↪portfolio, it would be too overly reliant on top partners)
#To first distinguish if there is significant merit to this claim, I will␣
 ↪assess RAIV again after removing China and controlling for this over␣
 ↪concentration
```



[11]:

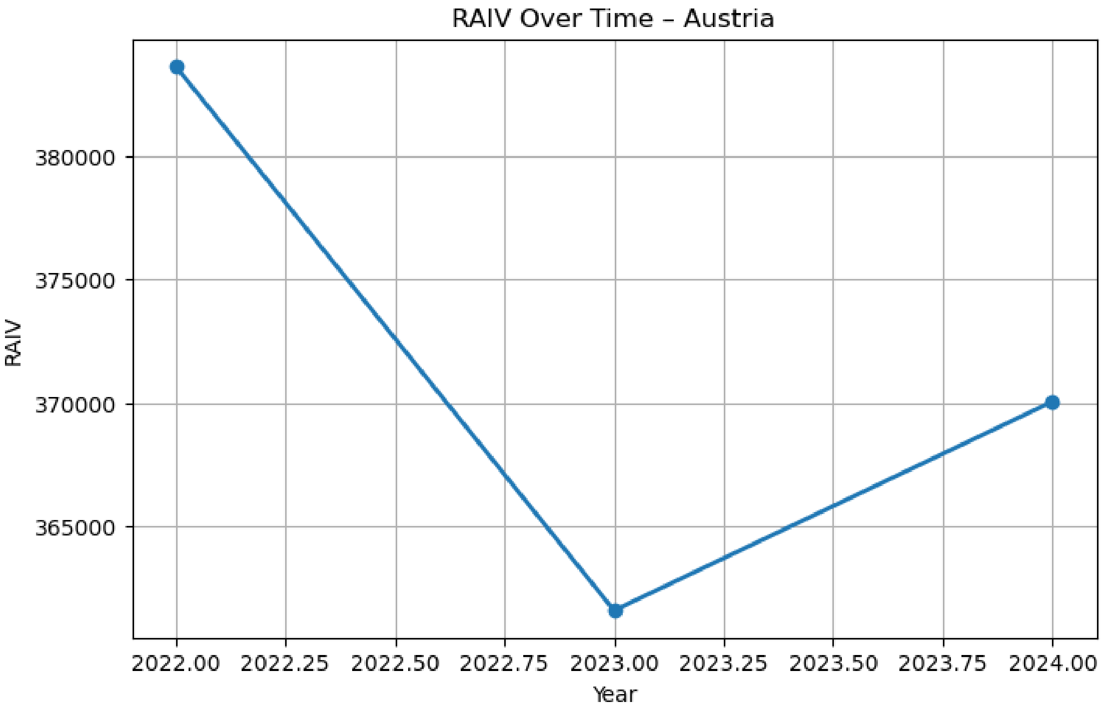
```
#Input a valid country to generate a 3 year line chart of RAIV, do not include␣
 ↪any spaces
import matplotlib.pyplot as plt

# Ask user for a country name
country_input = input("Enter a country name (PartnerName): ")

# Filter the RAIV data for the user’s country
country_data = raiv_all_years[raiv_all_years['PartnerName'].str.lower() ==␣
 ↪country_input.lower()]

if country_data.empty:
    print("No country data available.")
else:
    raiv_by_year = (
        country_data.groupby('Year')['RAIV']
        .sum()
        .reset_index()
        .sort_values('Year')
    )
    plt.figure(figsize=(8, 5))
    plt.plot(raiv_by_year['Year'], raiv_by_year['RAIV'], marker='o',␣
 ↪linewidth=2)
    plt.title(f"RAIV Over Time - {country_input}")
    plt.xlabel("Year")
    plt.ylabel("RAIV")
    plt.grid(True)
    plt.show()
```

```
Enter a country name (PartnerName):  Austria
```



```
[ ]: #Identifying concentration, based on output, China has a far larger share of␣
      ↪RAIV than intended
     country_shares = (
         raiv_all_years.groupby('PartnerName')['RAIV'].sum()
         / raiv_all_years['RAIV'].sum()
     ).sort_values(ascending=False)

     country_shares.head(10)
```

```
[ ]: #Herfindahl-Hirschim Index
     hhi = (country_shares ** 2).sum()
     print(f"HHI Score: {hhi:.4f}")
```

[ ]:

```
#Reccomendation query for procurement, based on user input risk adjusted␣
 ↪factors
recommendation_df = (
    raiv_all_years.groupby('PartnerName')
    .agg({
        'RAIV': 'sum',
        'RiskPremium': 'mean',
        'TimelinessScore': 'mean'
    })
    .reset_index()
)

# Composite Score: Maximize value and timeliness, minimize risk
recommendation_df['Score'] = (
    (recommendation_df['RAIV'] / recommendation_df['RAIV'].max()) * 0.4 +
    (recommendation_df['TimelinessScore'] / 5) * 0.4 -
    (recommendation_df['RiskPremium'] / recommendation_df['RiskPremium'].max())␣
 ↪* 0.2
)

top_recommendations = recommendation_df.sort_values(by='Score',␣
 ↪ascending=False).head(5)
top_recommendations
```

[14]:

```
# Remove China and recalculate HHI
raiv_no_china = raiv_all_years[raiv_all_years['PartnerName'] != 'China']

# Recalculate total RAIV and shares
total_raiv_no_china = raiv_no_china['RAIV'].sum()
raiv_no_china['RAIV_Share'] = raiv_no_china['RAIV'] / total_raiv_no_china

# HHI without China
hhi_no_china = (raiv_no_china['RAIV_Share'] ** 2).sum()
print(f"HHI Without China: {hhi_no_china:.4f}")

#Output of 0.0347 indicates significant improvement, RAIV is no longer skewed
```

```
HHI Without China: 0.0347
```

```
/tmp/ipykernel_1044/526038667.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  raiv_no_china['RAIV_Share'] = raiv_no_china['RAIV'] / total_raiv_no_china
```

```
[ ]: #Creation of weight scoring system, users can adjust for 3 types of criteria to␣
      ↪rank and assess the viability of suppliers
     #By imputing a higher # (0-100), users can place more emphasis on countries␣
      ↪with more established supply chains incorporating pet products with higher␣
      ↪export volumes,
     #or place more emphasis on LPI timeliness rankings from 2023 (valuing greater␣
      ↪efficiency, for pull systems where customers make custom requests or for␣
      ↪e-commerce businesses),
     #or finally placing more emphasis on less risk associated with the computed␣
      ↪Risk Premium

     # Prompt user for weights
     raiv_weight = float(input("Enter weight for RAIV (e.g. 40 for 40%): "))
     timeliness_weight = float(input("Enter weight for TimelinessScore (e.g. 40 for␣
      ↪40%): "))
     risk_weight = float(input("Enter weight for RiskPremium (e.g. 20 for 20%): "))

     # Normalize weights
     total_weight = raiv_weight + timeliness_weight + risk_weight
     raiv_weight /= total_weight
     timeliness_weight /= total_weight
     risk_weight /= total_weight

     # Group by and calculate aggregates
     recommendation_df = (
         raiv_no_china.groupby('PartnerName')
         .agg({
             'RAIV': 'sum',
             'RiskPremium': 'mean',
             'TimelinessScore': 'mean'
         })
         .reset_index()
     )

     # Calculate composite score based on weights
     recommendation_df['Score'] = (
         (recommendation_df['RAIV'] / recommendation_df['RAIV'].max()) * raiv_weight␣
      ↪+
         (recommendation_df['TimelinessScore'] / 5) * timeliness_weight -
         (recommendation_df['RiskPremium'] / recommendation_df['RiskPremium'].max())␣
      ↪* risk_weight
     )

     # Sort and return top recommendations
     top_recommendations = recommendation_df.sort_values(by='Score',␣
      ↪ascending=False).head(10)
```

```
top_recommendations
```

[ ]:

```
#Re-wiring the original "top_reccomendation tool to include a more complex,␣
 ↪thought out Risk Multiplier
#Instead of a RM based soley on conditional logic surrounding LPI Timeliness␣
 ↪Data, I decided to weigh each subcategory category at 0.15, and LPI␣
 ↪Timeliness weighted at 0.25
#This weighing gives a more comprehensive risk assessment based on 2023 LPI␣
 ↪Scores, yet still champions efficiency (more delicate balance of␣
 ↪effectiveness and efficiency)

# Load LPI data for 2023
lpi_df = pd.read_excel("RAIV.xlsx", sheet_name="LPI_2023")

# Normalize columns to avoid divide-by-zero and bring everything to the same␣
 ↪scale
for col in [
    "CustomsScore", "InfrastructureScore", "InternationalShipmentsScore",
    "TimelinessScore", "LogisticsCompetenceandQualityScore",␣
 ↪"TrackingandTracingScore"
]:
    lpi_df[col] = pd.to_numeric(lpi_df[col], errors='coerce')

# Compute the new RiskScore
lpi_df["RiskScore"] = 1 / (
    0.15 * lpi_df["CustomsScore"] +
    0.15 * lpi_df["InfrastructureScore"] +
    0.15 * lpi_df["InternationalShipmentsScore"] +
    0.25 * lpi_df["TimelinessScore"] +
    0.15 * lpi_df["LogisticsCompetenceandQualityScore"] +
    0.15 * lpi_df["TrackingandTracingScore"]
)

# Merge by country name - make sure the column names align
merged_df = raiv_no_china.merge(lpi_df[['Economy', 'RiskScore']],␣
 ↪left_on='PartnerName', right_on='Economy', how='left')

# User input for custom weighting
raiv_weight = float(input("Weight for RAIV (%): "))
timeliness_weight = float(input("Weight for Timeliness (%): "))
risk_weight = float(input("Weight for RiskScore (%): "))

# Normalize
total = raiv_weight + timeliness_weight + risk_weight
raiv_weight /= total
timeliness_weight /= total
```

```
risk_weight /= total

# Group + score
recommendation_df = (
    merged_df.groupby('Economy')
    .agg({
        'RAIV': 'sum',
        'TimelinessScore': 'mean',
        'RiskScore': 'mean'
    })
    .reset_index()
)

# Composite score
recommendation_df['Score'] = (
    (recommendation_df['RAIV'] / recommendation_df['RAIV'].max()) * raiv_weight␣
 ↪+
    (recommendation_df['TimelinessScore'] / 5) * timeliness_weight -
    (recommendation_df['RiskScore'] / recommendation_df['RiskScore'].max()) *␣
 ↪risk_weight
)

#Based on user priorities this assessment gives a top 10 which includes all␣
 ↪associated data points with the country, as well as the final score (a␣
 ↪higher composite score represents a better "fit" for your procurement based␣
 ↪on your criteria)
top_rec_RiskScore = recommendation_df.sort_values(by='Score', ascending=False).
 ↪head(10)
top_rec_RiskScore
```

[19]:

```
#To create a matching dataframe for lightweight streamlit application
merged_df.to_csv("adjusted_RM_raiv_no_china.csv", index= False)
```