Yaman

August 31, 2025

[5]:
```python
import pandas as pd

# Define Yaman scale ratios (Just Intonation, reference Sa=1/1)
ratios = {
    "Sa": 1.0,
    "Re": 9/8,      # major second
    "Ga": 5/4,      # major third
    "Ma": 45/32,    # tivra Ma (aug 4th)
    "Pa": 3/2,      # perfect fifth
    "Dha": 5/3,     # major sixth
    "Ni": 15/8,     # major seventh
    "Sa'": 2.0      # octave
}

# Build ratio matrix (fractions and decimals)
swara_list = list(ratios.keys())
matrix_fraction = []
matrix_decimal = []

for r1 in swara_list:
    row_frac = []
    row_dec = []
    for r2 in swara_list:
        ratio_val = ratios[r2] / ratios[r1]
        # simplify fraction form
        frac = f"{ratios[r2]}/{ratios[r1]}"
        row_frac.append(frac)
        row_dec.append(round(ratio_val, 4))
    matrix_fraction.append(row_frac)
    matrix_decimal.append(row_dec)

# Create DataFrames
df_fraction = pd.DataFrame(matrix_fraction, index=swara_list,␣
 ↪columns=swara_list)
df_decimal = pd.DataFrame(matrix_decimal, index=swara_list, columns=swara_list)

df_fraction, df_decimal
```

[5]: (

	Sa	Re
Sa	1.0/1.0	1.125/1.0
Re	1.0/1.125	1.125/1.125
Ga	1.0/1.25	1.125/1.25
Ma	1.0/1.40625	1.125/1.40625
Pa	1.0/1.5	1.125/1.5
Dha	1.0/1.6666666666666667	1.125/1.6666666666666667
Ni	1.0/1.875	1.125/1.875
Sa'	1.0/2.0	1.125/2.0

\

	Ga	Ma
Sa	1.25/1.0	1.40625/1.0
Re	1.25/1.125	1.40625/1.125
Ga	1.25/1.25	1.40625/1.25
Ma	1.25/1.40625	1.40625/1.40625
Pa	1.25/1.5	1.40625/1.5
Dha	1.25/1.6666666666666667	1.40625/1.6666666666666667
Ni	1.25/1.875	1.40625/1.875
Sa'	1.25/2.0	1.40625/2.0

\

	Pa	Dha
Sa	1.5/1.0	1.6666666666666667/1.0
Re	1.5/1.125	1.6666666666666667/1.125
Ga	1.5/1.25	1.6666666666666667/1.25
Ma	1.5/1.40625	1.6666666666666667/1.40625
Pa	1.5/1.5	1.6666666666666667/1.5
Dha	1.5/1.6666666666666667	1.6666666666666667/1.6666666666666667
Ni	1.5/1.875	1.6666666666666667/1.875
Sa'	1.5/2.0	1.6666666666666667/2.0

\

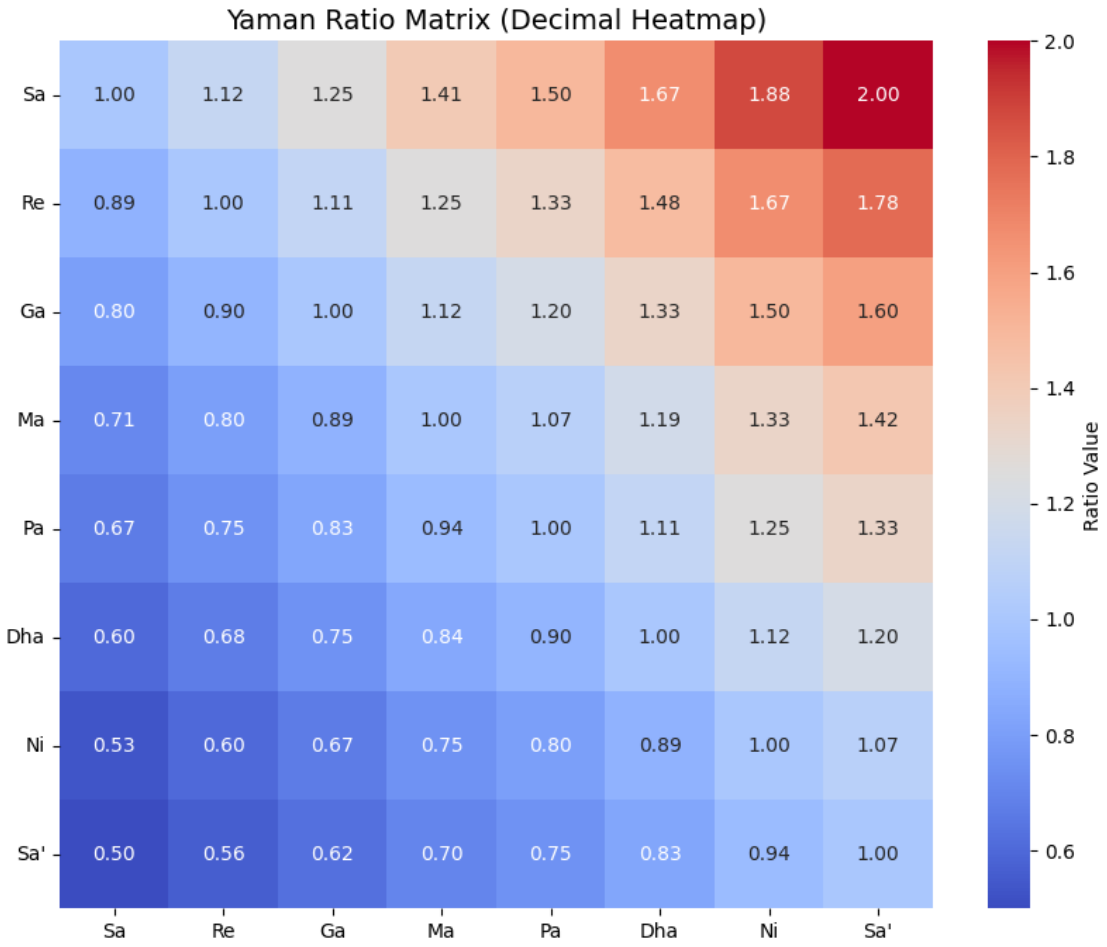
	Ni	Sa'
Sa	1.875/1.0	2.0/1.0
Re	1.875/1.125	2.0/1.125
Ga	1.875/1.25	2.0/1.25
Ma	1.875/1.40625	2.0/1.40625
Pa	1.875/1.5	2.0/1.5
Dha	1.875/1.6666666666666667	2.0/1.6666666666666667
Ni	1.875/1.875	2.0/1.875
Sa'	1.875/2.0	2.0/2.0

,

	Sa	Re	Ga	Ma	Pa	Dha	Ni	Sa'
Sa	1.0000	1.1250	1.2500	1.4062	1.5000	1.6667	1.8750	2.0000
Re	0.8889	1.0000	1.1111	1.2500	1.3333	1.4815	1.6667	1.7778
Ga	0.8000	0.9000	1.0000	1.1250	1.2000	1.3333	1.5000	1.6000
Ma	0.7111	0.8000	0.8889	1.0000	1.0667	1.1852	1.3333	1.4222
Pa	0.6667	0.7500	0.8333	0.9375	1.0000	1.1111	1.2500	1.3333
Dha	0.6000	0.6750	0.7500	0.8438	0.9000	1.0000	1.1250	1.2000
Ni	0.5333	0.6000	0.6667	0.7500	0.8000	0.8889	1.0000	1.0667

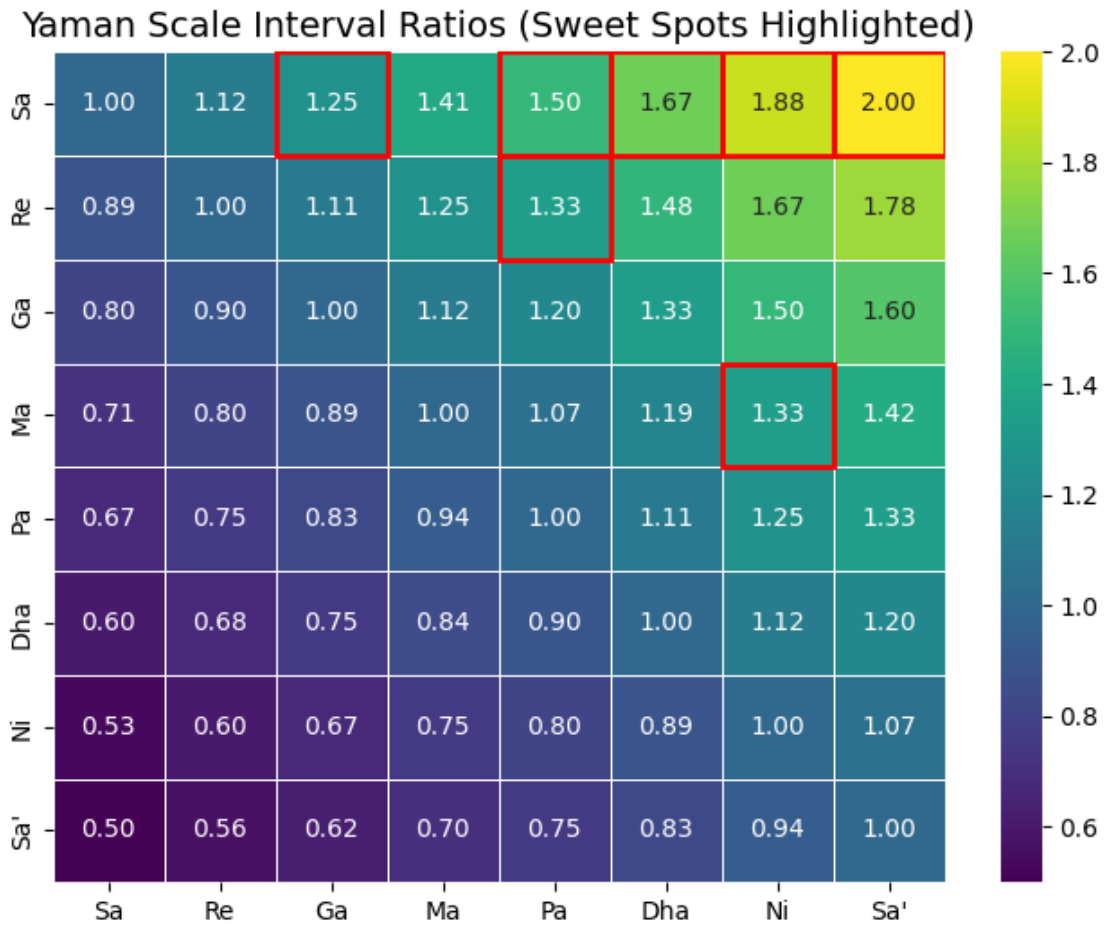
```
 Sa'   0.5000   0.5625   0.6250   0.7031   0.7500   0.8333   0.9375   1.0000)
```

[7]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Define Yaman scale ratios (Just Intonation, reference Sa=1/1)
ratios = {
    "Sa": 1.0,
    "Re": 9/8,      # major second
    "Ga": 5/4,      # major third
    "Ma": 45/32,    # tivra Ma (aug 4th)
    "Pa": 3/2,      # perfect fifth
    "Dha": 5/3,     # major sixth
    "Ni": 15/8,     # major seventh
    "Sa'": 2.0      # octave
}

# Build ratio matrix (fractions and decimals)
swara_list = list(ratios.keys())
matrix_fraction = []
matrix_decimal = []

for r1 in swara_list:
    row_frac = []
    row_dec = []
    for r2 in swara_list:
        ratio_val = ratios[r2] / ratios[r1]
        frac = f"{ratios[r2]}/{ratios[r1]}"
        row_frac.append(frac)
        row_dec.append(round(ratio_val, 4))
    matrix_fraction.append(row_frac)
    matrix_decimal.append(row_dec)

# Create DataFrames
df_fraction = pd.DataFrame(matrix_fraction, index=swara_list,␣
 ↪columns=swara_list)
df_decimal = pd.DataFrame(matrix_decimal, index=swara_list, columns=swara_list)

# --- Heatmap Visualization ---
plt.figure(figsize=(10, 8))
sns.heatmap(df_decimal, annot=True, cmap="coolwarm", cbar_kws={'label': 'Ratio␣
 ↪Value'}, fmt=".2f")
plt.title("Yaman Ratio Matrix (Decimal Heatmap)", fontsize=14)
plt.yticks(rotation=0)
plt.show()
```



[8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Define Yaman scale ratios (Just Intonation, reference Sa=1/1)
ratios = {
    "Sa": 1.0,
    "Re": 9/8,      # major second
    "Ga": 5/4,      # major third
    "Ma": 45/32,    # tivra Ma (aug 4th)
    "Pa": 3/2,      # perfect fifth
    "Dha": 5/3,     # major sixth
    "Ni": 15/8,     # major seventh
    "Sa'": 2.0      # octave
}

# Build ratio matrix
```

```
swara_list = list(ratios.keys())
matrix_decimal = []

for r1 in swara_list:
    row_dec = []
    for r2 in swara_list:
        ratio_val = ratios[r2] / ratios[r1]
        row_dec.append(round(ratio_val, 4))
    matrix_decimal.append(row_dec)

df_decimal = pd.DataFrame(matrix_decimal, index=swara_list, columns=swara_list)

# Define sweet consonant ratios to highlight
sweet_ratios = {
    ("Sa", "Pa"),  # 3:2
    ("Sa", "Ga"),  # 5:4
    ("Sa", "Dha"), # 5:3
    ("Sa", "Ni"),  # 15:8
    ("Re", "Pa"),  # 4:3
    ("Ma", "Ni"),  # 4:3
    ("Sa", "Sa'")  # Octave 2:1
}

# Plot heatmap
plt.figure(figsize=(8,6))
ax = sns.heatmap(df_decimal, annot=True, cmap="viridis", cbar=True, fmt=".2f",␣
 ↪linewidths=0.5)

# Highlight sweet intervals with rectangles
for (r1, r2) in sweet_ratios:
    i, j = swara_list.index(r1), swara_list.index(r2)
    ax.add_patch(plt.Rectangle((j, i), 1, 1, fill=False, edgecolor="red", lw=2))

plt.title("Yaman Scale Interval Ratios (Sweet Spots Highlighted)", fontsize=14)
plt.show()
```



[9]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Define Yaman scale ratios (Just Intonation, reference Sa=1/1)
ratios = {
    "Sa": 1.0,
    "Re": 9/8,      # major second
    "Ga": 5/4,      # major third
    "Ma": 45/32,    # tivra Ma (aug 4th)
    "Pa": 3/2,      # perfect fifth
    "Dha": 5/3,     # major sixth
    "Ni": 15/8,     # major seventh
    "Sa'": 2.0      # octave
}
```

```
# Build ratio matrix (fractions and decimals)
swara_list = list(ratios.keys())
matrix_fraction = []
matrix_decimal = []

for r1 in swara_list:
    row_frac = []
    row_dec = []
    for r2 in swara_list:
        ratio_val = ratios[r2] / ratios[r1]
        frac = f"{int(ratios[r2]*128)}/{int(ratios[r1]*128)}"  # approximate␣
 ↪ratio in integers
        row_frac.append(frac)
        row_dec.append(round(ratio_val, 4))
    matrix_fraction.append(row_frac)
    matrix_decimal.append(row_dec)

# DataFrames
df_fraction = pd.DataFrame(matrix_fraction, index=swara_list,␣
 ↪columns=swara_list)
df_decimal = pd.DataFrame(matrix_decimal, index=swara_list, columns=swara_list)

# Sweet consonant ratios to highlight (as decimals for matching)
sweet_ratios = {
    "3:2": 1.5,    # Sa-Pa
    "5:4": 1.25,   # Sa-Ga
    "5:3": 1.6667, # Sa-Dha
    "15:8": 1.875, # Sa-Ni
    "4:3": 1.3333, # Ma-Ni (or perfect fourth relationships)
    "2:1": 2.0     # Octave Sa-Sa'
}

# Plot heatmap with annotations
plt.figure(figsize=(10, 8))
ax = sns.heatmap(df_decimal, annot=df_fraction, fmt="", cmap="YlGnBu",␣
 ↪cbar=True, linewidths=0.5, linecolor="gray")

# Highlight sweet spots
for i, r1 in enumerate(swara_list):
    for j, r2 in enumerate(swara_list):
        val = df_decimal.iloc[i, j]
        for label, sweet_val in sweet_ratios.items():
            if np.isclose(val, sweet_val, atol=0.01):  # tolerance for matching
                ax.add_patch(plt.Rectangle((j, i), 1, 1, fill=False,␣
 ↪edgecolor="red", lw=2))
                ax.text(j+0.5, i+0.5, label, color="red", ha="center",␣
 ↪va="center", fontsize=10, weight="bold")
```

```
plt.title("Yaman Ratio Heatmap (Sweet Spots Highlighted)", fontsize=14)
plt.show()
```