Analysis of Chanting Effects of Hare Krishna Mantra with EEG Aquisition System

Comparative Study of Pre, During, and Post-Chanting Emotional Changes in Brain Activity through EEG

Karthik M Dani

2024-07-20

Context of the Data

A data aquisition session was done on `16` individuals, where they were monitored with `EEG Band`.

EEG Signal Aquisition of each of the subjects were split into 3 sub sessions:

1. Signal Aquisition Before Chant of Hare Krishna Mantra - For a duration of 5 mins
2. Signal Aquisition while the subject was chanting Hare Krishna Mantra - Duration of aquisition depended pace of chanting
3. Signal Aquisition after the subject successfully completed chanting Hare Krishna Mantra - For 5 mins

Data Analysis

Loading the `xlsx` formatted data (which was clubbed together by the earlier version of gui based python code with subject details) and then saving them into csv formats

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_pdf import PdfPages
import pdfkit
import seaborn as sns
```

```
xlsx_files = ("generated_before.xlsx", "generated_during.xlsx", "generated_after.xlsx")
csv_files = ("before.csv", "during.csv", "after.csv")

for i in range(len(xlsx_files)):
    df = pd.read_excel("data/" + xlsx_files[i])
```

```
    df.to_csv("data/" + csv_files[i], index=False)

print("Saved as csv formats!")
```

```
Saved as csv formats!
```

```
before_df = pd.read_csv('data/before.csv')
during_df = pd.read_csv('data/during.csv')
after_df = pd.read_csv('data/after.csv')
```

Splitting the data of each and every `subject` according to incremental values of `Sl No` column

```
def split_dataframe_by_increment(df, column_name = "Sl No"):
    split_indices = [0]
    sl_no_values = df[column_name].fillna(method='ffill').values  # Fill NaN values
    for i in range(1, len(sl_no_values)):
        if sl_no_values[i] > sl_no_values[i - 1]:
            split_indices.append(i)
    split_indices.append(len(df))

    return [df.iloc[split_indices[j]:split_indices[j+1]] for j in range(len(split_indices) -

before_chant_subject_dfs = split_dataframe_by_increment(before_df)
during_chant_subject_dfs = split_dataframe_by_increment(during_df)
after_chant_subject_dfs = split_dataframe_by_increment(after_df)

print(len(after_chant_subject_dfs))

before_chant_subject_dfs[0].head(10)
```

```
32
```

```
/var/folders/6x/c19tq81j2954h_g4n73r3lp40000gn/T/ipykernel_66103/2842426926.py:3: FutureWarn
  sl_no_values = df[column_name].fillna(method='ffill').values  # Fill NaN values
```

	Sl No	Subject ID	Name	Age	Gender	PhoneNumber	\
0	1.0	1_RajeshPanda_46_M	Rajesh Panda	46.0	Male	9.849274e+09	
1	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	NaN	
6	NaN	NaN	NaN	NaN	NaN	NaN	
7	NaN	NaN	NaN	NaN	NaN	NaN	
8	NaN	NaN	NaN	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	NaN	

```
                    Email               Occupation  \
```

0	rajeshpanda123@gmail.com	Founder: Fintech Startup
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN

	HKM Mantra Chanting streak (in years)	Session start time	...
0	2.5	14:24:53	...
1	NaN	NaN	...
2	NaN	NaN	...
3	NaN	NaN	...
4	NaN	NaN	...
5	NaN	NaN	...
6	NaN	NaN	...
7	NaN	NaN	...
8	NaN	NaN	...
9	NaN	NaN	...

	Baseline Relaxation index	Relaxation index	Theta peak frequency
0	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000
4	0.000000	0.000000	4.562830
5	0.000000	0.000000	4.872460
6	0.000000	0.000000	5.095028
7	0.243019	0.706599	4.607608
8	0.549890	2.031018	4.782931
9	0.549890	2.885415	5.315834

	Alpha peak frequency	Beta peak frequency	Chill	Stress	Focus
0	0.000000	0.000000	NaN	NaN	NaN
1	0.000000	0.000000	NaN	NaN	NaN
2	0.000000	0.000000	NaN	NaN	NaN
3	0.000000	0.000000	NaN	NaN	NaN
4	7.158896	15.327831	NaN	NaN	NaN
5	8.090403	15.491718	NaN	NaN	NaN
6	7.523340	16.930598	NaN	NaN	NaN
7	7.199132	15.970395	65.605411	44.334343	47.773791
8	7.849992	15.860481	74.506322	25.652510	29.231659
9	8.958558	17.519024	93.475621	3.019546	7.012908

```
       Anger  Self-control
0        NaN           NaN
1        NaN           NaN
2        NaN           NaN
3        NaN           NaN
4        NaN           NaN
5        NaN           NaN
6        NaN           NaN
7  46.585155     60.695328
8  23.845453     64.172992
9  19.107382     88.160316

[10 rows x 29 columns]
```

```
parameters_to_plot = [
    "IAPF", "Baseline Fatigue score", #"Fatigue score", "Baseline Alpha Gravity",
    #"Alpha Gravity",
    "Baseline Concentration index", "Concentration index",
    "Baseline Relaxation index", #"Relaxation index",
    "Theta peak frequency",
    "Alpha peak frequency", "Beta peak frequency", "Chill", "Stress",
    "Focus", "Anger", "Self-control"
]
```

There are two plans to analyse the data as dicussed:

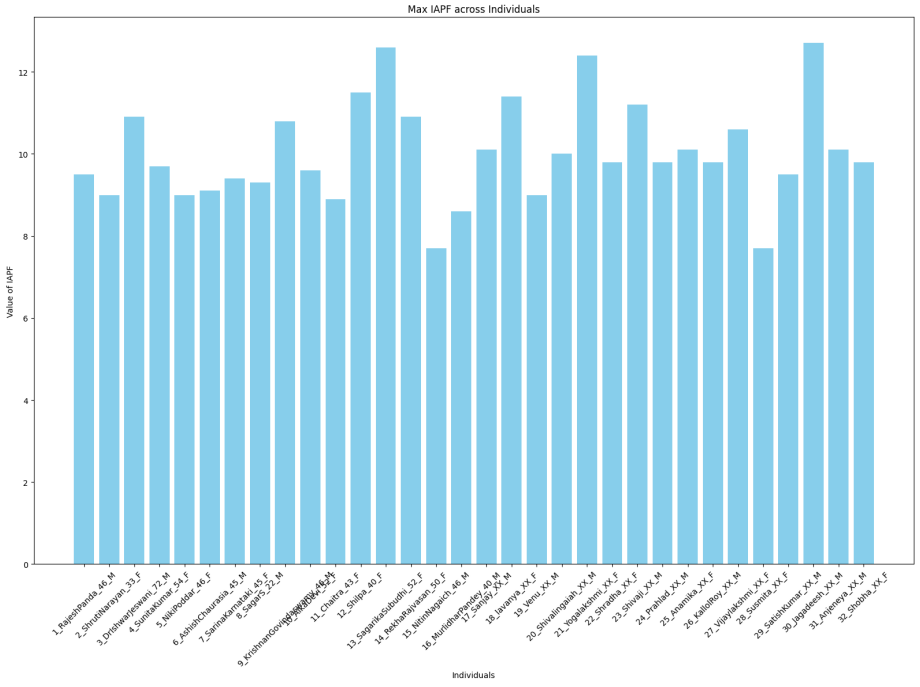
1. Analyse each subject's particular `parameter of interest` one by one, for this use the below written function: `plot_subjects_vs_parameter(..)`
2. Analyse all subject's `before`, `during` and `after` data all at once using a `triple bar plot`

The below cell does helps implement the first point above.

```
def plot_subjects_vs_parameter(chant_type_df=before_chant_subject_dfs, parameter_name = "IAF
    individuals = []
    parameter_values = []

    for idx, df in enumerate(chant_type_df):

        individual_id = df.iloc[0]['Subject ID']
        individuals.append(individual_id)

        parameter_value = df[parameter_name].median()
        parameter_values.append(parameter_value)

    print("individuals: ", individuals)
    print("Parameter values: ", parameter_values)
```

```
    # Create the bar plot
    plt.figure(figsize=(16, 12))
    plt.bar(individuals, parameter_values, color='skyblue')
    plt.xlabel('Individuals')
    plt.ylabel(f"Value of {parameter_name}")
    plt.title(f'Max {parameter_name} across Individuals')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

plot_subjects_vs_parameter(before_chant_subject_dfs, parameter_name="IAPF")
```

```
individuals:  ['1_RajeshPanda_46_M', '2_ShrutiNarayan_33_F', '3_DrIshwarJeswani_72_M', '4_Su
Parameter values:  [9.5, 9.0, 10.8999996185303, 9.69999980926514, 9.0, 9.10000038146973, 9.3
```



This following cell helps implement the second point:

```
plot_files = []
html_content = ""

def plot_subjects_vs_parameter(before_chant_subject_dfs, during_chant_subject_dfs, after_cha
    individuals = []
    before_values = []
    during_values = []
```

```
    after_values = []

    for idx, df in enumerate(before_chant_subject_dfs):
        individual_id = df.iloc[0]['Subject ID']
        individuals.append(individual_id)
        before_values.append(df[parameter_name].max())

    for idx, df in enumerate(during_chant_subject_dfs):
        during_values.append(df[parameter_name].max())

    for idx, df in enumerate(after_chant_subject_dfs):
        after_values.append(df[parameter_name].max())

    x = np.arange(len(individuals))
    width = 0.25

    fig, ax = plt.subplots(figsize=(16, 12))

    rects1 = ax.bar(x - width, before_values, width, label='Before Chant', color='skyblue')
    rects2 = ax.bar(x, during_values, width, label='During Chant', color='lightgreen')
    rects3 = ax.bar(x + width, after_values, width, label='After Chant', color='lightcoral')

    ax.set_xlabel('Individuals')
    ax.set_ylabel(f'Value of {parameter_name}')
    ax.set_title(f'Max {parameter_name} across Individuals')

    ax.set_xticks(x)
    ax.set_xticklabels(individuals, rotation=45)
    ax.legend()

    ax.text(0.5, 1.1, description, transform=ax.transAxes, ha='center')

    plt.tight_layout()
    plt.show()

    return fig
```

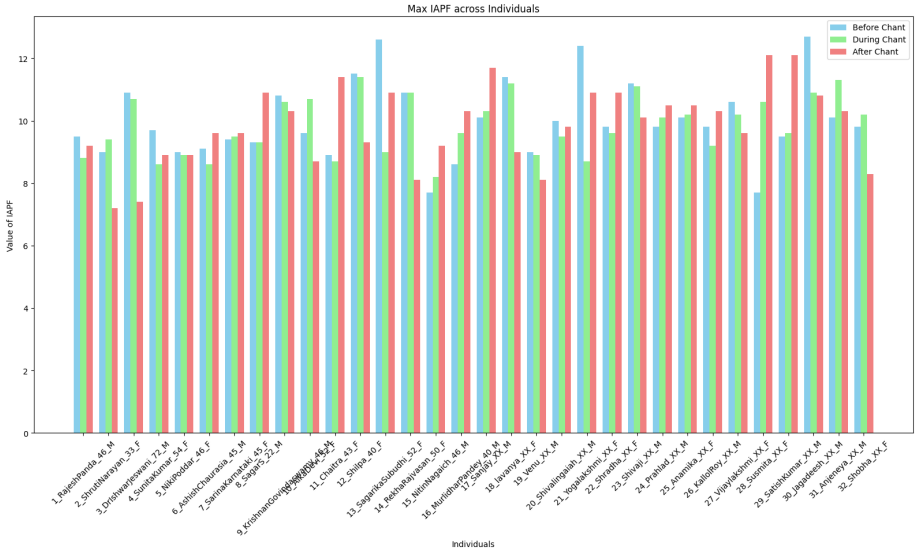
Along with each plot, the following code helps with `descriptive` statistics for each of the parameter. Then saves all the plots to a `plots.pdf` file

```
def compare_parameter_statistics(param_name, before_df, during_df, after_df):
    before_values = []
    during_values = []
    after_values = []

    for subject_before_df in before_df:
        before_values.extend(subject_before_df[param_name])
```

```
    for subject_during_df in during_df:
        during_values.extend(subject_during_df[param_name])

    for subject_after_df in after_df:
        after_values.extend(subject_after_df[param_name])

    before_stats = pd.Series(before_values).describe()
    during_stats = pd.Series(during_values).describe()
    after_stats = pd.Series(after_values).describe()

    comparison_df = pd.DataFrame({
        'Before Chant': before_stats,
        'During Chant': during_stats,
        'After Chant': after_stats
    })

    return comparison_df

pdf_filename = "plots.pdf"
pdf_pages = PdfPages(pdf_filename)
comparison_statistics_across_parameters = {}

for param_name in parameters_to_plot:
    comparison_statistics_across_parameters[param_name] = compare_parameter_statistics(paran
    statistics_df = comparison_statistics_across_parameters[param_name]
    description = f"Statistics for {param_name}:\n{statistics_df.to_string()}"
    fig = plot_subjects_vs_parameter(before_chant_subject_dfs, during_chant_subject_dfs, aft
    pdf_pages.savefig(fig)
    plt.close(fig)

pdf_pages.close()

print(f"Plots saved to {pdf_filename}")

comparison_df = pd.concat(comparison_statistics_across_parameters, axis=1)
#comparison_df.to_csv("comparison_data.csv")
```

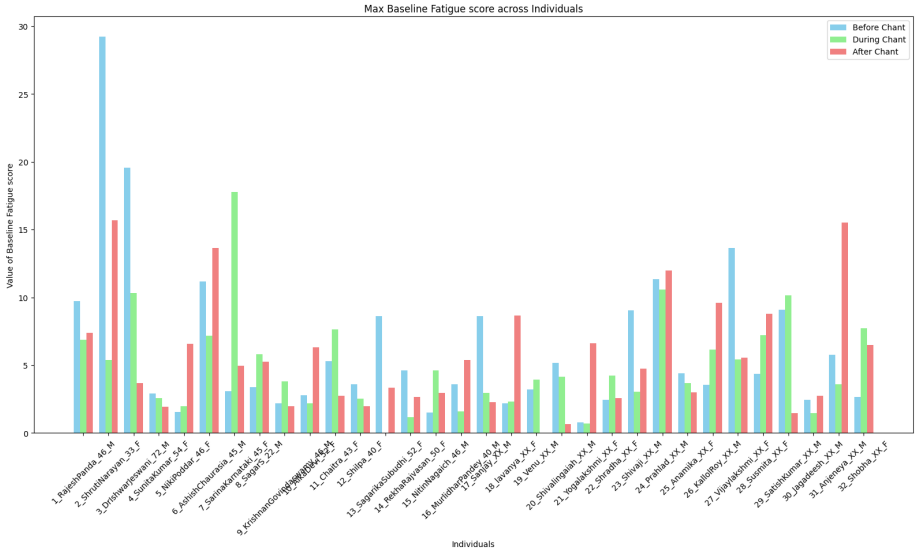
Statistics for IAPF:

	Before Chant	During Chant	After Chant
count	334.000000	595.000000	310.000000
mean	10.039222	9.821681	9.652258
std	1.236478	0.888299	1.258824
min	7.700000	8.200000	7.200000
25%	9.300000	9.000000	8.900000
50%	9.800000	9.600000	9.600000
75%	10.900000	10.600000	10.500000
max	12.700000	11.400000	12.100000



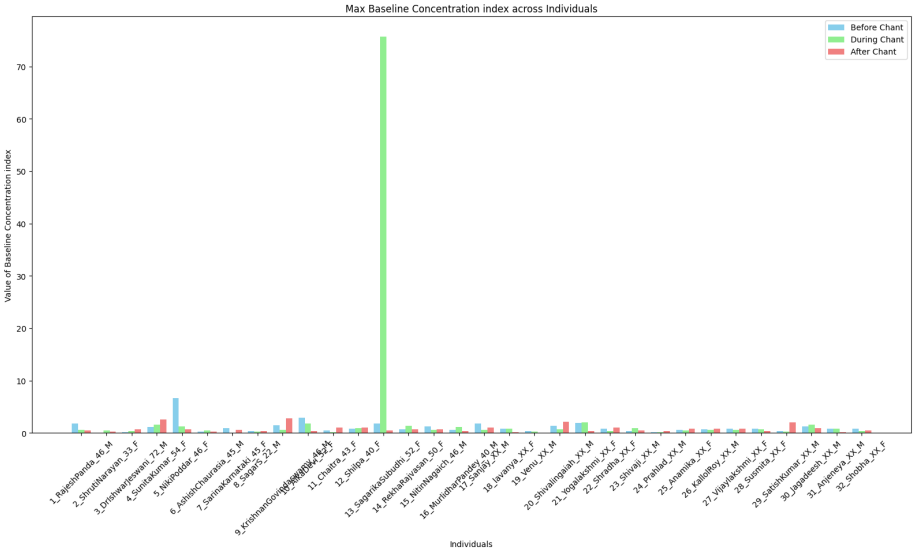
Statistics for Baseline Fatigue score:

	Before Chant	During Chant	After Chant
count	392.000000	618.000000	336.000000
mean	4.192437	4.032939	4.257030
std	5.361431	3.658173	4.121511
min	0.000000	0.000000	0.000000
25%	0.000000	1.606002	1.263017
50%	2.914013	3.035555	2.973773
75%	5.153500	6.139206	6.303939
max	29.250799	17.782393	15.679881



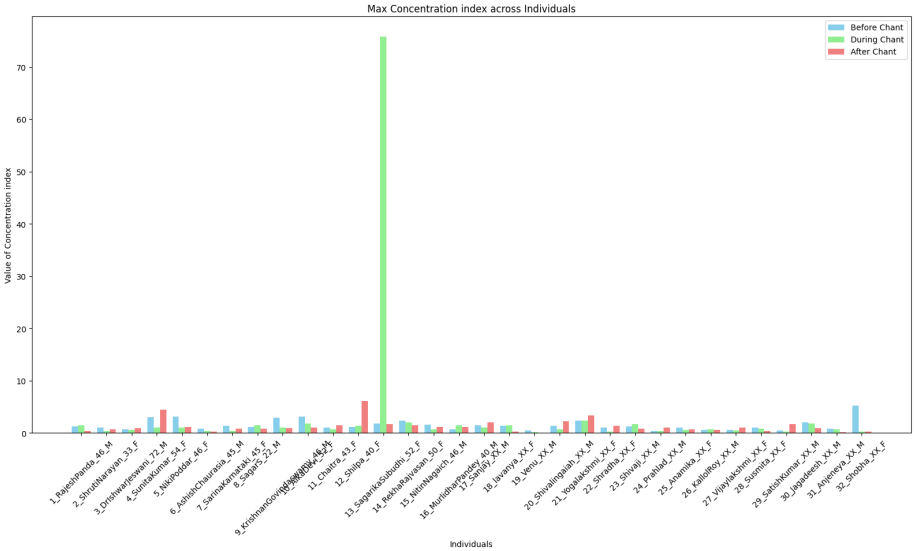
Statistics for Baseline Concentration index:

	Before Chant	During Chant	After Chant
count	392.000000	618.000000	336.000000
mean	0.737683	2.716582	0.617344
std	0.917037	12.302178	0.656463
min	0.000000	0.000000	0.000000
25%	0.000000	0.258525	0.111139
50%	0.591192	0.587758	0.451141
75%	0.889580	0.979584	0.801596
max	6.714382	75.737190	2.785440



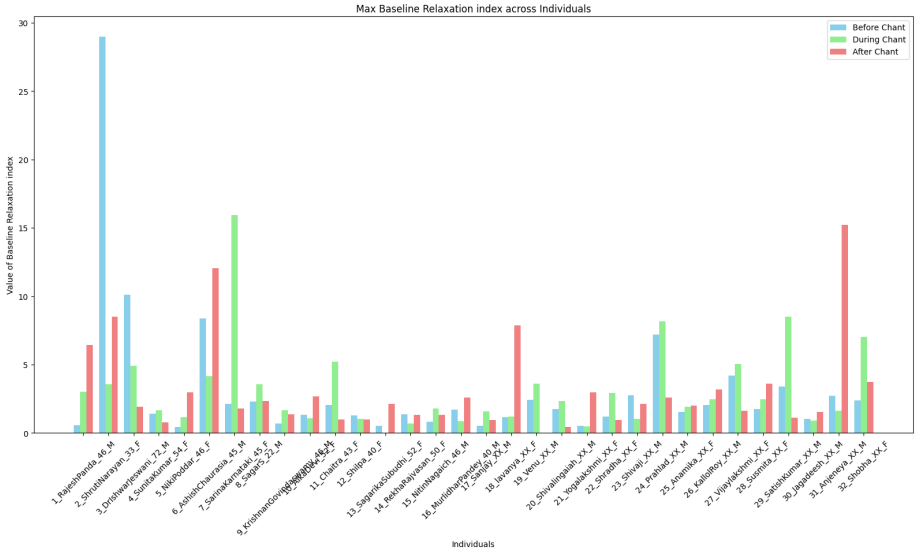
Statistics for Concentration index:

	Before Chant	During Chant	After Chant
count	392.000000	618.000000	336.000000
mean	0.722748	1.623227	0.625496
std	0.779313	8.835154	0.686651
min	0.000000	0.000000	0.000000
25%	0.000000	0.199540	0.115544
50%	0.580910	0.478858	0.534261
75%	1.045094	0.793152	0.867003
max	5.257327	75.867876	6.071821



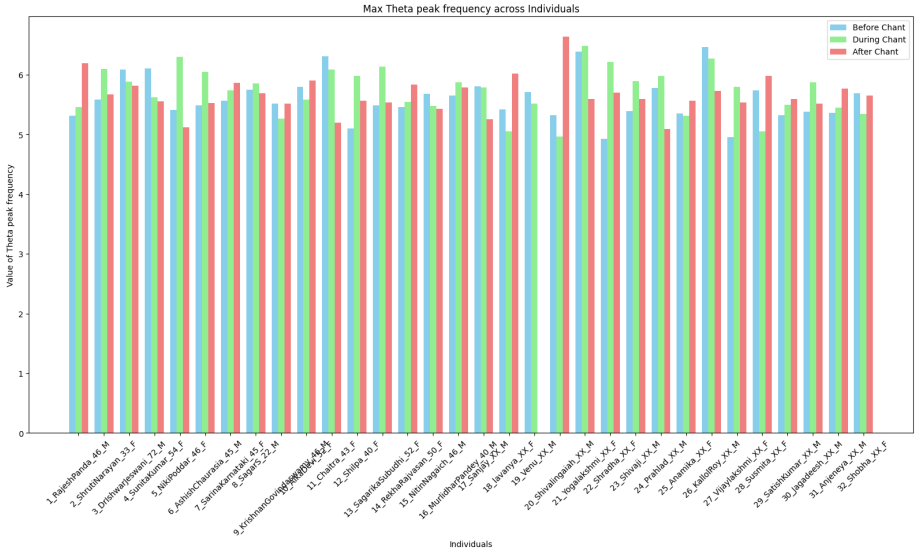
Statistics for Baseline Relaxation index:

	Before Chant	During Chant	After Chant
count	392.000000	618.000000	336.000000
mean	2.038923	2.549066	2.357484
std	4.278014	2.995885	3.056131
min	0.000000	0.000000	0.000000
25%	0.000000	0.876395	0.564075
50%	1.190451	1.642069	1.543795
75%	2.142207	2.999701	2.613924
max	29.002399	15.924937	15.222116



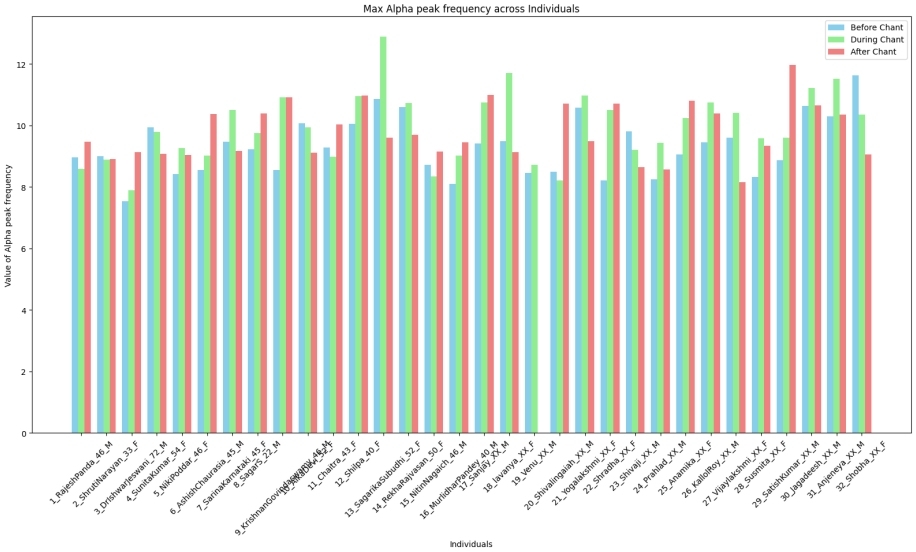
Statistics for Theta peak frequency:

	Before Chant	During Chant	After Chant
count	460.000000	682.000000	387.000000
mean	4.303540	4.752907	4.636373
std	1.787110	1.268195	1.513596
min	0.000000	0.000000	0.000000
25%	4.569895	4.715573	4.771008
50%	4.920182	5.001267	5.016770
75%	5.217192	5.321728	5.331485
max	6.466209	6.479138	6.642039



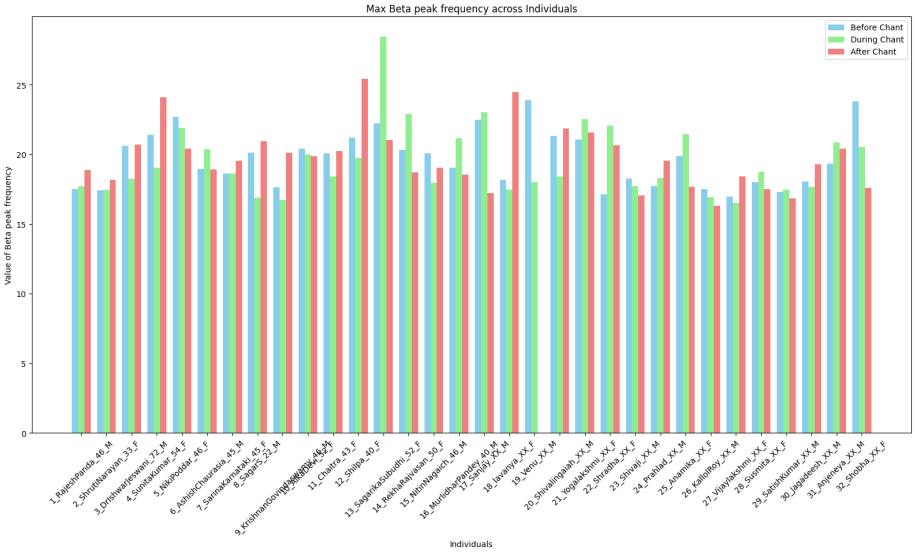
Statistics for Alpha peak frequency:

	Before Chant	During Chant	After Chant
count	460.000000	682.000000	387.000000
mean	6.946927	8.403006	7.628690
std	2.936936	2.413913	2.593925
min	0.000000	0.000000	0.000000
25%	7.130248	7.810342	7.298854
50%	7.635858	8.708757	7.976188
75%	8.468608	9.682545	9.063904
max	11.629287	12.900000	11.977126



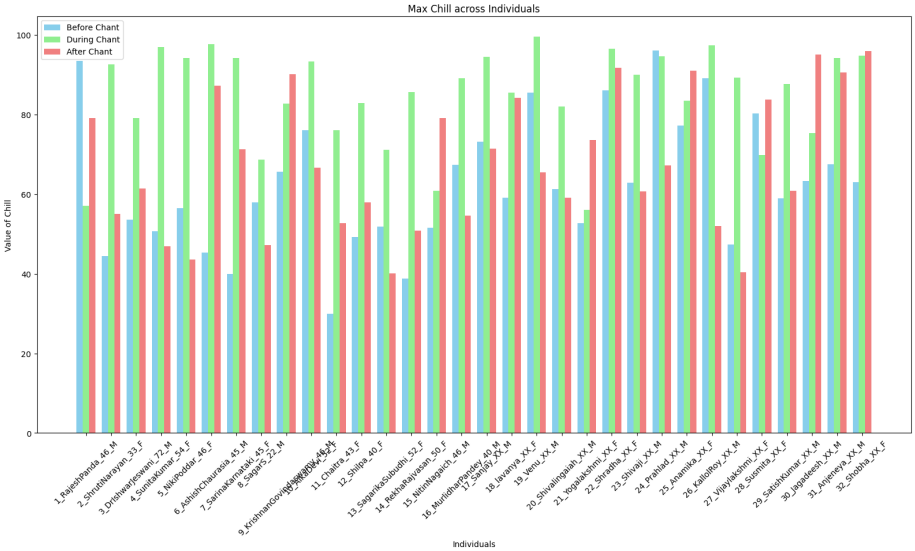
Statistics for Beta peak frequency:

	Before Chant	During Chant	After Chant
count	460.000000	682.000000	387.000000
mean	15.034728	16.466888	16.116269
std	6.268979	4.523518	5.347677
min	0.000000	0.000000	0.000000
25%	15.972246	16.194280	16.173075
50%	16.880273	16.927945	17.178912
75%	18.152009	18.181023	18.559596
max	23.903456	28.470807	25.407109



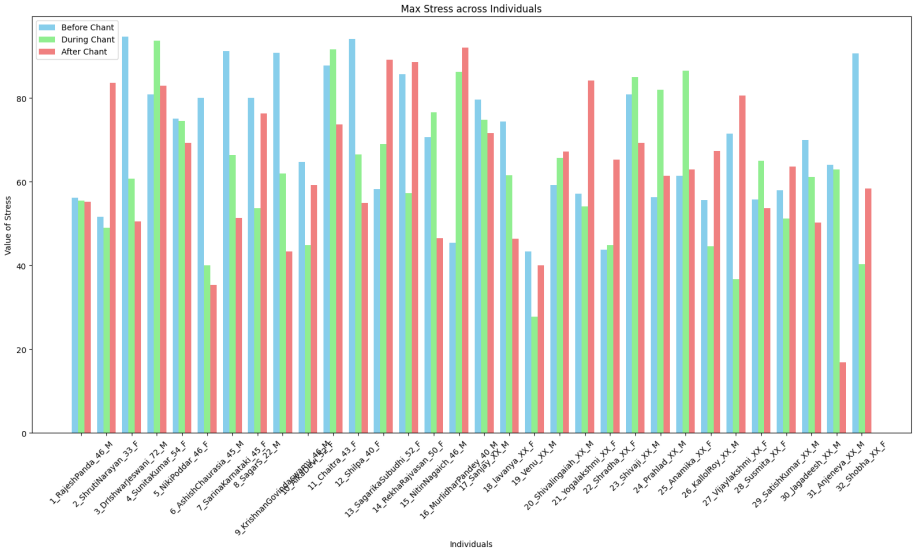
Statistics for Chill:

	Before Chant	During Chant	After Chant
count	289.000000	550.000000	270.000000
mean	46.170891	61.258701	51.356276
std	17.399539	19.702097	17.923574
min	3.227619	1.186601	17.900190
25%	34.395437	47.926035	38.765543
50%	46.177457	60.627199	47.561334
75%	57.107515	77.602774	62.003694
max	96.025777	99.613626	95.850656



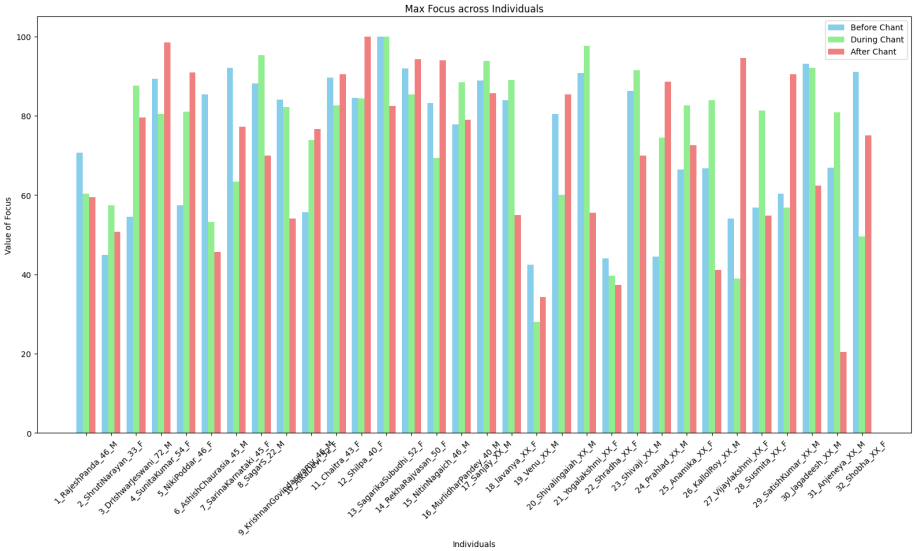
Statistics for Stress:

	Before Chant	During Chant	After Chant
count	289.000000	550.000000	270.000000
mean	53.167003	39.101402	48.363460
std	18.204966	17.653881	18.756183
min	3.019546	2.130805	3.766523
25%	41.646554	25.383382	35.810949
50%	53.670199	37.710612	47.309192
75%	65.059161	51.192149	61.144054
max	94.724699	93.633919	92.099810



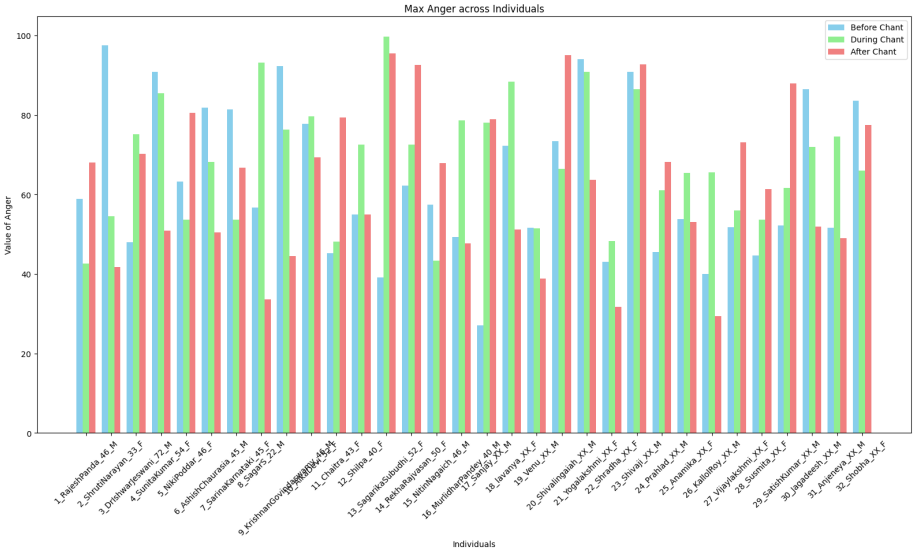
Statistics for Focus:

	Before Chant	During Chant	After Chant
count	289.000000	550.000000	270.000000
mean	53.651977	45.490437	51.297272
std	21.543044	22.782025	22.202750
min	3.060358	1.674587	6.316287
25%	38.452155	28.850424	34.308989
50%	51.499243	43.335869	50.353759
75%	68.484521	61.159837	66.244527
max	100.000000	99.873145	100.000000



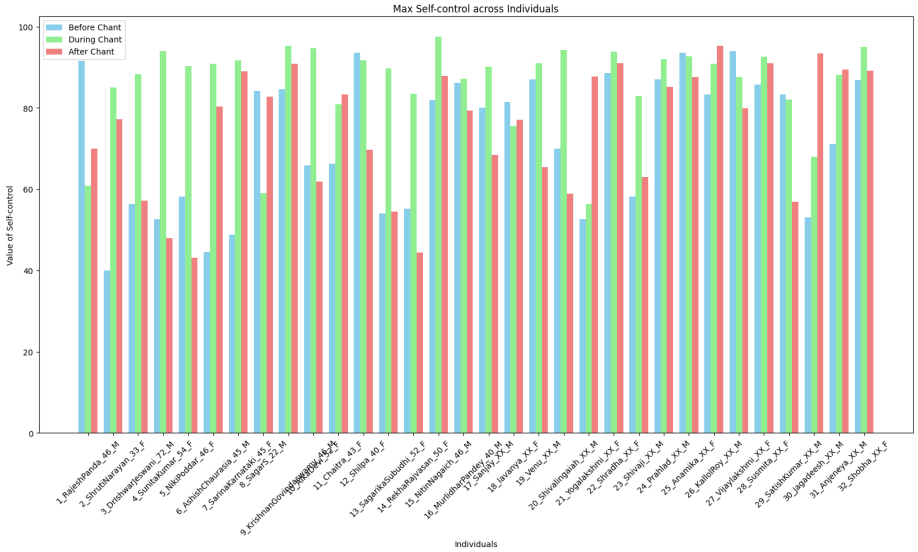
Statistics for Anger:

	Before Chant	During Chant	After Chant
count	289.000000	550.000000	270.000000
mean	43.685000	42.866209	45.595030
std	20.530652	18.822039	18.638525
min	6.177373	1.522244	7.696278
25%	27.940251	28.462784	31.038164
50%	40.404453	40.292564	44.484437
75%	56.655328	54.245156	58.350400
max	97.569615	99.777038	95.459231



Statistics for Self-control:

	Before Chant	During Chant	After Chant
count	289.000000	550.000000	270.000000
mean	52.042153	62.504224	55.253768
std	19.858987	19.665124	19.030316
min	2.381758	1.222838	13.279823
25%	37.732695	49.379835	40.142442
50%	51.964476	64.342505	54.524789
75%	65.792784	77.323400	68.686212
max	93.922688	97.602722	95.322922




```
Plots saved to plots.pdf
```

```
all_subjects_before = []
all_subjects_during = []
all_subjects_after = []

for subject in before_chant_subject_dfs:
    all_subjects_before.append(subject[parameters_to_plot].max(numeric_only=True))

for subject in during_chant_subject_dfs:
    all_subjects_during.append(subject[parameters_to_plot].max(numeric_only=True))

for subject in after_chant_subject_dfs:
    all_subjects_after.append(subject[parameters_to_plot].max(numeric_only=True))

before_mean = sum(all_subjects_before) / len(all_subjects_before)
during_mean = sum(all_subjects_during) / len(all_subjects_during)
after_mean = sum(all_subjects_after) / len(all_subjects_after)

percentage_change_during_before = ((during_mean - before_mean) / before_mean) * 100
percentage_change_after_during = ((after_mean - during_mean) / during_mean) * 100
percentage_change_after_before = ((after_mean - before_mean) / during_mean) * 100

# print("Percentage change from before to during chant:")
```

```
# print(percentage_change_during_before)

# print("\nPercentage change from during to after chant:")
# print(percentage_change_after_during)

# print("\nPercentage change from after to before chant:")
# print(percentage_change_after_before)
```

The following code gives the consolidated analysis for the HKM Data. One can easily note the percentange difference in eadch of the parameters before - during, during - after, and overall before - after percentage change (with Increase or Descrease specified)

This is the ultimate conclusion of the data which can be later plotted for a `correlation matrix` to show the relatioonship between the 3 events (before during and after) i,e, how all the parameters together are varying!

```
all_subjects_before = []
all_subjects_during = []
all_subjects_after = []

for subject in before_chant_subject_dfs:
    all_subjects_before.append(subject[parameters_to_plot].max(numeric_only=True))

for subject in during_chant_subject_dfs:
    all_subjects_during.append(subject[parameters_to_plot].max(numeric_only=True))

for subject in after_chant_subject_dfs:
    all_subjects_after.append(subject[parameters_to_plot].max(numeric_only=True))

before_mean = sum(all_subjects_before) / len(all_subjects_before)
during_mean = sum(all_subjects_during) / len(all_subjects_during)
after_mean = sum(all_subjects_after) / len(all_subjects_after)

percentage_change_during_before = ((during_mean - before_mean) / before_mean) * 100
percentage_change_after_during = ((after_mean - during_mean) / during_mean) * 100
percentage_change_after_before = ((after_mean - before_mean) / during_mean) * 100

increase_decrease_before = ["Increased" if change > 0 else "Decreased" for change in percent
increase_decrease_after = ["Increased" if change > 0 else "Decreased" for change in percenta
increase_decrease_after_before = ["Increased" if change > 0 else "Decreased" for change in p

description_before = [f"{abs(change):.2f} %" for change in percentage_change_during_before]
description_after = [f"{abs(change):.2f} %" for change in percentage_change_after_during]
description_after_before = [f"{abs(change):.2f} %" for change in percentage_change_after_bef

num_before = [change for change in percentage_change_during_before]
```

```
num_after = [change for change in percentage_change_after_during]
num_after_before = [change for change in percentage_change_after_before]

description_data = {
    "Parameter": parameters_to_plot,
    "% Change from Before to During Chant": [f"{increase_decrease} by {desc}" for increase_d
    "% Change from During to After Chant": [f"{increase_decrease} by {desc}" for increase_de
    "% Change from After to Before Chant": [f"{increase_decrease} by {desc}" for increase_de
}

numerical_data = {
       "Parameter": parameters_to_plot,
    "% Change Before-During Chant": [desc for desc in num_before],
    "% Change During-After Chant": [desc for desc in num_after],
    "% Change After-Before Chant": [desc for desc in num_after_before]
}

df_consolidated = pd.DataFrame(description_data)
df_num = pd.DataFrame(numerical_data)

df_consolidated.to_csv("consolidated_table.csv", index=False)
#df_num.to_csv("numerical_consolidated_table.csv", index=False)

print("CSV file saved successfully.")

df_consolidated.head(-1)
```

```
CSV file saved successfully.
```

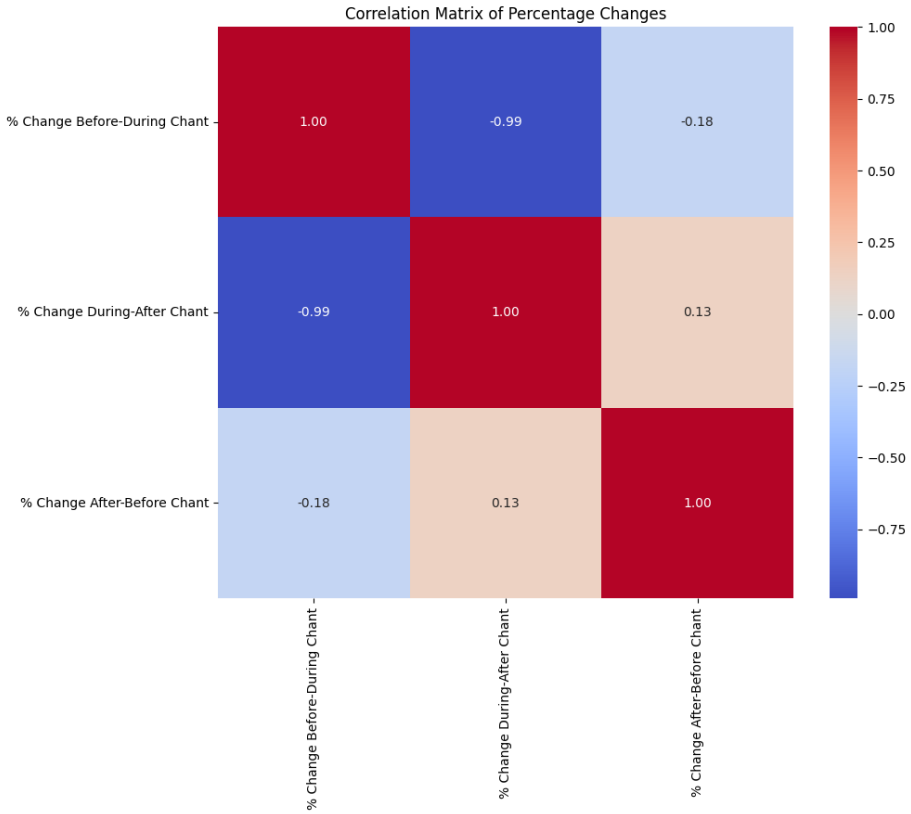
	Parameter	% Change from Before to During Chant	% Change from During to After Chant	% Change from After to Before Chant
0	IAPF	Decreased by 1.87 %	Increased by 0.13 %	Decreased by 1.78 %
1	Baseline Fatigue score	Decreased by 21.17 %	Increased by 11.50 %	Decreased by 15.36 %
2	Baseline Concentration index	Increased by 177.24 %	Decreased by 74.08 %	Decreased by 10.15 %
3	Concentration index	Increased by 115.90 %		
4	Baseline Relaxation index	Increased by 0.95 %		
5	Theta peak frequency	Increased by 2.30 %		
6	Alpha peak frequency	Increased by 6.98 %		
7	Beta peak frequency	Decreased by 0.93 %		
8	Chill	Increased by 35.94 %		
9	Stress	Decreased by 10.64 %		
10	Focus	Increased by 0.83 %		
11	Anger	Increased by 8.13 %		

```
3                 Decreased by 60.59 %                 Decreased by 6.91 %
4                  Decreased by 1.49 %                 Decreased by 0.55 %
5                  Decreased by 4.36 %                 Decreased by 2.11 %
6                  Decreased by 4.78 %                  Increased by 1.75 %
7                  Decreased by 1.97 %                 Decreased by 2.91 %
8                 Decreased by 20.14 %                  Increased by 6.30 %
9                  Increased by 0.97 %                Decreased by 10.94 %
10                 Decreased by 5.02 %                 Decreased by 4.20 %
11                 Decreased by 7.57 %                 Decreased by 0.05 %
```

Establishing correlation between different parameters using `correlation matrix`.

> The correlation being computed here takes into account all the parameter values together so that we see amount of net increase or net decrease between the before during and after events indicated in the `heatmap`

```
df_corr = df_num.drop(columns=['Parameter'])

correlation_matrix = df_corr.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0, fmt=".2f")
plt.title("Correlation Matrix of Percentage Changes")
plt.show()
```



Conclusion:

XXX