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

2024-06-05

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!")

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

```
Saved as csv formats!
```

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(before_chant_subject_dfs))

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

```
16
```

	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"
]
```

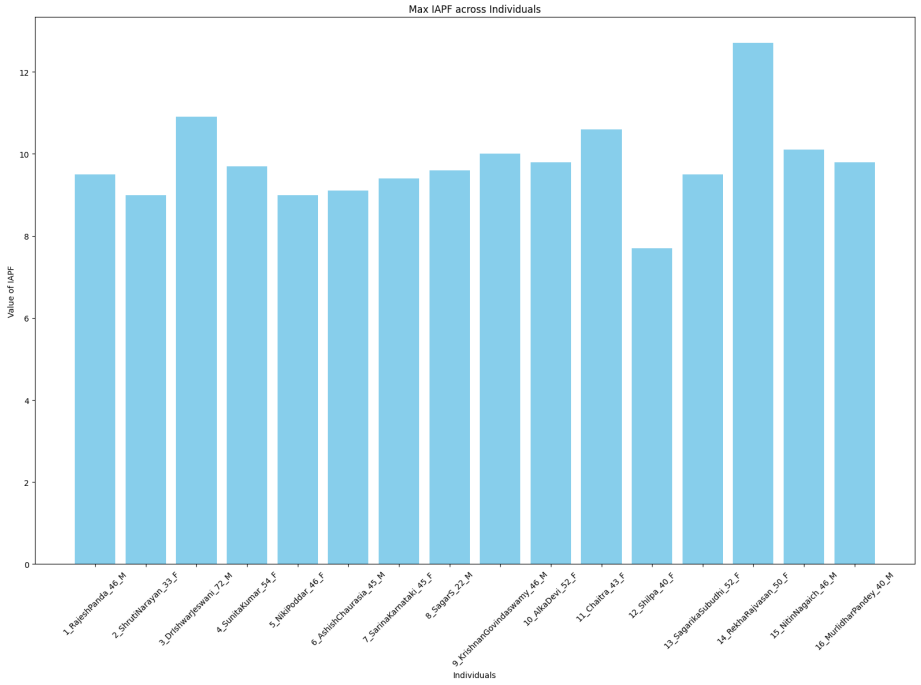
There are two plans to analyse the data as dicussed:

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].max()
        parameter_values.append(parameter_value)

    # 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")
```



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
```

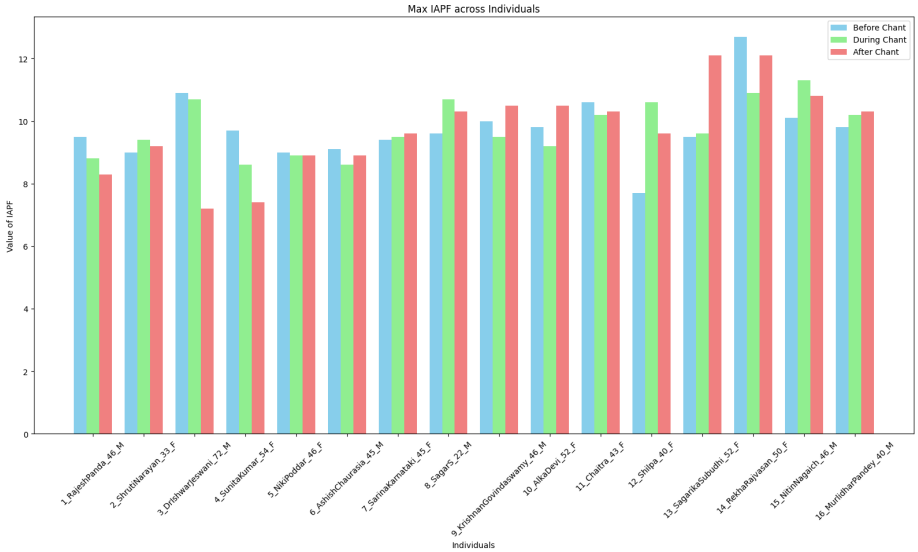
Along with each plot, the following code helps with `descriptive` statistics for each of the parameter. Then saves all the plots to a `plots.pdf` file

```
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")
```

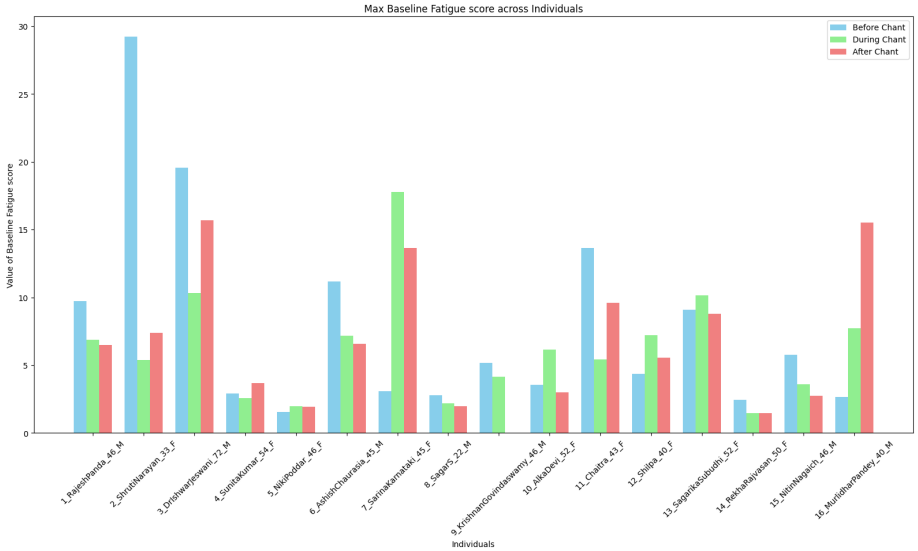
Statistics for IAPF:

	Before Chant	During Chant	After Chant
count	161.000000	279.000000	148.000000
mean	9.781988	9.732616	9.389865
std	1.067818	0.871229	1.390186
min	7.700000	8.600000	7.200000
25%	9.400000	8.900000	8.300000
50%	9.700000	9.500000	9.600000
75%	10.000000	10.600000	10.300000
max	12.700000	11.300000	12.100000



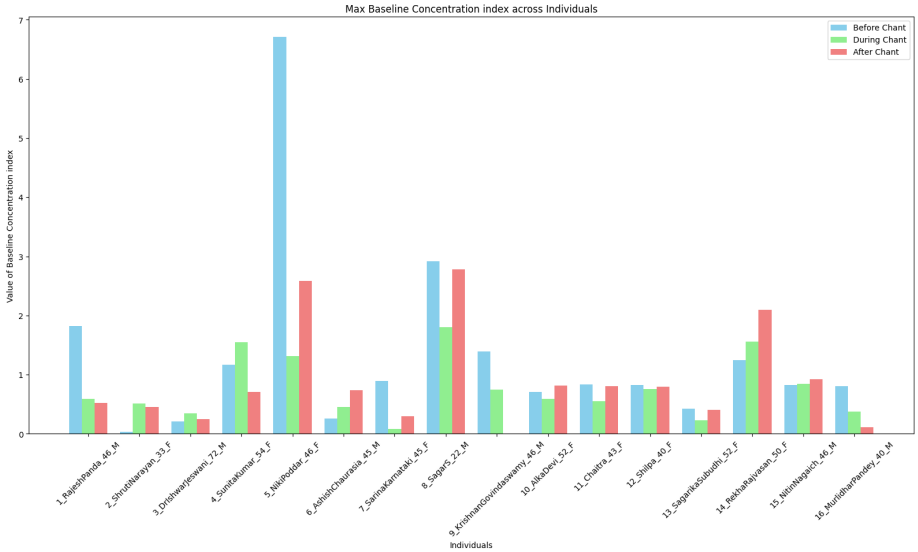
Statistics for Baseline Fatigue score:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	3.956803	4.361103	3.975602
std	6.409573	4.249902	4.875250
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	2.272455	3.594677	1.972323
75%	4.382785	6.865180	6.595374
max	29.250799	17.782393	15.679881



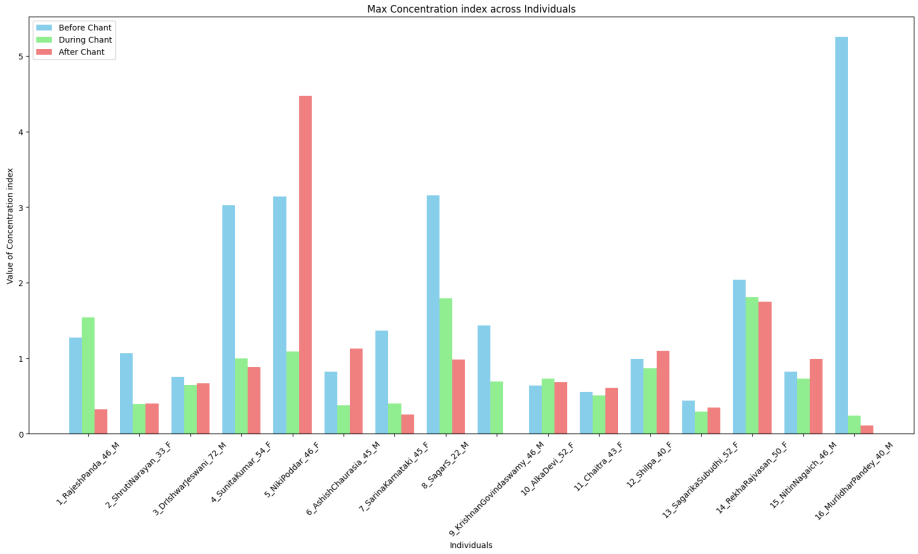
Statistics for Baseline Concentration index:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	0.629958	0.584415	0.569545
std	1.067518	0.554828	0.779431
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.208151	0.508997	0.295421
75%	0.834420	0.786671	0.796627
max	6.714382	1.807936	2.785440



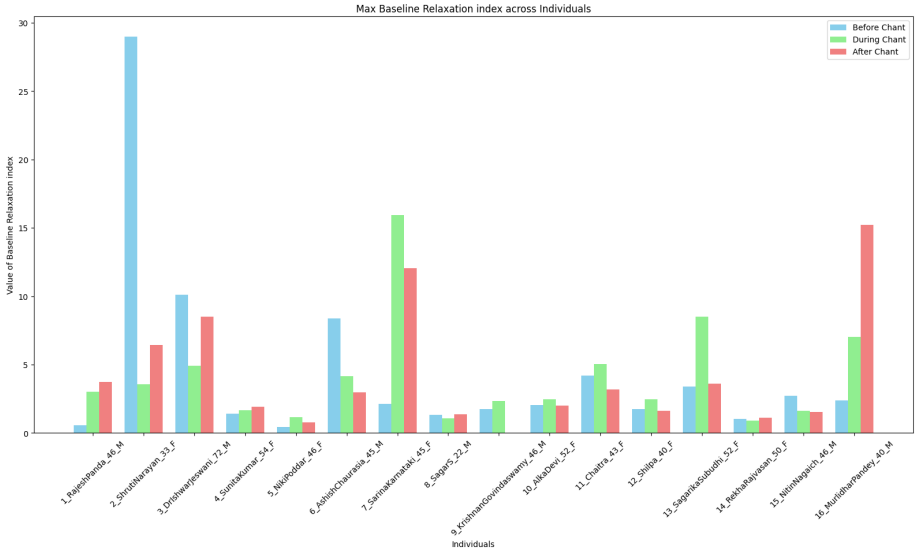
Statistics for Concentration index:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	0.560675	0.389630	0.391367
std	0.844234	0.389335	0.576515
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.246994	0.319203	0.250144
75%	0.824488	0.568499	0.608512
max	5.257327	1.806919	4.469898



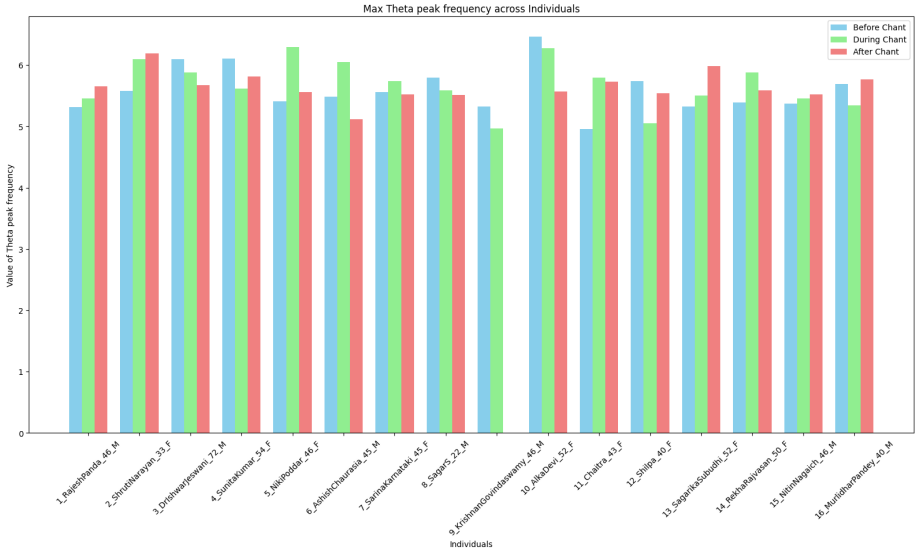
Statistics for Baseline Relaxation index:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	2.223890	2.773372	2.503410
std	5.303663	3.512386	3.758487
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.549890	1.669551	1.390875
75%	2.120126	3.571342	3.168781
max	29.002399	15.924937	15.222116



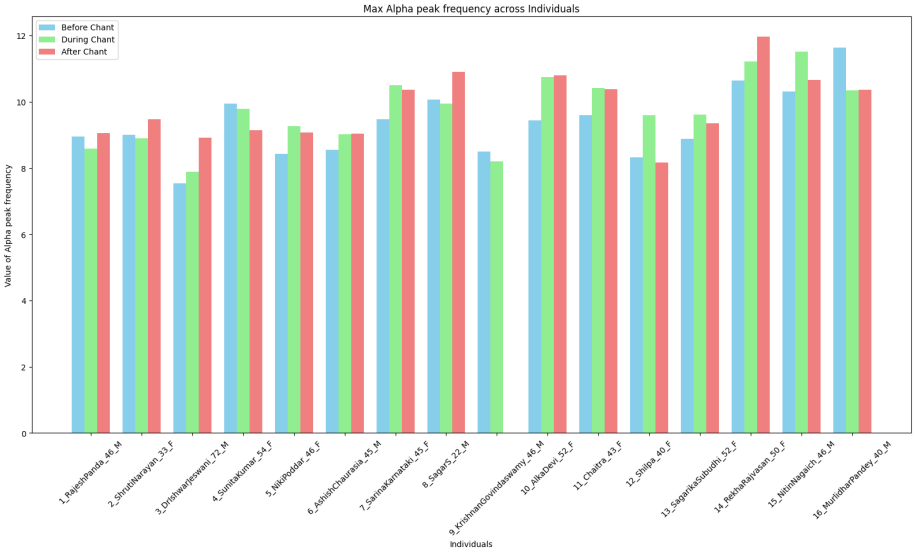
Statistics for Theta peak frequency:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	3.658075	4.374556	4.154471
std	2.235676	1.663213	1.974239
min	0.000000	0.000000	0.000000
25%	0.150036	4.587769	4.476290
50%	4.842435	4.900600	4.974892
75%	5.152733	5.176868	5.293926
max	6.466209	6.294402	6.189372



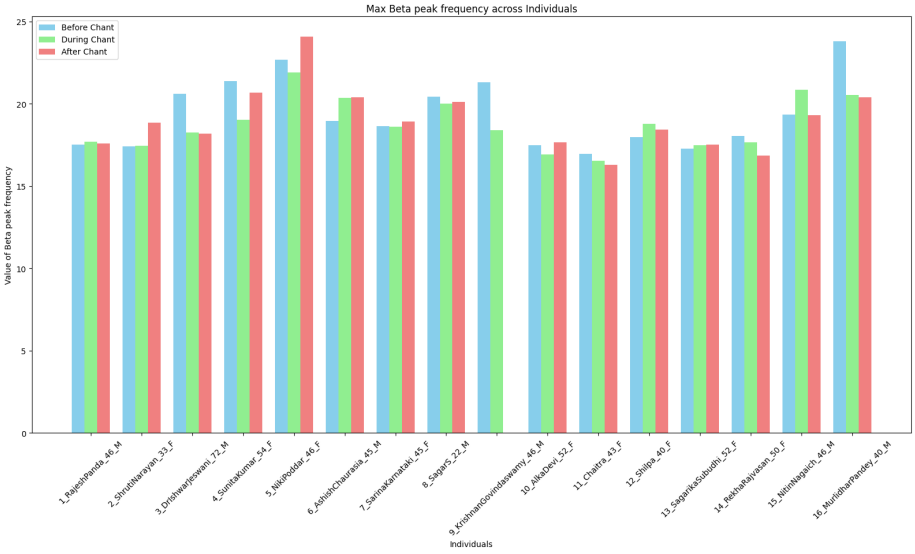
Statistics for Alpha peak frequency:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	6.026860	7.793896	7.103000
std	3.732415	3.048527	3.465148
min	0.000000	0.000000	0.000000
25%	0.271539	7.594431	7.101011
50%	7.550938	8.601785	8.062426
75%	8.582587	9.541385	9.423515
max	11.629287	11.514610	11.977126



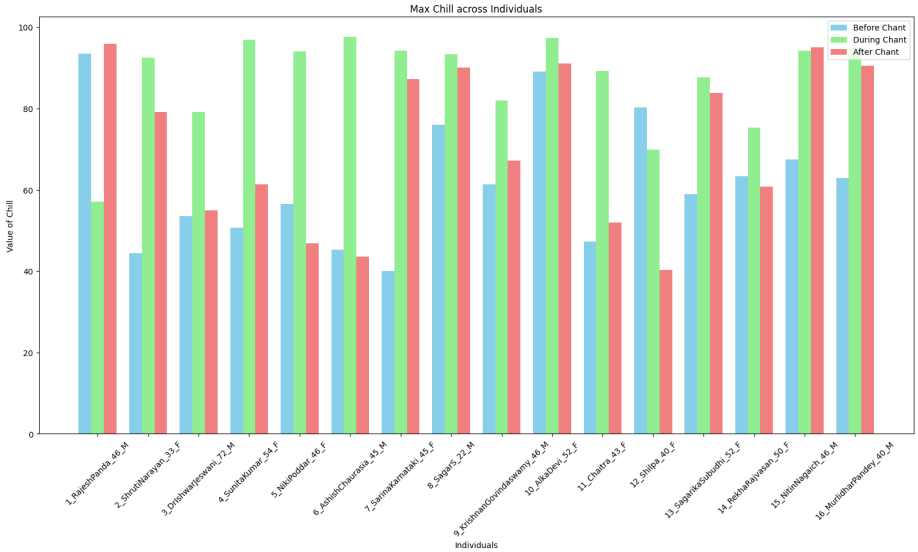
Statistics for Beta peak frequency:

	Before Chant	During Chant	After Chant
count	242.000000	336.000000	194.000000
mean	12.706238	15.145312	14.336154
std	7.821710	5.701655	6.876393
min	0.000000	0.000000	0.000000
25%	0.788237	15.844280	15.370041
50%	16.435954	16.756381	16.588946
75%	17.738460	18.027987	18.259240
max	23.817579	21.916888	24.099997



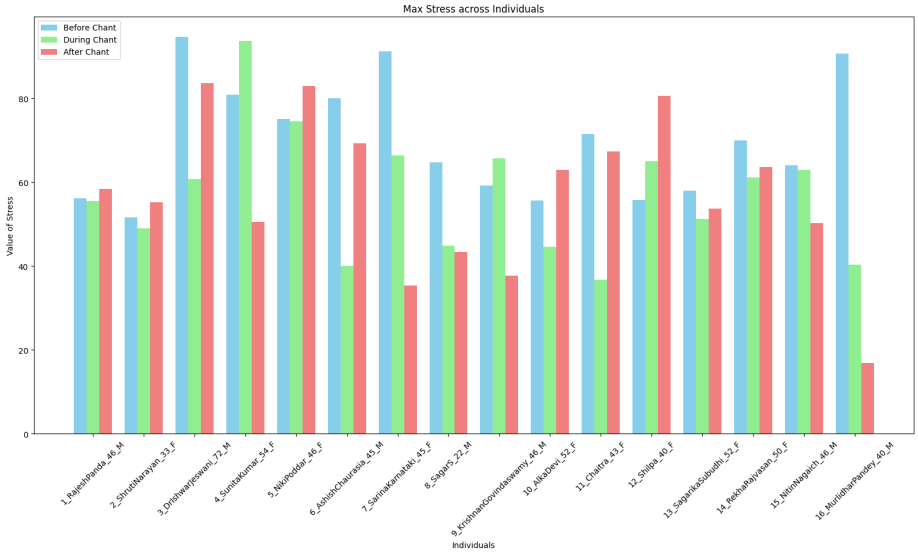
Statistics for Chill:

	Before Chant	During Chant	After Chant
count	136.000000	253.000000	125.000000
mean	46.132445	66.303961	56.205328
std	18.421590	16.456875	19.166030
min	3.227619	16.366082	20.663205
25%	34.537611	52.364497	42.610160
50%	45.858957	65.935385	50.622851
75%	57.180996	79.218613	71.065479
max	93.475621	97.677193	95.850656



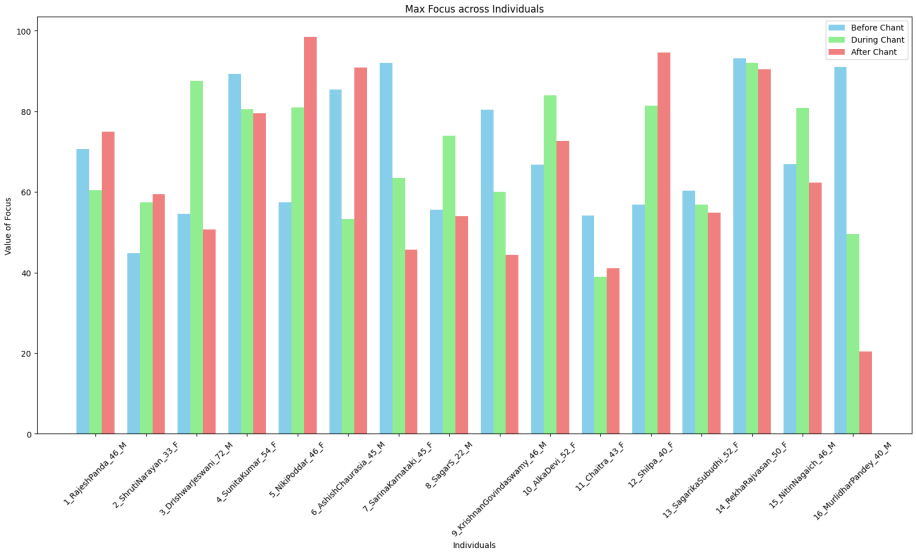
Statistics for Stress:

	Before Chant	During Chant	After Chant
count	136.000000	253.000000	125.000000
mean	51.279063	34.849934	45.501546
std	18.454142	16.053560	19.857996
min	3.019546	2.130805	3.766523
25%	40.712161	22.341565	30.379677
50%	53.572582	33.218662	46.098858
75%	62.696089	46.655828	59.275473
max	94.724699	93.633919	83.680188



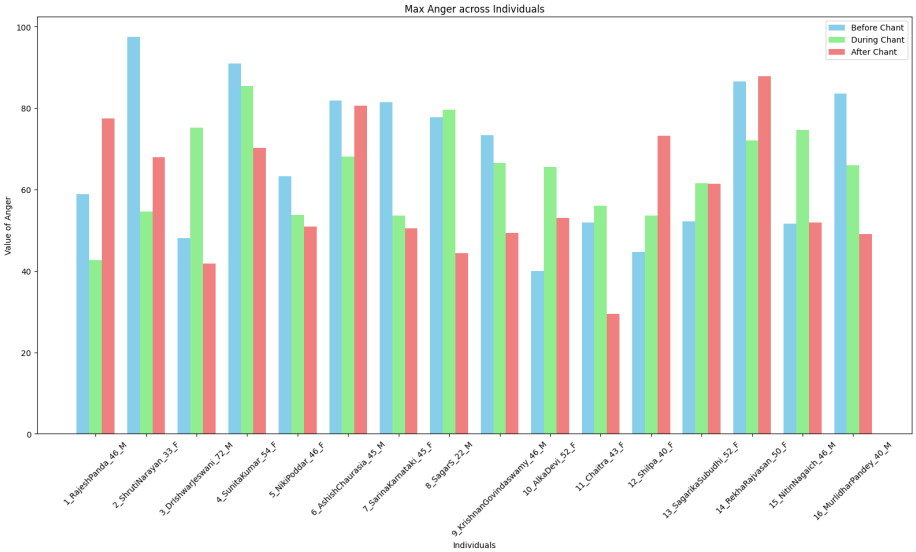
Statistics for Focus:

	Before Chant	During Chant	After Chant
count	136.000000	253.000000	125.000000
mean	51.696496	41.663471	46.736757
std	19.074890	20.521392	22.447643
min	7.012908	2.686942	6.333092
25%	39.542581	26.783556	30.623245
50%	50.691625	39.899692	45.628403
75%	63.307310	56.247078	61.013695
max	93.127893	92.029320	98.537728

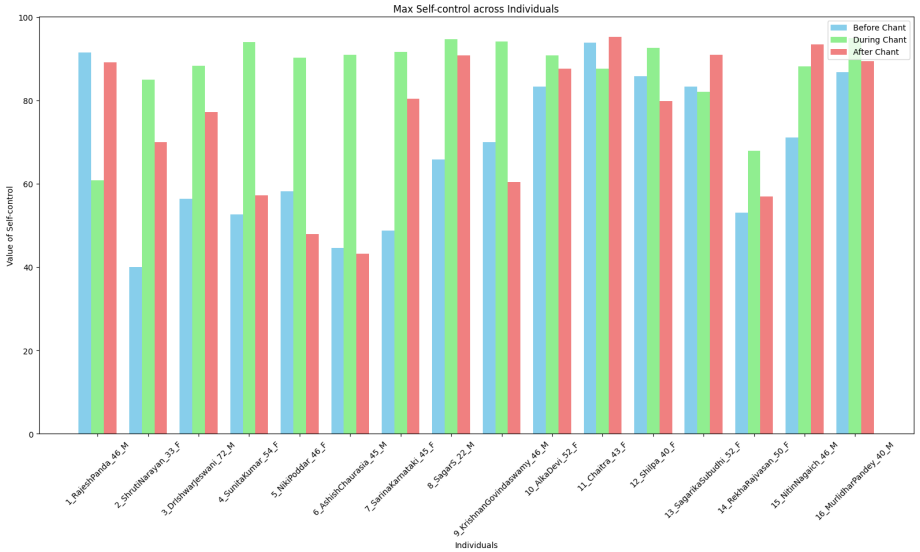


Statistics for Anger:

	Before Chant	During Chant	After Chant
count	136.000000	253.000000	125.000000
mean	48.915375	41.001529	43.137163
std	20.978398	15.655323	18.008514
min	6.501433	13.766655	7.696278
25%	34.003535	27.800049	28.775464
50%	48.190759	39.922180	41.165990
75%	62.871837	52.646343	55.080551
max	97.569615	85.436250	87.881216



```
Statistics for Self-control:
       Before Chant  During Chant  After Chant
count    136.000000    253.000000   125.000000
mean      49.121029     65.164421    58.912607
std       20.783841     16.539209    19.017216
min        2.381758     21.310864    21.983285
25%       35.031291     52.140358    43.295834
50%       47.763248     66.298207    57.963270
75%       63.831797     78.150202    74.936498
max       93.922688     94.977206    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_c
    "% 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.
```

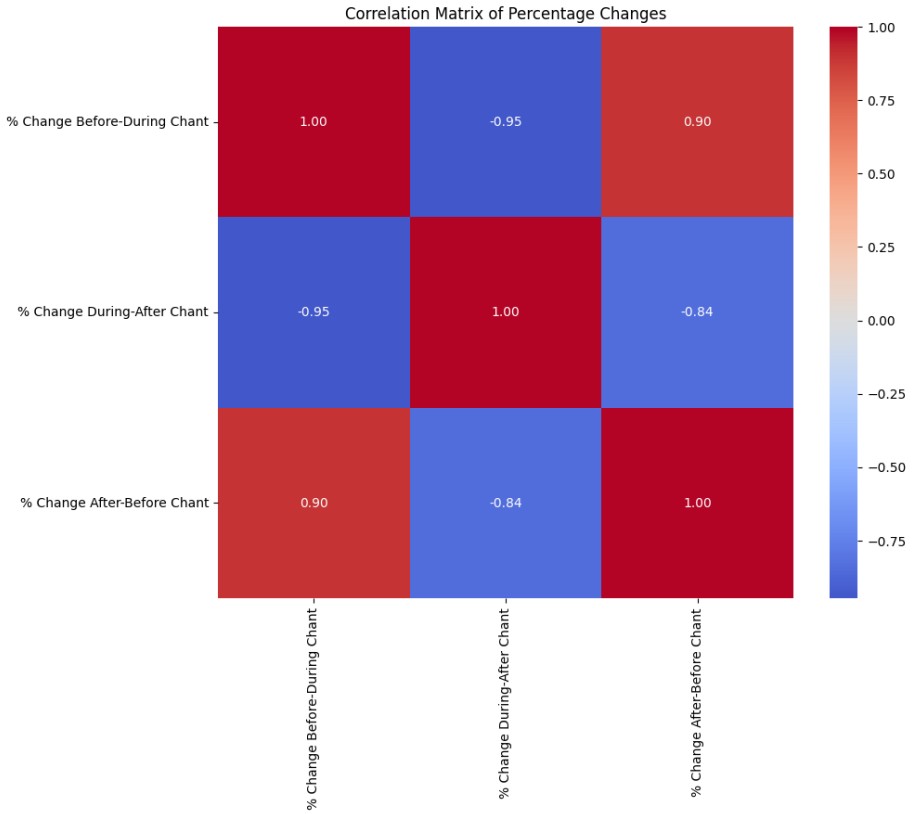
	Parameter	% Change from Before to During Chant	% Change from During to After Chant	% Change from After to Before Chant
0	IAPF	Increased by 0.19 %	Decreased by 0.45 %	Decreased by 0.26 %
1	Baseline Fatigue score	Decreased by 20.96 %	Increased by 3.85 %	Decreased by 22.66 %
2	Baseline Concentration index	Decreased by 41.64 %	Increased by 16.12 %	Decreased by 55.25 %
3	Concentration index	Decreased by 51.05 %		
4	Baseline Relaxation index	Decreased by 9.32 %		
5	Theta peak frequency	Increased by 1.55 %		
6	Alpha peak frequency	Increased by 4.20 %		
7	Beta peak frequency	Decreased by 3.00 %		
8	Chill	Increased by 40.83 %		
9	Stress	Decreased by 18.49 %		
10	Focus	Decreased by 1.63 %		
11	Anger	Decreased by 5.03 %		

```
3                 Increased by 12.20 %                Decreased by 92.11 %
4                  Increased by 0.36 %                 Decreased by 9.93 %
5                  Decreased by 6.88 %                 Decreased by 5.35 %
6                  Decreased by 5.07 %                 Decreased by 1.03 %
7                  Decreased by 5.06 %                 Decreased by 8.16 %
8                 Decreased by 18.30 %                Increased by 10.69 %
9                  Decreased by 0.05 %                Decreased by 22.74 %
10                 Decreased by 6.07 %                 Decreased by 7.73 %
11                 Decreased by 8.73 %                Decreased by 14.03 %
```

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:

The correlation clearly shows strong relationship between the events either positive and negative depending on the situation.

> Not all the parameters neccessarily show increase after chant for example decrease in `anger` after chant can lead to negative correlation which is also a good sign! Hence the experiment was a success!