

Hearing Well-being Analysis - Main Analysis file

Note: Generative AI was used mainly as a faster form of Googling

Ying Ki

2026-02-23

1 -Analysing Headphone Usage by Age

```
age_order = ['Under 18', '18 - 24', '25 - 34', '35 - 44', '45 - 54', '55 - 64']
headphone_usage_order = ['Less than 1 hour', '1-2 hours', '2-4 hours', 'More than 4 hours']

hearingcleaned['Age_group'] = pd.Categorical(
    hearingcleaned['Age_group'],
    categories = age_order,
    ordered = True
)

hearingcleaned['Daily_Headphone_Use'] = pd.Categorical(
    hearingcleaned['Daily_Headphone_Use'],
    categories = headphone_usage_order,
    ordered = True
)

#there's 3 cells with erroneous data for this column
#those cell will become NaNs they don't fall into the categories provided
ageheadphoneplot = hearingcleaned.dropna(subset = ['Daily_Headphone_Use'])
#Make a temporary dataframe that drops the NaN

ageheadphonect = pd.crosstab(ageheadphoneplot['Age_group'],
                             ageheadphoneplot['Daily_Headphone_Use'])
#crosstab is basically a pivot table (i.e. group + count + pivot)

fig, ax = plt.subplots(figsize=(8, 5), constrained_layout = True)
```

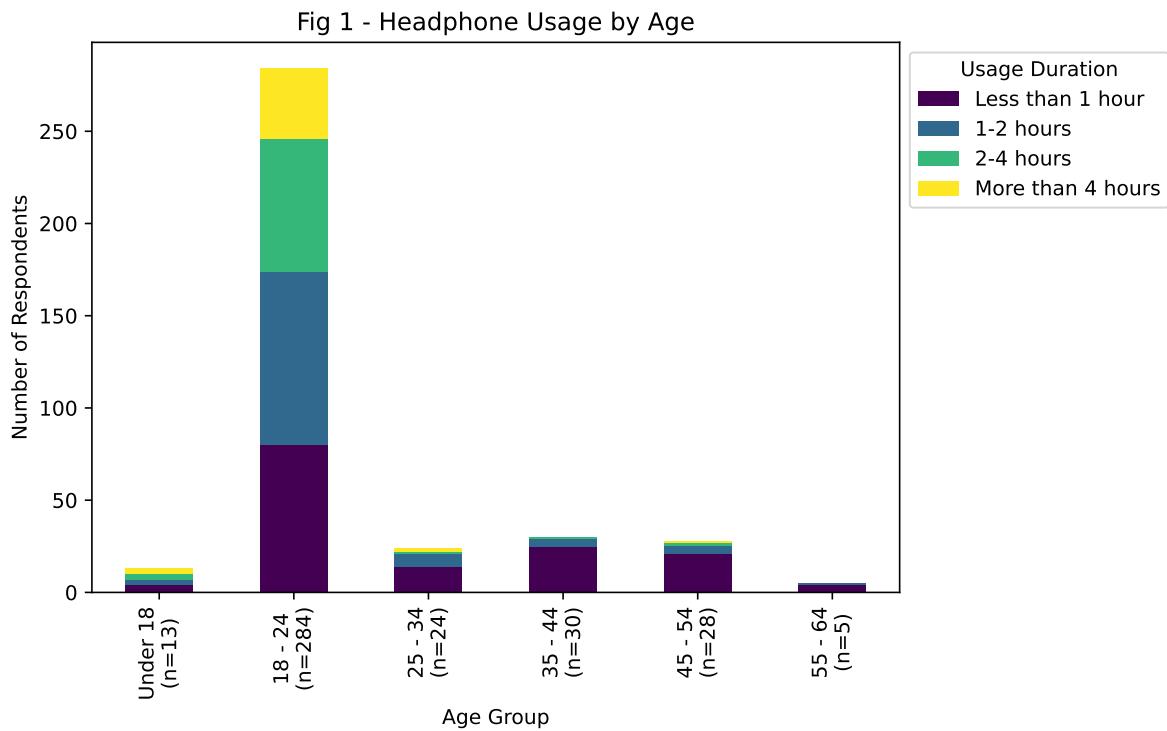
```

age_group_count = ageheadphoneplot['Age_group'].value_counts()
age_n_labels = [f'{age}\n(n={age_group_count.get(age,0)})' for age in age_order] #\n is escape character

ageheadphonect.plot(kind='bar', stacked = True, ax = ax, colormap = 'viridis')
ax.legend(title="Usage Duration", loc = 'upper left', bbox_to_anchor = (1.0, 1.0)) #top right
ax.set_xlabel("Age Group")
ax.set_ylabel("Number of Respondents")
ax.set_xticklabels(age_n_labels)
ax.set_title("Fig 1 - Headphone Usage by Age")

fig.set_constrained_layout_pads(h_pad = 0.05)
#using constrained layout and setting the padding to ensure everything is shown
plt.show()

```



From Fig.1, the '18–24' age group (hereafter referred to as young adults) demonstrates a sufficiently robust sample size ($n = 284$) to serve as the primary analytical base. While the '35–44' cohort follows with a sample size of $n = 30$, it was excluded from the baseline to maintain a more focused analysis.

```

young_adults_age_headphone = (
    ageheadphoneplot[ageheadphoneplot['Age_group'] == '18 - 24'] #slice
)

usage_18_24 = (
    young_adults_age_headphone['Daily_Headphone_Use']
    .value_counts()
    .reindex(headphone_usage_order)
)

def getsubpercentage(valuecount):
    total = valuecount.sum()
    return [f"{{(subgroup / total * 100)}:.1f}%" for subgroup in valuecount]
#to get fstrings for percentage of each subgroup to 1 d.p
#creating a function because I may need later

percent_18_24 = getsubpercentage(usage_18_24)

fig, ax = plt.subplots(figsize=(8, 5), constrained_layout = True)

hp_usage_color = ['seagreen', 'gold', 'orange', 'crimson']

young_adult_hp_bar = ax.bar(usage_18_24.index, usage_18_24.values, color=hp_usage_color)
ax.bar_label(young_adult_hp_bar, labels = percent_18_24, fontsize = 10)

ax.set_title("Fig. 2 - Young Adults (18 - 24 yr old): Daily Headphone Usage", loc = 'left', )
ax.set_xlabel(f"Usage Duration\n(n = 284)", fontsize = 12)
ax.set_ylabel("Number of Respondents")
plt.xticks(rotation = 0)

fig.set_constrained_layout_pads(h_pad = 0.05)
#using constrained layout and setting the padding to ensure everything is shown
plt.show()

```

Fig. 2 - Young Adults (18 - 24 yr old): Daily Headphone Usage

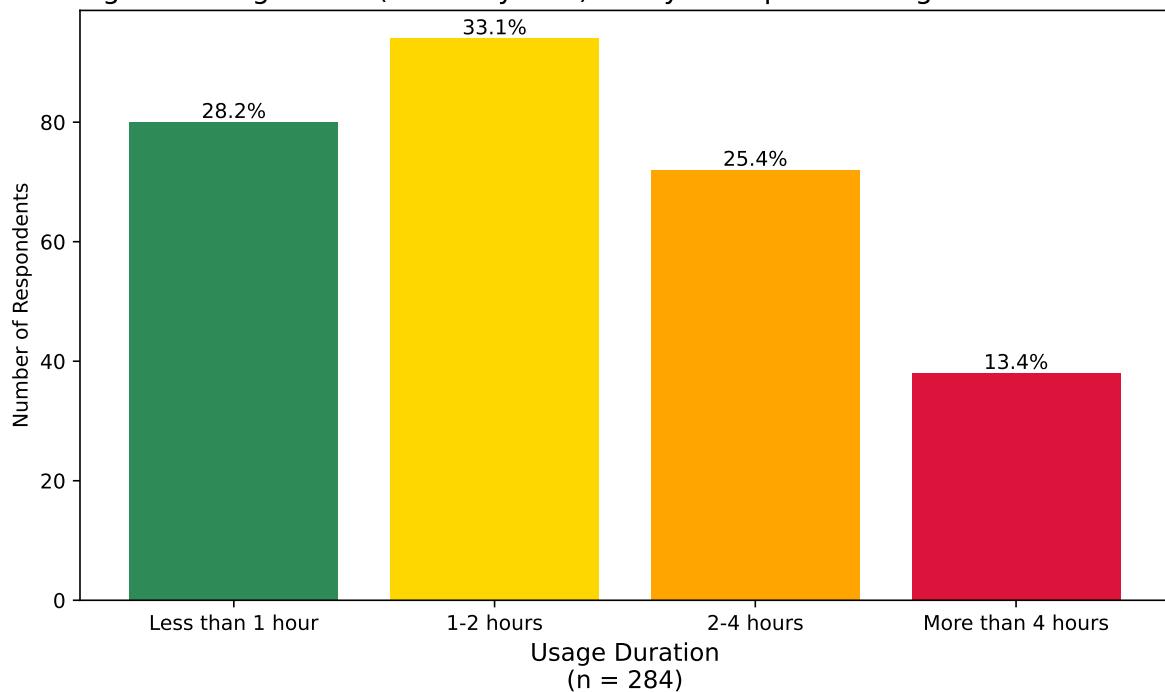


Fig. 2 shows the the daily headphone usage of young adults. Most fall into the 1-2 hour range.

2 - Analysing Headphone Usage by Important Sounds Missed

```
young_adults_only = hearingcleaned[hearingcleaned['Age_group'] == '18 - 24']
yesmaybeno_order = ["Yes", "Maybe", "No"] #to fix order for yes, maybe & no
yesmaybeno_colours = ["green", "steelblue", "red"]

missed_sounds_18_24_ct = (
    pd.crosstab(
        young_adults_only['Daily_Headphone_Use'],
        young_adults_only['Missed_Important_Sounds_clean']
    )
    .reindex(index = headphone_usage_order, columns = yesmaybeno_order)
)

missed_sounds_18_24_pct = missed_sounds_18_24_ct.div(missed_sounds_18_24_ct.sum(axis=1), axis=0)
#normalising. divides ind celss by row total, then * 100
missed_sounds_18_24_table = (
    missed_sounds_18_24_pct.map(lambda x: f"{x:.1f}%")
    .rename_axis("Daily Usage Duration", axis = 0)
    .rename_axis("Missed Important Sounds?", axis = 1)
)
#round to 1 d.p and add %
missed_sounds_18_24_table
```

Missed Important Sounds?	Yes	Maybe	No
Daily Usage Duration			
Less than 1 hour	50.0%	16.2%	33.8%
1-2 hours	52.1%	17.0%	30.9%
2-4 hours	33.3%	31.9%	34.7%
More than 4 hours	39.5%	21.1%	39.5%

Table 1 - Headphone Usage by Important Sounds Missed (n = 284)

From Table 1, it suggests that a higher frequency usage of headphone do not increase the frequency of missing important sounds. In fact, there seems to be a reduced frequency of missing important sounds with higher frequency usage of headphone. Deeper analysis (e.g. Chi Square) needs to be done.

3 - Analysing Headphone Usage by Belief To Care For Hearing Early

```
belief_18_24 = (
    young_adults_only.groupby('Daily_Headphone_Use', observed=False)[['Belief_Early_Hearing_Care']]
    #I am getting a Future Warning error for observed=False being deprecated
    .mean()
    .reindex(headphone_usage_order)
    .to_frame(name = 'Average Belief Score (1-5)')
)
#to_frame to convert a series of values into a DF for better presentation

belief_18_24
```

Daily_Headphone_Use	Average Belief Score (1-5)
Less than 1 hour	3.962500
1-2 hours	3.702128
2-4 hours	3.930556
More than 4 hours	4.000000

Table 2 - Headphone Usage vs. Belief for Early Hearing Care (n = 284)

From Table 2, it suggests that a higher frequency usage of headphone do not increase belief for early hearing care. We can do a one-way ANOVA to compare means (not in this report).

4 - Analysing interest & willingness to pay for hearing health app

```
yesmaybeno_order = ["Yes", "Maybe", "No"] #to fix order for yes, maybe & no
yesmaybeno_colours = ["green", "steelblue", "red"]

general_app_interest_18_24 = (
    young_adults_only['Interest_in_Hearing_App_clean']
    .value_counts()
    .reindex(yesmaybeno_order)
)
#showing code this way because code is too long
paid_app_interest_18_24 = (
    young_adults_only['Paid_App_Test_Interest_clean']
    .value_counts()
    .reindex(yesmaybeno_order)
)

general_interest_percentage = getsubpercentage(general_app_interest_18_24)
paid_interest_percentage = getsubpercentage(paid_app_interest_18_24)
#use previously defined function

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (8, 5), constrained_layout = True)

gen_bargraph = ax1.bar(
    general_app_interest_18_24.index,
    general_app_interest_18_24.values,
    color=yesmaybeno_colours
)

ax1.set_title('Interest in a Hearing Health App', fontsize = 8)
ax1.bar_label(gen_bargraph, labels=general_interest_percentage, fontsize = 8)
#ax1.set_xlabel('Interest Level', fontsize=10)
#ax1.set_ylabel('Number of Respondents', fontsize=10)

paid_bargraph = ax2.bar(
    paid_app_interest_18_24.index,
    paid_app_interest_18_24.values,
    color=yesmaybeno_colours
)

ax2.set_title('Interest in a paid Hearing Health App', fontsize = 8)
ax2.bar_label(paid_bargraph, labels=paid_interest_percentage, fontsize = 8)
```

```

#ax2.set_xlabel('Interest Level', fontsize=10)
#ax2.set_ylabel('Number of Respondents', fontsize=10)
#ax2.yaxis.set_label_position("right")

#using global labels instead, as labels are the same between the 2 graphs
fig.supxlabel("Interest Level", fontsize = 12)
fig.supylabel("Number of Respondents", fontsize = 12)
fig.suptitle("Fig.3 - Young Adults' Interest in a Hearing Health App: General vs. Paid (n=284)", fontweight='bold')

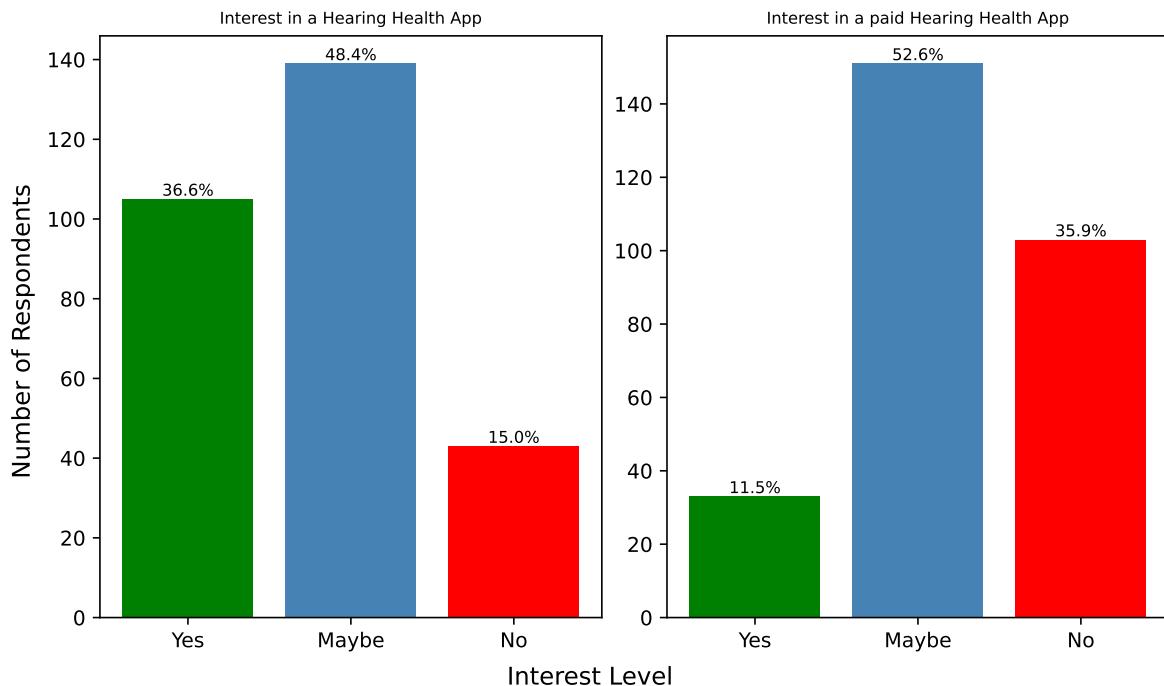
fig.set_constrained_layout_pads(h_pad = 0.05)
#using constrained layout and setting the padding to ensure everything is shown

plt.show()

#on hindsight, if I need to do more of the same graphs, I would create a function
#def plot_interest_bars(data_column, ax, title, order, color)

```

Fig.3 - Young Adults' Interest in a Hearing Health App: General vs. Paid (n=284)



From Fig.3, while nearly 85% of young adults respondents (combined ‘Yes’ and ‘Maybe’) demonstrate initial openness to a hearing health app, interest seems highly price-sensitive. Specifically, interest drops sharply from 36.6% to 11.5% for the “Yes” group when a cost is

introduced. Interestingly, the ‘Maybe’ segment remained stable at ~50%, suggesting a “wait-and-see” market that likely requires a stronger value proposition. Given the high general interest but low immediate willingness to pay, a freemium strategy (e.g. free screening with paid advanced features) is likely the most viable path for adoption.

5 - Analysis of Desired App Features for Hearing Health

```
desired_feature_18_24 = " ".join(young_adults_only['Desired_App_Features'])
# smashing individual responses into one massive paragraph so the Word Cloud
#wordcloud can ignore punctuations

wordcloud_desired = WordCloud(
    collocations = True, # Keeps phrases like "Noise Cancellation" together
    background_color = "white",
    max_words = 10
).generate(desired_feature_18_24)
#lesser words, more emphasis on top words

plt.figure(figsize = (8, 5))
plt.imshow(wordcloud_desired) #show image for wordcloud
plt.axis("off") # Hide x and y numbers because we are using plt
plt.title("Fig 4. - Wordcloud of Desired App Features for Hearing Health", fontsize = 14)
plt.show()
```

Fig 4. - Wordcloud of Desired App Features for Hearing Health



```
feature_counts_dict = {} #dict for easier tally. Key + value

for row in young_adults_only['Desired_App_Features']:
```

```

items = row.split(',')
for item in items:
    feature_clean = item.strip().lower() #remove space + make lowercase
    if feature_clean in feature_counts_dict:
        feature_counts_dict[feature_clean] += 1 #add tally
    else:
        feature_counts_dict[feature_clean] = 1 #new word

sort_features = sorted(feature_counts_dict.items(), key = lambda x: x[1], reverse = True)
#sort based on value, reverse for top
top_5 = sort_features[:5]

top_5_names = [item[0].title() for item in top_5]
top_5_counts = [item[1] for item in top_5] #just the count
top_5_perc = [(count / 284) * 100 for count in top_5_counts]
top_5_table = pd.DataFrame({
    'Top 5 Features': top_5_names,
    'Mentions (n)': top_5_counts,
    'Percentage (%)': top_5_perc
})
#DF for better presentation
top_5_table

```

	Top 5 Features	Mentions (n)	Percentage (%)
0	Quick Tests	183	64.436620
1	Doctor Consultation	143	50.352113
2	Game-Based Interaction	129	45.422535
3	Earphone Calibration	118	41.549296
4	Soft Guidance	115	40.492958

Table 3 - Top 5 Features

From Fig. 4, and Table 3, the most desired features is a quick hearing test, remote doctor consultation and gamified interaction.