```
In [ ]:
```

# Ling150 Project on Affirmation and Declination Responses

**Loading needed libraries**

```
In [1]:
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
import os
from pathlib import Path

plt.rcParams['figure.figsize'] = (12, 9)
plt.rcParams['font.size'] = 12

sns.set(style="whitegrid", palette="muted")
%matplotlib inline
```

```
In [ ]:
```

**Loading and Formatting Data**

```
In [2]:
# for Google Colab notebook

# run the following cells to mount the local drive and load the data
from google.colab import drive
drive.mount('/content/drive')
root_path = 'drive/My Drive/Ling150 Project/data'  #change dir to my project folder
```

```
In [3]:
# for Google Colab notebook

# Run this cell to load the data.
print("Importing training set...")
data_file = Path(root_path, "ling150_proj_data_2.csv")
df = pd.read_csv(data_file)

print("Done! Go model some data now!")
```

```
In [4]:
# # for Jupyter iPython notebook. Can run using the Anaconda environment into your browser

# # Run this cell to load the data.
# print("Importing training set...")
# data_file = Path("ling150_proj_data.csv")
# df = pd.read_csv(data_file)

# print("Done! Go model some data now!")
```

```
Importing training set...
Done! Go model some data now!
```

```
In [5]:
df.head()
```

Out[5]:

| | Response variants | Addresser | Sex_addresser | Race_addresser | Relationship | Addressee | Sex_addressee | Race_addressee | Situational Context | Location | Response type |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Kay | Jonathan | M | Chinese | good friend | Vincent | M | Chinese | NaN | in apartment | Affirmation |
| **1** | Yeah | Vincent | M | Chinese | good friend | Jonathan | M | Chinese | NaN | in apartment | Affirmation |
| **2** | Yeah | Jonathan | M | Chinese | good friend | Me | M | Chinese | NaN | in apartment | Affirmation |
| **3** | Yeah | Ethan | M | Chinese | good friend | Jonathan | M | Chinese | NaN | in apartment | Affirmation |
| **4** | Yeah | Ethan | M | Chinese | good friend | Me | M | Chinese | NaN | in apartment | Affirmation |

**Cleaning Data**

In [6]:
```
df = df.rename(columns={"Response variants" : "response", "Addresser" : "addresser", "Sex_addresser" : "sex_addresser",
                   "Race_addresser" : "race_addresser", "Relationship" : "relationship", "Addressee" : "addressee",
                   "Sex_addressee" : "sex_addressee", "Race_addressee" : "race_addressee",
                   "Situational Context" : "situational_context", "Location" : "location",
                   "Response type" : "response_type"})
```

In [7]:
```
column_list = list(df.columns)
print(column_list)
```

```
['response', 'addresser', 'sex_addresser', 'race_addresser', 'relationship', 'addressee', 'sex_addressee', 'race_addressee', 'situa
tional_context', 'location', 'response_type']
```

In [8]:
```
df['clean_response'] = df['response'].str.lower()
df['relationship'] = df['relationship'].str.lower()
```

In [9]:
```
df['clean_response'] = df['clean_response'].str.replace(",", "")    #Remove commas
df['clean_response'] = df['clean_response'].str.replace("?", "")    #Remove punctuation
df['clean_response'] = df['clean_response'].str.replace(".", "")    #Remove punctuation
df['clean_response'] = df['clean_response'].str.replace("so ", "")   #Remove leading uncertainty sound
df['clean_response'] = df['clean_response'].str.replace("uhh", "")  #Remove leading uncertainty sound
df['clean_response'] = df['clean_response'].str.replace("ohhh", "")   #Remove leading "oh" sound
df['clean_response'] = df['clean_response'].str.replace("oh", "")   #Remove leading "oh" sound
df['clean_response'] = df['clean_response'].str.replace("yeahhh", "yeah")   #Made length of "yeah" response uniform
df['clean_response'] = df['clean_response'].str.strip()             #Remove leading and trailing characters, like whitespace

df['clean_response'].loc[df['clean_response'] == 'ew no'] = "no"    #Removing the effect of leading and trailing words, and correcti
df['clean_response'].loc[df['clean_response'] == "eh it's okay"] = "it's okay"
df['clean_response'].loc[df['clean_response'] == "yeahyeahyeah"] = "yeah yeah yeah"
df['clean_response'].loc[df['clean_response'] == "yeah there we go"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "eh it's okay"] = "it's okay"
df['clean_response'].loc[df['clean_response'] == "but yeah"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "well yeah"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "eh it's okay"] = "it's okay"
df['clean_response'].loc[df['clean_response'] == "kay"] = "okay"
df['clean_response'].loc[df['clean_response'] == "yeah man"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "mmm not sure"] = "not sure"
df['clean_response'].loc[df['clean_response'] == "oh yeah?"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "uh yeah"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "yeah so"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "no it's okay"] = "no"
df['clean_response'].loc[df['clean_response'] == "yeah i know"] = "yeah"
df['clean_response'].loc[df['clean_response'] == "but yeah sure"] = "yeah sure"
df['clean_response'] = df['clean_response'].str.strip()             #Remove leading and trailing characters, like whitespace
```

In [10]:
```
print(df['clean_response'].value_counts())
```
```
yeah             111
no                19
okay              14
mm-hmm            12
yes               10
yeah yeah yeah     8
sure               7
alright            3
right              3
mm                 2
uh-huh             2
nice               2
yep                2
nah                2
yeah okay          2
yes yes            1
no it’s okay       1
exactly            1
guess so           1
naw it's fine      1
okay great         1
yeah exactly       1
yeah sure          1
that's cool        1
nice okay          1
nope               1
not sure           1
okay yeah          1
it's okay          1
no sorry           1
sounds good        1
think so           1
heck yeah          1
hopefully          1
aight              1
not really         1
no no no           1
right right        1
yeah yeah          1
of course          1
Name: clean_response, dtype: int64
```

In [ ]:

In [11]:
```
df['addresser'] = df['addresser'].str.strip()             #Remove leading and trailing characters, like whitespace
df['addresser'].loc[df['addresser'] == 'college student'] = "student"    #Removing the effect of leading and trailing words, and cor
df['addresser'].loc[df['addresser'] == 'Ling150 student'] = "student"
df['addresser'].loc[df['addresser'] == 'Confused student'] = "student"
df['addresser'].loc[df['addresser'] == 'Sarah (TA)'] = "Sarah"
df['addresser'].loc[df['addresser'] == 'Student'] = "student"
df['addresser'] = df['addresser'].str.strip()             #Remove leading and trailing characters, like whitespace
```

In [12]:
```
print(df['addresser'].value_counts())
```

```
student                 91
Gwen                    17
TA                      15
Tim Newman              12
Jonathan                11
Vincent                 10
Ethan                    9
Jenny Kim                7
Charis                   6
woman                    6
Amanda                   6
Sarah                    4
Joseph Park              4
Me                       3
Joyce                    3
MLK help desk            2
Inez                     2
Hispanic woman           2
father                   2
Amazon help desk         2
customer                 2
wife                     2
Tim Cruz                 1
photographer student     1
man                      1
Maryo                    1
restaurant cashier       1
Justin Hoong             1
Name: addresser, dtype: int64
```

In [13]:
```
df['addressee'] = df['addressee'].str.strip()             #Remove leading and trailing characters, like whitespace
df['addressee'].loc[df['addressee'] == 'college student'] = "student"     #Removing the effect of leading and trailing words, and cor
df['addressee'].loc[df['addressee'] == 'Ling150 student'] = "student"
df['addressee'].loc[df['addressee'] == 'Confused student'] = "student"
df['addressee'].loc[df['addressee'] == 'Sarah (TA)'] = "Sarah"
df['addressee'].loc[df['addressee'] == 'Student'] = "student"
df['addressee'].loc[df['addressee'] == 'female student'] = "student"
df['addressee'].loc[df['addressee'] == 'black woman'] = "woman"
df['addressee'].loc[df['addressee'] == 'male TA'] = "TA"
df['addressee'] = df['addressee'].str.strip()             #Remove leading and trailing characters, like whitespace
```

In [14]:
```
print(df['addressee'].value_counts())
```

```
student                       90
Me                            81
TA                            16
Jonathan                      10
woman                          8
Vincent                        5
Charis                         3
Joyce                          2
restaurant cashier             2
Tim Cruz                       2
man                            1
group of Caucasian friends     1
father                         1
Name: addressee, dtype: int64
```

In [15]:
```
df['relationship'] = df['relationship'].str.strip()             #Remove leading and trailing characters, like whitespace
# Removing the effect of leading and trailing words, and correcting non-uniform data input
df['relationship'].loc[df['relationship'] == 'on the phone, walking by'] = "on the phone"
```

In [16]:
```
print(df['relationship'].value_counts())
```

```
friend            87
good friend       71
teaching          32
fellow student    17
customer           7
parent             3
strangers          2
acquaintance       2
on the phone       1
Name: relationship, dtype: int64
```

In [17]:
```python
df.head()
```

Out[17]:

| | response | addresser | sex_addresser | race_addresser | relationship | addressee | sex_addressee | race_addressee | situational_context | location | response_type | cle |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Kay | Jonathan | M | Chinese | good friend | Vincent | M | Chinese | NaN | in apartment | Affirmation | |
| **1** | Yeah | Vincent | M | Chinese | good friend | Jonathan | M | Chinese | NaN | in apartment | Affirmation | |
| **2** | Yeah | Jonathan | M | Chinese | good friend | Me | M | Chinese | NaN | in apartment | Affirmation | |
| **3** | Yeah | Ethan | M | Chinese | good friend | Jonathan | M | Chinese | NaN | in apartment | Affirmation | |
| **4** | Yeah | Ethan | M | Chinese | good friend | Me | M | Chinese | NaN | in apartment | Affirmation | |

In [18]:
```python
for col in column_list:
    df[col] = df[col].str.strip()            #Remove leading and trailing characters, like whitespace)
```

In [19]:
```python
print(df['sex_addresser'].value_counts())
```

```
M    117
F    107
Name: sex_addresser, dtype: int64
```

In [20]:
```python
print(df['race_addresser'].value_counts())
```

```
Chinese        74
Caucasian      40
Indian         30
Hispanic       18
Asian Mixed    16
Korean         12
Asian          12
Black           2
Filipino        1
Vietnamese      1
Egyptian        1
Name: race_addresser, dtype: int64
```

In [21]:
```python
# I'm going to use the labels of race as determine by the US Census Bureau
# https://www.census.gov/mso/www/training/pdf/race-ethnicity-onepager.pdf
# So, my labels will comprise of "White", "Black or African American", "American Indian or Alaska Native",
# "Asian", and "Native Hawaiian or Other Pacific Islander"
# For the sake of my data results, I will further split the category "Asian" into "East Asian" and "Indian"
# Because I couldn't tell who was Chinese sometimes, and just put "Asian",
# I'm going to label all people of East Asian origin as "East Asian"

# Applying race category labels to addressers
# Normalizing East Asian people into the category "East Asian"
df['race_addresser'].loc[df['race_addresser'] == 'Asian'] = "East Asian"
df['race_addresser'].loc[df['race_addresser'] == 'Chinese'] = "East Asian"
df['race_addresser'].loc[df['race_addresser'] == 'Asian Mixed'] = "East Asian"
df['race_addresser'].loc[df['race_addresser'] == 'Korean'] = "East Asian"
df['race_addresser'].loc[df['race_addresser'] == 'Vietnamese'] = "East Asian"
df['race_addresser'].loc[df['race_addresser'] == 'Filipino'] = "Pacific Islander"
```

In [22]:
```python
# Going to apply the same labels to addressees as well

df['race_addressee'].loc[df['race_addressee'] == 'Asian'] = "East Asian"
df['race_addressee'].loc[df['race_addressee'] == 'Chinese'] = "East Asian"
df['race_addressee'].loc[df['race_addressee'] == 'Asian Mixed'] = "East Asian"
df['race_addressee'].loc[df['race_addressee'] == 'Korean'] = "East Asian"
df['race_addressee'].loc[df['race_addressee'] == 'Vietnamese'] = "East Asian"
df['race_addressee'].loc[df['race_addressee'] == 'Filipino'] = "Pacific Islander"
```

In [23]:
```python
print(df['sex_addressee'].value_counts())
```

```
M    172
F     49
Name: sex_addressee, dtype: int64
```

In [24]:
```python
print(df['race_addressee'].value_counts())
```

```
East Asian          149
Caucasian            24
Indian               16
Hispanic             16
Black                 4
Pacific Islander      2
Name: race_addressee, dtype: int64
```

In [25]:
```python
df['location'].loc[df['location'] == 'Data 8 lab'] = "in class"
df['location'] = df['location'].str.replace("in ", "")          #Remove "in "
df['location'] = df['location'].str.replace("outside ", "")     #Remove "outside "

print(df['location'].value_counts())
```

```
class         71
MLK           55
apartment     47
phone call    23
GBC           10
street        10
restaurant     5
RSF            3
Name: location, dtype: int64
```

In [26]:
```python
column_list = list(df.columns)
print(column_list)
```

```
['response', 'addresser', 'sex_addresser', 'race_addresser', 'relationship', 'addressee', 'sex_addressee', 'race_addressee', 'situa
tional_context', 'location', 'response_type', 'clean_response']
```

In [27]:
```python
# Final check that my data is cleaned and ready to explore now
for col in column_list:
    if col != 'response':
        print("Column is: '", col, "'")
        print(df[col].value_counts())
```

```
restaurant     5
RSF            3
Name: location, dtype: int64
Column is: ' response_type '
Affirmation    196
Declination     28
Name: response_type, dtype: int64
Column is: ' clean_response '
yeah              111
no                 19
okay               14
mm-hmm             12
yes                10
yeah yeah yeah      8
sure                7
alright             3
right               3
mm                  2
uh-huh              2
nice                2
```

In [ ]:
```python

```

## Data Exploration and Visualization

In [28]:
```python
yeses_df = df.loc[df['response_type'] == "Affirmation"]
nos_df = df.loc[df['response_type'] == "Declination"]
```

In [29]:
```python
df.head()
```

Out[29]:

| | response | addresser | sex_addresser | race_addresser | relationship | addressee | sex_addressee | race_addressee | situational_context | location | response_type | cle |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Kay | Jonathan | M | East Asian | good friend | Vincent | M | East Asian | NaN | apartment | Affirmation | |
| **1** | Yeah | Vincent | M | East Asian | good friend | Jonathan | M | East Asian | NaN | apartment | Affirmation | |
| **2** | Yeah | Jonathan | M | East Asian | good friend | Me | M | East Asian | NaN | apartment | Affirmation | |
| **3** | Yeah | Ethan | M | East Asian | good friend | Jonathan | M | East Asian | NaN | apartment | Affirmation | |
| **4** | Yeah | Ethan | M | East Asian | good friend | Me | M | East Asian | NaN | apartment | Affirmation | |

I am choosing to use this space to get counts and percentages of certain groupings, such as the number and percentage of "yes" response variants for female addressers. These will then be visualized in more aesthetically pleasing plots and tables in Microsoft Word and Excel. There exist libraries that can generate good plots in Python (such as matplotlib and seaborn) but in the interest of time and familiarity, I will use the Microsoft suite.

## Grouping response variants

```
In [30]:  print(len(df['clean_response'].value_counts()))
```

```
40
```

```
In [31]:  print(yeses_df['clean_response'].value_counts())
```

```
yeah               111
okay                14
mm-hmm              12
yes                 10
yeah yeah yeah       8
sure                 7
right                3
alright              3
yeah okay            2
mm                   2
yep                  2
nice                 2
uh-huh               2
yeah exactly         1
yeah sure            1
okay great           1
yes yes              1
guess so             1
exactly              1
```

```
In [32]:  print(nos_df['clean_response'].value_counts())
```

```
no              19
nah              2
nope             1
naw it's fine    1
no sorry         1
no it’s okay     1
not sure         1
no no no         1
not really       1
Name: clean_response, dtype: int64
```

In [33]:
```
# Because there are way too many types of response variants that I received,
# I am going to group them into certain general categories of response variants.


# For repeats of the same variant in one response, I will only count it as one response of that variant.
# (Ex. If the addresser used "yeah, yeah, yeah", I counted that as one response of "yeah")
# For mixtures of affirmation responses (i.e. "okay yeah"), I am going to group these into one category known as "Mixture."
# I was concerned with how I enumerated my data points, and in response with how I was to tally my data,
# the following argument convinced me to tally counts in this manner.
# If I chose to count my mixtures of affirmation responses as tallies for each affirmation
# (i.e. "okay yeah" means 1 count of "okay" and 1 count of "yeah"),
# then my counts of data points are no longer of the unique responses, but of the occurrence of each variant in an entire conversati
# I seek to look at each response as unique on its own, since repeats and mixtures of a variant serve the same purpose of
# affirmation and declination. So, I will choose to tally the number of unique response variants
# instead of the number of affirmation and declination words themselves.
# For that reason, I choose to group the above responses into categories which show the true frequency of response variants,
# and not that of response words themselves.

# I grouped "right" and "alright" responses together as the "Alright/Right" category
# For somewhat non-vocal affirmation and declination noises, I group them into the "Sounds" category. These include ...
# Finally, I grouped the other less common response variants into the "Other" category. These include all "yes" responses
# with count of two or lower.
# These include .....

# Grouping "Yes" responses
df['response_group'] = df['clean_response']
df['response_group'] = df['response_group'].str.replace(r"[\']", "")  #Remove apostrophes

df['response_group'].loc[df['response_group'] == 'yeah yeah'] = "yeah"
df['response_group'].loc[df['response_group'] == 'yeah yeah yeah'] = "yeah"
df['response_group'].loc[df['response_group'] == 'yeah yeah'] = "yeah"
df['response_group'].loc[df['response_group'] == 'yeah yeah yeah'] = "yeah"
df['response_group'].loc[df['response_group'] == 'heck yeah'] = "yeah"

df['response_group'].loc[df['response_group'] == 'yes yes'] = "yes"

df['response_group'].loc[df['response_group'] == 'okay yeah'] = "mixture"
df['response_group'].loc[df['response_group'] == 'yeah okay'] = "mixture"
df['response_group'].loc[df['response_group'] == 'nice okay'] = "mixture"
df['response_group'].loc[df['response_group'] == 'yeah exactly'] = "mixture"
df['response_group'].loc[df['response_group'] == 'okay great'] = "mixture"
df['response_group'].loc[df['response_group'] == 'yeah sure'] = "mixture"

df['response_group'].loc[df['response_group'] == 'its okay'] = "okay"

df['response_group'].loc[df['response_group'] == 'alright'] = "Alright/Right"
df['response_group'].loc[df['response_group'] == 'right'] = "Alright/Right"
df['response_group'].loc[df['response_group'] == 'right right'] = "Alright/Right"
df['response_group'].loc[df['response_group'] == 'aight'] = "Alright/Right"

df['response_group'].loc[df['response_group'] == 'mm-hmm'] = "Sounds"
df['response_group'].loc[df['response_group'] == 'mm'] = "Sounds"
df['response_group'].loc[df['response_group'] == 'uh-huh'] = "Sounds"

df['response_group'].loc[df['response_group'] == 'nice'] = "Other"
df['response_group'].loc[df['response_group'] == 'yep'] = "Other"
df['response_group'].loc[df['response_group'] == 'sounds good'] = "Other"
df['response_group'].loc[df['response_group'] == 'exactly'] = "Other"
df['response_group'].loc[df['response_group'] == 'think so'] = "Other"
df['response_group'].loc[df['response_group'] == 'of course'] = "Other"
df['response_group'].loc[df['response_group'] == 'guess so'] = "Other"
df['response_group'].loc[df['response_group'] == 'hopefully'] = "Other"
df['response_group'].loc[df['response_group'] == "thats cool"] = "Other"

df['response_group'] = df['response_group'].str.strip()             #Remove leading and trailing  whitespace


# Grouping "No" responses
# Groupings: No, not __, nah, "no" combination - degree of emotion is shown using surrounding words, nope

df['response_group'].loc[df['response_group'] == 'no no no'] = "no"
df['response_group'].loc[df['response_group'] == "no it’s okay"] = "'no' combination"
df['response_group'].loc[df['response_group'] == 'no sorry'] = "'no' combination"
df['response_group'].loc[df['response_group'] == "naw its fine"] = "nah"
df['response_group'].loc[df['response_group'] == 'not really'] = "not ___"
df['response_group'].loc[df['response_group'] == 'not sure'] = "not ___"

df['response_group'] = df['response_group'].str.strip()             #Remove leading and trailing  whitespace


# update my "yes" and "no" dataframes
yeses_df = df.loc[df['response_type'] == "Affirmation"]
nos_df = df.loc[df['response_type'] == "Declination"]
```

In [34]:
```
print(yeses_df['response_group'].value_counts())
```

```
yeah             121
Sounds            16
okay              15
yes               11
Other             11
Alright/Right      8
mixture            7
sure               7
Name: response_group, dtype: int64
```

In [35]:
```
print(nos_df['response_group'].value_counts())
```

```
no                  20
nah                  3
not ___              2
'no' combination     2
nope                 1
Name: response_group, dtype: int64
```

I have a fairly even distribution of male and female addressers. I'm pretty happy about this.

In [36]:
```
print(df['sex_addresser'].value_counts())
```

```
M    117
F    107
Name: sex_addresser, dtype: int64
```

### Exploring Gender of Addresser vs "Yes" and "No" Responses

In [37]:
```
print(df['sex_addresser'].value_counts())
```

```
M    117
F    107
Name: sex_addresser, dtype: int64
```

In [38]:
```
print("Female addresser, yeah: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "yeah")]))
print("Female addresser, sounds: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "Sounds")]))
print("Female addresser, okay: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "okay")]))
print("Female addresser, yes: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "yes")]))
print("Female addresser, alright/right: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "Alright/Right")]))
print("Female addresser, sure: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "sure")]))
print("Female addresser, mixture: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "mixture")]))
print("Female addresser, other: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "Other")]))
print("Total Female affirmation responses: ", len(yeses_df.loc[(yeses_df['sex_addresser'] == "F")]))

print()
print()

print("Female addresser, no: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "no")]))
print("Female addresser, nah: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "nah")]))
print("Female addresser, no comb: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "'no' combination")]))
print("Female addresser, not __: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "not ___")]))
print("Female addresser, nope: ", len(df.loc[(df['sex_addresser'] == "F") & (df['response_group'] == "nope")]))
print("Total Female declination responses: ", len(nos_df.loc[(nos_df['sex_addresser'] == "F")]))
```

```
Female addresser, yeah:  61
Female addresser, sounds:  8
Female addresser, okay:  8
Female addresser, yes:  5
Female addresser, alright/right:  4
Female addresser, sure:  1
Female addresser, mixture:  4
Female addresser, other:  3
Total Female affirmation responses:  94


Female addresser, no:  11
Female addresser, nah:  1
Female addresser, no comb:  0
Female addresser, not __:  1
Female addresser, nope:  0
Total Female declination responses:  13
```

In [39]:
```
print("Male addresser, yeah: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "yeah")]))
print("Male addresser, sounds: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "Sounds")]))
print("Male addresser, okay: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "okay")]))
print("Male addresser, yes: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "yes")]))
print("Male addresser, alright/right: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "Alright/Right")]))
print("Male addresser, sure: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "sure")]))
print("Male addresser, mixture: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "mixture")]))
print("Male addresser, other: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "Other")]))
print("Total Male affirmation responses: ", len(yeses_df.loc[(yeses_df['sex_addresser'] == "M")]))

print()
print()

print("Male addresser, no: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "no")]))
print("Male addresser, nah: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "nah")]))
print("Male addresser, no comb: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "'no' combination")]))
print("Male addresser, not __: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "not ___")]))
print("Male addresser, nope: ", len(df.loc[(df['sex_addresser'] == "M") & (df['response_group'] == "nope")]))
print("Total Male declination responses: ", len(nos_df.loc[(nos_df['sex_addresser'] == "M")]))
```

```
Male addresser, yeah:  60
Male addresser, sounds:  8
Male addresser, okay:  7
Male addresser, yes:  6
Male addresser, alright/right:  4
Male addresser, sure:  6
Male addresser, mixture:  3
Male addresser, other:  8
Total Male affirmation responses:  102


Male addresser, no:  9
Male addresser, nah:  2
Male addresser, no comb:  2
Male addresser, not __:  1
Male addresser, nope:  1
Total Male declination responses:  15
```

In [ ]:

### Exploring Gender of Addressee vs "Yes" and "No" Responses

In [40]:
```
print(df['sex_addressee'].value_counts())
```

```
M    172
F     49
Name: sex_addressee, dtype: int64
```

In [41]:
```
print("Female addressee, yeah: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "yeah")]))
print("Female addressee, sounds: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "Sounds")]))
print("Female addressee, okay: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "okay")]))
print("Female addressee, yes: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "yes")]))
print("Female addressee, alright/right: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "Alright/Right")]))
print("Female addressee, sure: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "sure")]))
print("Female addressee, mixture: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "mixture")]))
print("Female addressee, other: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "Other")]))
print("Total Female affirmation responses: ", len(yeses_df.loc[(yeses_df['sex_addressee'] == "F")]))

print()
print()

print("Female addressee, no: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "no")]))
print("Female addressee, nah: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "nah")]))
print("Female addressee, no comb: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "'no' combination")]))
print("Female addressee, not __: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "not ___")]))
print("Female addressee, nope: ", len(df.loc[(df['sex_addressee'] == "F") & (df['response_group'] == "nope")]))
print("Total Female declination responses: ", len(nos_df.loc[(nos_df['sex_addressee'] == "F")]))
```

```
Female addressee, yeah:  31
Female addressee, sounds:  2
Female addressee, okay:  2
Female addressee, yes:  1
Female addressee, alright/right:  3
Female addressee, sure:  2
Female addressee, mixture:  3
Female addressee, other:  2
Total Female affirmation responses:  46


Female addressee, no:  3
Female addressee, nah:  0
Female addressee, no comb:  0
Female addressee, not __:  0
Female addressee, nope:  0
Total Female declination responses:  3
```

In [42]:
```
print("Male addressee, yeah: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "yeah")]))
print("Male addressee, sounds: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "Sounds")]))
print("Male addressee, okay: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "okay")]))
print("Male addressee, yes: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "yes")]))
print("Male addressee, alright/right: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "Alright/Right")]))
print("Male addressee, sure: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "sure")]))
print("Male addressee, mixture: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "mixture")]))
print("Male addressee, other: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "Other")]))
print("Total Male affirmation responses: ", len(yeses_df.loc[(yeses_df['sex_addressee'] == "M")]))

print()
print()

print("Male addressee, no: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "no")]))
print("Male addressee, nah: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "nah")]))
print("Male addressee, no comb: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "'no' combination")]))
print("Male addressee, not __: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "not ___")]))
print("Male addressee, nope: ", len(df.loc[(df['sex_addressee'] == "M") & (df['response_group'] == "nope")]))
print("Total Male declination responses: ", len(nos_df.loc[(nos_df['sex_addressee'] == "M")]))
```

```
Male addressee, yeah:  88
Male addressee, sounds:  14
Male addressee, okay:  13
Male addressee, yes:  10
Male addressee, alright/right:  5
Male addressee, sure:  5
Male addressee, mixture:  4
Male addressee, other:  9
Total Male affirmation responses:  148


Male addressee, no:  16
Male addressee, nah:  3
Male addressee, no comb:  2
Male addressee, not __:  2
Male addressee, nope:  1
Total Male declination responses:  24
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

## Grouping formality of relationship

In [43]:
```
column_list = list(df.columns)
print(column_list)
```
```
['response', 'addresser', 'sex_addresser', 'race_addresser', 'relationship', 'addressee', 'sex_addressee', 'race_addressee', 'situa
tional_context', 'location', 'response_type', 'clean_response', 'response_group']
```

In [44]:
```
print(df['relationship'].value_counts())
```
```
friend            87
good friend       71
teaching          32
fellow student    17
customer           7
parent             3
strangers          2
acquaintance       2
on the phone       1
Name: relationship, dtype: int64
```

In [ ]:

In [45]:
```
# I decided to group the formality of relationships using the following metric.

# I grouped "friend", "good friend", "fellow student", and "on the phone" into the "Informal" category.
# I grouped "teaching", "customer", "parent", "acquaintance", "strangers" into the "Formal" category.

df['formality'] = df['relationship']

df['formality'].loc[df['formality'] == 'friend'] = "informal"
df['formality'].loc[df['formality'] == 'good friend'] = "informal"
df['formality'].loc[df['formality'] == 'fellow student'] = "informal"
df['formality'].loc[df['formality'] == 'on the phone'] = "informal"

df['formality'].loc[df['formality'] == 'teaching'] = "formal"
df['formality'].loc[df['formality'] == 'customer'] = "formal"
df['formality'].loc[df['formality'] == 'parent'] = "formal"
df['formality'].loc[df['formality'] == 'acquaintance'] = "formal"
df['formality'].loc[df['formality'] == 'strangers'] = "formal"

df['formality'] = df['formality'].str.strip()             #Remove leading and trailing  whitespace


# update my "yes" and "no" dataframes
yeses_df = df.loc[df['response_type'] == "Affirmation"]
nos_df = df.loc[df['response_type'] == "Declination"]
```

In [46]:
```
print(yeses_df['formality'].value_counts())
```
```
informal    152
formal       42
Name: formality, dtype: int64
```

In [47]:
```
print(nos_df['formality'].value_counts())
```
```
informal    24
formal       4
Name: formality, dtype: int64
```

In [48]:
```
print(df['formality'].value_counts())
```
```
informal    176
formal       46
Name: formality, dtype: int64
```

In [ ]:

### Exploring Formality of Relationship vs "Yes" and "No" Responses

In [49]:
```
print(df['formality'].value_counts())
```

```
informal    176
formal       46
Name: formality, dtype: int64
```

In [50]:
```
print("formal relationship, yeah: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "yeah")]))
print("formal relationship, sounds: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "Sounds")]))
print("formal relationship, okay: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "okay")]))
print("formal relationship, yes: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "yes")]))
print("formal relationship, alright/right: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "Alright/Right")]))
print("formal relationship, sure: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "sure")]))
print("formal relationship, mixture: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "mixture")]))
print("formal relationship, other: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "Other")]))
print("Total formal affirmation responses: ", len(yeses_df.loc[(yeses_df['formality'] =="formal")]))

print()
print()

print("formal relationship, no: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "no")]))
print("formal relationship, nah: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "nah")]))
print("formal relationship, no comb: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "'no' combination")]))
print("formal relationship, not __: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "not ___")]))
print("formal relationship, nope: ", len(df.loc[(df['formality'] =="formal") & (df['response_group'] == "nope")]))
print("Total formal declination responses: ", len(nos_df.loc[(nos_df['formality'] =="formal")]))
```

```
formal relationship, yeah:  24
formal relationship, sounds:  3
formal relationship, okay:  2
formal relationship, yes:  2
formal relationship, alright/right:  3
formal relationship, sure:  1
formal relationship, mixture:  2
formal relationship, other:  5
Total formal affirmation responses:  42


formal relationship, no:  2
formal relationship, nah:  1
formal relationship, no comb:  1
formal relationship, not __:  0
formal relationship, nope:  0
Total formal declination responses:  4
```

In [51]:
```
print("informal relationship, yeah: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "yeah")]))
print("informal relationship, sounds: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "Sounds")]))
print("informal relationship, okay: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "okay")]))
print("informal relationship, yes: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "yes")]))
print("informal relationship, alright/right: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "Alright/Right"
print("informal relationship, sure: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "sure")]))
print("informal relationship, mixture: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "mixture")]))
print("informal relationship, other: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "Other")]))
print("Total informal affirmation responses: ", len(yeses_df.loc[(yeses_df['formality'] =="informal")]))

print()
print()

print("informal relationship, no: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "no")]))
print("informal relationship, nah: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "nah")]))
print("informal relationship, no comb: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "'no' combination")])
print("informal relationship, not __: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "not ___")]))
print("informal relationship, nope: ", len(df.loc[(df['formality'] =="informal") & (df['response_group'] == "nope")]))
print("Total informal declination responses: ", len(nos_df.loc[(nos_df['formality'] =="informal")]))
```

```
informal relationship, yeah:  95
informal relationship, sounds:  13
informal relationship, okay:  13
informal relationship, yes:  9
informal relationship, alright/right:  5
informal relationship, sure:  6
informal relationship, mixture:  5
informal relationship, other:  6
Total informal affirmation responses:  152


informal relationship, no:  18
informal relationship, nah:  2
informal relationship, no comb:  1
informal relationship, not __:  2
informal relationship, nope:  1
Total informal declination responses:  24
```

In [ ]:

In [ ]:

In [ ]:

## Grouping publicity of location

In [52]:
```
column_list = list(df.columns)
print(column_list)
```

```
['response', 'addresser', 'sex_addresser', 'race_addresser', 'relationship', 'addressee', 'sex_addressee', 'race_addressee', 'situa
tional_context', 'location', 'response_type', 'clean_response', 'response_group', 'formality']
```

In [53]:
```
print(df['location'].value_counts())
```

```
class         71
MLK           55
apartment     47
phone call    23
GBC           10
street        10
restaurant     5
RSF            3
Name: location, dtype: int64
```

In [ ]:

In [54]:
```
# I decided to group the publicity of locations based on whether it was a public or private area and how crowded it was,
# in essence, giving a measure of how close other people excluded from the conversation are within the surrounding vicinity.

# I grouped "MLK", "class", and "restaurant" into the "Public & Crowded" category.
# I grouped "GBC", "street", and "RSF" into the "Public & Spacious" category.
# I grouped "apartment" into the "Private & Crowded" category.
# I grouped "phone call" into the "Private & Spacious" category.

df['publicity_of_location'] = df['location']

df['publicity_of_location'].loc[df['publicity_of_location'] == 'MLK'] = "Public and Crowded"
df['publicity_of_location'].loc[df['publicity_of_location'] == 'class'] = "Public and Crowded"
df['publicity_of_location'].loc[df['publicity_of_location'] == 'restaurant'] = "Public and Crowded"

df['publicity_of_location'].loc[df['publicity_of_location'] == 'GBC'] = "Public and Spacious"
df['publicity_of_location'].loc[df['publicity_of_location'] == 'street'] = "Public and Spacious"
df['publicity_of_location'].loc[df['publicity_of_location'] == 'RSF'] = "Public and Spacious"

df['publicity_of_location'].loc[df['publicity_of_location'] == 'apartment'] = "Private and Crowded"

df['publicity_of_location'].loc[df['publicity_of_location'] == 'phone call'] = "Private and Spacious"

df['publicity_of_location'] = df['publicity_of_location'].str.strip()             #Remove leading and trailing  whitespace


# update my "yes" and "no" dataframes
yeses_df = df.loc[df['response_type'] == "Affirmation"]
nos_df = df.loc[df['response_type'] == "Declination"]
```

In [55]:
```
print(yeses_df['publicity_of_location'].value_counts())
```
```
Public and Crowded      112
Private and Crowded      42
Public and Spacious      22
Private and Spacious     20
Name: publicity_of_location, dtype: int64
```

In [56]:
```
print(nos_df['publicity_of_location'].value_counts())
```
```
Public and Crowded      19
Private and Crowded      5
Private and Spacious     3
Public and Spacious      1
Name: publicity_of_location, dtype: int64
```

In [57]:
```
print(df['publicity_of_location'].value_counts())
```
```
Public and Crowded      131
Private and Crowded      47
Public and Spacious      23
Private and Spacious     23
Name: publicity_of_location, dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

## Exploring Publicity of Location vs "Yes" and "No" Responses

In [58]:
```
print(df['publicity_of_location'].value_counts())
```
```
Public and Crowded      131
Private and Crowded      47
Public and Spacious      23
Private and Spacious     23
Name: publicity_of_location, dtype: int64
```

In [59]:
```
blic and crowded, yeah: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "yeah")]))
blic and crowded, sounds: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "Sounds")])
blic and crowded, okay: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "okay")]))
blic and crowded, yes: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "yes")]))
blic and crowded, alright/right: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "Alri
blic and crowded, sure: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "sure")]))
blic and crowded, mixture: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "mixture")
blic and crowded, other: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "Other")]))
tal public & crowded affirmation responses: ", len(yeses_df.loc[(yeses_df['publicity_of_location'] == "Public and Crowded")]))




blic and crowded, no: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "no")]))
blic and crowded, nah: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "nah")]))
blic and crowded, no comb: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "'no' combi
blic and crowded, not __: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "not ___")]
blic and crowded, nope: ", len(df.loc[(df['publicity_of_location'] == "Public and Crowded") & (df['response_group'] == "nope")]))
tal public & crowded declination responses: ", len(nos_df.loc[(nos_df['publicity_of_location'] == "Public and Crowded")]))
```

```
public and crowded, yeah:  67
public and crowded, sounds:  10
public and crowded, okay:  8
public and crowded, yes:  7
public and crowded, alright/right:  4
public and crowded, sure:  3
public and crowded, mixture:  6
public and crowded, other:  7
Total public & crowded affirmation responses:  112


public and crowded, no:  14
public and crowded, nah:  2
public and crowded, no comb:  1
public and crowded, not __:  2
public and crowded, nope:  0
Total public & crowded declination responses:  19
```

In [60]:
```
spacious, yeah: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "yeah")]))
spacious, sounds: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "Sounds")]))
spacious, okay: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "okay")]))
spacious, yes: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "yes")]))
spacious, alright/right: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "Alright/Rig
spacious, sure: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "sure")]))
spacious, mixture: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "mixture")]))
spacious, other: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "Other")]))
c & crowded affirmation responses: ", len(yeses_df.loc[(yeses_df['publicity_of_location'] == "Public and Spacious")]))




spacious, no: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "no")]))
spacious, nah: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "nah")]))
spacious, no comb: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "'no' combination'
spacious, not __: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "not ___")]))
spacious, nope: ", len(df.loc[(df['publicity_of_location'] == "Public and Spacious") & (df['response_group'] == "nope")]))
c & crowded declination responses: ", len(nos_df.loc[(nos_df['publicity_of_location'] == "Public and Spacious")]))
```

```
public and spacious, yeah:  16
public and spacious, sounds:  1
public and spacious, okay:  1
public and spacious, yes:  2
public and spacious, alright/right:  1
public and spacious, sure:  0
public and spacious, mixture:  0
public and spacious, other:  1
Total public & crowded affirmation responses:  22


public and spacious, no:  1
public and spacious, nah:  0
public and spacious, no comb:  0
public and spacious, not __:  0
public and spacious, nope:  0
Total public & crowded declination responses:  1
```

In [61]:

```
 crowded, yeah: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "yeah")]))
 crowded, sounds: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "Sounds")]))
 crowded, okay: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "okay")]))
 crowded, yes: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "yes")]))
 crowded, alright/right: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "Alright/Rig
 crowded, sure: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "sure")]))
 crowded, mixture: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "mixture")]))
 crowded, other: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "Other")]))
te & crowded affirmation responses: ", len(yeses_df.loc[(yeses_df['publicity_of_location'] == "Private and Crowded")]))




 crowded, no: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "no")]))
 crowded, nah: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "nah")]))
 crowded, no comb: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "'no' combination'
 crowded, not __: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "not ___")]))
 crowded, nope: ", len(df.loc[(df['publicity_of_location'] == "Private and Crowded") & (df['response_group'] == "nope")]))
te & crowded declination responses: ", len(nos_df.loc[(nos_df['publicity_of_location'] == "Private and Crowded")]))
```

```
private and crowded, yeah:  25
private and crowded, sounds:  3
private and crowded, okay:  3
private and crowded, yes:  2
private and crowded, alright/right:  1
private and crowded, sure:  4
private and crowded, mixture:  1
private and crowded, other:  3
Total private & crowded affirmation responses:  42


private and crowded, no:  2
private and crowded, nah:  1
private and crowded, no comb:  1
private and crowded, not __:  0
private and crowded, nope:  1
Total private & crowded declination responses:  5
```

In [62]:

```
print("private and spacious, yeah: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] ==
print("private and spacious, sounds: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] =
print("private and spacious, okay: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] ==
print("private and spacious, yes: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] == "
print("private and spacious, alright/right: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_gr
print("private and spacious, sure: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] ==
print("private and spacious, mixture: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group']
print("private and spacious, other: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] ==
print("Total private & spacious affirmation responses: ", len(yeses_df.loc[(yeses_df['publicity_of_location'] == "Private and Spacio

print()
print()

print("private and spacious, no: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] == "r
print("private and spacious, nah: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] == "
print("private and spacious, no comb: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group']
print("private and spacious, not __: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] =
print("private and spacious, nope: ", len(df.loc[(df['publicity_of_location'] == "Private and Spacious") & (df['response_group'] ==
print("Total private & spacious declination responses: ", len(nos_df.loc[(nos_df['publicity_of_location'] == "Private and Spacious")
```

```
private and spacious, yeah:  13
private and spacious, sounds:  2
private and spacious, okay:  3
private and spacious, yes:  0
private and spacious, alright/right:  2
private and spacious, sure:  0
private and spacious, mixture:  0
private and spacious, other:  0
Total private & spacious affirmation responses:  20


private and spacious, no:  3
private and spacious, nah:  0
private and spacious, no comb:  0
private and spacious, not __:  0
private and spacious, nope:  0
Total private & spacious declination responses:  3
```

In [63]:

```
# Plotting code - for another day perhaps....
# yes_count_female = yeses_df.loc[yeses_df['sex_addresser'] == "F"].groupby('response').count()
# yes_count_female = yes_count_female.reset_index()
# sns.barplot(x='response', y='sex_addresser', data=yes_count_female, order=order, ax=ax1)
# ax1.set_title("Frequency Distribution of 'Yes' Responses of Females")
# ax1.set_xlabel("Affirmation Variants")
# ax1.set_ylabel('Frequency (%)')

# plt.show();
```

In [ ]:

In [ ]: