MENTAL HEALTH PREDICITION

In [1]:
```
#exceptions to warnings
import warnings
warnings.filterwarnings('ignore')
```

IMPORTING NECESSARY LIBRARIES

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly as pio
import matplotlib.colors as mcolors
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from IPython.display import display
import ipywidgets as widgets
from fuzzywuzzy import fuzz
import scipy as misc
from category_encoders.ordinal import OrdinalEncoder
```

In [3]:
```
import plotly
```

To enable the rendering of plotly visualizations

In [4]:
```
plotly.offline.init_notebook_mode (connected = True)
```

PROVIDING LIST OF COLOR CODES

palette = sky blue, powder blue, light green, pale turquoise, thistle, deep taupe, light pink, pink, light peach, lemon chiffon, tan, sandy brown, rosy brown, dusty rose

palette1 = deep taupe, copper rose, light pink, pink, pale turquoise, thistle, mauve, light pink, pink, light peach, lemon chiffon, tan, sandy brown, rosy brown

In [5]:
```
palette = ['#87CEEB','#B0E0E6','#C1F8C1', '#C1F8D8', '#D8BFD8','#8D4C66', '#FFB6C1','#F0B2B2','#F8D8C1', '#FFFACD', '#D2B48C','#d6b39f', '#c48d86', '#ad6b75']
palette1 = ['#8D4C66', '#AD6B75','#FFB6C1','#F0B2B2', '#C1F8D8', '#D8BFD8','#E0B0FF','#FFB6C1','#F0B2B2','#F8D8C1', '#FFFACD', '#D2B48C','#d6b39f', '#c48d86']
```

IMPORTING DATASET

In [6]:
```
df=pd.read_csv('survey.csv')
df.head(10)
```

Out[6]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequence	phys_health_consequence	coworkers	supervisor	mental_health_inte
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	No	No	Some of them	Yes	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Maybe	No	No	No	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	No	No	Yes	Yes	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	Yes	Yes	Some of them	No	M
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	No	No	Some of them	Yes	
5	2014-08-27 11:31:22	33	Male	United States	TN	NaN	Yes	No	Sometimes	6-25	...	Don't know	No	No	Yes	Yes	
6	2014-08-27 11:31:50	35	Female	United States	MI	NaN	Yes	Yes	Sometimes	1-5	...	Somewhat difficult	Maybe	Maybe	Some of them	No	
7	2014-08-27 11:32:05	39	M	Canada	NaN	NaN	No	No	Never	1-5	...	Don't know	No	No	No	No	
8	2014-08-27 11:32:39	42	Female	United States	IL	NaN	Yes	Yes	Sometimes	100-500	...	Very difficult	Maybe	No	Yes	Yes	
9	2014-08-27 11:32:43	23	Male	Canada	NaN	NaN	No	No	Never	26-100	...	Don't know	No	No	Yes	Yes	M

10 rows × 27 columns

In [7]:

```
print(f'DataTypes in given dataset: \n{df.dtypes}')
```

```
DataTypes in given dataset: 
Timestamp                    object
Age                           int64
Gender                       object
Country                      object
state                        object
self_employed                object
family_history               object
treatment                    object
work_interfere               object
no_employees                 object
remote_work                  object
tech_company                 object
benefits                     object
care_options                 object
wellness_program             object
seek_help                    object
anonymity                    object
leave                        object
mental_health_consequence    object
phys_health_consequence      object
coworkers                    object
supervisor                   object
mental_health_interview      object
phys_health_interview        object
mental_vs_physical           object
obs_consequence              object
comments                     object
dtype: object
```

In [8]:

```
df.describe()
```

Out[8]:

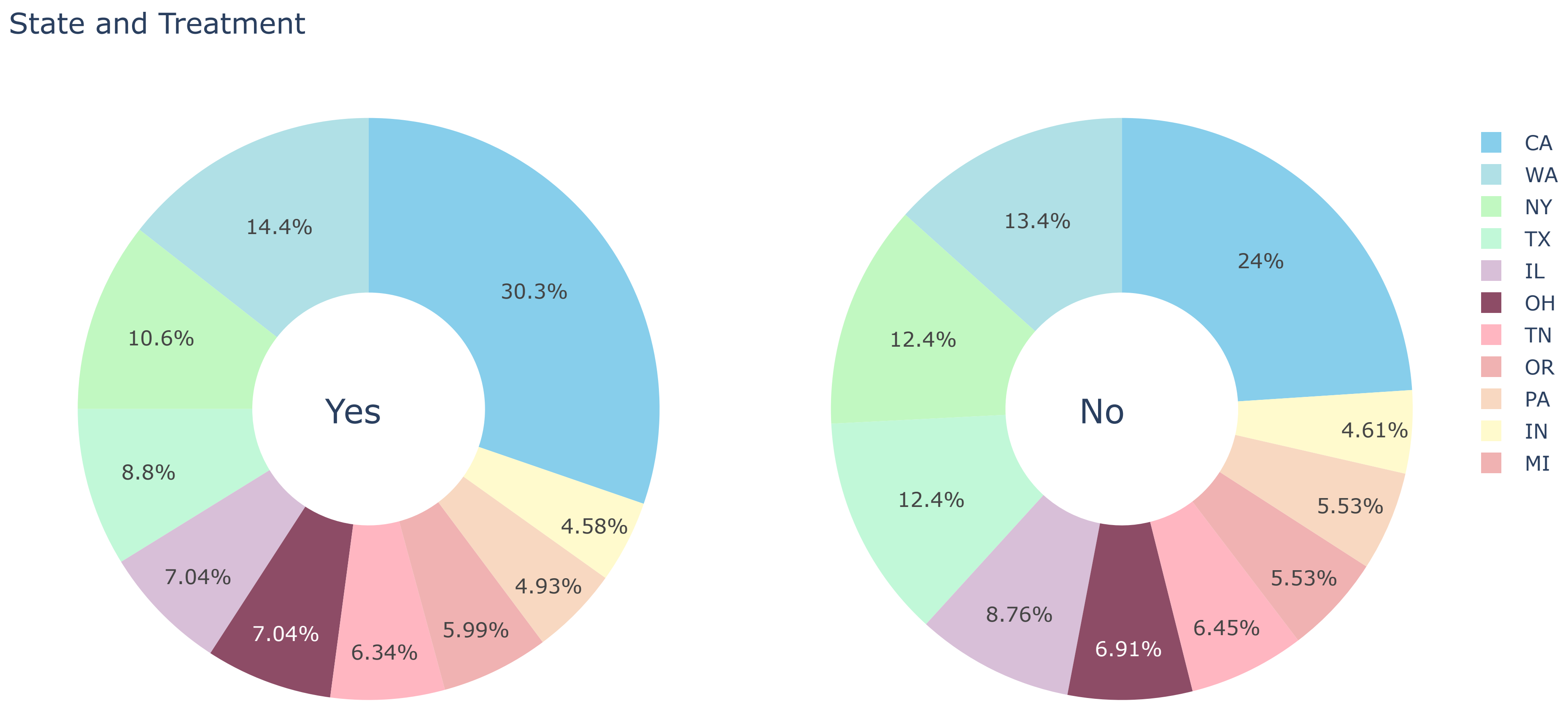
	Age
count	1.259000e+03
mean	7.942815e+07
std	2.818299e+09
min	-1.726000e+03
25%	2.700000e+01
50%	3.100000e+01
75%	3.600000e+01
max	1.000000e+11

CREATING PIE CHARTS

Comparing the distribution of states among survey respondents who answers 'yes' or 'no'

In [9]:

```python
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=df['state'].loc[df['treatment'] == 'Yes'].value_counts().index.to_list()[:10], values=df['state'].loc[df['treatment'] == 'Yes'].value_counts()[:10], name="Treatment - Yes", marker=dict(colors=palette)),
              1, 1)
fig.add_trace(go.Pie(labels=df['state'].loc[df['treatment'] == 'No'].value_counts().index.to_list()[:10], values=df['state'].loc[df['treatment'] == 'No'].value_counts()[:10], name="Treatment - No", marker=dict(colors=palette)),
              1, 2)

fig.update_traces(hole=0.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="State and Treatment",
    annotations=[dict(text='Yes', x=0.19, y=0.5, font_size=20, showarrow=False),
                 dict(text='No', x=0.78, y=0.5, font_size=20, showarrow=False)]
)

fig.show()
```



DROPPING UNNECESSARY COLUMNS

In [10]:

```
df = df.drop(columns=['state','comments', 'Timestamp'])
print('\033[1m' + 'Columns in updated Dataframe :' + '\033[0m', len(df.columns))
```

```
Columns in updated Dataframe : 24
```

DATA INTERPRETATION

Imputing missing values and displaying the total count of remaining empty values

In [11]:

```
df['self_employed'] = df['self_employed']\
                      .fillna(pd.Series(np.random.choice(['Yes', 'No'], p=[0.117647, 0.882353], size=len(df))))

df['work_interfere'] = df['work_interfere']\
                      .fillna(pd.Series(np.random.choice(['Sometimes', 'Never', 'Rarely', 'Often']
                                                         , p=[0.467337, 0.214070, 0.173869, 0.144724], size=len(df))))
print('\033[1m' + 'Total empty values in the Dataset :' + '\033[0m' , df.isnull().sum().sum())
```

```
Total empty values in the Dataset : 0
```

DATA ENCODING

In [12]:

```
df_encoding = df
encoder = OrdinalEncoder()
df_encoding = encoder.fit_transform(df.drop(['Age'], axis=1))
```

In [13]:

```
df_encoding['Age'] = df.Age
df_encoding.head(10)
```

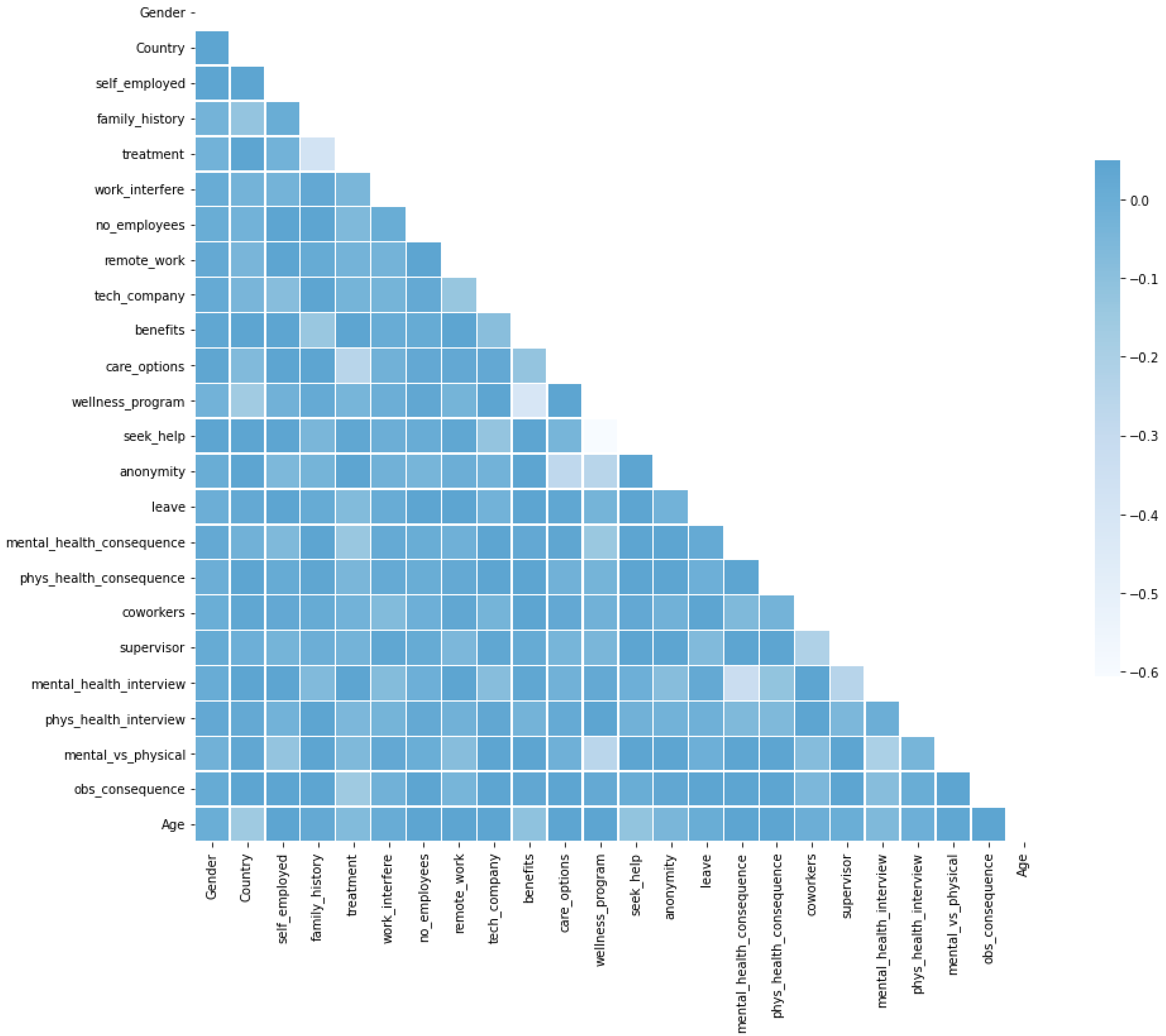
Out[13]:

employees	remote_work	tech_company	benefits	...	leave	mental_health_consequence	phys_health_consequence	coworkers	supervisor	mental_health_interview	phys_health_interview	mental_vs_physical	obs_consequence	Age
1	1	1	1	...	1	1	1	1	1	1	1	1	1	37
2	1	2	2	...	2	2	1	2	2	1	2	2	1	44
1	1	1	3	...	3	1	1	3	1	2	3	3	1	32
3	1	1	3	...	3	3	2	1	2	3	1	3	2	31
4	2	1	1	...	2	1	1	1	1	2	3	2	1	31
1	1	1	1	...	2	1	1	3	1	1	1	2	1	33
5	2	1	3	...	3	2	3	1	2	1	2	2	1	35
5	2	1	3	...	2	1	1	2	2	1	2	3	1	39
4	1	1	1	...	4	2	1	3	1	1	1	3	1	42
3	1	1	2	...	2	1	1	3	1	3	1	1	1	23

CREATING A HEATMAP

In [14]:

```
corr = df_encoding.corr(method ='spearman')
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 15))
cmap = sns.diverging_palette(220, 5, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap='Blues', vmax=.05, center=0,square=True, linewidths=.4, cbar_kws={"shrink": .5})
plt.show()
```


Gender
Country
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
care_options
wellness_program
seek_help
anonymity
leave
mental_health_consequence
phys_health_consequence
coworkers
supervisor
mental_health_interview
phys_health_interview
mental_vs_physical
obs_consequence
Age
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6

CALCULATING VALUES

In [15]:
```
gender_values = df.Gender.value_counts().sort_values(ascending=False).to_frame()
gender_values = gender_values.rename(columns={'Gender': 'count'})
table_gender = gender_values.style.background_gradient(cmap=cmap)
table_gender
```

Out[15]:

	count
Male	615
male	206
Female	121
M	116
female	62
F	38
m	34
f	15
Make	4
Woman	3
Male	3
Female	2
Man	2
Female (trans)	2
Cis Male	2
cis male	1
cis-female/femme	1
Agender	1
msle	1
Neuter	1
woman	1
A little about you	1
maile	1
something kinda male?	1
fluid	1
Mal	1
Female (cis)	1
All	1
Male (CIS)	1
Trans-female	1
Cis Female	1
Guy (-ish) ^_^	1
ostensibly male, unsure what that really means	1
non-binary	1
Mail	1

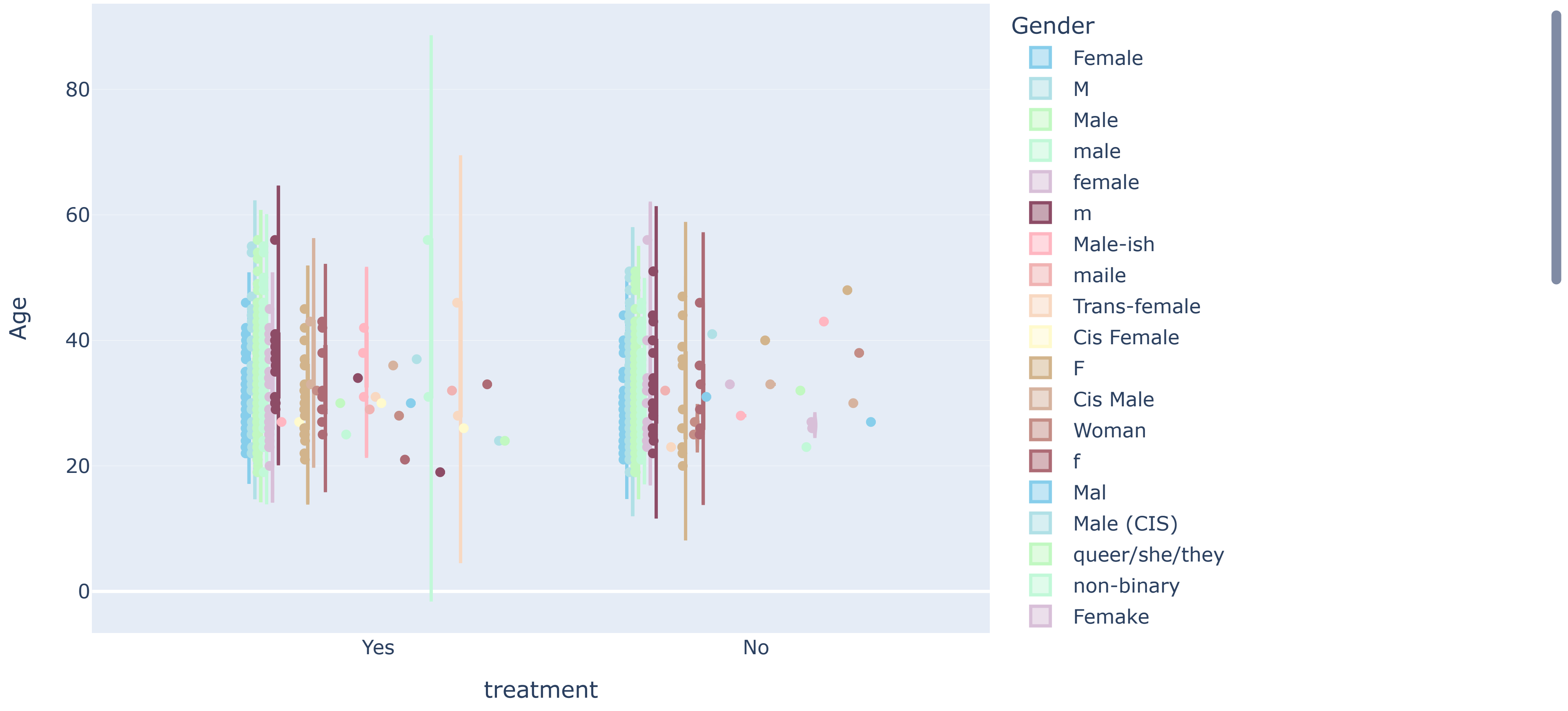
	count
Enby	1
p	1
male leaning androgynous	1
Male-ish	1
Nah	1
Cis Man	1
Genderqueer	1
Malr	1
femail	1
queer/she/they	1
Femake	1
Trans woman	1
queer	1
Androgyne	1

REMOVING OUTLIERS

In [16]:

```
def winsorization_outliers(df):
    out=[]
    for i in df:
        q1 = np.percentile(df , 1)
        q3 = np.percentile(df , 99)
        if i > q3 or i < q1:
            out.append(i)
    print("Outliers:",out)
    return out
outliers = winsorization_outliers(df.Age)
data_age = df.loc[~df.Age.isin(outliers)]
fig = px.violin(data_age, y="Age", x="treatment", color="Gender", box=True, points="all", color_discrete_sequence=palette)
fig.show()
```

```
Outliers: [18, 18, 18, -29, 18, 18, 60, 329, 99999999999, 57, 58, 57, 18, 18, 62, 65, 57, -1726, 5, 61, 8, 11, -1, 72, 60]
```



AGE WINSORIZATION & GENDER CATEGORIZATION

In [17]:

```
age = []
for i in df.Age:
    if (i<18) or (i>99):
        age.append(31)
    else:
        age.append(i)
df['Age'] = age

other  = ['A little about you', 'p', 'Nah', 'Enby', 'Trans-female','something kinda male?','queer/she/they','non-binary','All','fluid', 'Genderqueer','Androgyne', 'Agender','Guy (-ish)
^_^', 'male leaning androgynous','Trans woman','Neuter', 'Female (trans)','queer','ostensibly male, unsure what that really means','trans']
male   = ['male', 'Maleleaning androgynous', 'cis Male','something kinda Male?', 'Male(CIS)','ostensibly Male, unsure what that really means','cis Male','Male','M', 'm', 'Male-ish', 'm
aile','Cis Male','Mal', 'Male (CIS)','Make','Male ', 'Man', 'msle','cis male', 'Cis Man','Malr','Mail']
female = ['Female', 'Female(cis)','Trans-Female','cis-Female/femme','Trans Female','Female(trans)', 'Female(cis)''female','Cis Female', 'F','f','Femake', 'woman','Female ','cis-female/
femme','Female (cis)','femail','Woman','female']

df['Gender'].replace(to_replace = other, value = 'other', inplace=True)
df['Gender'].replace(to_replace = male, value = 'M', inplace=True)
df['Gender'].replace(to_replace = female, value = 'F', inplace=True)

print('\033[1m' + 'Unique values in updated Gender column :' + '\033[0m', df.Gender.unique())
print('\033[1m' + 'Range of column Age (Before) :' + '\033[0m', (df.Age.min(), df.Age.max()))
print('\033[1m' + 'Range of column Age :' + '\033[0m', (df.Age.min(), df.Age.max()))
```

```
Unique values in updated Gender column : ['F' 'M' 'other']
Range of column Age (Before) : (18, 72)
Range of column Age : (18, 72)
```

NORMALISING VALUE COUNTS

In [18]:

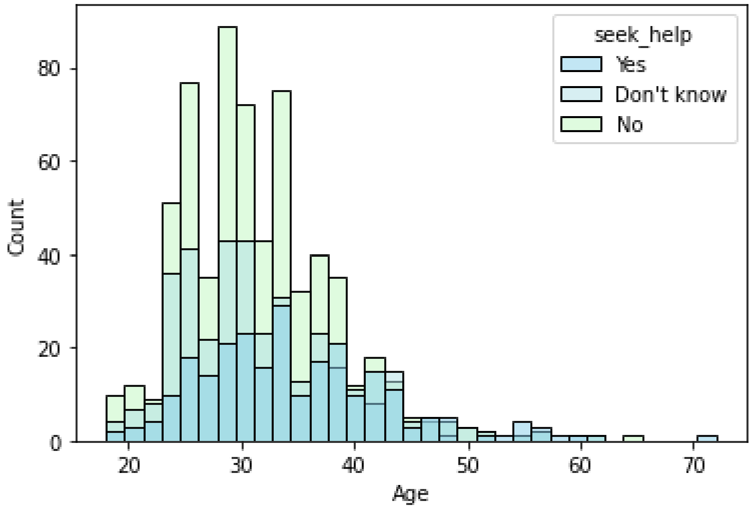
```
prop = pd.DataFrame(df.Gender.value_counts(normalize=True))
prop
```

Out[18]:

	Gender
M	0.787133
F	0.196187
other	0.016680

In [19]:

```
a= sns.histplot(df,x='Age',hue='seek_help', palette=palette)
```



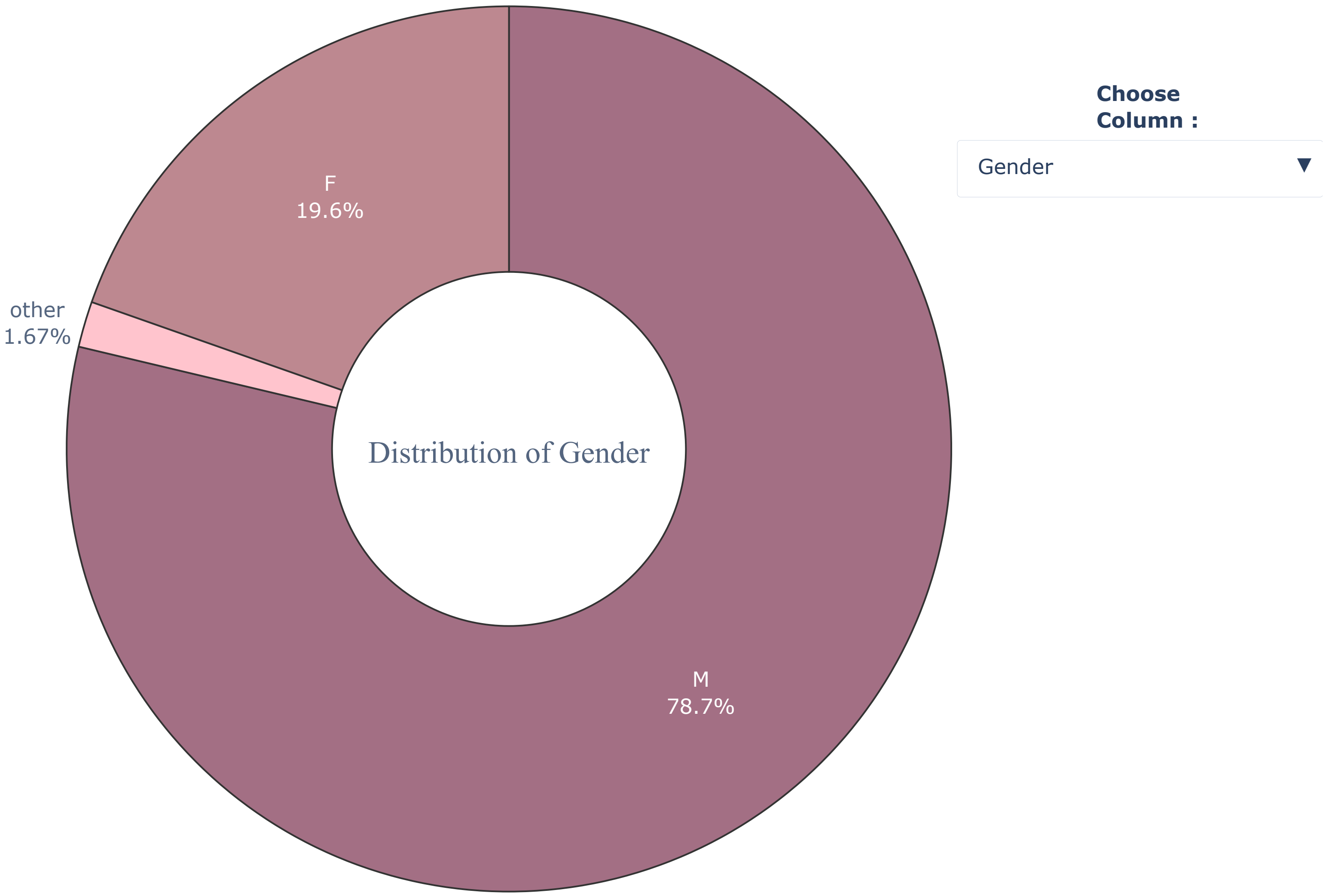
In [20]:

```
df_ = df.drop(['Age', 'Country'], axis=1)
```

DISTRIBUTION OF GENDER

In [21]:

```
import plotly.graph_objects as go

buttons = []
i = 0
vis = [False] * 24

for col in df_.columns:
    vis[i] = True
    buttons.append({
        'label': col,
        'method': 'update',
        'args': [{'visible': vis}, {'title': col}]
    })
    i += 1
    vis = [False] * 24

fig = go.Figure()

for col in df_.columns:
    fig.add_trace(go.Pie(
        values=df_[col].value_counts(),
        labels=df_[col].value_counts().index,
        title=dict(text='Distribution of {}'.format(col),
                   font=dict(size=18, family='Times New Roman')),
        hole=0.4,
        hoverinfo='label+percent',
    ))

fig.update_traces(
    hoverinfo='label+percent',
    textinfo='label+percent',
    textfont_size=12,
    opacity=0.8,
    showlegend=False,
    marker=dict(colors=sns.color_palette(palette1).as_hex(),
                line=dict(color='#000000', width=1))
)

fig.update_layout(
    margin=dict(t=0, b=0, l=0, r=0),
    updatemenus=[dict(
        type='dropdown',
        x=1.15,
        y=0.85,
        showactive=True,
        active=0,
        buttons=buttons
    )],
    annotations=[
        dict(text="<b>Choose<br>Column<b> : ",
             showarrow=False,
             x=1.06, y=0.92, yref="paper", align="left")
    ]
)

for i in range(1, 22):
    fig.data[i].visible = False

fig.show()
```



In [22]:

```
male   = df[df.Gender == 'M'].drop(['Gender', 'Age', 'Country'], axis=1)
female = df[df.Gender == 'F'].drop(['Gender', 'Age', 'Country'], axis=1)
other  = df[df.Gender == 'other'].drop(['Gender', 'Age', 'Country'], axis=1)
```

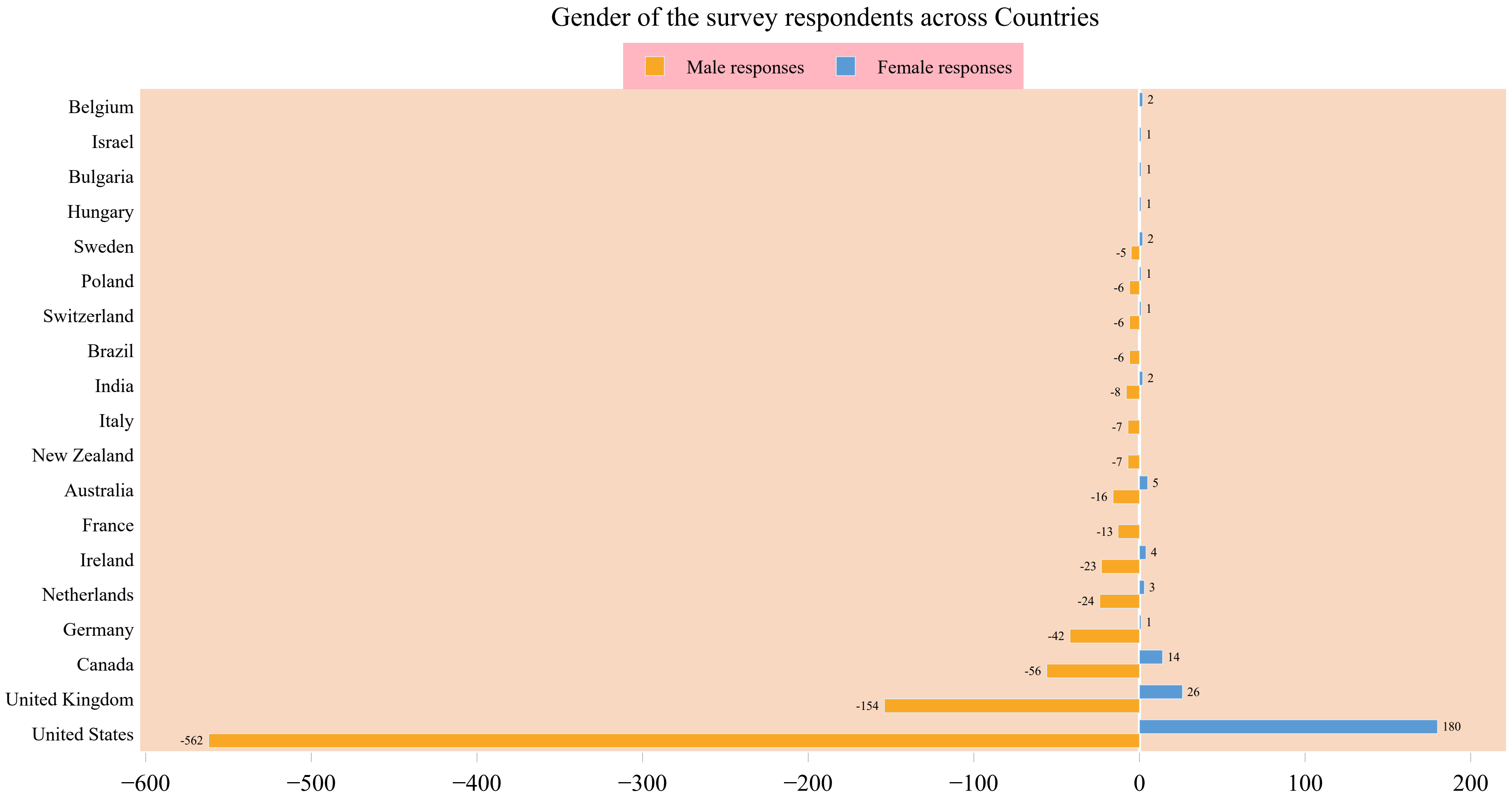
PLOTTING TOP 15 COUNTRIES

In [23]:

```
male_country = df[df['Gender'] == 'M'][['Country', 'Gender']]
female_country = df[df['Gender'] == 'F'][['Country', 'Gender']]
male_country_counts = male_country['Country'].value_counts().reset_index().rename(columns={'index': 'Country', 'Country': 'count'}).head(15)
female_country_counts = female_country['Country'].value_counts().reset_index().rename(columns={'index': 'Country', 'Country': 'count'}).head(15)
male_country_counts['count'] = male_country_counts['count'] * -1
```

In [24]:

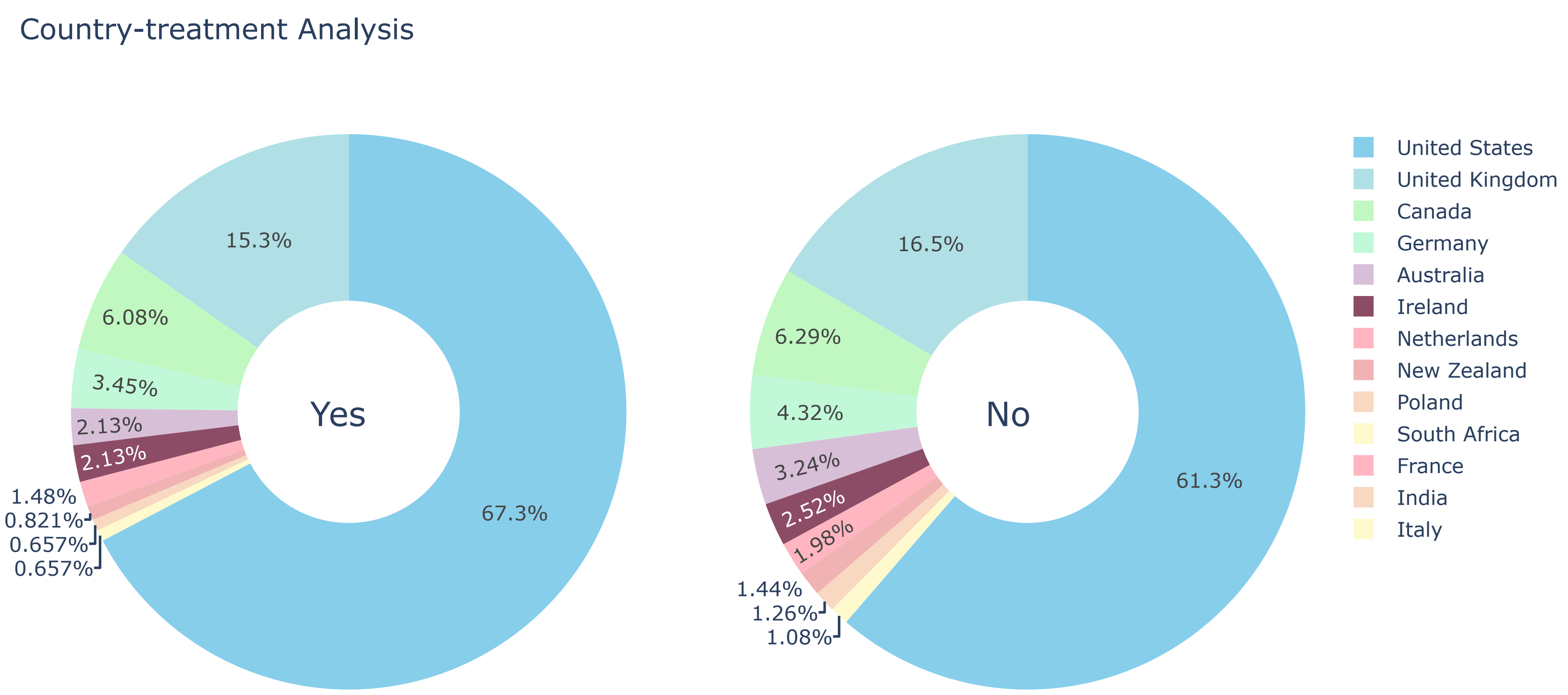
```
import plotly.graph_objects as go
male_country_counts = male_country['Country'].value_counts().reset_index().rename(columns={'index': 'Country', 'Country': 'count'}).head(15)
male_country_counts['count'] = male_country_counts['count'] * -1
fig = go.Figure()
fig.add_trace(go.Bar(
    y=male_country_counts['Country'],
    x=male_country_counts['count'],
    text=male_country_counts['count'],
    textfont=dict(size=10, color='black'),
    textposition='outside',
    name='Male responses',
    marker_color="#F9A825",
    orientation='h'
))

fig.add_trace(go.Bar(
    y=female_country_counts['Country'],
    x=female_country_counts['count'],
    text=female_country_counts['count'],
    textfont=dict(size=10, color='black'),
    textposition='outside',
    name='Female responses',
    marker_color="#5A9BD5",
    orientation='h'
))

fig.update_xaxes(
    tickfont=dict(size=15),
    tickmode='array',
    ticklen=6,
    showline=False,
    showgrid=False,
    ticks='outside'
)
fig.update_yaxes(
    showgrid=False,
    categoryorder='total ascending',
    ticksuffix=' ',
    showline=False
)

fig.update_layout(
    font_family='Times New Roman',
    title=dict(text='Gender of the survey respondents across Countries', x=0.525),
    margin=dict(t=80, b=0, l=70, r=40),
    hovermode="y unified",
    plot_bgcolor="#F8D8C1",
    paper_bgcolor="#FFB6C1",
    font=dict(color='black'),
    legend=dict(orientation="h", yanchor="bottom", y=1, xanchor="center", x=0.5),
    hoverlabel=dict(bgcolor="#FFFFCC", font_size=13, font_family="Times New Roman")
)

fig.show()
```



COUNTRY-WISE ANALYSIS

Comparing the distribution of 'treatment' variable amongst different countries

In [25]:
```
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=df.Country.loc[df.treatment == 'Yes'].value_counts().index.to_list()[:10], values = df.Country.loc[df.treatment == 'Yes'].value_counts()[:10], name="Treatment -Yes", marker=dict(colors=palette)),
              1, 1)
fig.add_trace(go.Pie(labels=df.Country.loc[df.treatment == 'No'].value_counts().index.to_list()[:10], values = df.Country.loc[df.treatment == 'No'].value_counts()[:10], name="Treatment - No", marker=dict(colors=palette)),
              1, 2)

fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="Country-treatment Analysis",

    annotations=[dict(text='Yes', x=0.19, y=0.5, font_size=20, showarrow=False),
                 dict(text='No', x=0.78, y=0.5, font_size=20, showarrow=False)])
fig.show()
```



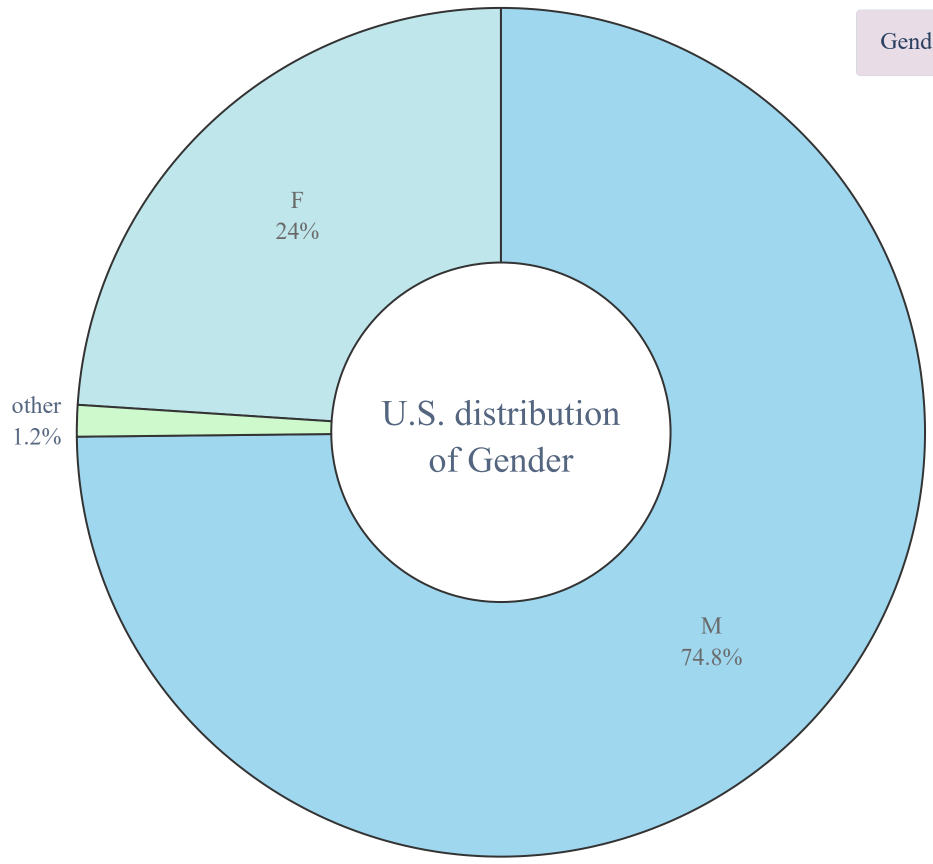
In [26]:
```
us = df[df.Country == 'United States'].drop(['Age', 'Country'], axis=1)
uk = df[df.Country == 'United Kingdom'].drop(['Age', 'Country'], axis=1)
cd = df[df.Country == 'Canada'].drop(['Age', 'Country'], axis=1)
gr = df[df.Country == 'Germany'].drop(['Age', 'Country'], axis=1)
```

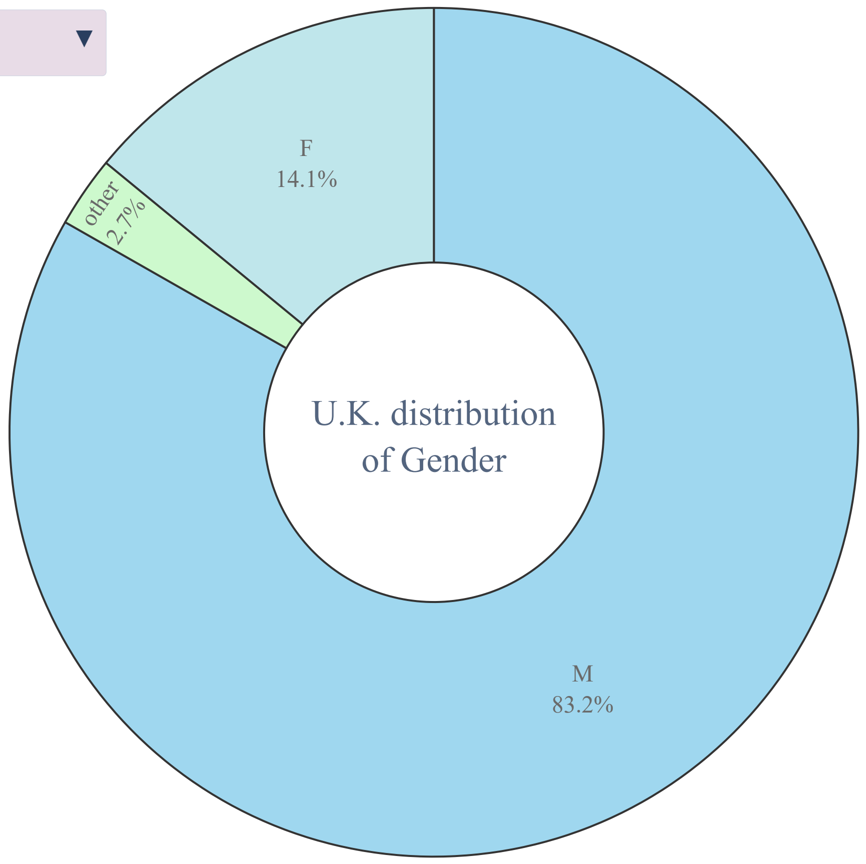
In [27]:

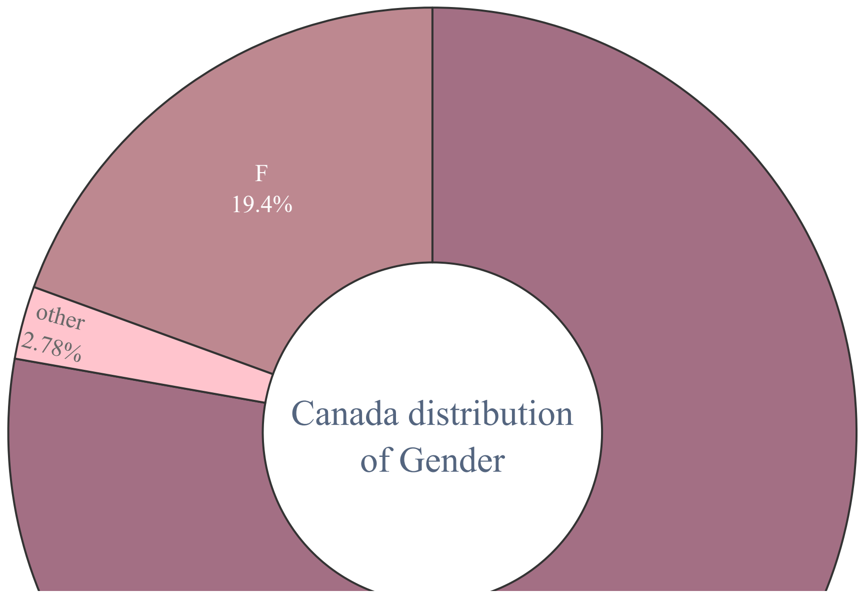
```
buttons = []
i = 0
vis = [False] * 22
for col in us.columns:
    vis[i] = True
    buttons.append({'label' : col,
             'method' : 'update',
             'args'   : [{'visible' : vis},
             {'title'  : col}] })
    i+=1
    vis = [False] * 22
fig = make_subplots(rows=2, cols=2,
                    specs=[[{'type':'domain'}, {'type':'domain'}], [{'type':'domain'}, {'type':'domain'}]],
                    vertical_spacing = 0.07)
for col in us.columns:
    fig.add_trace(go.Pie(
             values = us[col].value_counts(),
             labels = us[col].value_counts().index,
             title = dict(text = 'U.S. distribution<br>of {}'.format(col),
                          font = dict(size=18, family = 'Times New Roman'),
                          ),
             hole = 0.4,
             hoverinfo='label+percent',),1,1)
for col in uk.columns:
    fig.add_trace(go.Pie(
             values = uk[col].value_counts(),
             labels = uk[col].value_counts().index,
             title = dict(text = 'U.K. distribution<br>of {}'.format(col),
                          font = dict(size=18, family = 'Times New Roman'),
                          ),
             hole = 0.4,
             hoverinfo='label+percent',),1,2)
for col in cd.columns:
    fig.add_trace(go.Pie(
             values = cd[col].value_counts(),
             labels = cd[col].value_counts().index,
             title = dict(text = 'Canada distribution<br>of {}'.format(col),
                          font = dict(size=18, family = 'Times New Roman'),
                          ),
             hole = 0.4,
             hoverinfo='label+percent',),2,1)
for col in gr.columns:
    fig.add_trace(go.Pie(
             values = gr[col].value_counts(),
             labels = gr[col].value_counts().index,
             title = dict(text = 'Germany distribution<br>of {}'.format(col),
                          font = dict(size=18, family = 'Times New Roman'),
                          ),
             hole = 0.4,
             hoverinfo='label+percent',),2,2)
fig.update_traces(hoverinfo='label+percent',
                   textinfo='label+percent',
                  textfont_size=12,
                  opacity = 0.8,
                  showlegend = False,
                  marker = dict(colors = sns.color_palette(palette).as_hex(),
                              line=dict(color='#000000', width=1)))
fig.update_traces(row=2, col=1, hoverinfo='label+percent',
                  textinfo='label+percent',
                  textfont_size=12,
                  opacity = 0.8,
                  showlegend = False,
                  marker = dict(colors = sns.color_palette(palette1).as_hex(),
                                line=dict(color='#000000', width=1)))
fig.update_traces(row=2, col=2, hoverinfo='label+percent',
                  textinfo='label+percent',
                  textfont_size=12,
                  opacity = 0.8,
                  showlegend = False,
                  marker = dict(colors = sns.color_palette(palette1).as_hex(),
                              line=dict(color='#000000', width=1)))
fig.update_layout(margin=dict(t=0, b=0, l=0, r=0),
                   paper_bgcolor = "#E8DCE7",
                  height = 1200,
                  font_family   = 'Times New Roman',
                  updatemenus = [dict(
                        type = 'dropdown',
                        x = 0.60,
                        y = 0.96,
                        showactive = True,
                        active = 0,
                        buttons = buttons)],
                 annotations=[
                             dict(text = "<b>Choose<br>Column<b> : ",
                                  font = dict(size = 14),
                             showarrow=False,
                             x = 0.48, y = 1, yref = "paper", align = "right")])
for i in range(0,88):
    fig.data[i].visible = False
fig.data[0].visible = True
fig.data[22].visible = True
fig.data[44].visible = True
fig.data[66].visible = True
fig.show()
```

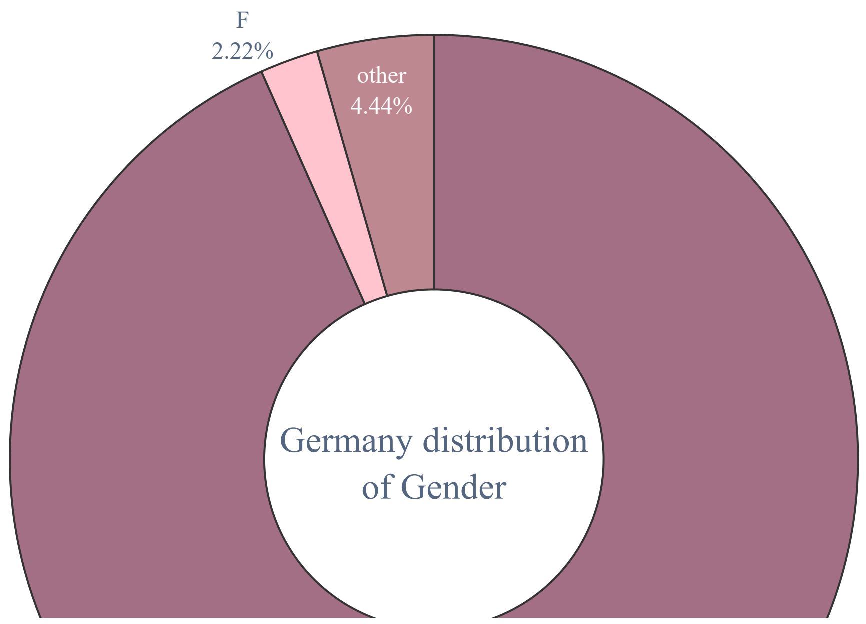
Choose Column :

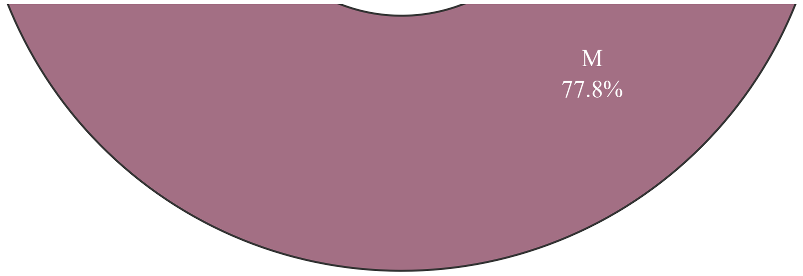
Gender ▼

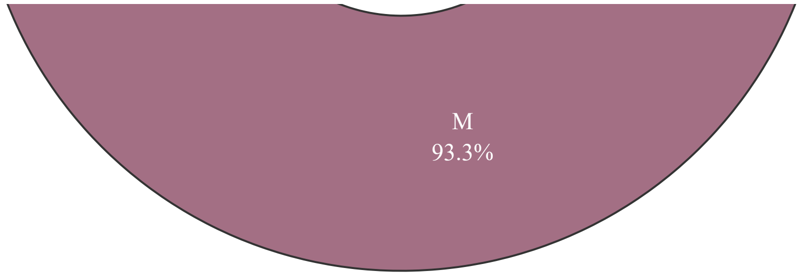












```
In [ ]:
```