Business Center Effectivness analysis

The outputs were cleard since I don't have the rights over the data...

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib
!pip install linearmodels
from linearmodels import PanelOLS
from linearmodels import RandomEffects
pd.set_option('display.max_colwidth', 999,'max_rows',999,'display.max_rows',999)
```

Import and declare as Panel Data

In [ ]:
```
df = pd.read_excel("••2019 מסד מאוחד - שבר 2017-2018 וותיקים-PARETO54.xlsx", encoding='utf
df=df.sort_values(by=['id',"שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)"])
survey_year = pd.Categorical(df["שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)"])
df = df.set_index(['id', "שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)"], drop=Fals
df.head()
```

In [ ]:
```
print("There are %s observations and %s features."% (df.shape[0],df.shape[1]))
print("The columns numbers and names are:")
for i in range(0, df.shape[1]):
    print("%s. %s" % (i, df.columns[i]))
cols = df.columns
```

Data Pre-processing

convert qunatitiave features to floats

These are some questions who were classified as objects even though they are numbers (float, int or datetime):

In [ ]:
```
non_informative_features = 'id', 'טלפון','שם לקוח','מס"ד לסקר הספציפי בשנה הספציפית','אימייל','ה
df[df.columns.difference(non_informative_features)].dtypes.value_counts()
```

In [ ]:
```
object_questions=df[df.columns.difference(non_informative_features)].dtypes[(df.dtypes
object_questions
for i in range(len(object_questions)):
    print("%s. %s" % (i, object_questions[i]))
```

In [ ]:
```
df[object_questions].T
```

In [ ]:
```
df[object_questions[6]].value_counts()
```

In [ ]:
```
df[object_questions[6]].loc[df[object_questions[6]]=="1=["לא
df[object_questions[6]]=pd.to_numeric(df[object_questions[6]] , downcast='integer')
```

In [ ]:
```
df[object_questions[36]].unique()
```

In [ ]:
```
df[object_questions[36]].loc[df[object_questions[36]]=='2013-2014']=2014
df[object_questions[36]].loc[df[object_questions[36]]=='12/2017']=2017
df[object_questions[36]].loc[df[object_questions[36]]=='12/2017']=2017
```

In [ ]:
```
df.loc[df[object_questions[36]]=='2017-2018']
```

Education

copy education of 2019 to same people in 2017-2018

In [ ]:
```
df["participated_in_2019"]=(df["1*(2019 == ["(סקר ותיקים - 2019 ,שביעות רצון - 2017-2018) 
df.participated_in_2019.loc[df.id.isin(df.loc[df["participated_in_2019"]==1].id.tolist
# if the subject participated in 2 survey's in which one of the is 2019, update the no
to_udpate=df[(df["2==["מספר הסקרים בהם השתתף) & (df["participated_in_2019"]==True)]["בלבד
df.loc[(df["2==["מספר הסקרים בהם השתתף) & (df["participated_in_2019"]==True),"בלבד 2019 '
# if the subject participated in 3 survey's, update the non-2019 to the same eduction
to_udpate=df[(df["3==["מספר הסקרים בהם השתתף) & (df["participated_in_2019"]==True)]["בלבד
df.loc[(df["3==["מספר הסקרים בהם השתתף) & (df["participated_in_2019"]==True),"בלבד 2019 '
```

unify duplicate education categories

In [ ]:
```
print(df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].value_counts(dropna=False))
df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].loc[df["בלבד 2019 'ש  - Q25 תיכון"=["תיכונית"==["?השכלתך
df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].loc[df["בלבד 2019 'ש  - Q25 תעודה, סמינר או קורס מקצועי 
df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].loc[df["בלבד 2019 'ש  - Q25 תעודה, סמינר או קורס מקצועי 
df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].loc[df["בלבד 2019 'ש  - Q25 תעודה, סמינר או קורס מקצועי 
df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].loc[df["בלבד 2019 'ש  - Q25 מסרב"==["?מהי השכלתך"]=np
df["בלבד 2019 'ש  - Q25 השכלתך מהי?"].value_counts(dropna=False)
```

unify education categories to academic/non academic

In [ ]:
```
df["is_academic_education"]=df["בלבד 2019 'ש  - Q25 השכלתך מהי?"]
df["is_academic_education"].loc[df["is_academic_education"]=="אקדמית (תואר ראשון ומעלה"=["
df["is_academic_education"].loc[df["is_academic_education"]=="אקדמית (תואר ראשון ומעלה"=["
df["is_academic_education"].loc[df["is_academic_education"]=="לא-אקדמית (תיכונית ועל-תיכונית"
df["is_academic_education"].loc[df["is_academic_education"]=="לא-אקדמית (תיכונית ועל-תיכונית"
print("All years:")
print(df["is_academic_education"].value_counts(dropna=False))
#print("2009:")
#df_2019["is_academic_education"].value_counts(dropna=False)
```

In [ ]:
```python
def reverse(s):
    str = ""
    for i in s:
        str = i + str
    return str
def reverse_index(dataframe):
    arr=[]
    for j in dataframe.index:
        arr.append(reverse(j))
    return arr
df_educ = df["ש' 2019 בלבד  - Q25 מהי השכלתך?"].value_counts()
df_educ.index=reverse_index(df_educ)
df_educ.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='highest diploma, all y
df_educ.index=reverse_index(df_educ)
```

Sex

Identify gender by name

In [ ]:
```python
print(df["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].value_counts(dr
df["שם פרטי"]=df["שם לקוח"].str.split(expand=True)[0]
df["שם משפחה"]=df["שם לקוח"].str.split(expand=True)[1]
df_sex_nan_names=df["שם פרטי"].loc[df["ש' 2019 בלבד - Q26 מין המרואיין : (להקריא אלא למלא לבד)
df_sex_nan_family_names=df["שם משפחה"].loc[df["ש' 2019 בלבד - Q26 מין המרואיין : (למלא לבד)
not_identified_names = df_sex_nan_names[[2,4,6,7,11,14,15,19,21,23,24,39,40,45,47,51,5
print("not_identified_names:",not_identified_names)
identified_femals = df_sex_nan_names[[13,26,27,30,36,37,38,44,50,76,83,97,100,104,109,
print("identified_femals:",identified_femals)
not_males = []
not_males.append(list(not_identified_names))
not_males.append(list(identified_femals))
flat_not_males = [item for sublist in not_males for item in sublist]
df["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].loc[(~df['שם פרטי'].is
df["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].loc[df["שם פרטי"].isin
df["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].value_counts(dropna=F
#df_R3_R5_2019["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].value_coun
```

In [ ]:
```python
df_2019["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].value_counts(dro
```

In [ ]:
```python
df_R1_R2_2019["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].value_coun
```

In [ ]:
```python
df_R3_R5_2019["ש' 2019 בלבד - Q26 מין המרואיין : (סוקר נא לא להקריא אלא למלא לבד) "].value_coun
```

Age

copy age of 2019 to same people in 2017-2018 (minus 1 or 2)

(first create participated_in_2018 and participated_in_2017)

In [ ]:
```
print(df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].value_counts(dropna=False,bins=3).sor
df["participated_in_2018"]=(df["1*(2018 == ["(ותיקים סקר - 2019 ,רצון שביעות - 2017-2018)
df.participated_in_2018.loc[df.id.isin(df.loc[df["participated_in_2018"]==1].id.tolist
df["participated_in_2018"].value_counts()

df["participated_in_2017"]=(df["1*(2017 == ["(ותיקים סקר - 2019 ,רצון שביעות - 2017-2018)
df.participated_in_2017.loc[df.id.isin(df.loc[df["participated_in_2017"]==1].id.tolist

# 2019 to 2018
to_udpate=df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==1)&(df["pa
to_update=df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==1)&(df["pa

# 2018 to 2017
to_udpate=df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==1)&(df["pa
to_update=df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==1)&(df["pa

# 2019 to 2017
to_udpate=df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==0)&(df["pa
to_update=df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].loc[(df["participated_in_2018"]==0)&(df["pa

df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].value_counts(dropna=False,bins=3).sort_inde
```

In [ ]:
```
pd.cut(df["בלבד 2019 'ש  - KEY14 16,34,66] ,["?ה/את כמה בת /בן]).value_counts().sort_ind
#pd.cut(df_2019["בלבד 2019 'ש  - KEY14 16,34,67] ,["?ה/את כמה בת /בן]).value_counts().sor
```

In [ ]:
```
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].value_counts().sort_index().plot(kind='bar'
```

In [ ]:
```
print(df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].std())
df["בלבד 2019 'ש  - KEY14 ה/את כמה בת /בן?"].mean()
```

Business sector

domain is the participants reported domain (it's very diverse and unorganized), sector is 6 category sector using the 1993 CBS classification

copy business domain of 2019 to same people in 2017-2018

In [ ]:
```
print(df["מדווח - עיסוק תחום"].loc[(df["2019 =! ["(ותיקים סקר - 2019 ,רצון שביעות - 2017-2018)
to_udpate=df[(df["2==["השתתף בהם הסקרים מספר) & (df["participated_in_2019"]==True)]["מדווח
df.loc[(df["2==["השתתף בהם הסקרים מספר) & (df["participated_in_2019"]==True),"מדווח - עיסוק
to_udpate=df[(df["3==["השתתף בהם הסקרים מספר) & (df["participated_in_2019"]==True)]["מדווח
df.loc[(df["3==["השתתף בהם הסקרים מספר) & (df["participated_in_2019"]==True),"מדווח - עיסוק
df["מדווח - עיסוק תחום"].loc[(df["2019 =! ["(ותיקים סקר - 2019 ,רצון שביעות - 2017-2018) הסקר ת
```

unify businees domain categories to economic sector (1993 classification)

[https://old.cbs.gov.il/publications13/1544_seker_mischar_2010/pdf/t01.pdf
(https://old.cbs.gov.il/publications13/1544_seker_mischar_2010/pdf/t01.pdf)](https://old.cbs.gov.il/publications13/1544_seker_mischar_2010/pdf/t01.pdf)

In [ ]:
```
מסחר_תיקון_כלי_רכב_ותיקונים_אחרים=df["תחום עיסוק - מדווח"].unique()[[4,16,23,68,74,76,79,110,83,5
print("מסחר_תיקון_כלי_רכב_ותיקונים_אחרים:",מסחר_תיקון_כלי_רכב_ותיקונים_אחרים)
תחבורה_אחסנה_ותקשורת=df["תחום עיסוק - מדווח"].unique()[[29,41,52,70,86,95,164,139,141]].tolis
print("")
print("תחבורה_אחסנה_ותקשורת:",תחבורה_אחסנה_ותקשורת)
בנקאות_ביטוח_ומוסדות_פיננסיים_אחרים=df["תחום עיסוק - מדווח"].unique()[[48,36,81,118]].tolist()
print("")
print("בנקאות_ביטוח_ומוסדות_פיננסיים_אחרים:",בנקאות_ביטוח_ומוסדות_פיננסיים_אחרים)
שירותי_חינוך_בריאות_וסעד_עסקיים=df["תחום עיסוק - מדווח"].unique()[[22,21,27,33,100,102,28,113,34,
print("")
print("שירותי_חינוך_בריאות_וסעד_עסקיים:",שירותי_חינוך_בריאות_וסעד_עסקיים)
שירותים_חברתיים_אישיים_ואחרים=df["תחום עיסוק - מדווח"].unique()[[1,9,14,20,51,88,94,98,46,162,40
print("")
print("שירותים_חברתיים_אישיים_ואחרים:",שירותים_חברתיים_אישיים_ואחרים)


אחסנה_ותקשורת,בנקאות_ביטוח_ומוסדות_פיננסיים_אחרים,שירותי_חינוך_בריאות_וסעד_עסקיים,שירותים_חברתיים_אישיים_ואחרים
flattened = [item for sublist in הכל_חוץ_משירותים_עסקיים for item in sublist]
הכל_חוץ_משירותים_עסקיים=flattened
def diff(first, second):
        second = set(second)
        return [item for item in first if item not in second]
שירותים_עסקיים=diff(df["תחום עיסוק - מדווח"].unique(),הכל_חוץ_משירותים_עסקיים)[1:]

print("")
print("שירותים_עסקיים:",שירותים_עסקיים)

df["business_sector"]=df["תחום עיסוק - מדווח"]
df['business_sector'].loc[df['business_sector'].isin(חרים)]="מסחר_תיקון_כלי_רכב_ותיקונים_אחרים"
df['business_sector'].loc[df['business_sector'].isin(אחסנה_ותקשורת)]="תחבורה_אחסנה_ותקשורת"
df['business_sector'].loc[df['business_sector'].isin(רים)]="בנקאות_ביטוח_ומוסדות_פיננסיים_אחרים"
df['business_sector'].loc[df['business_sector'].isin(עסקיים)]="שירותי_חינוך_בריאות_וסעד_עסקיים"
df['business_sector'].loc[df['business_sector'].isin(ואחרים)]="שירותים_חברתיים_אישיים_ואחרים"
df['business_sector'].loc[df['business_sector'].isin(שירותים_עסקיים)]="שירותים_עסקיים"
df['business_sector'].loc[df['business_sector'].isin(["פיננסיים_אחרים_וגם_תחבורה_אחסנה_ותקשורת
df['business_sector'].value_counts()
```

In [ ]:
```
'''bla=pd.DataFrame(df_2019['business_sector'].value_counts())
bla.index=reverse_index(bla)
bla
xlabels=reverse_index(pd.DataFrame(df_2019['business_sector'].value_counts()))
df_2019['business_sector'].value_counts().plot(kind='bar',fontsize=15,figsize=(19,6),r
new_df_2019.index=reverse_index(new_df_2019)
#plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='2019 %s
plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(12,6),rot=0,title='(2019 (
plot_2019.axhline(15,color='black',linestyle='dashed')
plot_2019.text(-0.64, 15,'15',fontsize=13)
new_df_2019.index=reverse_index(new_df_2019)
'''
df['ש' 2019 בלבד - KEY1 מהו תחום הפעילות של העסק?']
```

Population groups

copy population group of 2019 to same people in 2017-2018

In [ ]:
```
print(df["אוכלוסיה"].loc[(df["שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)"] != 2019
to_udpate=df[(df["מספר הסקרים בהם השתתף"]==2) & (df["participated_in_2019"]==True)]["לוסיה
df.loc[(df["מספר הסקרים בהם השתתף"]==2) & (df["participated_in_2019"]==True),"אוכלוסיה"]=tc
to_udpate=df[(df["מספר הסקרים בהם השתתף"]==3) & (df["participated_in_2019"]==True)]["לוסיה
df.loc[(df["מספר הסקרים בהם השתתף"]==3) & (df["participated_in_2019"]==True),"אוכלוסיה"]=tc
df["אוכלוסיה"].loc[(df["שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)"] != 2019)].valu
```

unify categories into jewish or non jewish

In [ ]:
```
print(df["אוכלוסיה"].value_counts(dropna=False))
df["is_jewish"]=df["אוכלוסיה"]
jewish=df["אוכלוסיה"].value_counts(dropna=False).index[[2,3]]
non_jewish=df["אוכלוסיה"].value_counts(dropna=False).index[[0,4,5]]
print("jewish:",jewish)
print("non jewish:",non_jewish)
df["is_jewish"].loc[df["is_jewish"].isin(jewish)]="יהודים"
df["is_jewish"].loc[df["is_jewish"].isin(non_jewish)]="לא-יהודים"
df["is_jewish"].value_counts(dropna=False)
```

idenity jewish and non-jewish from NaNs using name

In [ ]:
```
df_nan_names=pd.DataFrame(df["שם לקוח"].loc[df["is_jewish"].isna()].unique())
jewish_names=df_nan_names.iloc[[0,4,7,8,9,11,13,14,15,17,18,19,20,22,23,25,28,29,30,31
flattened_jewish_names = [item for sublist in jewish_names for item in sublist]
jewish_names=flattened_jewish_names
non_jewish_names=df_nan_names[0].loc[~df_nan_names[0].isin(flattened_jewish_names)][3:
non_jewish_names.append(df_nan_names[0].loc[~df_nan_names[0].isin(flattened_jewish_nam
#flattened_non_jewish_names = [item for sublist in non_jewish_names for item in sublis
#flattened_non_jewish_names
print("non_jewish_names:",non_jewish_names)
print("jewish_names:",jewish_names)
df["is_jewish"].loc[df["שם לקוח"].isin(jewish_names)]="יהודים"
df["is_jewish"].loc[df["שם לקוח"].isin(non_jewish_names)]="לא-יהודים"
df["is_jewish"].value_counts(dropna=False)
```

Business size

unify to 2 categories only, medium and small

In [ ]:
```
print(df["סוג מספר זיהוי עסק"].value_counts(dropna=False))
df["business_size"]=df["סוג מספר זיהוי עסק"]
medium=df["סוג מספר זיהוי עסק"].unique()[[1,3,4,5]].tolist()
small=df["סוג מספר זיהוי עסק"].unique()[2]
df["business_size"].loc[df["business_size"].isin(medium)]="פרטית, שותפות ותעודת זהות (בינוני)
df["business_size"].loc[df["business_size"]==small]="עוסק פטור (קטן)"
df["business_size"].value_counts(dropna=False)
```

add business to the 2 categories, using the income question from before the business entered the business center

In [ ]:
```
df["מה היה המחזור השנתי של העסק בזמן שנכנס למרכז העסקים ? Q27 - ש' 2019 בלבד"].replace("לא זוכר"
df["מה היה המחזור השנתי של העסק בזמן שנכנס למרכז העסקים ? Q27 - ש' 2019 בלבד"] = pd.to_numeric(
df["business_size"].loc[(df["של העסק בזמן שנכנס למרכז העסקים ? Q27 - ש' 2019 בלבד"]<=98707
df["business_size"].loc[(df["של העסק בזמן שנכנס למרכז העסקים ? Q27 - ש' 2019 בלבד"]>98707
df["business_size"].value_counts(dropna=False)
```

Lastly - Create different dataframe for every year

And in every year keep only relevant questions in which there are any answers (using the questions_responsiveness_year dataframe)

In [ ]:
```
questions_responsiveness = pd.DataFrame(df.count().sort_values(ascending=False))
df_2017 = df.loc[df['שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)'] == 2017]
questions_responsiveness_2017 = pd.DataFrame(df_2017.count().sort_values(ascending=Fal
df_2017 = df_2017[questions_responsiveness_2017[questions_responsiveness_2017.iloc[:,0
questions_responsiveness_2017 = pd.DataFrame(df_2017.count().sort_values(ascending=Fal
df_2018 = df.loc[df['שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)'] == 2018]
questions_responsiveness_2018 = pd.DataFrame(df_2018.count().sort_values(ascending=Fal
df_2018 = df_2018[questions_responsiveness_2018[questions_responsiveness_2018.iloc[:,0
questions_responsiveness_2018 = pd.DataFrame(df_2018.count().sort_values(ascending=Fal
df_2019 = df.loc[df['שנת הסקר (2017-2018 - שביעות רצון, 2019 - סקר ותיקים)'] == 2019]
questions_responsiveness_2019 = pd.DataFrame(df_2019.count().sort_values(ascending=Fal
df_2019 = df_2019[questions_responsiveness_2019[questions_responsiveness_2019.iloc[:,0
questions_responsiveness_2019 = pd.DataFrame(df_2019.count().sort_values(ascending=Fal
df_2017_full = df_2017.loc[df_2017['ענה על חצי מהשאלות בשאלון או יותר'] == 1]
df_2018_full = df_2018.loc[df_2018['ענה על חצי מהשאלות בשאלון או יותר'] == 1]
df_2019_full = df_2019.loc[df_2019['ענה על חצי מהשאלות בשאלון או יותר'] == 1]
df_full = df.loc[df['ענה על חצי מהשאלות בשאלון או יותר'] == 1]
```

Descriptive Statistics - Sample

In [ ]:
```

df[df.columns.difference(non_informative_features)].describe(include='all').T.round(1)

```

The number of times each unique participated in the surverys:

In [ ]:
```
times_participated = pd.DataFrame(columns=["Pooled"])
times_participated["Pooled"] = df["מספר הסקרים בהם השתתף"].value_counts(dropna=False)
# Divide in number of surveys, since for participants and participated twice (thrice)
times_participated.loc[2] = times_participated.loc[2]//2
times_participated.loc[3] = times_participated.loc[3]//3
times_participated=times_participated.iloc[:3]
times_participated.append = times_participated.sum
times_participated=times_participated.reindex([1,2,3])
times_participated.rename(index={1:'once',2:'twice', 3:'thrice'}, inplace=True)
summation = pd.Series(times_participated.sum(axis=0),name='SUM of *unique* participant
times_participated=times_participated.append(summation)
times_participated

```

In [ ]:
```
times_participated.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='times_parti
```

In [ ]:
```
def create_feature_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 2019
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df,df_full
    pooled_df = df
    pooled_df_full = df_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_fu
    years_dfs = df_2019,df_2019_full,df_2018,df_2018_full,df_2017,df_2017_full
    year_2019_dfs = df_2019,df_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'
    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)

    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns*
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/r
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(c
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].valu
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
        count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"

    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='201
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(12,6),rot=0,title='(201
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)

    return new_df_name,new_df_2019,plot_2019
```

In [ ]:
```
groups,groups_2019,groups_2019_plot=create_feature_specific_df("אוכלוסיה","groups")
groups
```

In [ ]:
```
are_jewish,are_jewish_2019,are_jewish_2019_plot=create_feature_specific_df("is_jewish"
are_jewish
```

In [ ]:
```
sex,sex_2019,sex_2019_plot=create_feature_specific_df(": מין המרואיין) Q26 - ש' 2019 בלבד"
sex
```

In [ ]:
```
education,education_2019,education_2019_plot=create_feature_specific_df("ש' 2019 בלבד"
education
```

In [ ]:
```
academic_education,academic_education_2019,academic_education_2019_plot=create_feature
academic_education
```

In [ ]:
```
business_sector,business_sector_2019,business_sector_2019_plot=create_feature_specific
business_sector
```

In [ ]:
```

```

In [ ]:
```
business_domain,business_domain_2019,business_domain_2019_plot=create_feature_specific
business_domain=business_domain.drop(columns=['% 2018','% 2017','% 2018_full','% 2017_
business_domain1 = business_domain.iloc[:22,]
business_domain2 = business_domain.iloc[22:,]
business_domain1
```

In [ ]:
```
business_domain2
```

In [ ]:
```
business_registration,business_registration_2019,business_registration_2019_plot=creat
business_registration=business_registration.drop(columns=['% 2018','% 2017','% 2018_fu
business_registration
```

In [ ]:
```
business_size,business_size_2019,business_size_2019_plot=create_feature_specific_df("b
business_size=business_size.drop(columns=['% 2018','% 2017','% 2018_full','% 2017_full
business_size
```

In [ ]:
```
region,region_2019,region_2019_plot=create_feature_specific_df("אשכול","region")
#region=region.reindex([1,2,3,5,'SUM (not Nan)'])
#region.rename(index={1:'R1. North East (גולן, נצרת, ראש פינה וסכנין)',2:'R2. North West (3 ,
region
```

In [ ]:
```
business_center,business_center_2019,business_center_2019_plot=create_feature_specific
business_center
```

Region specific statistics

In [ ]:
```
df.is_jewish.value_counts()
```

In [ ]:
```
df_2019.is_jewish.value_counts()
```

In [ ]:

```
df_R1 = df.loc[df['1 == ['אשכול']
df_R2 = df.loc[df['2 == ['אשכול']
df_R1_R2 = df.loc[(df['1 == ['אשכול') | (df['2 == ['אשכול')]
df_R3 = df.loc[df['3 == ['אשכול']
df_R5 = df.loc[df['5 == ['אשכול']
df_R3_R5 = df.loc[(df['3 == ['אשכול') | (df['5 == ['אשכול')]

df_R1_full = df_R1.loc[df_R1['1 == ['ענה על חצי מהשאלות בשאלון או יותר']
df_R2_full = df_R2.loc[df_R2['1 == ['ענה על חצי מהשאלות בשאלון או יותר']
df_R1_R2_full = df_R1_R2.loc[df_R1_R2['1 == ['ענה על חצי מהשאלות בשאלון או יותר']
df_R3_full = df_R3.loc[df_R3['1 == ['ענה על חצי מהשאלות בשאלון או יותר']
df_R5_full = df_R5.loc[df_R5['1 == ['ענה על חצי מהשאלות בשאלון או יותר']
df_R3_R5_full = df_R3_R5.loc[df_R3_R5['1 == ['ענה על חצי מהשאלות בשאלון או יותר']

# Create empty regions dfs
df_years = "2017","2018","2019"
df_regions = "R1","R2","R3","R5"
full="","_full"
df_names=[]
for region in df_regions:
    for year in df_years:
        for full_or_not in full:
            df_names.append("df_%s_%s%s"%(region,year,full_or_not))
for df_to_create in df_names:
    exec('{} = pd.DataFrame()'.format(df_to_create))

# Fill empty regions dfs
df_R1_2017 = df_2017.loc[df_2017['1 == ['אשכול']
df_R1_2017_full = df_2017_full.loc[df_2017_full['1 == ['אשכול']
df_R1_2018 = df_2018.loc[df_2018['1 == ['אשכול']
df_R1_2018_full = df_2018_full.loc[df_2018_full['1 == ['אשכול']
df_R1_2019 = df_2019.loc[df_2019['1 == ['אשכול']
df_R1_2019_full = df_2019_full.loc[df_2019_full['1 == ['אשכול']
df_R2_2017 = df_2017.loc[df_2017['2 == ['אשכול']
df_R2_2017_full = df_2017_full.loc[df_2017_full['2 == ['אשכול']
df_R2_2018 = df_2018.loc[df_2018['2 == ['אשכול']
df_R2_2018_full = df_2018_full.loc[df_2018_full['2 == ['אשכול']
df_R2_2019 = df_2019.loc[df_2019['2 == ['אשכול']
df_R2_2019_full = df_2019_full.loc[df_2019_full['2 == ['אשכול']
df_R3_2017 = df_2017.loc[df_2017['3 == ['אשכול']
df_R3_2017_full = df_2017_full.loc[df_2017_full['3 == ['אשכול']
df_R3_2018 = df_2018.loc[df_2018['3 == ['אשכול']
df_R3_2018_full = df_2018_full.loc[df_2018_full['3 == ['אשכול']
df_R3_2019 = df_2019.loc[df_2019['3 == ['אשכול']
df_R3_2019_full = df_2019_full.loc[df_2019_full['3 == ['אשכול']
df_R5_2017 = df_2017.loc[df_2017['5 == ['אשכול']
df_R5_2017_full = df_2017_full.loc[df_2017_full['5 == ['אשכול']
df_R5_2018 = df_2018.loc[df_2018['5 == ['אשכול']
df_R5_2018_full = df_2018_full.loc[df_2018_full['5 == ['אשכול']
df_R5_2019 = df_2019.loc[df_2019['5 == ['אשכול']
df_R5_2019_full = df_2019_full.loc[df_2019_full['5 == ['אשכול']
df_R1_R2_2017 = df_2017.loc[(df_2017['1 == ['אשכול')|(df_2017['2 == ['אשכול')]
df_R1_R2_2018 = df_2018.loc[(df_2018['1 == ['אשכול')|(df_2018['2 == ['אשכול')]
df_R1_R2_2019 = df_2019.loc[(df_2019['1 == ['אשכול')|(df_2019['2 == ['אשכול')]
df_R3_R5_2017 = df_2017.loc[(df_2017['3 == ['אשכול')|(df_2017['5 == ['אשכול')]
df_R3_R5_2018 = df_2018.loc[(df_2018['3 == ['אשכול')|(df_2018['5 == ['אשכול')]
df_R3_R5_2019 = df_2019.loc[(df_2019['3 == ['אשכול')|(df_2019['5 == ['אשכול')]
df_R1_R2_2017_full = df_2017_full.loc[(df_2017_full['1 == ['אשכול')|(df_2017_full['2 ==
df_R1_R2_2018_full = df_2018_full.loc[(df_2018_full['1 == ['אשכול')|(df_2018_full['2 ==
df_R1_R2_2019_full = df_2019_full.loc[(df_2019_full['1 == ['אשכול')|(df_2019_full['2 ==
df_R3_R5_2017_full = df_2017_full.loc[(df_2017_full['3 == ['אשכול')|(df_2017_full['5 ==
```

```python
df_R3_R5_2018_full = df_2018_full.loc[(df_2018_full['3 == ['אשכול')|(df_2018_full['5 ==
df_R3_R5_2019_full = df_2019_full.loc[(df_2019_full['3 == ['אשכול')|(df_2019_full['5 ==

def create_R1_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 201
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df_R1,df_R1_full
    pooled_df = df_R1
    pooled_df_full = df_R1_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_f
    years_dfs = df_R1_2019,df_R1_2019_full,df_R1_2018,df_R1_2018_full,df_R1_2017,df_R
    year_2019_dfs = df_R1_2019,df_R1_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'

    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)
    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].val
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
        count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"

    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='20
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='1 2
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='1
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)

    return new_df_name,new_df_2019,plot_2019

def create_R2_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 201
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df_R2,df_R2_full
    pooled_df = df_R2
    pooled_df_full = df_R2_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_fu
    years_dfs = df_R2_2019,df_R2_2019_full,df_R2_2018,df_R2_2018_full,df_R2_2017,df_R2
    year_2019_dfs = df_R2_2019,df_R2_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'
    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)
    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns*
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/r
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(d
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].valu
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
        count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"

    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='201
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='2 20
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='2 2
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)

    return new_df_name,new_df_2019,plot_2019

def create_R3_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 2019
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df_R3,df_R3_full
    pooled_df = df_R3
    pooled_df_full = df_R3_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_fu
    years_dfs = df_R3_2019,df_R3_2019_full,df_R3_2018,df_R3_2018_full,df_R3_2017,df_R3
    year_2019_dfs = df_R3_2019,df_R3_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'
    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)
    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].valu
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
        count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"
    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='20
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='3 2
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='3
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)
    return new_df_name,new_df_2019,plot_2019

def create_R5_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 201
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df_R5,df_R5_full
    pooled_df = df_R5
    pooled_df_full = df_R5_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_f
    years_dfs = df_R5_2019,df_R5_2019_full,df_R5_2018,df_R5_2018_full,df_R5_2017,df_R
    year_2019_dfs = df_R5_2019,df_R5_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'
    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)
    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].valu
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
            count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"
    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='20
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='5 20
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='5 
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)
    return new_df_name,new_df_2019,plot_2019

def create_R1_R2_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 2019
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df_R1_R2,df_R1_R2_full
    pooled_df = df_R1_R2
    pooled_df_full = df_R1_R2_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_fu
    years_dfs = df_R1_R2_2019,df_R1_R2_2019_full,df_R1_R2_2018,df_R1_R2_2018_full,df_R
    year_2019_dfs = df_R1_R2_2019,df_R1_R2_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'

    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)
    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].valu
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_cou
        count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"

    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='20
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='1 
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='1+
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='1+2
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)

    return new_df_name,new_df_2019,plot_2019

def create_R3_R5_specific_df(feature_name,new_df_name):
    # Define Variables
    columns_names = ["No. Pooled","% 2019","% 2018","% 2017","No. Pooled_full","% 201
    pooled_column = 'No. Pooled'
    pooled_column_full = 'No. Pooled_full'
    pooled_columns = [pooled_column,pooled_column_full]
    pooled_dfs = df_R3_R5,df_R3_R5_full
    pooled_df = df_R3_R5
    pooled_df_full = df_R3_R5_full
    years_columns = ["% 2019","% 2019_full","% 2018","% 2018_full","% 2017","% 2017_f
    years_dfs = df_R3_R5_2019,df_R3_R5_2019_full,df_R3_R5_2018,df_R3_R5_2018_full,df_
    year_2019_dfs = df_R3_R5_2019,df_R3_R5_2019_full
    year_2019_columns = "2019","2019_full"
    sum_column = 'SUM (not Nan)'

    # Insert Data
    new_df_name = pd.DataFrame(columns=columns_names)
    new_df_2019 = pd.DataFrame(columns=year_2019_columns)
    for column,dataframe in zip(pooled_columns,pooled_dfs):
        new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)
    for column,dataframe,pooled_columns in zip(years_columns,years_dfs,pooled_columns
        if feature_name in dataframe.columns:
            new_df_name[column] = dataframe[feature_name].value_counts(dropna=False)/
    new_df_name=new_df_name.append(pd.Series(name=sum_column))
    new_df_name.loc[sum_column,pooled_column] = pooled_df[feature_name].value_counts(
    new_df_name.loc[sum_column,pooled_column_full] = pooled_df_full[feature_name].val
    count=0
    for column,dataframe in zip(years_columns,years_dfs):
        if feature_name in dataframe.columns:
            if count%2==0:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
            elif count%2==1:
                new_df_name.loc[sum_column,column] = dataframe[feature_name].value_co
        count+=1
    new_df_name = new_df_name.fillna(0)
    new_df_name = new_df_name.astype('int64', copy=False)
    for column in years_columns:
        new_df_name[column] = new_df_name[column].map(str) + "%"

    # 2019
    for column,dataframe in zip(year_2019_columns,year_2019_dfs):
        new_df_2019[column] = dataframe[feature_name].value_counts(dropna=True)
    new_df_2019.index=reverse_index(new_df_2019)
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(19,6),rot=0,title='20
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='1 
    #plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='3+
    plot_2019=new_df_2019.plot(kind='bar',fontsize=15,figsize=(16,6),rot=0,title='3+5
    plot_2019.axhline(15,color='black',linestyle='dashed')
    plot_2019.text(-0.64, 15,'15',fontsize=13)
    new_df_2019.index=reverse_index(new_df_2019)

    return new_df_name,new_df_2019,plot_2019
```

Sex

In [ ]:
```
feature_name=" סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד"
sex_R1,sex_R1_2019,sex_R1_2019_pooled=create_R1_specific_df(feature_name,sex)
sex_R1
```

In [ ]:
```
feature_name=" סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד"
sex_R2,sex_R2_2019,sex_R2_2019_pooled=create_R2_specific_df(feature_name,sex)
sex_R2
```

In [ ]:
```
feature_name=" סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד"
sex_R3,sex_R3_2019,sex_R3_2019_pooled=create_R3_specific_df(feature_name,sex)
sex_R3
```

In [ ]:
```
feature_name=" סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד"
sex_R5,sex_R5_2019,sex_R5_2019_pooled=create_R5_specific_df(feature_name,sex)
sex_R5
```

In [ ]:
```
feature_name=" סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד"
sex_R1_R2,sex_R1_R2_2019,sex_R1_R2_2019_pooled=create_R1_R2_specific_df(feature_name,s
sex_R1_R2
```

In [ ]:
```
feature_name=" סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד"
sex_R3_R5,sex_R3_R5_2019,sex_R3_R5_2019_pooled=create_R3_R5_specific_df(feature_name,s
sex_R3_R5
```

population groups

In [ ]:
```
feature_name="אוכלוסיה"
groups_R1,groups_R1_2019,groups_R1_2019_pooled=create_R1_specific_df(feature_name,grou
groups_R1
```

In [ ]:
```
feature_name="אוכלוסיה"
groups_R2,groups_R2_2019,groups_R2_2019_pooled=create_R2_specific_df(feature_name,grou
groups_R2
```

In [ ]:
```
feature_name="אוכלוסיה"
groups_R3,groups_R3_2019,groups_R3_2019_pooled=create_R3_specific_df(feature_name,grou
groups_R3
```

In [ ]:
```
feature_name="אוכלוסיה"
groups_R5,groups_R5_2019,groups_R5_2019_pooled=create_R5_specific_df(feature_name,grou
groups_R5
```

In [ ]:
```
feature_name="אוכלוסיה"
is_jewish_R1_R2,is_jewish_R1_R2_2019,is_jewish_R1_R2_2019_pooled=create_R1_R2_specific
is_jewish_R1_R2
```

In [ ]:
```
feature_name="אוכלוסיה"
is_jewish_R3_R5,is_jewish_R3_R5_2019,is_jewish_R3_R5_2019_pooled=create_R3_R5_specific
is_jewish_R3_R5
```

education

In [ ]:
```
feature_name="is_academic_education"
is_academic_R1_R2,is_academic_R1_R2_2019,is_academic_R1_R2_2019_pooled=create_R1_R2_sp
is_academic_R1_R2
```

In [ ]:
```
feature_name="is_academic_education"
is_academic_R3_R5,is_academic_R3_R5_2019,is_academic_R3_R5_2019_pooled=create_R3_R5_sp
is_academic_R3_R5
```

Jewish or not

In [ ]:
```
feature_name="is_jewish"
are_jewish_R1,are_jewish_R1_2019,are_jewish_R1_2019_plot=create_R1_specific_df("is_jew
are_jewish_R1
```

In [ ]:
```
feature_name="is_jewish"
are_jewish_R2,are_jewish_R2_2019,are_jewish_R2_2019_plot=create_R2_specific_df("is_jew
are_jewish_R2
```

In [ ]:
```
feature_name="is_jewish"
are_jewish_R3,are_jewish_R3_2019,are_jewish_R3_2019_plot=create_R3_specific_df("is_jew
are_jewish_R3
```

In [ ]:
```
feature_name="is_jewish"
are_jewish_R5,are_jewish_R5_2019,are_jewish_R5_2019_plot=create_R5_specific_df("is_jew
are_jewish_R5
```

In [ ]:
```
feature_name="is_jewish"
are_jewish_R1_R2,are_jewish_R1_R2_2019,are_jewish_R1_R2_2019_plot=create_R1_R2_specifi
are_jewish_R1_R2
```

In [ ]:
```
feature_name="is_jewish"
are_jewish_R3_R5,are_jewish_R3_R5_2019,are_jewish_R3_R5_2019_plot=create_R3_R5_specifi
are_jewish_R3_R5
```

business sector

In [ ]:
```
feature_name="business_sector"
business_sector_R1_R2,business_sector_R1_R2_2019,business_sector_R1_R2_2019_plot=creat
business_sector_R1_R2
```

```
In [ ]: feature_name="business_sector"
        business_sector_R3_R5,business_sector_R3_R5_2019,business_sector_R3_R5_2019_plot=creat
        business_sector_R3_R5
```

business size

```
In [ ]: feature_name="business_size"
        business_size_R1_R2,business_size_R1_R2_2019,business_size_R1_R2_2019_plot=create_R1_R
        business_size_R1_R2=business_size_R1_R2.drop(columns=['% 2018','% 2017','% 2018_full',
        business_size_R1_R2
        business_size_R1_R2
```

```
In [ ]: feature_name="business_size"
        business_size_R3_R5,business_size_R3_R5_2019,business_size_R3_R5_2019_plot=create_R3_R
        business_size_R3_R5=business_size_R3_R5.drop(columns=['% 2018','% 2017','% 2018_full',
        business_size_R3_R5
```

Descriptive Statistics - Questions responsivness

```
In [ ]: df_2019.dtypes.value_counts()
```

```
In [ ]: df_2019.dtypes[(df_2019.dtypes!="int64")&(df_2019.dtypes!="int32")].value_counts()
```

```
In [ ]: df_2019.dtypes[(df_2019.dtypes=="int64")]
```

```
In [ ]: questions_responsiveness_2019
```

```
In [ ]: questions_responsiveness_2018
```

```
In [ ]: questions_responsiveness_2017
```

```
In [ ]: df_2019[df_2019.columns.difference(non_informative_features)].describe().T.round(1)
        df_2019[df_2019.columns.difference(non_informative_features)].count
```

2018 Survey:

```
In [ ]: questions_responsiveness_2018 = pd.DataFrame(df_2018[df_2018.columns.difference(non_ir
        df_2018=df_2018[questions_responsiveness_2018[questions_responsiveness_2018.iloc[:,0]>
```

```
In [ ]: questions_responsiveness_2018.index
```

```
In [ ]: df_2018["סוקר נא לא להקריא אלא למלא לבד) : מין המרואיין) Q26 - ש' 2019 בלבד "].value_counts(dro
```

In [ ]:
```
df_2018[df_2018.columns.difference(non_informative_features)].describe().T.round(1)
```

2017 Survey:

In [ ]:
```
df_2017[df_2017.columns.difference(non_informative_features)].describe().T.round(1)
```

In [ ]:
```
questions_responsiveness_2017 = pd.DataFrame(df_2017[df_2017.columns.difference(non_ir
questions_responsiveness_2017
```

Convert qualitative features into dummies

In [ ]:
```
#df.select_dtypes(include=["bool_","object_"])
df=pd.concat([df,pd.get_dummies(df[cols[11]])],axis=1)
df
```

Random Effect Model explaining number of employees

In [ ]:
```
# Need to create the independent var!
exog_vars = ['change_in_number_of_employees', ‘employ’]
exog = sm.add_constant(data[exog_vars])
mod = RandomEffects(data.clscrap, exog)
re_res = mod.fit()
print(re_res)
```

Random Effect Model explaining overall satisfaction

In [ ]:
```
exog_vars = ['general_satisfaction_center_contribution_to_business_development', ‘empl
exog = sm.add_constant(data[exog_vars])
mod = RandomEffects(data.clscrap, exog)
re_res = mod.fit()
print(re_res)
```