מבוא לניתוח נתונים בפייתון - 094202

אביב תשפ"ג - תרגיל בית 3

חלק א' - תיאור קובץ הנתונים

קובץ הנתונים מכיל מדגם של משכורות של עובדים במקצועות הקשורים בעיבוד נתונים (לצורך קיצור נקרא להם מעתה עובדי "מקצועות הנתונים") שנאסף בתחילת שנת 2023 בארצות הברית.

feature	Description
experience_level	The experience level in the job during the year. (SE : Senior, EN : Entry level, EX : Executive level, MI : Mid/Intermediate level)
employment_type	The type of employment for the role. (CT: Contract worker, FL: Freelancer, FT: Full Time, PT: Part Time)
job_title	The role worked in during the year.
Salary_in_usd	The salary in USD ($)
remote_ratio	The overall amount of work done remotely. (Fully Remote: 100% of work is remote, Hybrid: less than 100% remote work)
company_size	The median number of people that worked for the company during the year

שאלות

*הערה - כאשר נבקש להסביר את קטע הקוד שכתבתם נצפה לתיעוד הקוד במידה מספקת.

פונקציות: הסבירו מי הם המשתנים ומה סוגם, מה הפונקציה עושה ומהי התוצאה המתקבלת (משמעותה וסוג המשתנה).

מקטעי קוד/תאים - הסבר מילולי קצר (משפט/שני משפטים).

In [1]:
```
#Import libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.style as style
style.use('tableau-colorblind10')
import seaborn as sns
sns.set_palette("viridis")
from tqdm import tqdm
```

In [2]:
```
#Reading the 'ds_salaries.csv' file:
df_salaries = pd.read_csv('ds_salaries.csv')
df_salaries
```

Out[2]:

	experience_level	employment_type	job_title	salary_in_usd	remote_ratio	company_size
0	MI	CT	ML Engineer	30000	fully remote	S
1	MI	CT	ML Engineer	25500	fully remote	S
2	SE	FT	Applied Scientist	222200	hybrid	L
3	SE	FT	Applied Scientist	136000	hybrid	L
4	SE	FT	Data Scientist	147100	hybrid	M
...	...	...	...	...	...	...
1560	SE	FT	Machine Learning Engineer	134500	hybrid	L
1561	MI	FT	Data Scientist	130000	hybrid	M
1562	MI	FT	Data Scientist	90000	hybrid	M
1563	EN	FT	Data Engineer	160000	hybrid	M
1564	EN	FT	Data Engineer	135000	hybrid	M

1565 rows × 6 columns

שאלה מס' 1:

1. סקר שנערך בחודש פברואר 2023 קבע כי אחוז האמריקאים שעובדים מהבית באופן מוחלט הוא 35%:

https://www.pewresearch.org/short-reads/2023/03/30/about-a-third-of-us-workers-who-can-work-from-home-do-so-all-the-time/

נניח שנתון זה נמדד גם בקרב עובדי "מקצועות הנתונים" בארה"ב ונמצא דומה. האם קובץ הנתונים מייצג היטב את האוכלוסייה של עובדי "מקצועות הנתונים" בארה"ב או שהוא כולל תת ייצוג לעובדים שעובדים מהבית ב-100% מהזמן? בדקו באמצעות המשתנה *remote_ratio* (רמת מובהקות נדרשת: 0.05)

סעיף א'

א. ציינו באופן ברור את השערת האפס וההשערה האלטרנטיבית

Solution

We were told that it was measured among the workers of the "data professions" in the United States that the percentage of workers from home absolutely is 35%.
Therefore, in order to test whether our dataset is a good representation of the population of "data professions" workers in the United States, we will test the following hypothesis:

H_{0}: The percentage of workers from home among "data professions" workers in the United States in the data file is equal to 35%

H_{1}: The percentage of workers from home among "data professions" workers in the United States in the data file is less then 35%

סעיף ב'

ב. מהו סטטיסטי המבחן

Solution

In [3]:
```
#Find the number of employees according to their work location
value_counts = df_salaries[['remote_ratio']].value_counts()

#Find the percentage of employees that work from home
test_statistic = (value_counts['fully remote'] / len(df_salaries[['remote_ratio']])) * 100

#Leave 2 digits after the decimal point
formatted_number = "{:.2f}".format(test_statistic)

print('The test statistic is the percentage of workers from home full time among' +
      '"data professions" workers')
print(f'in the United States in the data file, that is equal to {formatted_number}%')
```

```
The test statistic is the percentage of workers from home full time among"data professions" workers
in the United States in the data file, that is equal to 32.46%
```

סעיף ג'

ג. כתבו קוד לבחינת ההשערה באמצעות סימולציות. הסבירו את הקוד שכתבתם.

Solution

In [4]:
```
work_location = ['fully remote', 'hybrid'] # possible work location
prob_for_item = [35/100, 65/100] # probabilities GIVEN THE MODEL IS TRUE
sample_size = df_salaries.shape[0] # the number of employees sampled

# simulate one value
def prob_fully_remote():
    '''This function returns a number that represents the percentage of "data professions"
    that work from home full time in each sample'''

    #Sample according to the model
    sample_work_location = np.random.choice(work_location,
                                            p=prob_for_item, size=sample_size)

    #Find the number of employees in the sample that work from home
    num_fully_remote = np.count_nonzero(sample_work_location == 'fully remote')

    #returns the percentage of employees that work from home from in the sample
    return (num_fully_remote/sample_size) * 100
```

סעיף ד'

ד. מהי המסקנה שלכם? הציגו תוצאה מספרית וכן גרף המדגים את תוצאת המבחן

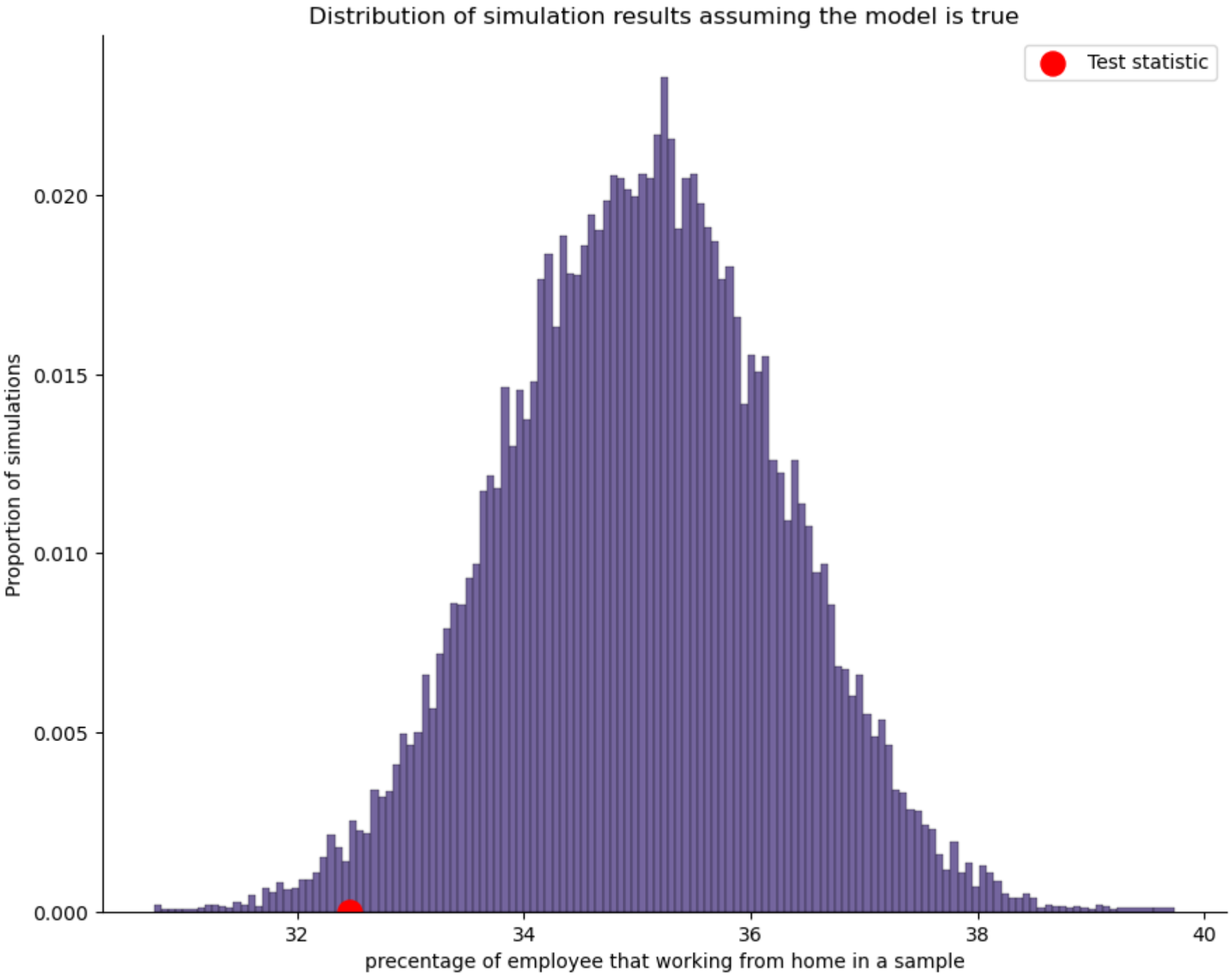
Solution

In [5]:
```
# run multiple simulations
num_repetitions = 20000
many_prob_fully_remote = np.array([prob_fully_remote() for i in range(num_repetitions)])

# plot the empirical distribution of the statistic
```

```
facetgrid_obj = sns.displot(many_prob_fully_remote, bins=np.unique(many_prob_fully_remote),
                            stat='probability') # Defining the bins

facetgrid_obj.fig.set_size_inches(10, 7)
facetgrid_obj.set(title='Distribution of simulation results assuming the model is true',
                  xlabel='precentage of employee that working from home in a sample',
                  ylabel='Proportion of simulations')

# Add a red point on the plot marking our data
facetgrid_obj.axes[0, 0].scatter(test_statistic, 0, s=150, color='red') # draw observed value
facetgrid_obj.axes[0, 0].legend(['Test statistic'])
```

Out[5]:

```
<matplotlib.legend.Legend at 0x11fdc18a3a0>
```


Distribution of simulation results assuming the model is true
Test statistic
Proportion of simulations
0.020
0.015
0.010
0.005
0.000
32
34
36
38
40
precentage of employee that working from home in a sample

In [6]:
```
# Count the number of simulations less than or equal to the observed test statistic
num_simulations = np.count_nonzero(many_prob_fully_remote <= test_statistic)

# Calculate the p-value by dividing the count by the total number of simulations
p_value = num_simulations / num_repetitions

# Print the calculated p-value
print(f'The p-value is {p_value}')
```

```
The p-value is 0.01775
```

According to the P-value we received, it can be seen that at a significance level of 5% the null hypothesis can definitely be rejected, and this is because the P-value we received is significantly smaller than 5%.
Therefore, our dataset is **not** a good representation of the population of "data professions" workers in the United State.

שאלה מס' 2:

2. קיימת טענה בתעשייה לפיה Data Scientists מקבלים בממוצע משכורת שווה למשכורת שמקבלים בממוצע Data Engineers. בדקו את הטענה בהסתמך על הדאטה הנתון, בהנחה שהוא מייצג את אוכלוסיית עובדי "מקצועות הנתונים" בארה"ב.

סעיף א'

א. ציינו באופן ברור את השערת האפס וההשערה האלטרנטיבית.

Solution

H_{0}: The mean salery of Data Scientists is equal to the mean salery of Data Engineers
Therfore, the mean salery of Data Scientists less the mean salery of Data Engineers is equal to 0.

H_{1}: The mean salery of Data Scientists is **not** equal to the mean salery of Data Engineerss
Therfore, the mean salery of Data Scientists less the mean salery of Data Engineers is **not** equal to 0.

סעיף ב'

ב. מהו סטטיסטי המבחן?

Solution

In [7]:
```
# function that returns the difference in averages
def diff_of_avgs(df, column_name, grouping_var):
    '''This function returns the difference between the average salary of the Data Scientists,
        and the average salary of the Data Engineers'''

    #Creates GroupBy according to 'grouping_var'
    grpby_var = df.groupby(grouping_var)

    #Finds the average value of each category in 'column_name'
    avgs = grpby_var[column_name].mean()

    #Returns the difference between thethe average salary of the Data Scientists
    #and the average salary of the Data Engineers
    return avgs.loc['Data Scientist'] - avgs.loc['Data Engineer']

#Gets the test statistic and Leave 2 digits after the decimal point
formatted_number = "{:.2f}".format(diff_of_avgs(df_salaries, 'salary_in_usd', 'job_title'))

print('Under the assumption that the given data represents the population\n' +
      'of "data professions" workers in the United States,\n' +
      'Our test statistic is the difference between the average salary of the Data Scientists\n' +
      f'and the average salary of the Data Engineers in the given data: {formatted_number}')
```

```
Under the assumption that the given data represents the population
of "data professions" workers in the United States,
Our test statistic is the difference between the average salary of the Data Scientists
and the average salary of the Data Engineers in the given data: 12686.68
```

סעיף ג'

ג. כתבו קוד לבחינת ההשערה באמצעות רווח סמך (עם לפחות 5000 רפליקציות). הסבירו את הקוד שכתבתם.

Solution

In [8]:
```
def bootstrap_mean_difference(original_sample, column_name, grouping_var, num_replications):
    '''This function returns an array of bootstrapped differences between two sample averages:
      original_sample: df containing the original sample
      column_name: name of column containing the variable to average
      grouping_var: name of variable according to which to group
      num_replications: number of bootstrap samples'''

    # we need to replicate with the same sample size
    original_sample_size = original_sample.shape[0]

    # Filters the 'original_sample' according to the 'column_name' and 'grouping_var'
    original_sample_cols_of_interest = original_sample[[column_name, grouping_var]]

    # collection array for our estimates
    bstrap_mean_diffs = np.empty(num_replications)

    # iterations
    for i in tqdm(range(num_replications)):

        # Gets a random sample at the size of 'original_sample'
        # note WITH REPLACEMENT!
        bootstrap_sample = original_sample_cols_of_interest.sample(original_sample_size,
                                                                   replace=True)

        # Gets the difference in averages
        resampled_mean_diff = diff_of_avgs(bootstrap_sample,
                                           column_name, grouping_var)

        # Insert to the array
        bstrap_mean_diffs[i] = resampled_mean_diff

    # Returns an array of bootstrapped average differences
    return bstrap_mean_diffs
```

```
# run the bootstrap procedure
bstrap_diffs = bootstrap_mean_difference(df_salaries, 'salary_in_usd',
                                         'job_title',  5000)
```

```
100%|██████████████████████████████████████████████████████████| 5000/5000 [00:07<00:00, 702.44it/s]
```

סעיף ד'

ד. מצאו רווחי סמך לערכי הפרמטר שחיפשתם ברמות ביטחון של 0.99 ו-0.95.

Solution

In [9]:
```
# Get the endpoints of the 99% confidence interval
left_end_99 = np.percentile(bstrap_diffs, 0.5, method='higher')
right_end_99 = np.percentile(bstrap_diffs, 99.5, method='higher')
formatted_number_left = "{:.2f}".format(left_end_99)
formatted_number_right = "{:.2f}".format(right_end_99)
print('The 99% boostsrap confidence interval for difference' +
      'between the average salary of the Data Scientists,')
print('and the average salary of the Data Engineers',
      [formatted_number_left, formatted_number_right])
```

```
The 99% boostsrap confidence interval for differencebetween the average salary of the Data Scientists,
and the average salary of the Data Engineers ['3120.41', '22444.70']
```

In [10]:
```
# Get the endpoints of the 95% confidence interval
left_end_95 = np.percentile(bstrap_diffs, 2.5, method='higher')
right_end_95 = np.percentile(bstrap_diffs, 97.5, method='higher')
formatted_number_left = "{:.2f}".format(left_end_95)
formatted_number_right = "{:.2f}".format(right_end_95)
print('The 95% boostsrap confidence interval for difference' +
      ' between the average salary of the Data Scientists,')
print('and the average salary of the Data Engineers',
      [formatted_number_left, formatted_number_right])
```

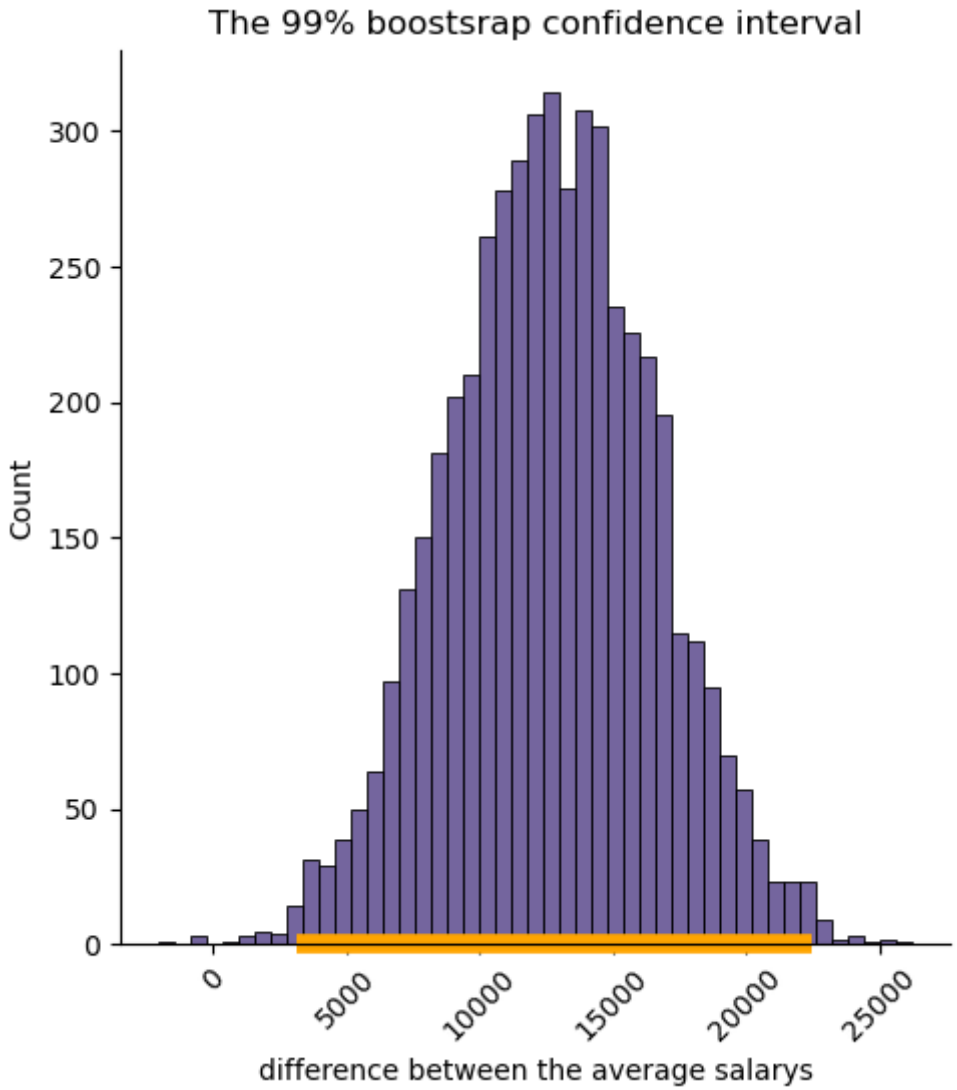
```
The 95% boostsrap confidence interval for difference between the average salary of the Data Scientists,
and the average salary of the Data Engineers ['5098.07', '20228.74']
```

סעיף ה'

ה. מהי המסקנה שלכם? הציגו תוצאות מספרית וכן גרפים המדגימים את תוצאת המבחן לשתי רמות הביטחון המבוקשות.

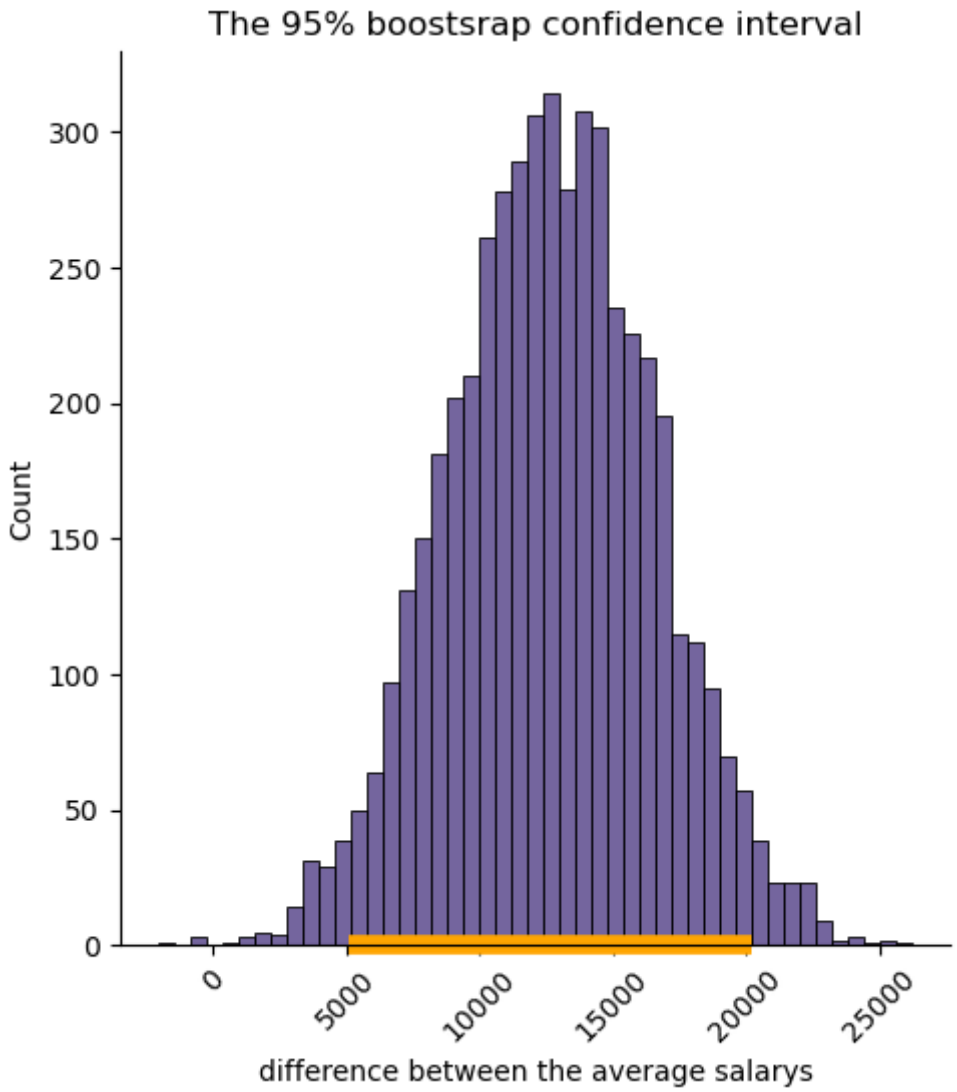
Solution

In [11]:
```
# visualize results
ax = sns.displot(bstrap_diffs)
plt.xticks(rotation=45)
plt.title('The 99% boostsrap confidence interval')
plt.xlabel('difference between the average salarys')

# draw observed value
facetgrid_obj.axes[0, 0].scatter(test_statistic, 0, s=150, color='red')

# show line of values between 0.5 and 99.5 percentiles
plt.hlines(y=0, xmin=left_end_99, xmax=right_end_99, colors='orange',
           linestyles='solid', lw=7, clip_on=False);
```



In [12]:
```
# visualize results
ax = sns.displot(bstrap_diffs)
plt.xticks(rotation=45)
plt.title('The 95% boostsrap confidence interval')
plt.xlabel('difference between the average salarys')

# draw observed value
facetgrid_obj.axes[0, 0].scatter(test_statistic, 0, s=150, color='red')
```

```
# show line of values between 2.5 and 97.5 percentiles
plt.hlines(y=0, xmin=left_end_95, xmax=right_end_95, colors='orange',
           linestyles='solid', lw=7, clip_on=False);
```



For each of the confidence intervals(for a safety level of 95% and 99%) we found that the value 0 is not in the interval, and since each value in the distribution represents the difference between the average salaries for each sample, then the difference that is equal to 0 is not in the interval,

and therefore we can reject the null hypothesis according to which the average salaries are equal (the difference between the average salaries is equal to 0).

שאלה מס' 3:

3. חשבו רווח סמך של 95% עבור ממוצע המשכורות של עובדים בחברות בינוניות (M). הסבירו את הקוד שכתבתם וציינו באופן מפורש את רווח הסמך (ערך עליון ותחתון).

Solution

In [13]:
```
def avg_of_salary_of_M_workers(df, column_name, grouping_var):
    '''This function returns the average salary of "data professions" workers in M size companies'''

    #Creates GroupBy according to 'grouping_var'
    grpby_var = df.groupby(grouping_var)

    #Finds the average value of each category in 'column_name'
    avgs = grpby_var[column_name].mean()

    # Returns the the average salary of "data professions" workers in M size companies
    return avgs.loc['M']

def bootstrap_mean_M_workers(original_sample, column_name, grouping_var, num_replications):
    '''This function returns an array of bootstrapped average
    of salary of "data professions" workers in M size companies:
    original_sample: df containing the original sample
    column_name: name of column containing the variable to average
    grouping_var: name of variable according to which to group
    num_replications: number of bootstrap samples'''

    # we need to replicate with the same sample size
    original_sample_size = original_sample.shape[0]

    # Filters the 'original_sample' according to the 'column_name' and 'grouping_var'
    original_sample_cols_of_interest = original_sample[[column_name, grouping_var]]

    # collection array for our estimates
```

```
    bstrap_mean_M_workers = np.empty(num_replications)

    # iterations
    for i in tqdm(range(num_replications)):

        # Gets a random sample at the size of 'original_sample'
        # note WITH REPLACEMENT!
        bootstrap_sample = original_sample_cols_of_interest.sample(original_sample_size,
                                                                   replace=True)

        # Gets the average of employees of M sized companies
        resampled_mean_M_workers = avg_of_salary_of_M_workers(bootstrap_sample,
                                                              column_name, grouping_var)

        # Insert to the array
        bstrap_mean_M_workers[i] = resampled_mean_M_workers

    # Returns an array of bootstrapped average
    return bstrap_mean_M_workers

# run the bootstrap procedure
bstrap_mean_M_workers = bootstrap_mean_M_workers(df_salaries, 'salary_in_usd',
                                                 'company_size',  5000)
```

```
100%|██████████████████████████████████████████████████████████| 5000/5000 [00:05<00:00, 896.15it/s]
```

In [14]:

```
# Get the endpoints of the 95% confidence interval
left_end_95 = np.percentile(bstrap_mean_M_workers, 2.5, method='higher')
right_end_95 = np.percentile(bstrap_mean_M_workers, 97.5, method='higher')
formatted_number_left = "{:.2f}".format(left_end_95)
formatted_number_right = "{:.2f}".format(right_end_95)
print('The 95% boostsrap confidence interval for the average salary'
      + f'of "data professions" workers\nin M size companies',
      [formatted_number_left, formatted_number_right])
```

```
The 95% boostsrap confidence interval for the average salaryof "data professions" workers
in M size companies ['153715.75', '159453.21']
```

שאלה מס' 4:

4. חברת ניתוח נתונים כלכליים עולמית רוצה לדעת מה המשכורת החציונית שמשולמת לעובדים בתחום הדאטה. מנתח נתונים חדש ומפוזר בחברה מחק בטעות את קובץ הנתונים. כדי למצוא את הערך החציוני, הוא החליט לערוך מדגם ולחשב את החציון באמצעות בוטסטראפ. הוא מצא קובץ עם נתונים על עובדים בתחום הדאטה באחד המיילים שקיבל, מבלי שידע שהאנשים המופיעים בקובץ הם רק אנשים שאינם עובדים מהבית לחלוטין (hybrid). העובד בחר באקראי 150 אנשים מתוך קובץ העובדים ושאל אותם מהי המשכורת הנוכחית שלהם. מכיוון שהבטיח פרס כספי למשיבים, כל האנשים שקיבלו את המייל ענו על השאלה. הניחו שהנתונים עליהם דיווחו המשיבים הם נתוני אמת התואמים את קובץ הנתונים המקורי.

סעיף א'

א. כדי לבדוק מה הסיכוי שהעובד הצליח ליצור מתוך המדגם (של העובדים ההיברידיים) רווח סמך של 95% המכיל את החציון האמיתי של משכורות כל העובדים, הריצו 100 סימולציות הדוגמות כל פעם 150 אנשים מתוך אוכלוסיית העובדים ההיברידיים ומחשבות על כל אחד מהמדגמים רווחי סמך של 95% עבור הערך החציוני של המשכורת שלהם. בכמה מתוך 100 רווחי הסמך שקיבלתם מוכל החציון האמיתי של עובדים בתחום הדאטה כפי שניתן לחשבו מקובץ הנתונים המקורי המלא?

Solution

In [15]:
```
def bootstrap_median(original_sample, column_name, num_replications):
    '''This function returns an array of bootstrapped sample medians:
    original_sample: df containing the original sample
    column_name: name of column containing the variable of interest
    num_replications: number of bootstrap samples to draw '''

    # we need to replicate with the same sample size
    original_sample_size = original_sample.shape[0]
```

```
    # Filters the 'original_sample' according to the 'column_name'
    original_sample_var_of_interest = original_sample[[column_name]]

    # collection array for our estimates
    bstrap_medians = np.empty(num_replications)

    # iterations
    for i in range(num_replications):

        # Gets a random sample at the size of 'original_sample'
        # note WITH REPLACEMENT!
        bootstrap_sample = original_sample_var_of_interest.sample(n=original_sample_size,
                                                                  replace=True)

        # Gets the median
        resampled_median = bootstrap_sample.quantile(0.5, interpolation='higher')

        # Insert to the array
        bstrap_medians[i] = resampled_median

    # Returns an array of bootstrapped medians
    return bstrap_medians
```

In [16]:

```
# BIG simulation, may take a few minutes
filt = (df_salaries['remote_ratio'] == 'hybrid') # Creates a filter for 'hybrid'
df_hybrid = df_salaries[filt] # Uses the filter on the data
# Gets the mediam from the data
data_median = df_hybrid[['salary_in_usd']].quantile(0.5, interpolation='higher')[0]

left_ends = []
right_ends = []
median_in_interval = []

# iterations
for i in tqdm(range(100)):

    # sampling from the population
    median_hybrid_sample = df_hybrid.sample(150, replace=False)

    # then using the bootstrap method to get confidence interval
    medians = bootstrap_median(median_hybrid_sample, 'salary_in_usd', 5000)
    left = np.percentile(medians, 2.5, method='higher')
```

```
    left_ends.append(left)
    right = np.percentile(medians, 97.5, method='higher')
    right_ends.append(right)
    median_in_interval.append((data_median < right) and (data_median > left))

print('in', np.count_nonzero(median_in_interval),
      'of 100 runs of the bootstrap process,the data salary median was inside the interval')
```

```
100%|██████████████████████████████████████████████████████████| 100/100 [05:21<00:00,  3.22s/it]
in 95 of 100 runs of the bootstrap process,the data salary median was inside the interval
```

סעיף ב'

ב. מנתח הנתונים השתמש במדגם שלא נלקח באקראי מהאוכלוסייה אותה הוא בוחן (אלא מאוכלוסייה אחרת). האם, למרות זאת, מתקיימת בפועל ההבטחה התיאורטית של שיטת הבוטסטראפ ליצירת רווח סמך? מדוע ההבטחה מתקיימת או לא מתקיימת? הסבירו את תשובתכם. בסעיף זה אין צורך בכתיבת קוד.

Solution

From a theoretical point of view,
the promise of the bootstrap method to create a confidence interval is not fulfilled
since the population from which the data analyzer samples in advance was not taken at random
but rather a specific group (hybrid employees) was selected
and therefore it is not representative of the general population which is all employees.

סעיף ג'

ג. איך היו משתנות תשובותיכם לסעיפים א' ו-ב' אם חברת ניתוח הנתונים הייתה מעוניינת לדעת מהי המשכורת הרבעונית של עובדים בתחום (כלומר במקום לחשב חציון, מנתח הנתונים היה מחשב את הרבעון הראשון של המשכורות). הסבירו.

רמז: ציירו את התפלגויות המשכורות או סיכום שלהן בנפרד עבור שתי קבוצות העובדים.

Solution

In [19]:
```
def bootstrap_quantile(original_sample, column_name, num_replications):
    '''This function returns an array of bootstrapped sample quantiles:
    original_sample: df containing the original sample
    column_name: name of column containing the variable of interest
    num_replications: number of bootstrap samples to draw '''
    original_sample_size = original_sample.shape[0] # we need to replicate with the same sample size
    original_sample_var_of_interest = original_sample[[column_name]] # the use of [[]] will return a df rather than series
    bstrap_quantiles = np.empty(num_replications) # collection array for our estimates
    for i in range(num_replications):
        bootstrap_sample = original_sample_var_of_interest.sample(n=original_sample_size, replace=True) # note WITH REPLACEMENT!
        resampled_quantile = bootstrap_sample.quantile(0.25, interpolation='higher')
        bstrap_quantiles[i] = resampled_quantile

    return bstrap_quantiles
```

In [20]:
```
# BIG simulation, may take a few minutes
data_quantile = df_hybrid[['salary_in_usd']].quantile(0.25, interpolation='higher')[0] # Gets the Q1 from the data

left_ends = []
right_ends = []
quantile_in_interval = []

# iterations
for i in tqdm(range(100)):

    # sampling from the population
    quantile_hybrid_sample = df_hybrid.sample(150, replace=False)
```

```
    # then using the bootstrap method to get confidence interval
    quantiles = bootstrap_quantile(quantile_hybrid_sample, 'salary_in_usd', 5000)
    left = np.percentile(quantiles, 2.5, method='higher')
    left_ends.append(left)
    right = np.percentile(quantiles, 97.5, method='higher')
    right_ends.append(right)
    quantile_in_interval.append((data_quantile < right) and (data_quantile > left))

print('in', np.count_nonzero(quantile_in_interval), 'of 100 runs of the bootstrap process, the data salary Q1 was inside the inte
```

```
100%|██████████████████████████████████████████████████████████| 100/100 [04:52<00:00,  2.93s/it]
in 85 of 100 runs of the bootstrap process, the data salary Q1 was inside the interval
```

For section A:
It can be seen that when we perform the bootstrap for the first quarter instead of the median, the true value is in smaller confidence intervals.

For section B:
In accordance with section b, even in this case, the population from which the data analyzer samples (hybrid employees)
is not taken at random from all employees, therefore the promise to create the bootstrap is not fulfilled either.

חלק ב'

שאלה מס' 1:

1. מדוע בעייתי להשתמש בבוטסטראפ כדי לחשב רווח סמך בהתבסס על מדגם מאוד קטן? הסבירו ב-2-4 משפטים.

Solution

It is problematic to use bootstrap to calculate a confidence interval based on a very small sample
because if the sample is too small then its empirical distribution does not reliably represent the probability distribution of the population.

שאלה מס' 2:

2. חוקרים בארה"ב בחנו את ההשפעה של גזע על ההחלטות של שופטים לדון למוות אדם שהורשע ברצח. הנתונים שלהם כללו 702 הרשעות ברצח במדינת לואיזיאנה. נמצא כי, בסך הכל, 16.4% מהמורשעים הלבנים נידונו למוות, לעומת 15.2% מהמורשעים השחורים. עם זאת, כאשר הם כללו באנליזה גם את גזע הקורבן, הם מצאו כי מורשעים שחורים נידונים למוות באחוזים גבוהים יותר מאשר מורשעים לבנים כאשר הקורבן לבן וגם כאשר הקורבן שחור. ראו פירוט בטבלה:

		נידון למוות?		
גזע הקורבן	**גזע הנאשם המורשע**	**כן**	**לא**	**אחוז נידונים למוות**
לבן	לבן	42	176	19.2
	שחור	29	84	25.7
שחור	לבן	1	43	2.3
	שחור	38	289	11.6
סך הכל	לבן	43	219	16.4
	שחור	67	373	15.2

סעיף א'

א. איזו תופעה (סטטיסטית) יכולה להסביר את הפער בתוצאות בין המצב בו מתייחסים למצב בו לא מתייחסים לגזע הקורבן? הסבירו ב-2-3 משפטים.

Solution

The statistical phenomenon that can explain the gap in the results in which the race of the victim is not considered is "Simpson's Paradox".
The explanation for this is that when the race of the victim is not taken into account,
then you get a fairly equal percentage of those sentenced to death between blacks and whites (and even higher than one percent).
However, when the race of the victim is considered, it can certainly be seen that the percentage of blacks sentenced to death is significantly higher than that of whites.
Therefore, we are not considering an important variable when studying the relationship.

סעיף ב'

ב. בהתבסס על הטבלה:

תת סעיף 1

1. איזו נטייה של השופטים יכולה לעזור להסביר את הפער בתוצאות? רמז: באיזה מצב יותר סביר שיינתן גזר דין מוות?

Solution

Based on the table, the tendency of the judges that can explain the gap in the results is to make the sentence worse when the victim is white, and this is in accordance with the fact that it can be seen in the table that the percentage of those sentenced to death when
the victim is white is significantly higher in the case where the victim is black, regardless of the race of the accused.

תת סעיף 2

2. איזו נטייה של המורשעים יכולה לעזור להסביר את הפער בתוצאות?
רמז: מה הקשר בין גזע הקורבן לבין גזע הנאשם?

Solution

Based on the table, the tendency of the convicts that can explain the gap in the results is to kill mainly people of the same race as the accused,
and according to this in the table you can see that mainly whites kill whites,
and according to what we saw in the previous section (the tendency to punish more severely accused who killed whites),
then more whites are found themselves condemned to death.