Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Case Study - Confidence Intervals (Nap no Nap)

Name: Chong Mun Chen

IC Number: 960327-07-5097

Date : 11/7/2023

Introduction : Solving a confidence interval exercise using a case study on toddlers that nap against toddlers that do not.

Conclusion : This exercise gave me better practice on calculating the confidence interval.

Case Study - Confidence Intervals (Nap no Nap)

In this assessment, you will look at data from a study on toddler sleep habits. The confidence intervals you create and the questions you answer in this Jupyter notebook will be used to answer questions in the cell below.

In [7]:
```
import numpy as np
import pandas as pd
from scipy.stats import t
pd.set_option('display.max_columns', 30) # set so can see all columns of the Da
```

Your goal is to analyse data which is the result of a study that examined differences in a number of sleep variables between napping and non-napping toddlers. Some of these sleep variables included: Bedtime (lights-off time in decimalized time), Night Sleep Onset Time (in decimalized time), Wake Time (sleep end time in decimalized time), Night Sleep Duration (interval between sleep onset and sleep end in minutes), and Total 24-Hour Sleep Duration (in minutes). Note: [Decimalized time (https://en.wikipedia.org/wiki/Decimal_time)](https://en.wikipedia.org/wiki/Decimal_time) is the representation of the time of day using units which are decimally related.

The 20 study participants were healthy, normally developing toddlers with no sleep or behavioral problems. These children were categorized as napping or non-napping based upon parental report of children's habitual sleep patterns. Researchers then verified napping status with data from actigraphy (a non-invasive method of monitoring human rest/activity cycles by wearing of a sensor on the wrist) and sleep diaries during the 5 days before the study assessments were made.

You are specifically interested in the results for the Bedtime, Night Sleep Duration, and Total 24-Hour Sleep Duration.

ref: Akacem LD, Simpkin CT, Carskadon MA, Wright KP Jr, Jenni OG, Achermann P, et al. (2015) The Timing of the Circadian Clock and Sleep Differ between Napping and Non-Napping Toddlers. PLoS ONE 10(4): e0125181. [https://doi.org/10.1371/journal.pone.0125181 (https://doi.org/10.1371/journal.pone.0125181)](https://doi.org/10.1371/journal.pone.0125181)

In [1]:
```
import pandas as pd
import numpy as np
```

In [8]:
```
# Import the data
df = pd.read_csv("../data_samples/nap_no_nap.csv")
```

In [9]:
```
# First, look at the DataFrame to get a sense of the data
df
```

Out[9]:

	id	sex	age (months)	dlmo time	days napped	napping	nap lights outl time	nap sleep onset	nap midsleep	nap sleep offset	nap wake time	na duratio
0	1	female	33.7	19.24	0	0	NaN	NaN	NaN	NaN	NaN	Na
1	2	female	31.5	18.27	0	0	NaN	NaN	NaN	NaN	NaN	Na
2	3	male	31.9	19.14	0	0	NaN	NaN	NaN	NaN	NaN	Na
3	4	female	31.6	19.69	0	0	NaN	NaN	NaN	NaN	NaN	Na
4	5	female	33.0	19.52	0	0	NaN	NaN	NaN	NaN	NaN	Na
5	6	female	36.2	18.22	4	1	14.00	14.22	15.00	15.78	16.28	93.7
6	7	male	36.3	19.28	1	1	14.75	15.03	15.92	16.80	16.08	106.0
7	8	male	30.0	21.06	5	1	13.09	13.43	14.44	15.46	15.82	121.6
8	9	male	33.2	19.38	2	1	14.41	14.42	15.71	17.01	16.60	155.5
9	10	female	37.1	19.93	3	1	13.12	13.42	14.31	15.19	15.30	106.6
10	11	male	32.9	18.79	4	1	13.99	14.03	14.85	15.68	16.10	98.7
11	12	female	35.0	19.65	5	1	13.18	13.45	14.33	15.21	15.35	105.8
12	13	male	35.1	19.83	3	1	13.94	14.48	15.26	16.03	15.78	93.3
13	14	female	35.6	19.88	4	1	12.68	13.08	13.92	14.76	15.00	100.7
14	15	female	36.6	19.94	4	1	12.71	12.88	13.80	14.72	14.88	110.7
15	16	male	36.5	20.25	3	1	13.74	14.68	15.66	16.64	16.45	117.3
16	17	female	33.7	20.33	5	1	13.15	13.87	14.49	15.11	15.40	74.2
17	18	male	36.4	20.16	5	1	12.47	12.56	13.30	14.05	14.25	89.8
18	19	female	33.6	19.68	3	1	14.71	14.85	15.46	16.07	16.20	73.0
19	20	male	33.8	20.51	3	1	12.68	13.54	14.30	15.07	15.23	91.6

Question: What variable is used in the column 'napping' to indicate a toddler takes a nap?

In [10]:
```
df[['napping']].describe()
```

Out[10]:

	napping
count	20.000000
mean	0.750000
std	0.444262
min	0.000000
25%	0.750000
50%	1.000000
75%	1.000000
max	1.000000

Question: What is the sample size n? What is the sample size for toddlers who nap, n_1 , and toddlers who don't nap, n_2 ?

In [11]:
```
n1 = len(df[df['napping'] == 1])
n2 = len(df[df['napping'] == 0])
print(n1, n2)
```

```
15 5
```

Average bedtime confidence interval for napping and non napping toddlers

Create two 95% confidence intervals for the average bedtime, one for toddler who nap and one for toddlers who don't.

Before any analysis, we will convert 'night bedtime' into decimalized time.

In [12]:
```
# Convert 'night bedtime' into decimalized time
df.loc[:,'night bedtime'] = np.floor(df['night bedtime'])*60 + np.round(df['ni
```

Now, isolate the column 'night bedtime' for those who nap into a new variable, and those who didn't nap into another new variable.

In [13]:
```
bedtime_nap = df[df['napping'] == 1]['night bedtime']
```

In [14]:
```
bedtime_nap
```

Out[14]:
```
5     1235.0
6     1260.0
7     1321.0
8     1224.0
9     1278.0
10    1185.0
11    1218.0
12    1222.0
13    1226.0
14    1228.0
15    1246.0
16    1243.0
17    1202.0
18    1190.0
19    1218.0
Name: night bedtime, dtype: float64
```

In [15]:
```
bedtime_no_nap = df[df['napping'] == 0]['night bedtime']
```

In [16]:
```
bedtime_no_nap
```

Out[16]:
```
0    1245.0
1    1163.0
2    1200.0
3    1186.0
4    1161.0
Name: night bedtime, dtype: float64
```

Now find the sample mean bedtime for nap and no_nap.

In [17]:
```
nap_mean_bedtime = bedtime_nap.mean()
nap_mean_bedtime
```

Out[17]:
```
1233.0666666666666
```

In [18]:
```
no_nap_mean_bedtime = bedtime_no_nap.mean()
no_nap_mean_bedtime
```

Out[18]:
```
1191.0
```

Constructing Confidence Intervals

Now that we have the population proportions of male and female smokers, we can begin to calculate confidence intervals. From lecture, we know that the equation is as follows:

$$Best\ Estimate \pm Margin\ of\ Error$$

Where the *Best Estimate* is the **observed population proportion or mean** from the sample and the *Margin of Error* is the **t-multiplier**.

The equation to create a 95% confidence interval can also be shown as:

$$Population\ Proportion\ or\ Mean\ \pm (t - multiplier * \ Standard\ Error)$$

The Standard Error is calculated differenly for population proportion and mean:

$$Standard\ Error\ for\ Population\ Proportion = \sqrt{\frac{Population\ Proportion * (1 - Population\ Proportion)}{Number\ Of\ Observations}}$$

$$Standard\ Error\ for\ Mean = \frac{Standard\ Deviation}{\sqrt{Number\ Of\ Observations}}$$

Now find the standard error for $\bar{X}_{nap}$ and $\bar{X}_{no\ nap}$.

In [25]:
```
import math

n1 = len(df[df['napping'] == 1])
se_nap = bedtime_nap.std() / math.sqrt(n1)
se_nap
```

Out[25]: 8.893800230479801

In [26]:
```
import math

n2 = len(df[df['napping'] == 0])
se_no_nap = bedtime_no_nap.std() / math.sqrt(n2)
se_no_nap
```

Out[26]: 15.339491516996253

Question: Given our sample sizes of n_1 and n_2 for napping and non napping toddlers respectively, how many degrees of freedom (df) are there for the associated t distributions?

In [27]:
```
dof_nap = n1 - 1
dof_no_nap = n2 - 1
print(dof_nap, dof_no_nap)
```

```
14 4
```

To build a 95% confidence interval, what is the value of t*? You can find this value using the percent point function:

```
from scipy.stats import t

t.ppf(probabiliy, df)
```

This will return the quantile value such that to the left of this value, the tail probabiliy is equal to the input probabiliy (for the specified degrees of freedom).

Example: to find the t^* for a 90% confidence interval, we want t^* such that 90% of the density of the t distribution lies between $-t^*$ and t^*. Or in other words if $X \sim t(df)$:

$P(-t^* < X < t^*) = .90$

Which, because the t distribution is symmetric, is equivalent to finding t^* such that:

$P(X < t^*) = .95$

So the t^* for a 90% confidence interval, and lets say df=10, will be:

```
t_star = t.ppf(.95, df=10)
```

Quesion: What is t^* for nap and no nap?

In [28]:
```
from scipy.stats import t
```

In [30]:
```
# Find the t_stars for the 95% confidence intervals
nap_t_star = t.ppf(0.975, dof_nap)
nap_t_star
```

Out[30]:
```
2.1447866879169273
```

In [31]:
```
no_nap_t_star = t.ppf(0.975, dof_no_nap)
no_nap_t_star
```

Out[31]:
```
2.7764451051977987
```

Now to create our confidence intervals. For the average bedtime for nap and no nap, find the upper and lower bounds for the respective confidence intervals.

Question: What are the 95% confidence intervals, rounded to the nearest ten, for the average bedtime (in decimalized time) for toddlers who nap and for toddlers who don't nap?

CI = $\bar{X} \pm t^* \cdot s.e.(\bar{X})$

In [33]:
```
ci_nap = nap_mean_bedtime + np.array([-1, +1]) * nap_t_star * se_nap
lower_nap, upper_nap = ci_nap
nap_ci_rounded = (round(lower_nap), round(upper_nap))
print(nap_ci_rounded)
tuple(ci_nap)
```

```
(1214, 1252)
```

Out[33]:
```
(1213.991362327341, 1252.1419710059922)
```

In [34]:
```
ci_no_nap = no_nap_mean_bedtime + np.array([-1, +1]) * no_nap_t_star * se_no_na
lower_no_nap, upper_no_nap = ci_no_nap
no_nap_ci_rounded = (round(lower_no_nap), round(upper_no_nap))
print(no_nap_ci_rounded)
tuple(ci_no_nap)
```

```
(1148, 1234)
```

Out[34]:
```
(1148.4107438614126, 1233.5892561385874)
```

Challenge problem 1: Write a function that inputs the column containing the data you want to build your confidence interval from and returns the confidence interval as a list or tuple (i.e. [upper, lowe] or (upper, lower)).

In [35]:
```
import numpy as np
from scipy.stats import t

def find_ci(data_column, confidence):
    sample_mean = np.mean(data_column)
    sample_std = np.std(data_column)

    n = len(data_column)
    t_value = t.ppf((1 + confidence) / 2, df = n - 1)
    margin_of_error = t_value * sample_std / np.sqrt(n)

    lower_bound = sample_mean - margin_of_error
    upper_bound = sample_mean + margin_of_error

    return (lower_bound, upper_bound)

# find_ci(df[df['napping'] == 1]['night bedtime'], 0.95)
```

Challenge problem 2: Create the intervals using the statsmodels function stats.weightstats.DescrStatsW:

```
stats.weightstats.DescrStatsW(data).tconfint_mean(alpha=.05)
```

In [36]:
```
from statsmodels.stats.weightstats import DescrStatsW

DescrStatsW(bedtime_nap).tconfint_mean(alpha=0.05)
```

Out[36]:
```
(1213.991362327341, 1252.1419710059922)
```

In [37]:
```
from statsmodels.stats.weightstats import DescrStatsW

DescrStatsW(bedtime_no_nap).tconfint_mean(alpha=0.05)
```

Out[37]:
```
(1148.4107438614126, 1233.5892561385874)
```