3.1 Challenge

Monday, August 15, 2022 10:08 PM

Myroslava Sánchez Andrade A01730712 | 15/08/2022

3 Hypothesis testing - comparing the mean of 2 groups

3.1 CHALLENGE - IS AMD MEAN RETURN HIGHER THAN INTEL MEAN RETURN?

Do a t-test to check whether the mean monthly cc return of AMD (AMD) is greater than the mean monthly return of Intel. Use data from Jan 2017 to date.

```
# Collecting the real data from currency of AMD and INTC
AMD = pdr.get_data_yahoo("AMD", start = "01/01/2017", end = "08/15/2022", interval = "m")
AMD = AMD["Adj Close"]
INTC = pdr.get_data_yahoo("INTC", start = "01/01/2017", end = "08/15/2022", interval = "m")
INTC = INTC["Adj Close"]
```

```
# Calculating the monthly cc returns
ccr_AMD = np.log(AMD) - np.log(AMD.shift(1))
ccr_AMD = ccr_AMD.dropna()
ccr_INTC = np.log(INTC) - np.log(INTC.shift(1))
ccr_INTC = ccr_INTC.dropna()
print("AMD CCr: ", ccr_AMD, " INTC CCr: ", ccr_INTC)
```

```
AMD CCr:  Date
2017-02-01    0.332469
2017-03-01    0.006205
2017-04-01   -0.089827
2017-05-01   -0.172744
2017-06-01    0.109107
                 ...
2022-04-01   -0.245712
2022-05-01    0.174849
2022-06-01   -0.286701
2022-07-01    0.211384
2022-08-01    0.015545
Name: Adj Close, Length: 67, dtype: float64  INTC CCr:  Date
2017-02-01   -0.016982
2017-03-01    0.003516
2017-04-01    0.002216
2017-05-01   -0.001107
2017-06-01   -0.060474
                 ...
```

```python
# Calculating the mean of the cc returns
AMD_mean = ccr_AMD.mean()
INTC_mean = ccr_INTC.mean()
print(f"AMD mean: {AMD_mean}, INTC mean: {INTC_mean}")
```

```
AMD mean: 0.03320761461854227, INTC mean: 0.0015393332495637387
```

```python
# Calculating the variance of the cc returns
N = ccr_AMD.count()
AMD_var = ccr_AMD.var()
INTC_var = ccr_INTC.var()
print(f"AMD var: {AMD_var}, INTC var: {INTC_var}")
```

```
AMD var: 0.026250250294288773, INTC var: 0.005644972280669387
```

```python
# H0: (mean(rAMD) - mean(rINTEL)) = 0
# Ha: (mean(rAMD) - mean(rINTEL)) <> 0

# Calculating the standard error of the difference of the means
t = (AMD_mean - INTC_mean - 0) / np.sqrt((1/N) * (AMD_var + INTC_var))
t
```

```
1.451439967108968
```

SINCE THE DIFFERENCE OF THE MEANS OF THE CC RETURNS IS ONLY 1.45 STANDARD DEVIATIONS, WE CANNOT ASSUME THAT NEITHER OF THE HYPOTHESIS ARE CORRECT. THE T-RESULT DEMONSTRATES THAT THE MEAN OF AMD IS JUST 1.45 STANDARD DEVIATIONS AWAY FROM THE INTEL'S, WHICH REPRESENTS LESS THAN THE 95% OF THE CASES.

```python
# Calculating the pvalue from the t-Statistic
from scipy import stats as st

# the degrees of freedom for 2-independent-means t-test is calculated with the following formula:
df = ( ((N-1) / N**2) * (AMD_var + INTC_var)**2  / ( (AMD_var/N)**2 + (INTC_var/N)**2  ) )
print(f"Degrees of freedom: {df}")
pvalue = 2 * st.t.sf(np.abs(t), df)
print(f"2-tailed pvalue: {pvalue}")
```

```
Degrees of freedom: 93.13121585600648
2-tailed pvalue: 0.15001768658500247
```

```
# Using the ttest function from stats to calculate the t value and the 2-tailed p-value
st.ttest_ind(ccr_AMD,ccr_INTC,equal_var=False)
```

```
Ttest_indResult(statistic=1.451439967108968, pvalue=0.15001768658500247)
```

```
import researchpy as rp
# Using the ttest function from researchpy:
rp.ttest(ccr_AMD,ccr_INTC, equal_variances = False)
# We got the same result as above!
# With this function we avoid calculating all steps of the hypothesis test!
```

```
C:\Users\myros\anaconda3\envs\Statistics\lib\site-packages\researchpy\ttest.py:18: FutureWarning: The serie
s.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instea
d.
  groups = group1.append(group2, ignore_index= True)
```

```
(    Variable      N      Mean        SD        SE  95% Conf.  Interval
 0  Adj Close   67.0  0.033208  0.162019  0.019794  -0.006312  0.072727
 1  Adj Close   67.0  0.001539  0.075133  0.009179  -0.016787  0.019866
 2   combined  134.0  0.017373  0.126808  0.010955  -0.004294  0.039041,
                        Welch's t-test  results
 0  Difference (Adj Close - Adj Close) =    0.0317
 1                  Degrees of freedom =   93.9534
 2                                   t =    1.4514
 3               Two side test p value =    0.1500
 4              Difference < 0 p value =    0.9250
 5              Difference > 0 p value =    0.0750
 6                           Cohen's d =    0.2508
 7                           Hedge's g =    0.2493
 8                       Glass's delta =    0.1955
 9                         Pearson's r =    0.1481)
```

3.1 CHALLENGE - IS AAPL MEAN RETURN HIGHER THAN MSFT MEAN RETURN?

Run a t-test to compare whether the average monthly return of AAPL is greater than the average monthly returns of MSFT H0 : mean(r_AAPL) > mean(r_MSFT) ==> H0: mean(r_AAPL) - mean(r_MSFT) = 0 Ha: mean(r_AAPL) < mean(r_MSFT) ==> Ha: mean(r_AAPL) - mean(r_MSFT) != 0

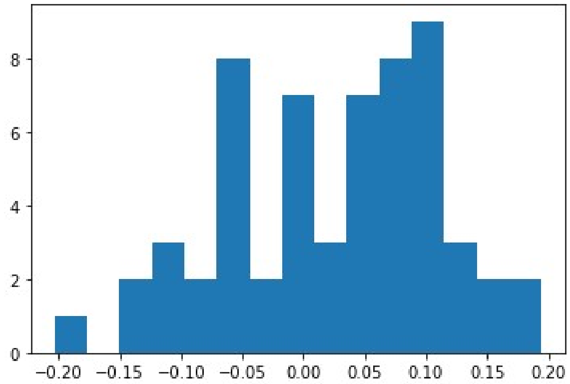
VARIABLE OF STUDY = DIFFERENCE OF BOTH MEAN RETURNS dif = mean(r_AAPL) - mean(r_MSFT) H0: dif = 0 Ha: dif != 0

AAPL

```
# Collecting the real data from AAPL from July 2019 until July 2022 monthly (60 months)
AAPL = pdr.get_data_yahoo("AAPL", start = "2017-07-01", end = "2022-05-31", interval = "m")
AAPL.tail()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2022-02-01	176.649994	152.000000	174.009995	165.119995	1.627516e+09	164.439545
2022-03-01	179.610001	150.100006	164.699997	174.610001	2.180800e+09	174.111984
2022-04-01	178.490005	155.380005	174.029999	157.649994	1.687796e+09	157.200348
2022-05-01	166.479996	132.610001	156.710007	148.839996	2.401040e+09	148.415466
2022-06-01	151.740005	129.039993	149.899994	136.720001	1.749100e+09	136.530350

```python
# Calculating the monthly cc returns
AAPL["r"] = (np.log(AAPL["Adj Close"]) - np.log(AAPL["Adj Close"].shift(1)))
# Plotting the cc returns
plt.hist(AAPL['r'], bins=15)
plt.show()
```




```python
# We calculate the mean of the cc returns
AAPL_mean = AAPL['r'].mean()
AAPL_mean
```

```
0.022998397443621706
```

```python
# We calculate the std of the group
AAPL_var = AAPL['r'].var()
AAPL_var
```

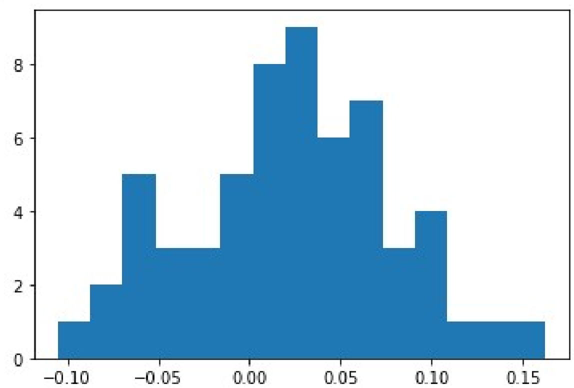
```
0.007557693200442321
```

MSFT

```
# Collecting the real data from returns of MSFT from July 2019 until July 2022 monthly
MSFT = pdr.get_data_yahoo("MSFT", start = "2017-07-01", end = "2022-05-31", interval = "m")
MSFT.tail()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2022-02-01	315.119995	271.519989	310.410004	298.790009	697050600.0	296.850494
2022-03-01	315.950012	270.000000	296.399994	308.309998	734334200.0	306.942047
2022-04-01	315.109985	270.000000	309.369995	277.519989	627343400.0	276.288666
2022-05-01	290.880005	246.440002	277.709991	271.869995	742902000.0	270.663757
2022-06-01	277.690002	241.509995	275.200012	256.829987	621372300.0	256.285980

```
# Calculating the monthly cc returns
MSFT["r"] = np.log(MSFT["Adj Close"]) - np.log(MSFT["Adj Close"].shift(1))
# Plotting the cc returns
plt.hist(MSFT["r"], bins = 15)
plt.show()
```



```
# Calculating the mean of the cc returns
MSFT_mean = MSFT["r"].mean()
MSFT_mean
```

0.02250023878448932

```
# Calculating the standard deviation of the cc returns
MSFT_var = MSFT["r"].var()
MSFT_var
```

0.0032421207830642673

Calculating the t-statistic

t = ((mean(r_AAPL) - mean(r_MSFT) - 0) / se => Remember that the standard error is the standard deviation of the variable of study.

Then: t = ((mean(r_AAPL) - mean(r_MSFT) - 0) / SD(mean(r_APPL) - mean(r_MSFT)) => The standard deviation can be calculated with the squared root of the variance of this difference

Then: t = ((mean(r_AAPL) - mean(r_MSFT) - 0) / sqrt((1/N)(Var(r_AAPL) + Var(r_MSFT)))

```
# Calculating the t-statisitc
t = (AAPL_mean - MSFT_mean -0) / np.sqrt((1 / MSFT["r"].count()) * (AAPL_var + MSFT_var))
t
```

0.036820149874669555

SINCE THE RESULT OF THE T-STATISTIC IS THAT THE DIFFERENCE BETWEEN THE TWO MEANS IS 0.037 STANDARD DEVIATIONS, WE CANNOT DETERMINE THAT NEITHER OF THE HYPOTHESIS ARE CONFIRMED.